4.2 Prefix Codes

Definition 4.9 A code is called a prefix-free code if no codeword is a prefix of any other codeword. For brevity, a prefix-free code will be referred to as a prefix code.

Definition 4.9 A code is called a prefix-free code if no codeword is a prefix of any other codeword. For brevity, a prefix-free code will be referred to as a prefix code.

Example 4.10

x	$\mathcal{C}'(x)$
A	0
B	10
C	110
D	1111

Definition 4.9 A code is called a prefix-free code if no codeword is a prefix of any other codeword. For brevity, a prefix-free code will be referred to as a prefix code.

Example 4.10

x	$\mathcal{C}'(x)$
A	0
B	10
C	110
D	1111

Definition 4.9 A code is called a prefix-free code if no codeword is a prefix of any other codeword. For brevity, a prefix-free code will be referred to as a prefix code.

Example 4.10

x	$\mathcal{C}'(x)$
A	0
B	10
C	110
D	1111

Definition 4.9 A code is called a prefix-free code if no codeword is a prefix of any other codeword. For brevity, a prefix-free code will be referred to as a prefix code.

Example 4.10

x	$\mathcal{C}'(x)$
A	0
B	10
C	110
D	1111

Definition 4.9 A code is called a prefix-free code if no codeword is a prefix of any other codeword. For brevity, a prefix-free code will be referred to as a prefix code.

Example 4.10

x	$\mathcal{C}'(x)$
A	0
B	10
C	110
D	1111

Code Tree for Prefix Code

- A D-ary tree is a graphical representation of a collection of finite sequences of D-ary symbols.

Code Tree for Prefix Code

- A D-ary tree is a graphical representation of a collection of finite sequences of D-ary symbols.
- A node is either an internal node or a leaf.

Code Tree for Prefix Code

- A D-ary tree is a graphical representation of a collection of finite sequences of D-ary symbols.
- A node is either an internal node or a leaf.
- The tree representation of a prefix code is called a code tree.

Instantaneous Decoding

- Using the code $\mathcal{C}'$ in Example 4.10:

$$BCDAC \cdots \rightarrow 1011011110110 \cdots$$

x	$\mathcal{C}'(x)$
A	0
B	10
C	110
D	1111

Instantaneous Decoding

- Using the code $\mathcal{C}'$ in Example 4.10:

$$\underline{B}CDAC \cdots \rightarrow 1011011110110 \cdots$$

x	$\mathcal{C}'(x)$
A	0
B	10
C	110
D	1111

Instantaneous Decoding

- Using the code $\mathcal{C}'$ in Example 4.10:

$$BCDAC\cdots \rightarrow \underline{10}11011110110\cdots$$

x	$\mathcal{C}'(x)$
A	0
B	10
C	110
D	1111

Instantaneous Decoding

- Using the code $\mathcal{C}'$ in Example 4.10:

$$B\underline{C}DAC \cdots \to 1011011110110 \cdots$$

x	$\mathcal{C}'(x)$
A	0
B	10
C	110
D	1111

Instantaneous Decoding

- Using the code $\mathcal{C}'$ in Example 4.10:

$$BCDAC \cdots \rightarrow 10\underline{110}11110110 \cdots$$

x	$\mathcal{C}'(x)$
A	0
B	10
C	110
D	1111

Instantaneous Decoding

- Using the code $\mathcal{C}'$ in Example 4.10:

$$BC\underline{D}AC \cdots \rightarrow 1011011110110 \cdots$$

x	$\mathcal{C}'(x)$
A	0
B	10
C	110
D	1111

Instantaneous Decoding

- Using the code $\mathcal{C}'$ in Example 4.10:

$$BCDAC \cdots \rightarrow 10110\underline{1111}0110 \cdots$$

x	$\mathcal{C}'(x)$
A	0
B	10
C	110
D	1111

Instantaneous Decoding

- Using the code $\mathcal{C}'$ in Example 4.10:

$$BCD\underline{A}C\cdots \to 1011011110110\cdots$$

x	$\mathcal{C}'(x)$
A	0
B	10
C	110
D	1111

Instantaneous Decoding

- Using the code $\mathcal{C}'$ in Example 4.10:

$$BCDAC \cdots \rightarrow 1011011110110 \cdots$$

x	$\mathcal{C}'(x)$
A	0
B	10
C	110
D	1111

Instantaneous Decoding

- Using the code $\mathcal{C}'$ in Example 4.10:

$$BCDA\underline{C}\cdots \rightarrow 1011011110110\cdots$$

x	$\mathcal{C}'(x)$
A	0
B	10
C	110
D	1111

Instantaneous Decoding

- Using the code $\mathcal{C}'$ in Example 4.10:

$$BCDAC\cdots \rightarrow 1011011110\underline{110}\cdots$$

x	$\mathcal{C}'(x)$
A	0
B	10
C	110
D	1111

Instantaneous Decoding

- Using the code $\mathcal{C}'$ in Example 4.10:

$$BCDAC \cdots \to 1011011110110 \cdots$$

- Upon concatenating the codewords, their boundaries are no longer explicit.

x	$\mathcal{C}'(x)$
A	0
B	10
C	110
D	1111

Instantaneous Decoding

- Using the code $\mathcal{C}'$ in Example 4.10:

$$BCDAC \cdots \to 1011011110110 \cdots$$

- Upon concatenating the codewords, their boundaries are no longer explicit.
- The stream of coded symbols are then transmitted to the receiver.

x	$\mathcal{C}'(x)$
A	0
B	10
C	110
D	1111

Instantaneous Decoding

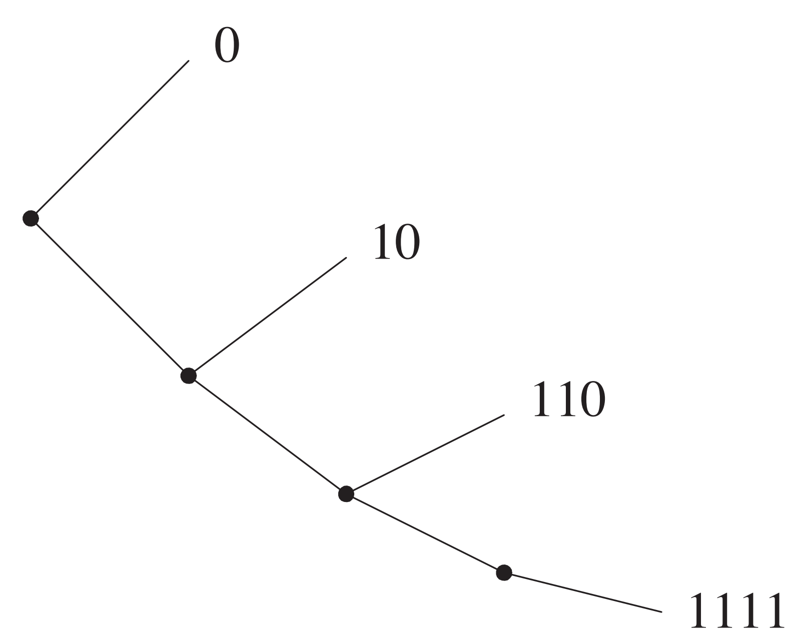
- Since $\mathcal{C}'$ is a prefix code, the codewords can be represented by a code tree.

x	$\mathcal{C}'(x)$
A	0
B	10
C	110
D	1111

Instantaneous Decoding

- Since $\mathcal{C}'$ is a prefix code, the codewords can be represented by a code tree.

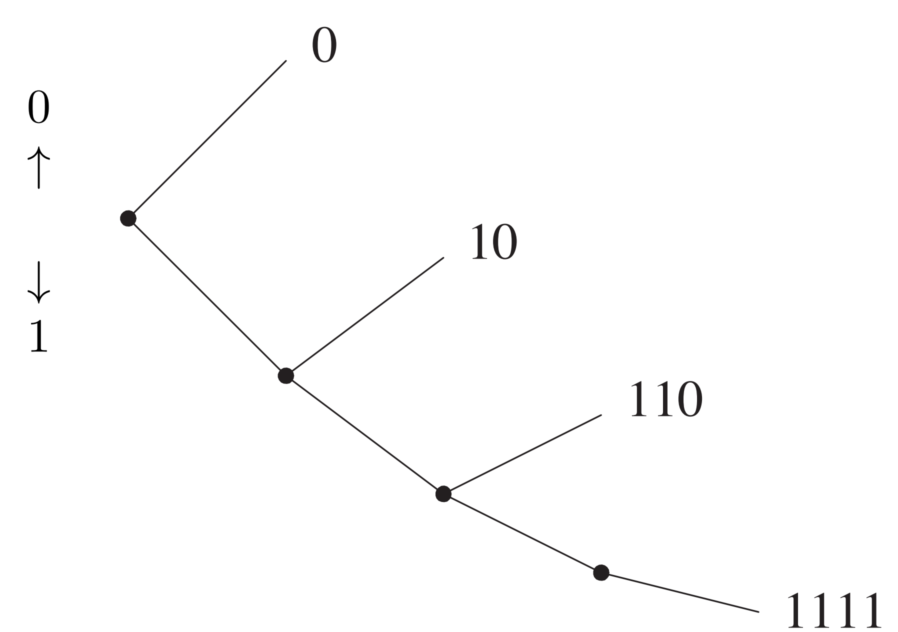
x	$\mathcal{C}'(x)$
A	0
B	10
C	110
D	1111



Instantaneous Decoding

- Since $\mathcal{C}'$ is a prefix code, the codewords can be represented by a code tree.

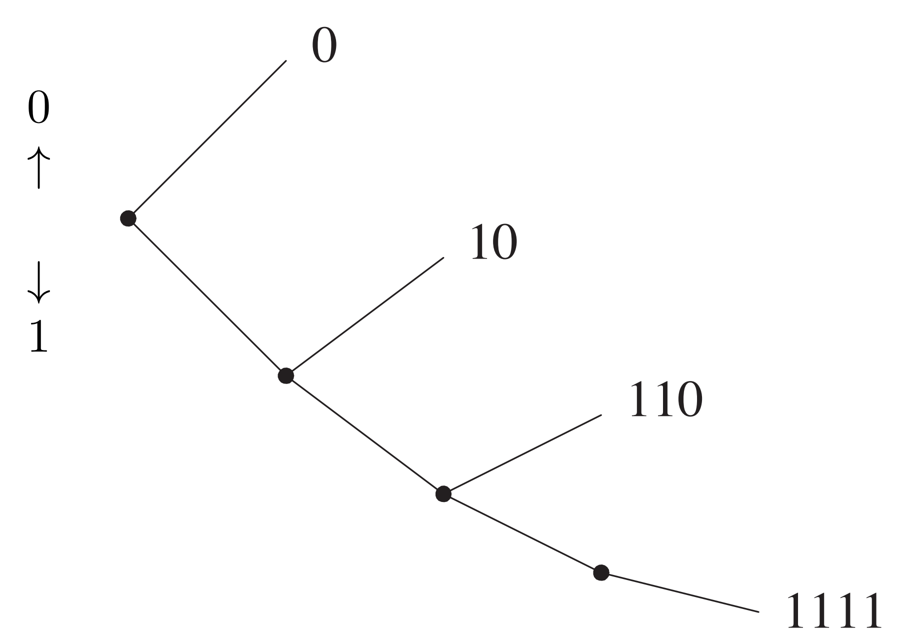
x	$\mathcal{C}'(x)$
A	0
B	10
C	110
D	1111



Instantaneous Decoding

- Since $\mathcal{C}'$ is a prefix code, the codewords can be represented by a code tree.

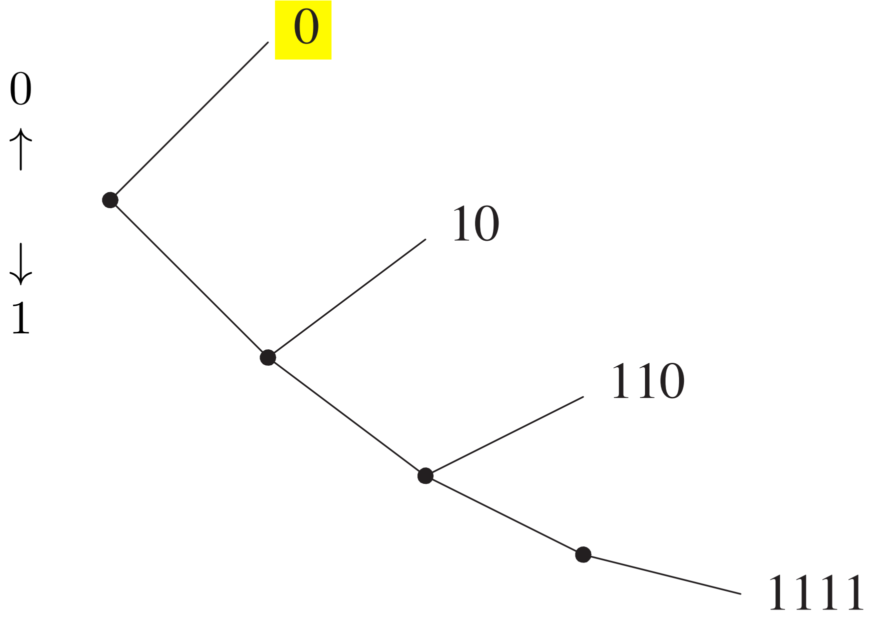
x	$\mathcal{C}'(x)$
A	0
B	10
C	110
D	1111



Instantaneous Decoding

- Since $\mathcal{C}'$ is a prefix code, the codewords can be represented by a code tree.

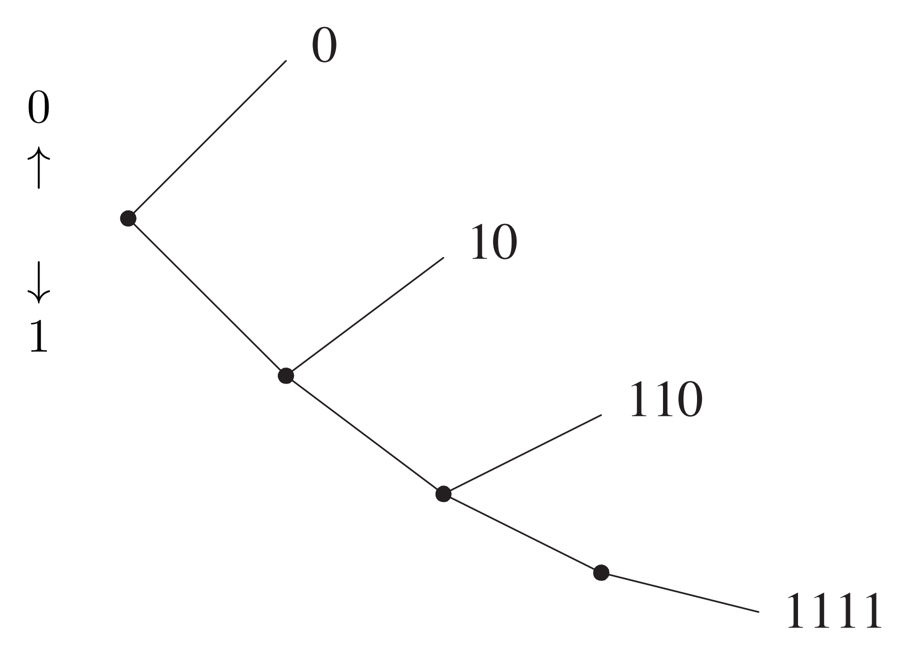
x	$\mathcal{C}'(x)$
A	0
B	10
C	110
D	1111



Instantaneous Decoding

- Since $\mathcal{C}'$ is a prefix code, the codewords can be represented by a code tree.

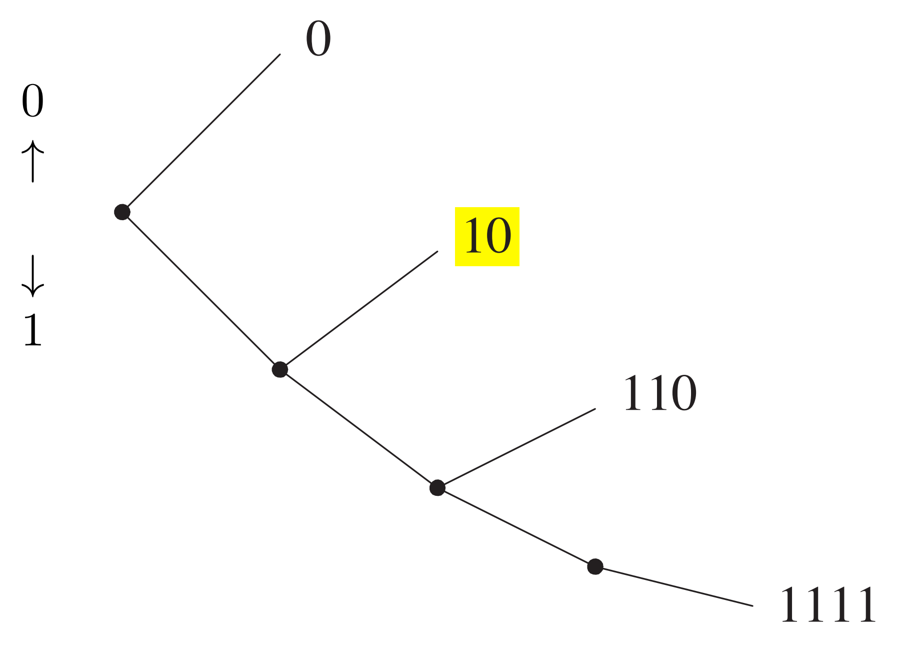
x	$\mathcal{C}'(x)$
A	0
B	10
C	110
D	1111



Instantaneous Decoding

- Since $\mathcal{C}'$ is a prefix code, the codewords can be represented by a code tree.

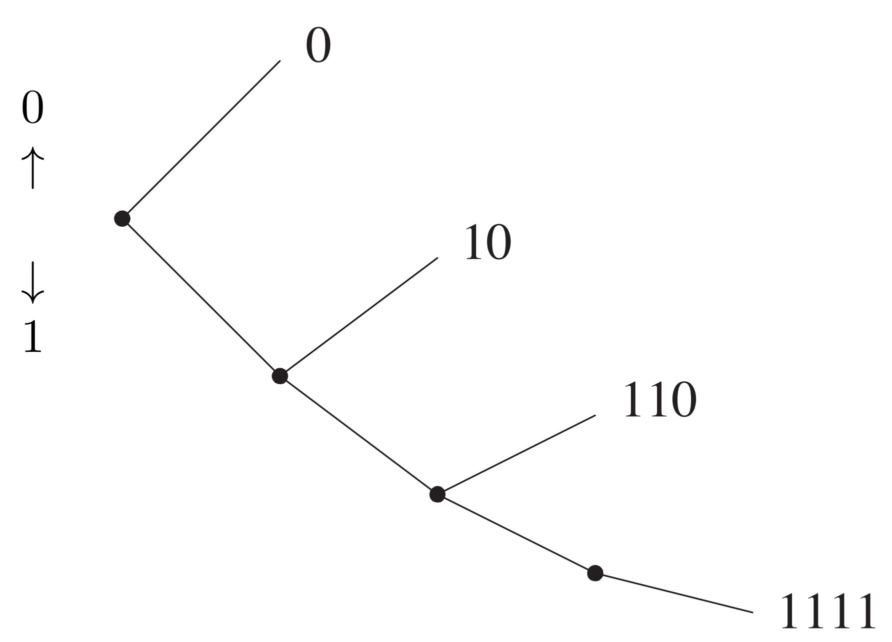
x	$\mathcal{C}'(x)$
A	0
B	10
C	110
D	1111



Instantaneous Decoding

- Since $\mathcal{C}'$ is a prefix code, the codewords can be represented by a code tree.

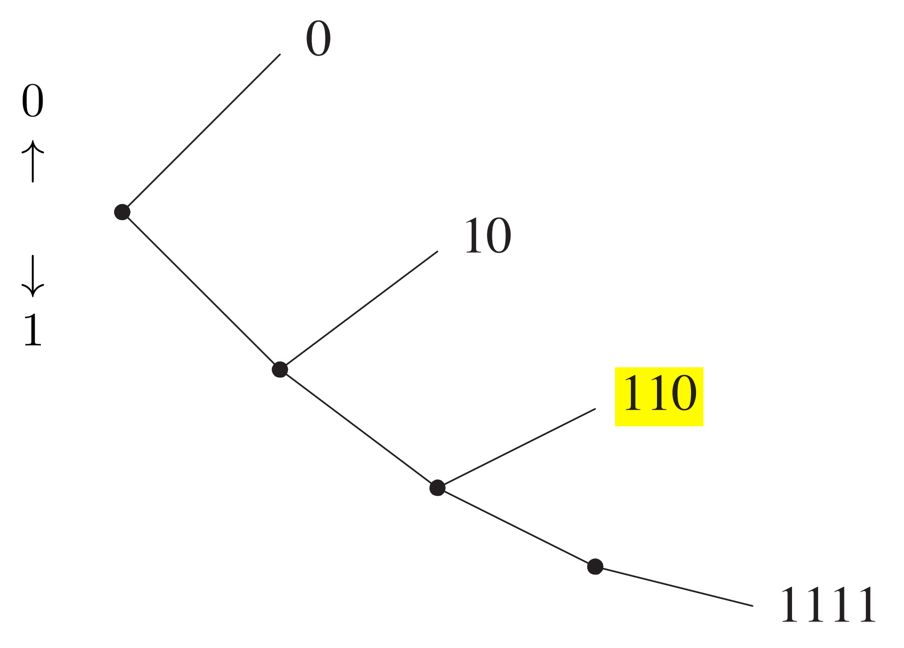
x	$\mathcal{C}'(x)$
A	0
B	10
C	110
D	1111



Instantaneous Decoding

- Since $\mathcal{C}'$ is a prefix code, the codewords can be represented by a code tree.

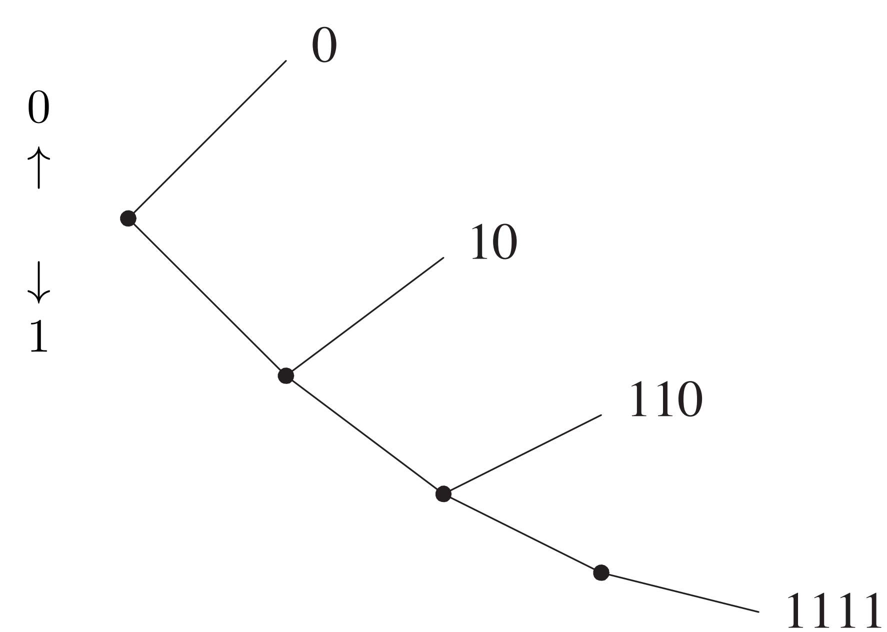
x	$\mathcal{C}'(x)$
A	0
B	10
C	110
D	1111



Instantaneous Decoding

- Since $\mathcal{C}'$ is a prefix code, the codewords can be represented by a code tree.

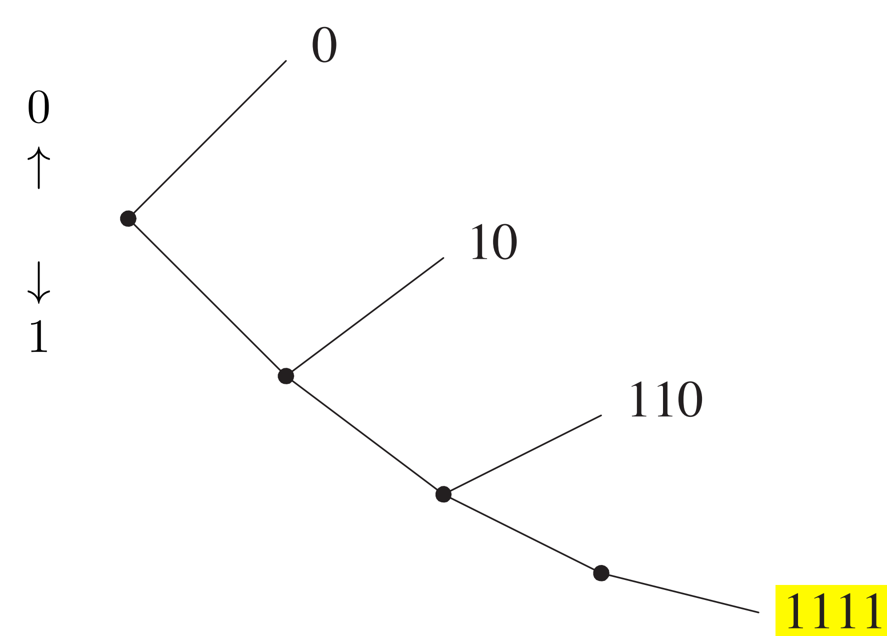
x	$\mathcal{C}'(x)$
A	0
B	10
C	110
D	1111



Instantaneous Decoding

- Since $\mathcal{C}'$ is a prefix code, the codewords can be represented by a code tree.

x	$\mathcal{C}'(x)$
A	0
B	10
C	110
D	1111

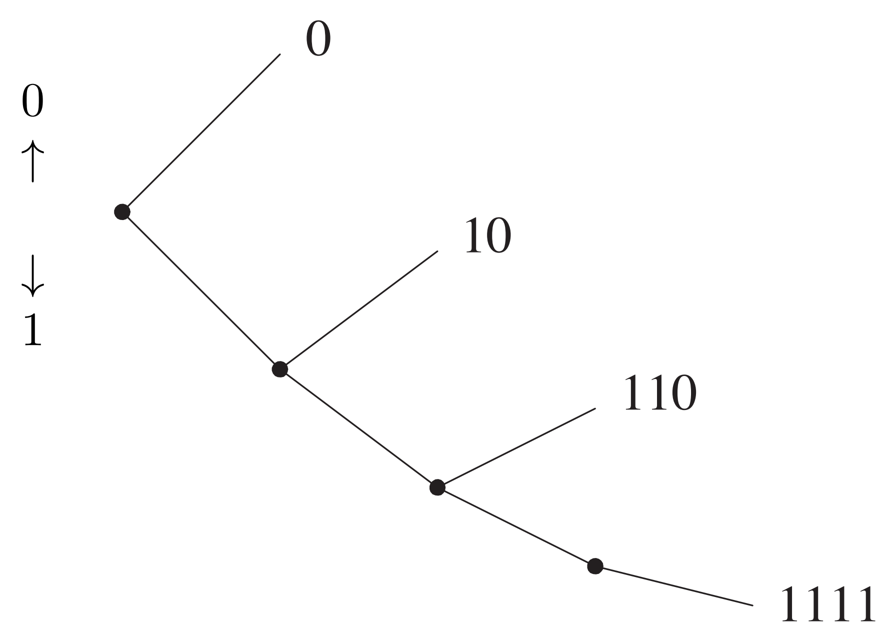


Instantaneous Decoding

- Since $\mathcal{C}'$ is a prefix code, the codewords can be represented by a code tree.
- Instantaneous decoding at the receiver can be done by tracing the code tree from the root:

$$1011011110110\cdots \to 10, 110, 1111, 0, 110, \cdots$$

x	$\mathcal{C}'(x)$
A	0
B	10
C	110
D	1111

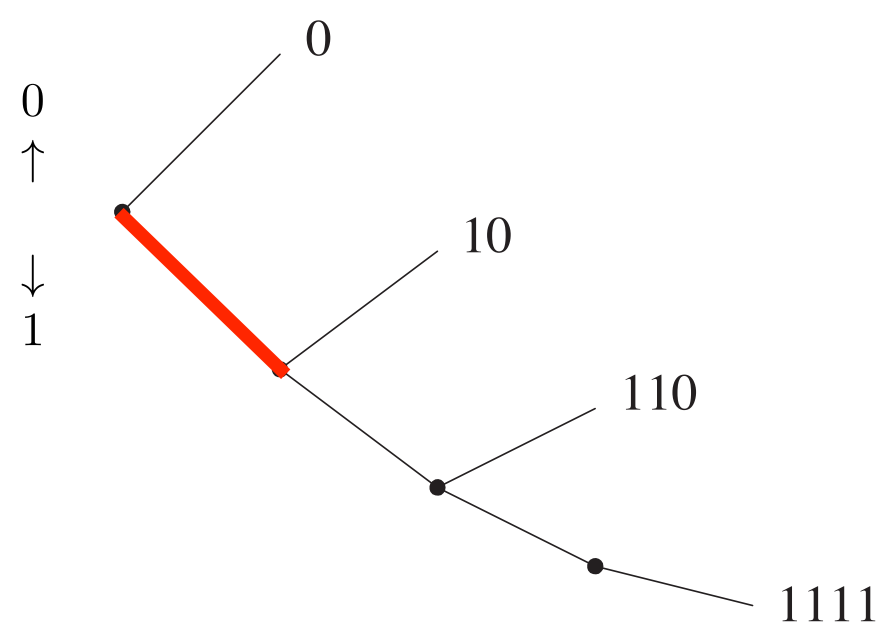


Instantaneous Decoding

- Since $\mathcal{C}'$ is a prefix code, the codewords can be represented by a code tree.
- Instantaneous decoding at the receiver can be done by tracing the code tree from the root:

$$1011011110110\cdots \to 10, 110, 1111, 0, 110, \cdots$$

x	$\mathcal{C}'(x)$
A	0
B	10
C	110
D	1111

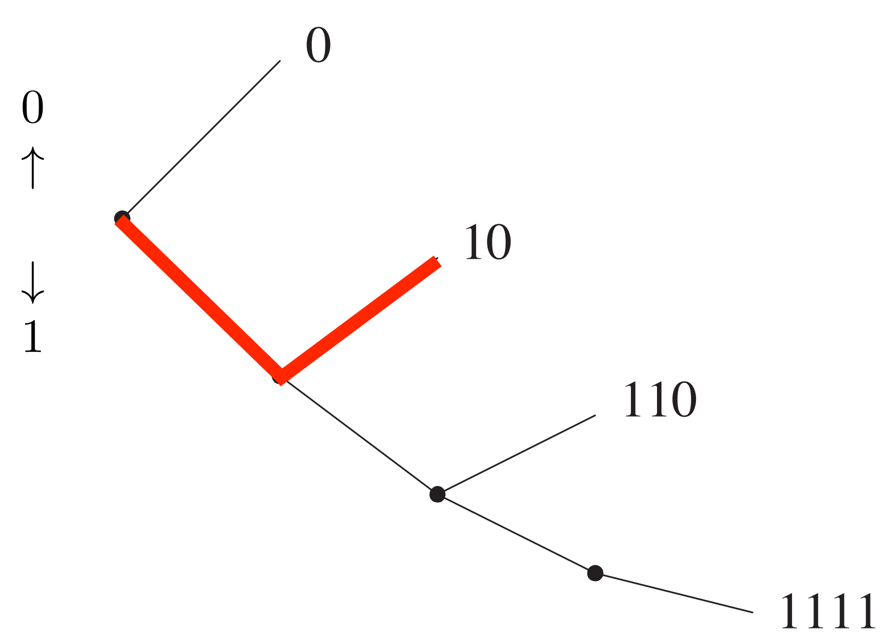


Instantaneous Decoding

- Since $\mathcal{C}'$ is a prefix code, the codewords can be represented by a code tree.
- Instantaneous decoding at the receiver can be done by tracing the code tree from the root:

$$1011011110110\cdots \to 10, 110, 1111, 0, 110, \cdots$$

x	$\mathcal{C}'(x)$
A	0
B	10
C	110
D	1111

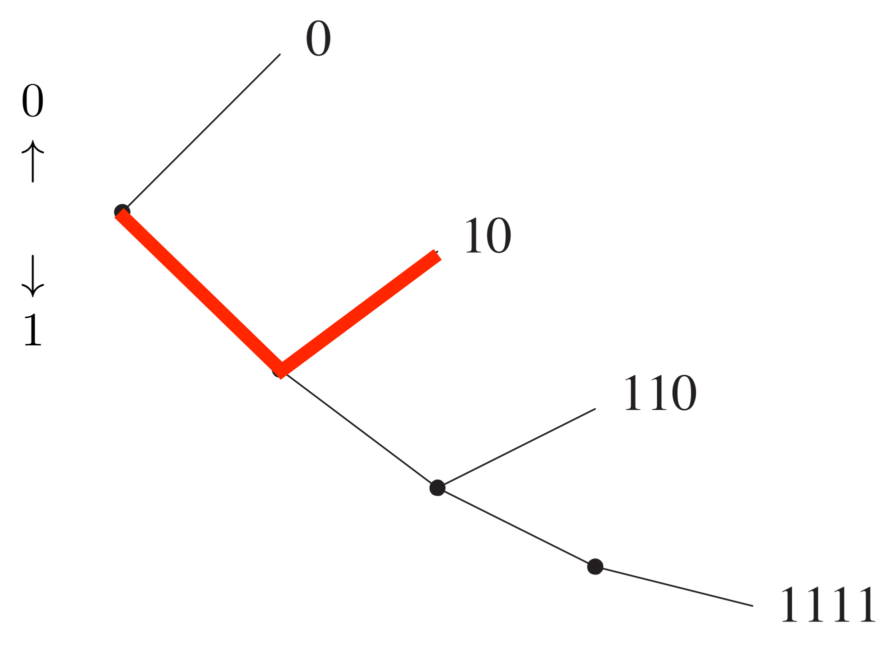


Instantaneous Decoding

- Since $\mathcal{C}'$ is a prefix code, the codewords can be represented by a code tree.
- Instantaneous decoding at the receiver can be done by tracing the code tree from the root:

$$1011011110110\cdots \rightarrow \underline{10}, 110, 1111, 0, 110, \cdots$$

x	$\mathcal{C}'(x)$
A	0
B	10
C	110
D	1111

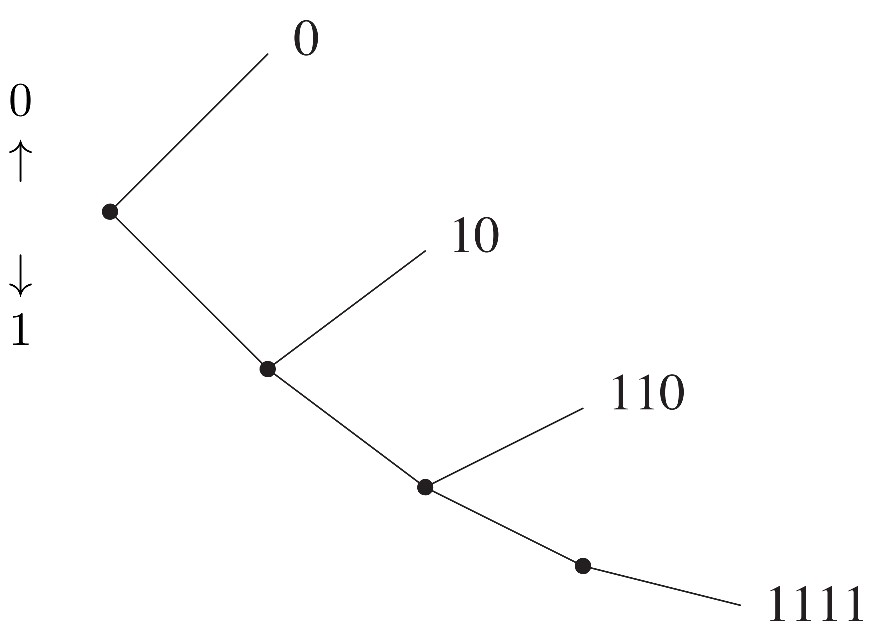


Instantaneous Decoding

- Since $\mathcal{C}'$ is a prefix code, the codewords can be represented by a code tree.
- Instantaneous decoding at the receiver can be done by tracing the code tree from the root:

$$1011011110110 \cdots \to 10, 110, 1111, 0, 110, \cdots$$

x	$\mathcal{C}'(x)$
A	0
B	10
C	110
D	1111

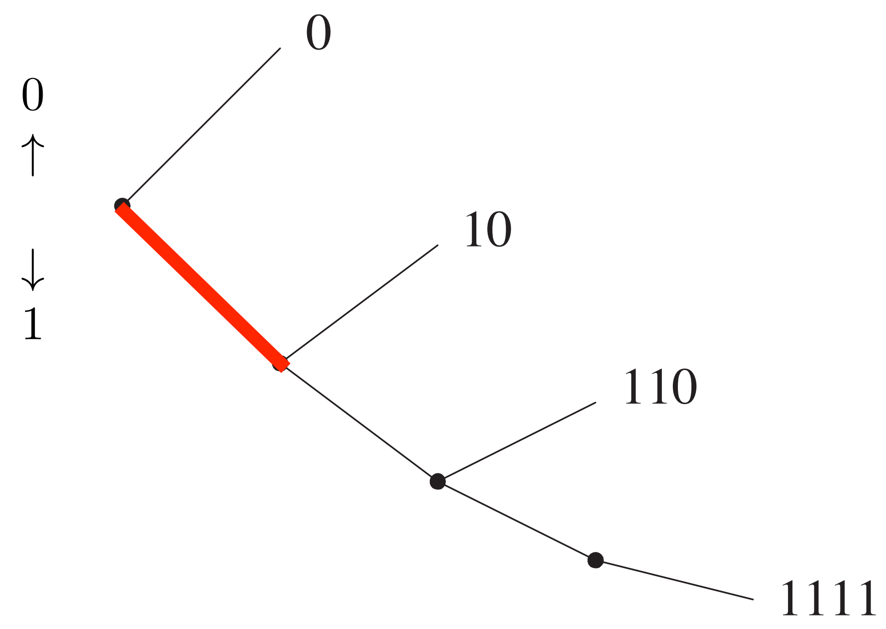


Instantaneous Decoding

- Since $\mathcal{C}'$ is a prefix code, the codewords can be represented by a code tree.
- Instantaneous decoding at the receiver can be done by tracing the code tree from the root:

$$1011011110110\cdots \to 10, 110, 1111, 0, 110, \cdots$$

x	$\mathcal{C}'(x)$
A	0
B	10
C	110
D	1111

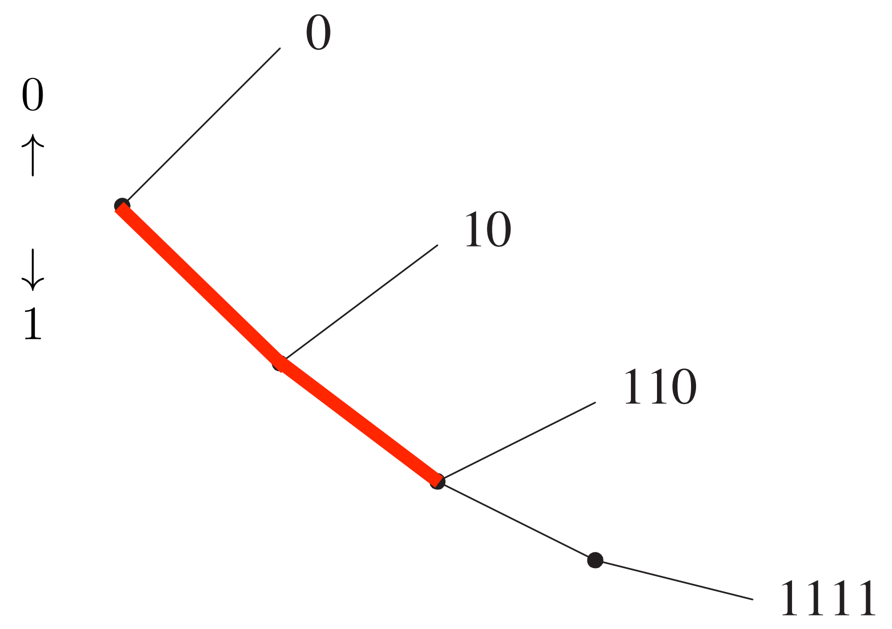


Instantaneous Decoding

- Since $\mathcal{C}'$ is a prefix code, the codewords can be represented by a code tree.
- Instantaneous decoding at the receiver can be done by tracing the code tree from the root:

$$1011011110110\cdots \to 10, 110, 1111, 0, 110, \cdots$$

x	$\mathcal{C}'(x)$
A	0
B	10
C	110
D	1111

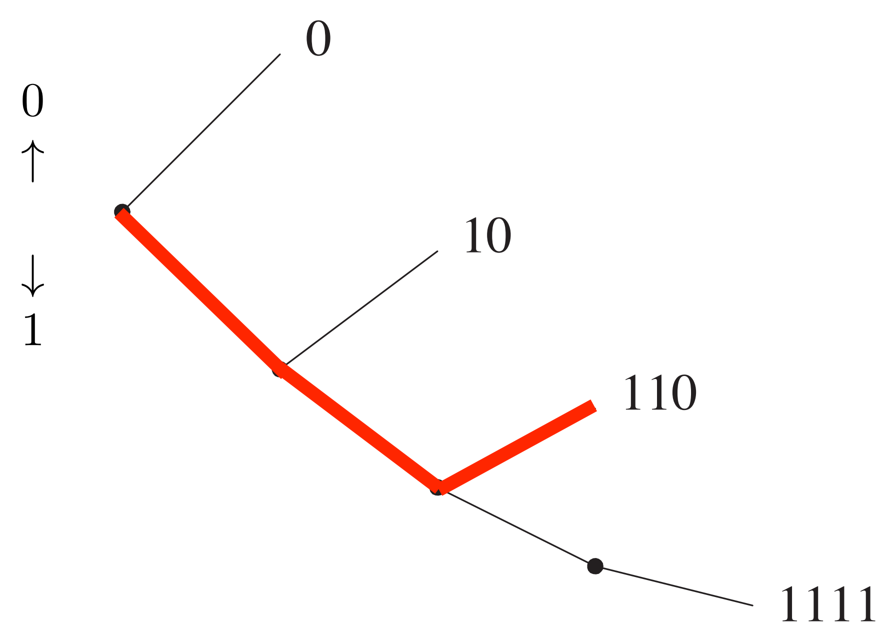


Instantaneous Decoding

- Since $\mathcal{C}'$ is a prefix code, the codewords can be represented by a code tree.
- Instantaneous decoding at the receiver can be done by tracing the code tree from the root:

$$1011011110110\cdots \to 10, 110, 1111, 0, 110, \cdots$$

x	$\mathcal{C}'(x)$
A	0
B	10
C	110
D	1111

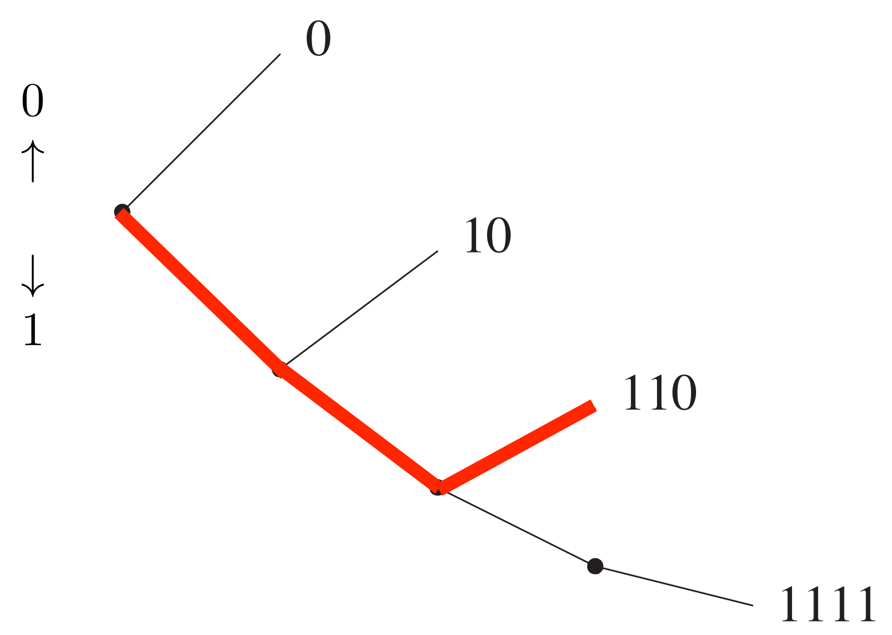


Instantaneous Decoding

- Since $\mathcal{C}'$ is a prefix code, the codewords can be represented by a code tree.
- Instantaneous decoding at the receiver can be done by tracing the code tree from the root:

$$1011011110110\cdots \to 10, \underline{110}, 1111, 0, 110, \cdots$$

x	$\mathcal{C}'(x)$
A	0
B	10
C	110
D	1111

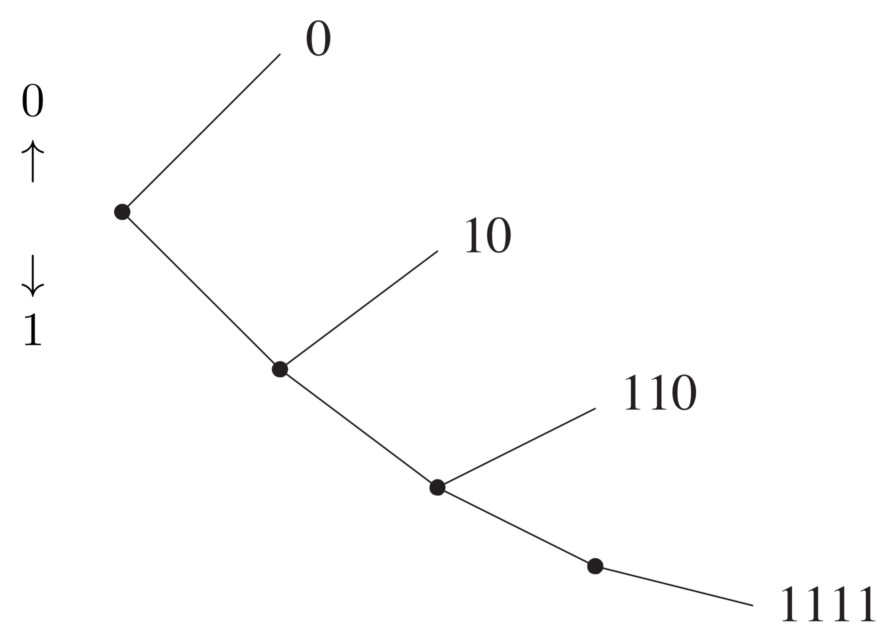


Instantaneous Decoding

- Since $\mathcal{C}'$ is a prefix code, the codewords can be represented by a code tree.
- Instantaneous decoding at the receiver can be done by tracing the code tree from the root:

$$1011011110110\cdots \to 10, 110, 1111, 0, 110, \cdots$$

x	$\mathcal{C}'(x)$
A	0
B	10
C	110
D	1111

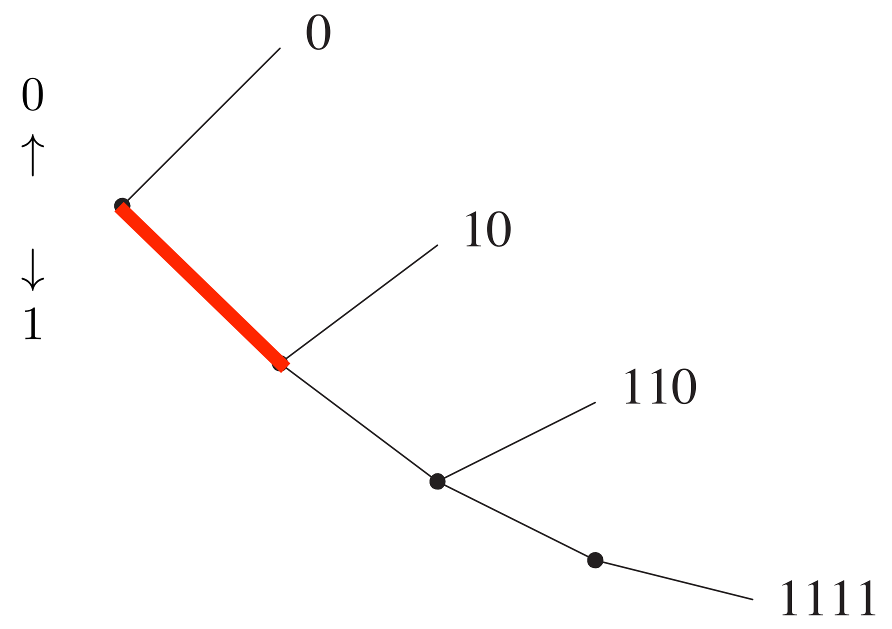


Instantaneous Decoding

- Since $\mathcal{C}'$ is a prefix code, the codewords can be represented by a code tree.
- Instantaneous decoding at the receiver can be done by tracing the code tree from the root:

$$1011011110110 \cdots \to 10, 110, 1111, 0, 110, \cdots$$

x	$\mathcal{C}'(x)$
A	0
B	10
C	110
D	1111

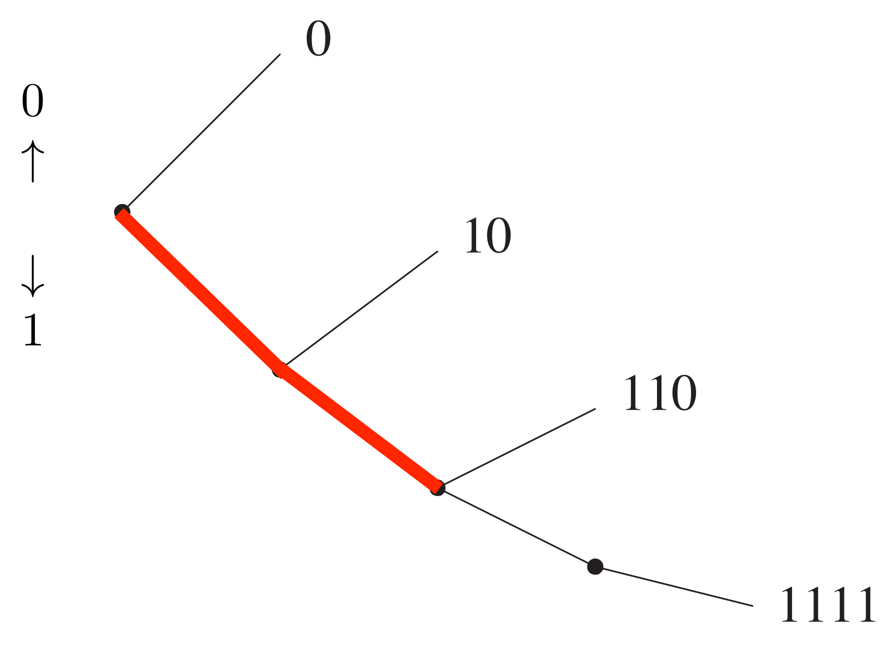


Instantaneous Decoding

- Since $\mathcal{C}'$ is a prefix code, the codewords can be represented by a code tree.
- Instantaneous decoding at the receiver can be done by tracing the code tree from the root:

$$1011011110110\cdots \to 10, 110, 1111, 0, 110, \cdots$$

x	$\mathcal{C}'(x)$
A	0
B	10
C	110
D	1111

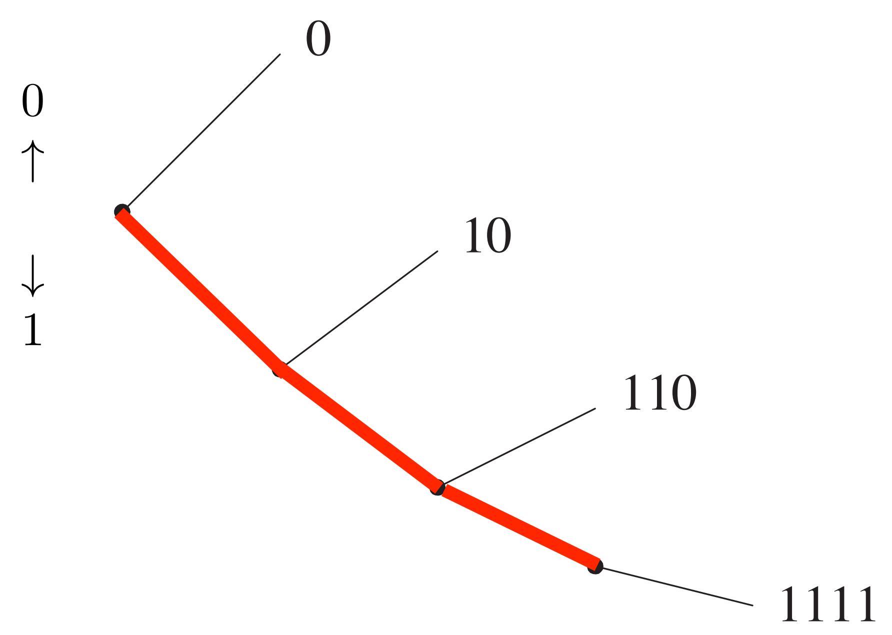


Instantaneous Decoding

- Since $\mathcal{C}'$ is a prefix code, the codewords can be represented by a code tree.
- Instantaneous decoding at the receiver can be done by tracing the code tree from the root:

$$1011011110110\cdots \to 10, 110, 1111, 0, 110, \cdots$$

x	$\mathcal{C}'(x)$
A	0
B	10
C	110
D	1111

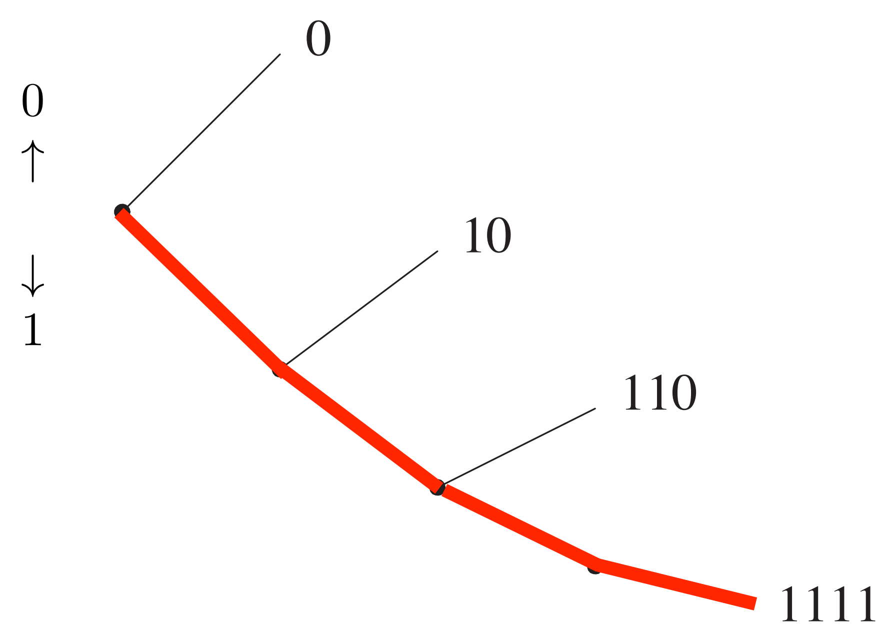


Instantaneous Decoding

- Since $\mathcal{C}'$ is a prefix code, the codewords can be represented by a code tree.
- Instantaneous decoding at the receiver can be done by tracing the code tree from the root:

$$1011011110110\cdots \to 10, 110, 1111, 0, 110, \cdots$$

x	$\mathcal{C}'(x)$
A	0
B	10
C	110
D	1111

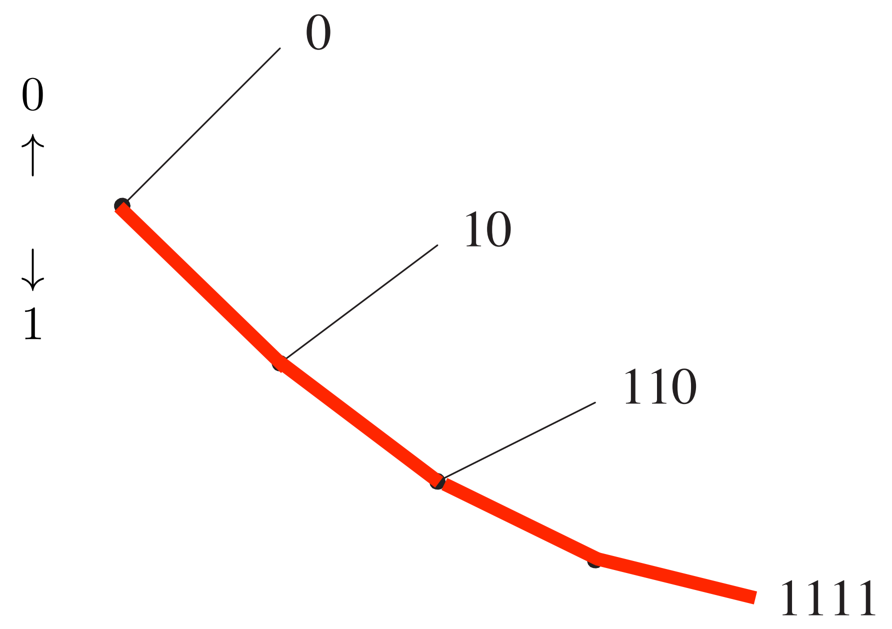


Instantaneous Decoding

- Since $\mathcal{C}'$ is a prefix code, the codewords can be represented by a code tree.
- Instantaneous decoding at the receiver can be done by tracing the code tree from the root:

$$1011011110110\cdots \rightarrow 10, 110, \underline{1111}, 0, 110, \cdots$$

x	$\mathcal{C}'(x)$
A	0
B	10
C	110
D	1111

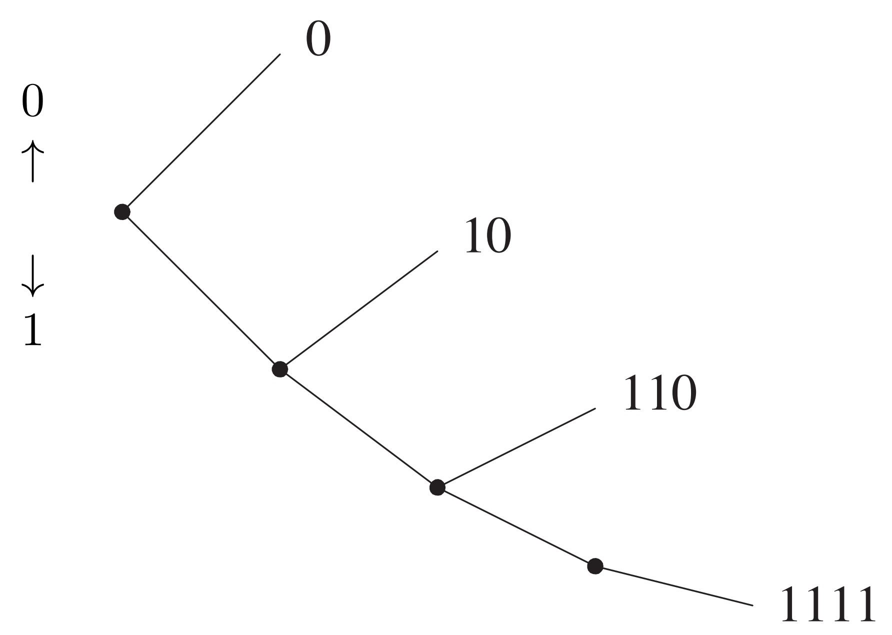


Instantaneous Decoding

- Since $\mathcal{C}'$ is a prefix code, the codewords can be represented by a code tree.
- Instantaneous decoding at the receiver can be done by tracing the code tree from the root:

$$1011011110110\cdots \rightarrow 10, 110, 1111, 0, 110, \cdots$$

x	$\mathcal{C}'(x)$
A	0
B	10
C	110
D	1111

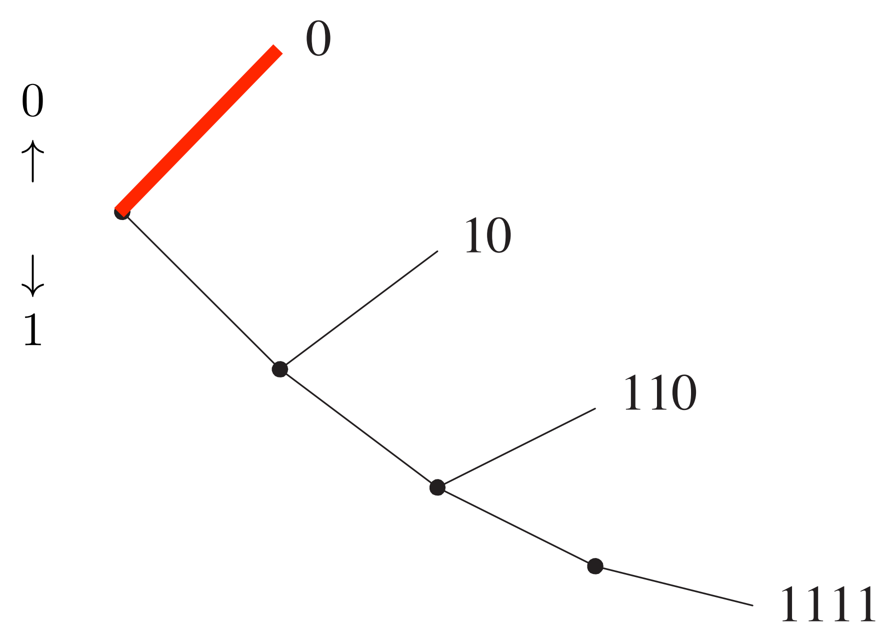


Instantaneous Decoding

- Since $\mathcal{C}'$ is a prefix code, the codewords can be represented by a code tree.
- Instantaneous decoding at the receiver can be done by tracing the code tree from the root:

$$1011011110110\cdots \rightarrow 10, 110, 1111, 0, 110, \cdots$$

x	$\mathcal{C}'(x)$
A	0
B	10
C	110
D	1111

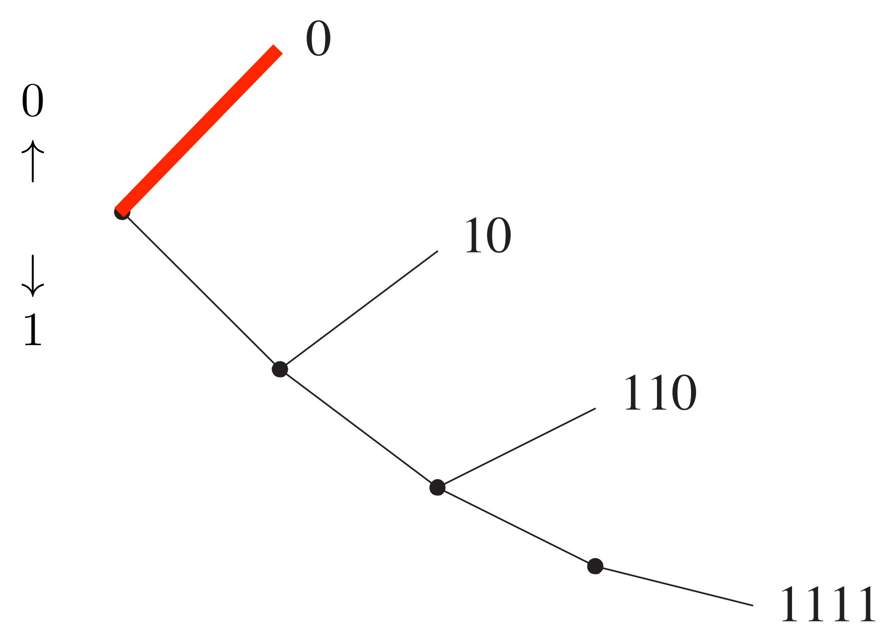


Instantaneous Decoding

- Since $\mathcal{C}'$ is a prefix code, the codewords can be represented by a code tree.
- Instantaneous decoding at the receiver can be done by tracing the code tree from the root:

$$1011011110110\cdots \rightarrow 10, 110, 1111, \underline{0}, 110, \cdots$$

x	$\mathcal{C}'(x)$
A	0
B	10
C	110
D	1111

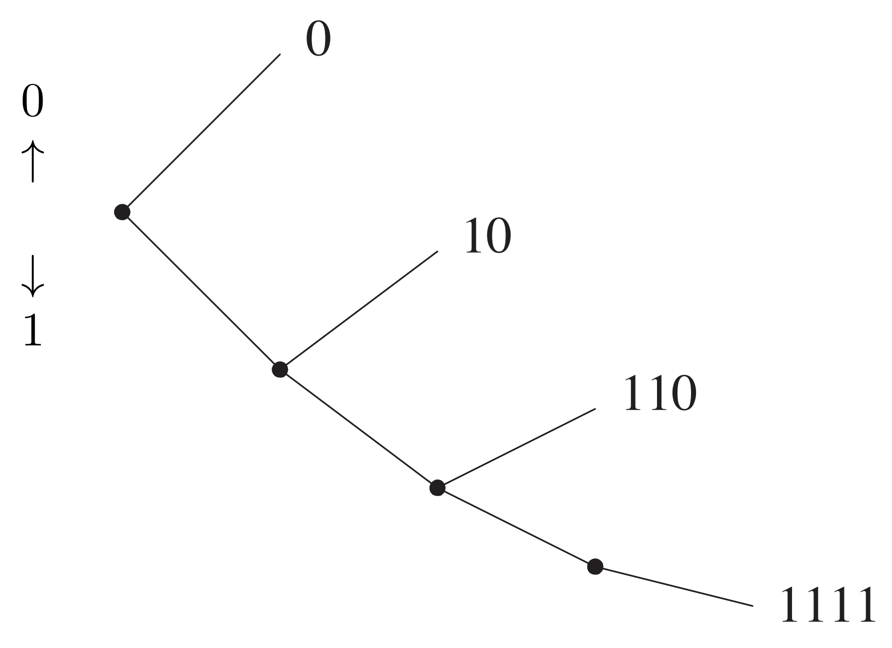


Instantaneous Decoding

- Since $\mathcal{C}'$ is a prefix code, the codewords can be represented by a code tree.
- Instantaneous decoding at the receiver can be done by tracing the code tree from the root:

$$1011011110110\cdots \to 10, 110, 1111, 0, 110, \cdots$$

x	$\mathcal{C}'(x)$
A	0
B	10
C	110
D	1111

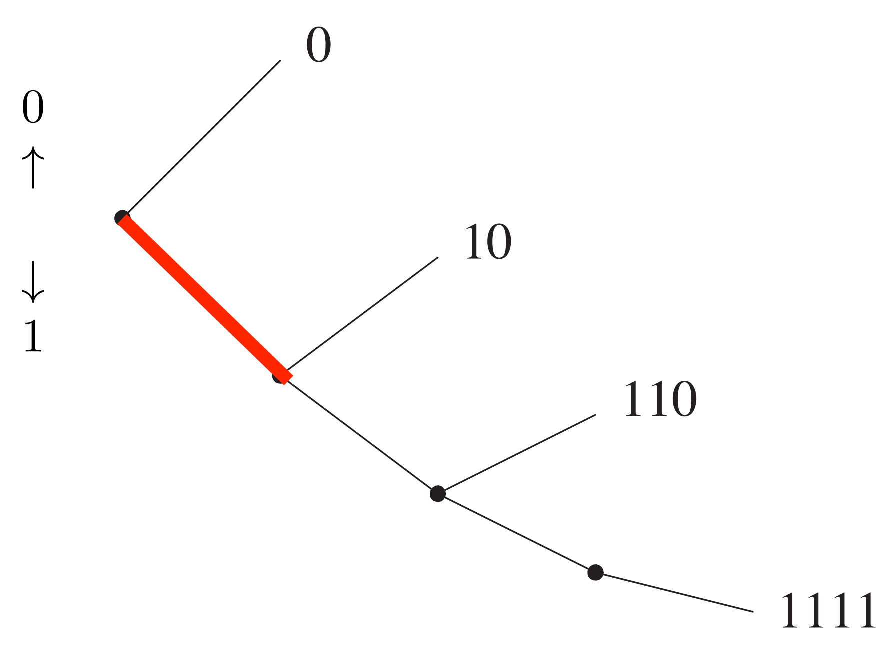


Instantaneous Decoding

- Since $\mathcal{C}'$ is a prefix code, the codewords can be represented by a code tree.
- Instantaneous decoding at the receiver can be done by tracing the code tree from the root:

$$1011011110110\cdots \to 10, 110, 1111, 0, 110, \cdots$$

x	$\mathcal{C}'(x)$
A	0
B	10
C	110
D	1111

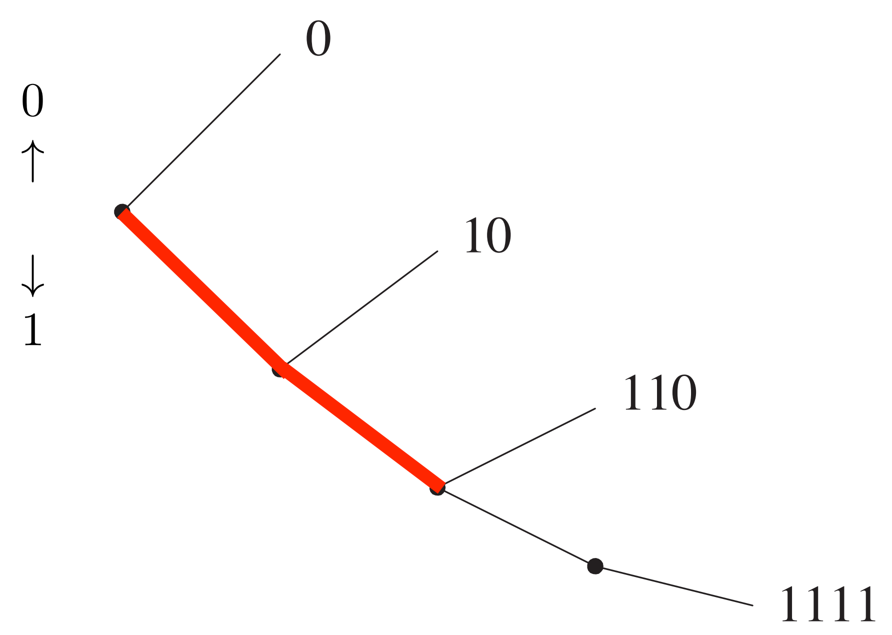


Instantaneous Decoding

- Since $\mathcal{C}'$ is a prefix code, the codewords can be represented by a code tree.
- Instantaneous decoding at the receiver can be done by tracing the code tree from the root:

$$1011011110110\cdots \to 10, 110, 1111, 0, 110, \cdots$$

x	$\mathcal{C}'(x)$
A	0
B	10
C	110
D	1111

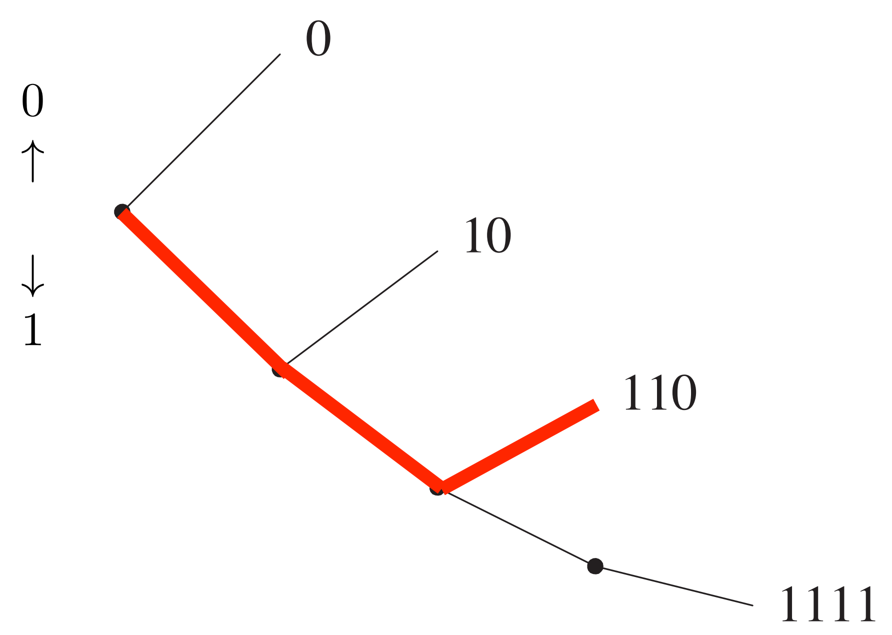


Instantaneous Decoding

- Since $\mathcal{C}'$ is a prefix code, the codewords can be represented by a code tree.
- Instantaneous decoding at the receiver can be done by tracing the code tree from the root:

$$1011011110110\cdots \to 10, 110, 1111, 0, 110, \cdots$$

x	$\mathcal{C}'(x)$
A	0
B	10
C	110
D	1111

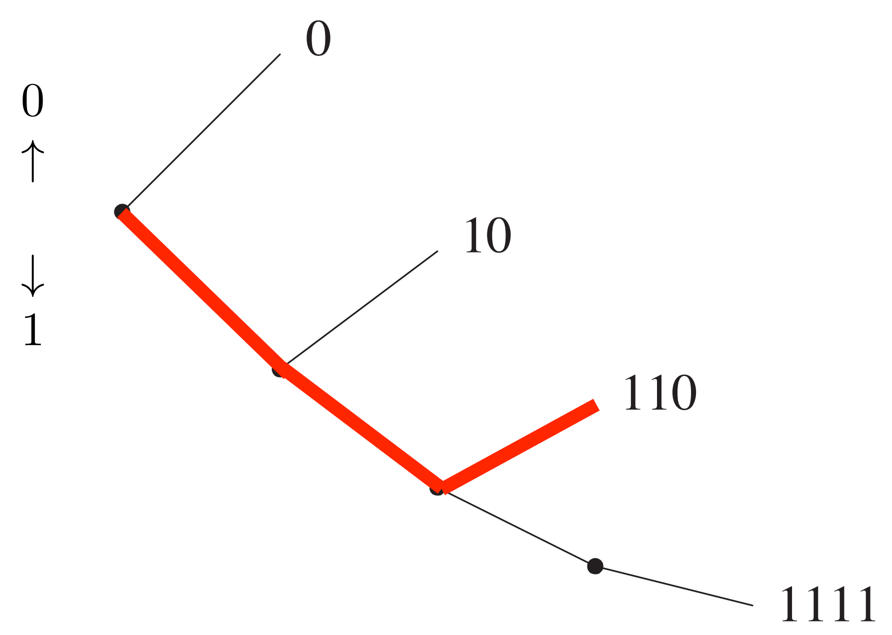


Instantaneous Decoding

- Since $\mathcal{C}'$ is a prefix code, the codewords can be represented by a code tree.
- Instantaneous decoding at the receiver can be done by tracing the code tree from the root:

$$1011011110110\cdots \rightarrow 10, 110, 1111, 0, \underline{110}, \cdots$$

x	$\mathcal{C}'(x)$
A	0
B	10
C	110
D	1111

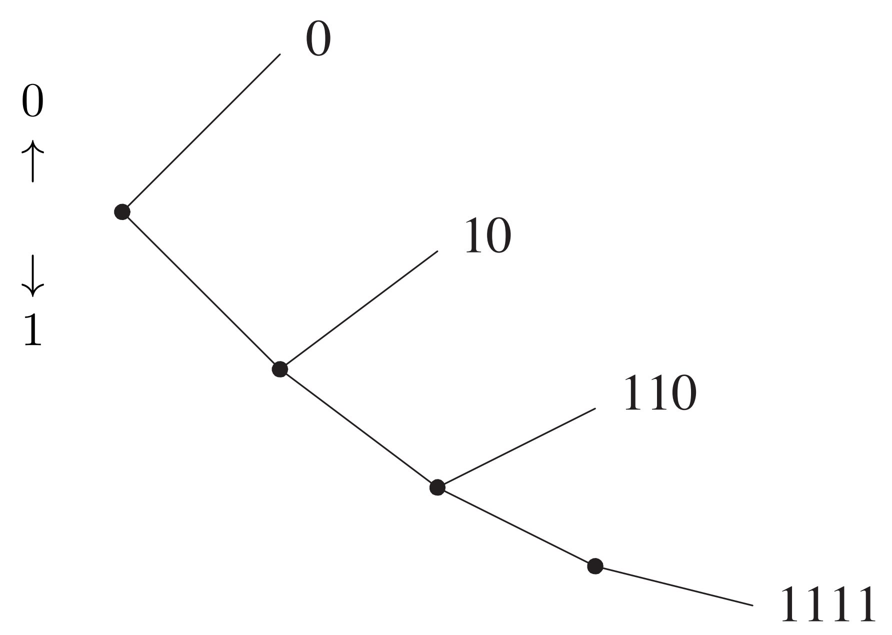


Instantaneous Decoding

- Since $\mathcal{C}'$ is a prefix code, the codewords can be represented by a code tree.
- Instantaneous decoding at the receiver can be done by tracing the code tree from the root:

$$1011011110110\cdots \to 10, 110, 1111, 0, 110, \cdots$$

x	$\mathcal{C}'(x)$
A	0
B	10
C	110
D	1111



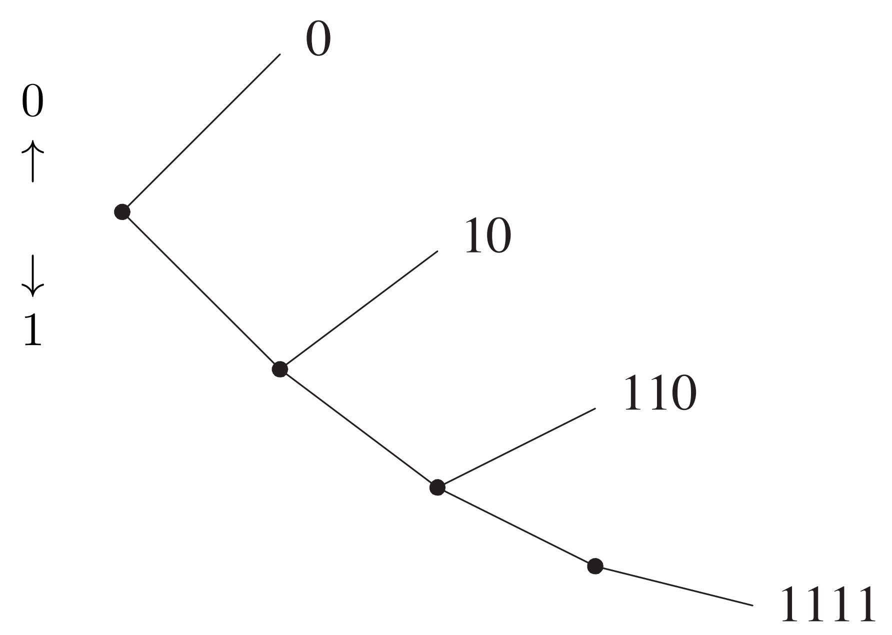
Instantaneous Decoding

- Since $\mathcal{C}'$ is a prefix code, the codewords can be represented by a code tree.
- Instantaneous decoding at the receiver can be done by tracing the code tree from the root:

$$1011011110110 \cdots \rightarrow 10, 110, 1111, 0, 110, \cdots$$

- The boundaries of the codewords can thus be recovered. In this sense, a prefix code is said to be self-punctuating.

x	$\mathcal{C}'(x)$
A	0
B	10
C	110
D	1111



Theorem 4.11 There exists a D-ary prefix code with codeword lengths l_1, $l_2, \cdots, l_m$ if and only if the Kraft inequality

$$\sum_{k=1}^{m} D^{-l_k} \leq 1$$

is satisfied.

Theorem 4.11 There exists a D-ary prefix code with codeword lengths l_1, $l_2, \cdots, l_m$ if and only if the Kraft inequality

$$\sum_{k=1}^{m} D^{-l_k} \leq 1$$

is satisfied.

Proof Direct part follows because a prefix code is uniquely decodable and hence satisfies Kraft's inequality.

Theorem 4.11 There exists a D-ary prefix code with codeword lengths $l_1, l_2, \cdots, l_m$ if and only if the Kraft inequality

$$\sum_{k=1}^{m} D^{-l_k} \leq 1$$

is satisfied.

Proof (Converse)

Theorem 4.11 There exists a D-ary prefix code with codeword lengths $l_1, l_2, \cdots, l_m$ if and only if the Kraft inequality

$$\sum_{k=1}^{m} D^{-l_k} \leq 1$$

is satisfied.

Proof (Converse)

1. We need to prove the existence of a D-ary prefix code with codeword lengths $l_1, l_2, \cdots, l_m$ if these lengths satisfy the Kraft inequality. Without loss of generality, assume that

Theorem 4.11 There exists a D-ary prefix code with codeword lengths $l_1, l_2, \cdots, l_m$ if and only if the Kraft inequality

$$\sum_{k=1}^{m} D^{-l_k} \leq 1$$

is satisfied.

Proof (Converse)

1. We need to prove the existence of a D-ary prefix code with codeword lengths $l_1, l_2, \cdots, l_m$ if these lengths satisfy the Kraft inequality. Without loss of generality, assume that

Theorem 4.11 There exists a D-ary prefix code with codeword lengths $l_1, l_2, \cdots, l_m$ if and only if the Kraft inequality

$$\sum_{k=1}^{m} D^{-l_k} \leq 1$$

is satisfied.

Proof (Converse)

1. We need to prove the existence of a D-ary prefix code with codeword lengths $l_1, l_2, \cdots, l_m$ if these lengths satisfy the Kraft inequality. Without loss of generality, assume that

$$l_1 \leq l_2 \leq \cdots \leq l_m.$$

Theorem 4.11 There exists a D-ary prefix code with codeword lengths $l_1,\ l_2, \cdots, l_m$ if and only if the Kraft inequality

$$\sum_{k=1}^{m} D^{-l_k} \leq 1$$

is satisfied.

Proof (Converse)

1. We need to prove the existence of a D-ary prefix code with codeword lengths $l_1, l_2, \cdots, l_m$ if these lengths satisfy the Kraft inequality. Without loss of generality, assume that

$$l_1 \leq l_2 \leq \cdots \leq l_m.$$

2. Consider all the D-ary sequences of lengths less than or equal to l_m and regard them as the nodes of the full D-ary tree of depth l_m. We will refer to a sequence of length l as a node of order l.

Theorem 4.11 There exists a D-ary prefix code with codeword lengths $l_1, l_2, \cdots, l_m$ if and only if the Kraft inequality

$$\sum_{k=1}^{m} D^{-l_k} \leq 1$$

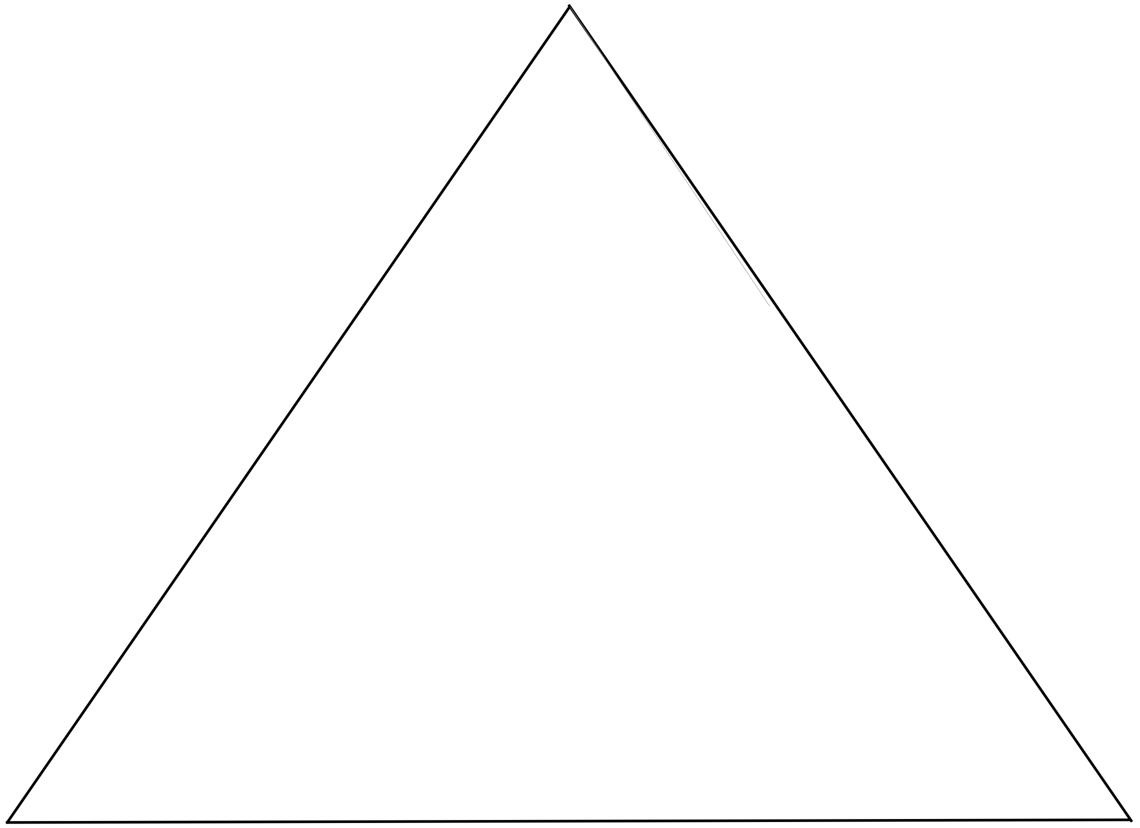
is satisfied.

Proof (Converse)

1. We need to prove the existence of a D-ary prefix code with codeword lengths $l_1, l_2, \cdots, l_m$ if these lengths satisfy the Kraft inequality. Without loss of generality, assume that

$$l_1 \leq l_2 \leq \cdots \leq l_m.$$

2. Consider all the D-ary sequences of lengths less than or equal to l_m and regard them as the nodes of the full D-ary tree of depth l_m. We will refer to a sequence of length l as a node of order l.



Theorem 4.11 There exists a D-ary prefix code with codeword lengths l_1, $l_2, \cdots, l_m$ if and only if the Kraft inequality

$$\sum_{k=1}^{m} D^{-l_k} \leq 1$$

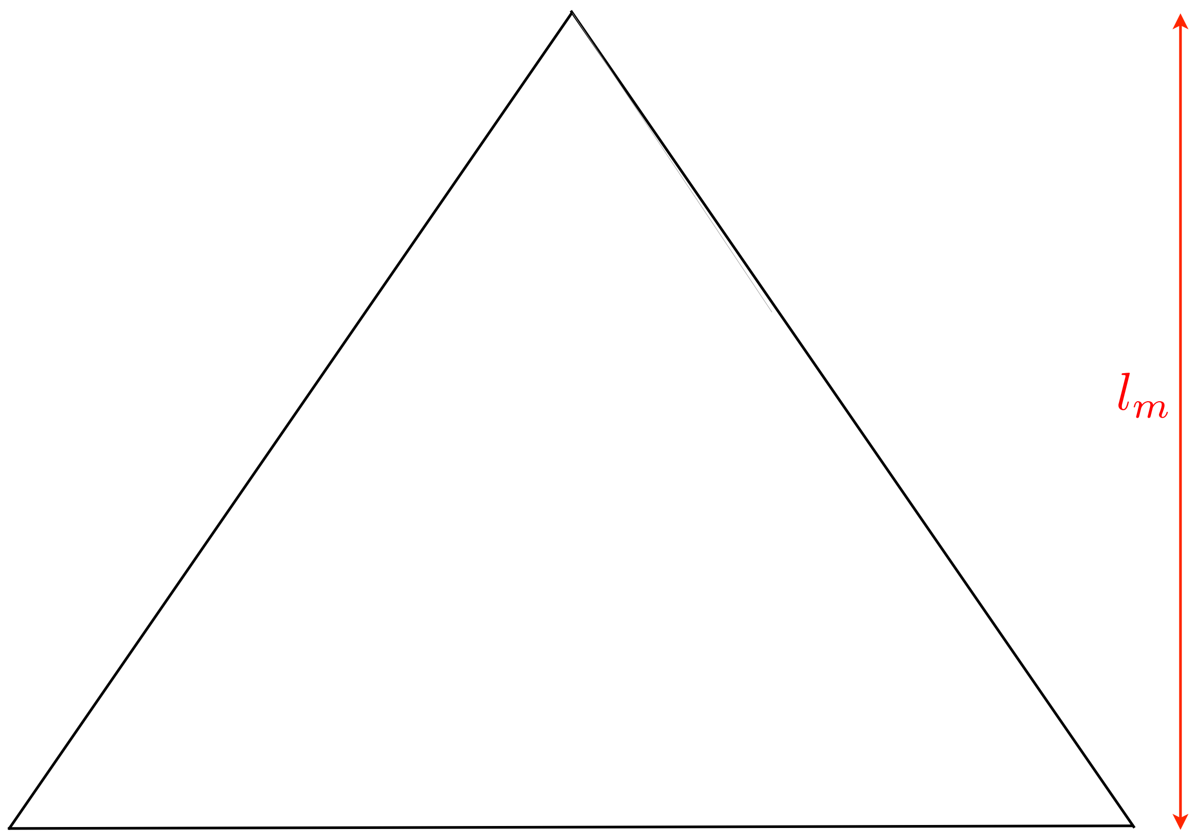
is satisfied.

Proof (Converse)

1. We need to prove the existence of a D-ary prefix code with codeword lengths $l_1, l_2, \cdots, l_m$ if these lengths satisfy the Kraft inequality. Without loss of generality, assume that

$$l_1 \leq l_2 \leq \cdots \leq l_m.$$

2. Consider all the D-ary sequences of lengths less than or equal to l_m and regard them as the nodes of the full D-ary tree of depth l_m. We will refer to a sequence of length l as a node of order l.



Theorem 4.11 There exists a D-ary prefix code with codeword lengths l_1, $l_2, \cdots, l_m$ if and only if the Kraft inequality

$$\sum_{k=1}^{m} D^{-l_k} \leq 1$$

is satisfied.

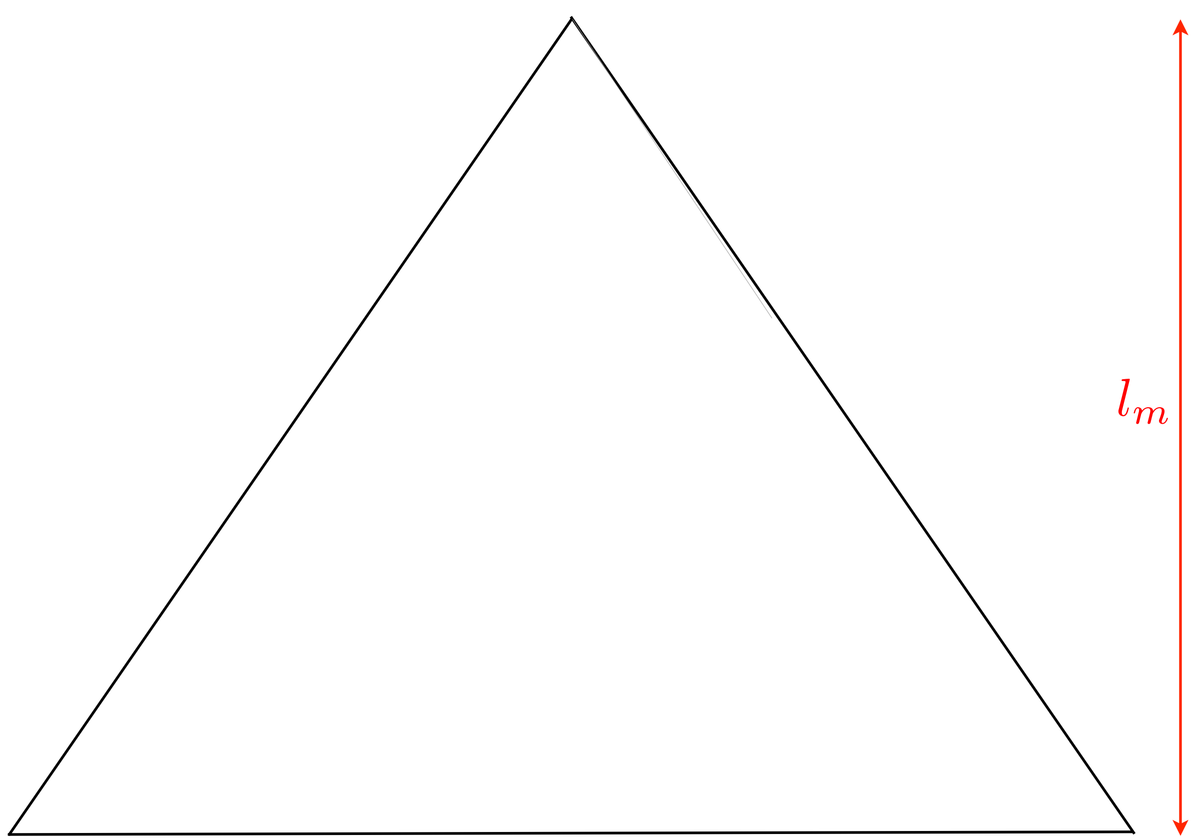
Proof (Converse)

1. We need to prove the existence of a D-ary prefix code with codeword lengths $l_1, l_2, \cdots, l_m$ if these lengths satisfy the Kraft inequality. Without loss of generality, assume that

$$l_1 \leq l_2 \leq \cdots \leq l_m.$$

2. Consider all the D-ary sequences of lengths less than or equal to l_m and regard them as the nodes of the full D-ary tree of depth l_m. We will refer to a sequence of length l as a node of order l.

3. There are $D^{l_1} > 1$ (since $l_1 \geq 1$) nodes of order l_1 which can be chosen as the first codeword. Thus choosing the first codeword is always possible.



Theorem 4.11 There exists a D-ary prefix code with codeword lengths $l_1,\ l_2, \cdots, l_m$ if and only if the Kraft inequality

$$\sum_{k=1}^{m} D^{-l_k} \leq 1$$

is satisfied.

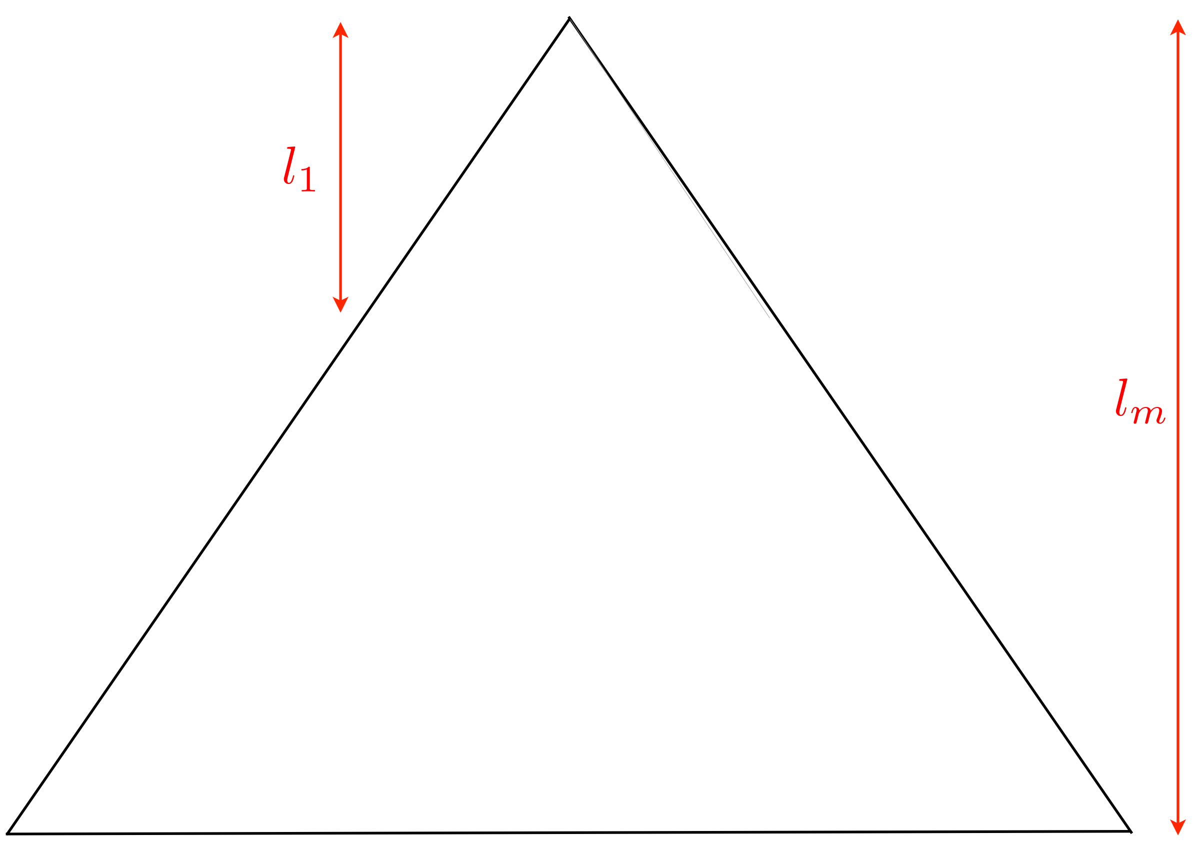
Proof (Converse)

1. We need to prove the existence of a D-ary prefix code with codeword lengths $l_1, l_2, \cdots, l_m$ if these lengths satisfy the Kraft inequality. Without loss of generality, assume that

$$l_1 \leq l_2 \leq \cdots \leq l_m.$$

2. Consider all the D-ary sequences of lengths less than or equal to l_m and regard them as the nodes of the full D-ary tree of depth l_m. We will refer to a sequence of length l as a node of order l.

3. There are $D^{l_1} > 1$ (since $l_1 \geq 1$) nodes of order l_1 which can be chosen as the first codeword. Thus choosing the first codeword is always possible.



Theorem 4.11 There exists a D-ary prefix code with codeword lengths l_1, $l_2, \cdots, l_m$ if and only if the Kraft inequality

$$\sum_{k=1}^{m} D^{-l_k} \leq 1$$

is satisfied.

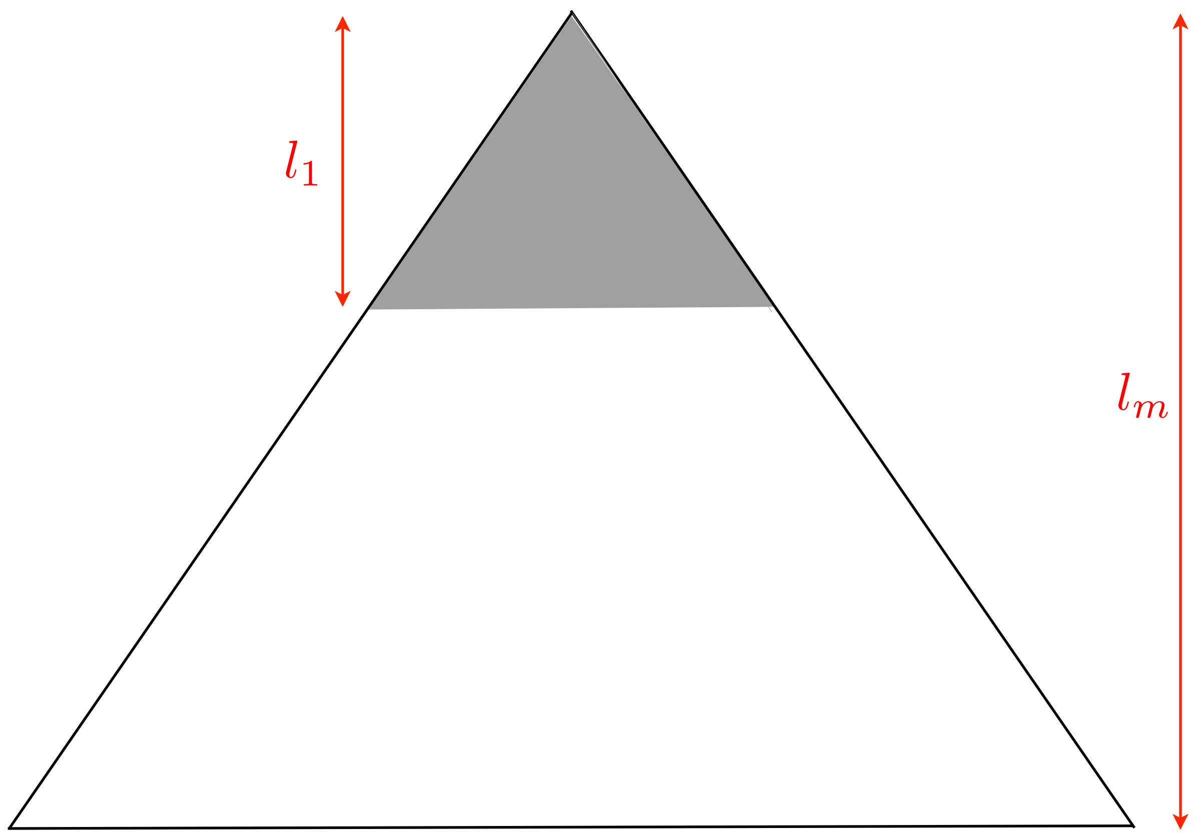
Proof (Converse)

1. We need to prove the existence of a D-ary prefix code with codeword lengths $l_1, l_2, \cdots, l_m$ if these lengths satisfy the Kraft inequality. Without loss of generality, assume that

$$l_1 \leq l_2 \leq \cdots \leq l_m.$$

2. Consider all the D-ary sequences of lengths less than or equal to l_m and regard them as the nodes of the full D-ary tree of depth l_m. We will refer to a sequence of length l as a node of order l.

3. There are $D^{l_1} > 1$ (since $l_1 \geq 1$) nodes of order l_1 which can be chosen as the first codeword. Thus choosing the first codeword is always possible.



Theorem 4.11 There exists a D-ary prefix code with codeword lengths l_1, $l_2, \cdots, l_m$ if and only if the Kraft inequality

$$\sum_{k=1}^{m} D^{-l_k} \leq 1$$

is satisfied.

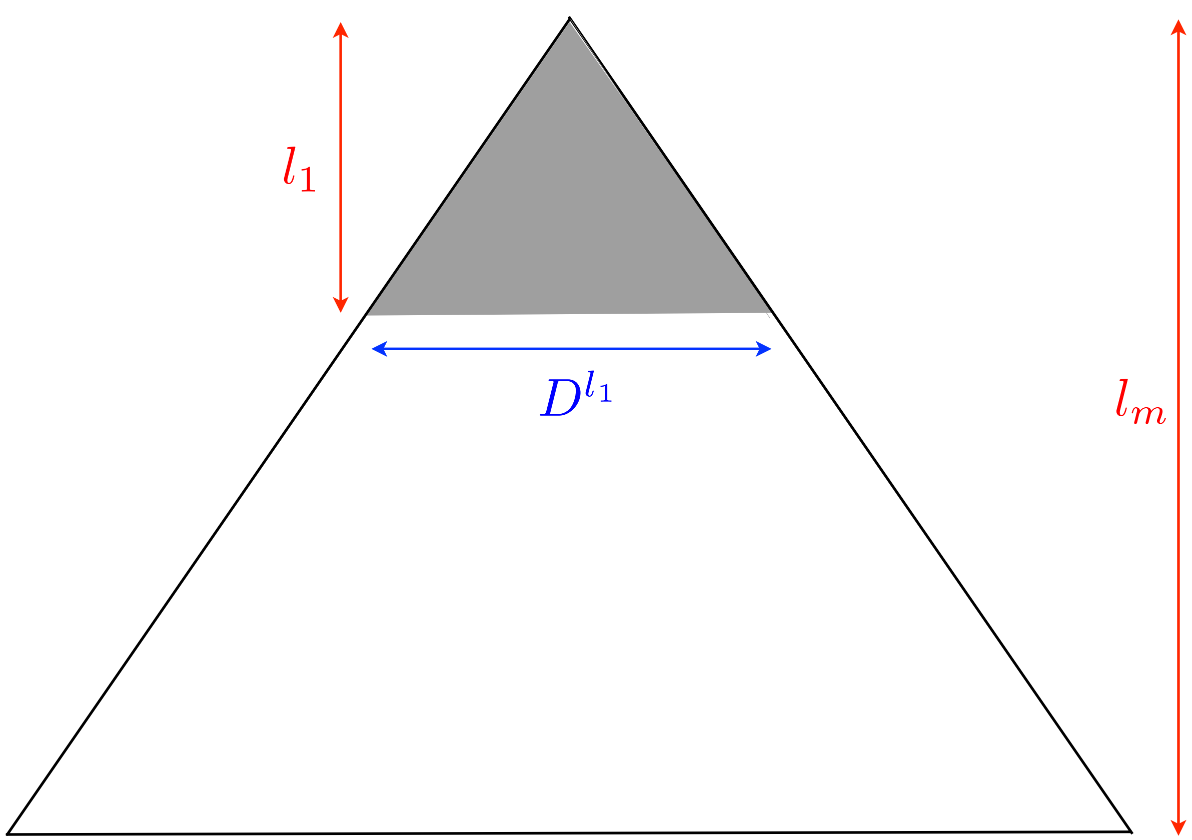
Proof (Converse)

1. We need to prove the existence of a D-ary prefix code with codeword lengths $l_1, l_2, \cdots, l_m$ if these lengths satisfy the Kraft inequality. Without loss of generality, assume that

$$l_1 \leq l_2 \leq \cdots \leq l_m.$$

2. Consider all the D-ary sequences of lengths less than or equal to l_m and regard them as the nodes of the full D-ary tree of depth l_m. We will refer to a sequence of length l as a node of order l.

3. There are $D^{l_1} > 1$ (since $l_1 \geq 1$) nodes of order l_1 which can be chosen as the first codeword. Thus choosing the first codeword is always possible.



Theorem 4.11 There exists a D-ary prefix code with codeword lengths $l_1, l_2, \cdots, l_m$ if and only if the Kraft inequality

$$\sum_{k=1}^{m} D^{-l_k} \le 1$$

is satisfied.

Proof (Converse)

1. We need to prove the existence of a D-ary prefix code with codeword lengths $l_1, l_2, \cdots, l_m$ if these lengths satisfy the Kraft inequality. Without loss of generality, assume that

$$l_1 \le l_2 \le \cdots \le l_m.$$

2. Consider all the D-ary sequences of lengths less than or equal to l_m and regard them as the nodes of the full D-ary tree of depth l_m. We will refer to a sequence of length l as a node of order l.

3. There are $D^{l_1} > 1$ (since $l_1 \ge 1$) nodes of order l_1 which can be chosen as the first codeword. Thus choosing the first codeword is always possible.

4. Assume that the first i codewords have been chosen successfully, where $1 \le i \le m-1$, and we want to choose a node of order l_{i+1} as the $(i+1)$st codeword such that it is not prefixed by any of the previously chosen codewords.

Theorem 4.11 There exists a D-ary prefix code with codeword lengths $l_1, l_2, \cdots, l_m$ if and only if the Kraft inequality

$$\sum_{k=1}^{m} D^{-l_k} \leq 1$$

is satisfied.

Proof (Converse)

1. We need to prove the existence of a D-ary prefix code with codeword lengths $l_1, l_2, \cdots, l_m$ if these lengths satisfy the Kraft inequality. Without loss of generality, assume that

$$l_1 \leq l_2 \leq \cdots \leq l_m.$$

2. Consider all the D-ary sequences of lengths less than or equal to l_m and regard them as the nodes of the full D-ary tree of depth l_m. We will refer to a sequence of length l as a node of order l.

3. There are $D^{l_1} > 1$ (since $l_1 \geq 1$) nodes of order l_1 which can be chosen as the first codeword. Thus choosing the first codeword is always possible.

4. Assume that the first i codewords have been chosen successfully, where $1 \leq i \leq m-1$, and we want to choose a node of order l_{i+1} as the $(i+1)$st codeword such that it is not prefixed by any of the previously chosen codewords.

5. Since all the previously chosen codewords are not prefixes of each other, their descendants of order l_{i+1} do not overlap. The $(i+1)$st node to be chosen cannot be a descendant of any of the previously chosen codewords.

Theorem 4.11 There exists a D-ary prefix code with codeword lengths $l_1,\ l_2, \cdots, l_m$ if and only if the Kraft inequality

$$\sum_{k=1}^{m} D^{-l_k} \leq 1$$

is satisfied.

Proof (Converse)

1. We need to prove the existence of a D-ary prefix code with codeword lengths $l_1, l_2, \cdots, l_m$ if these lengths satisfy the Kraft inequality. Without loss of generality, assume that

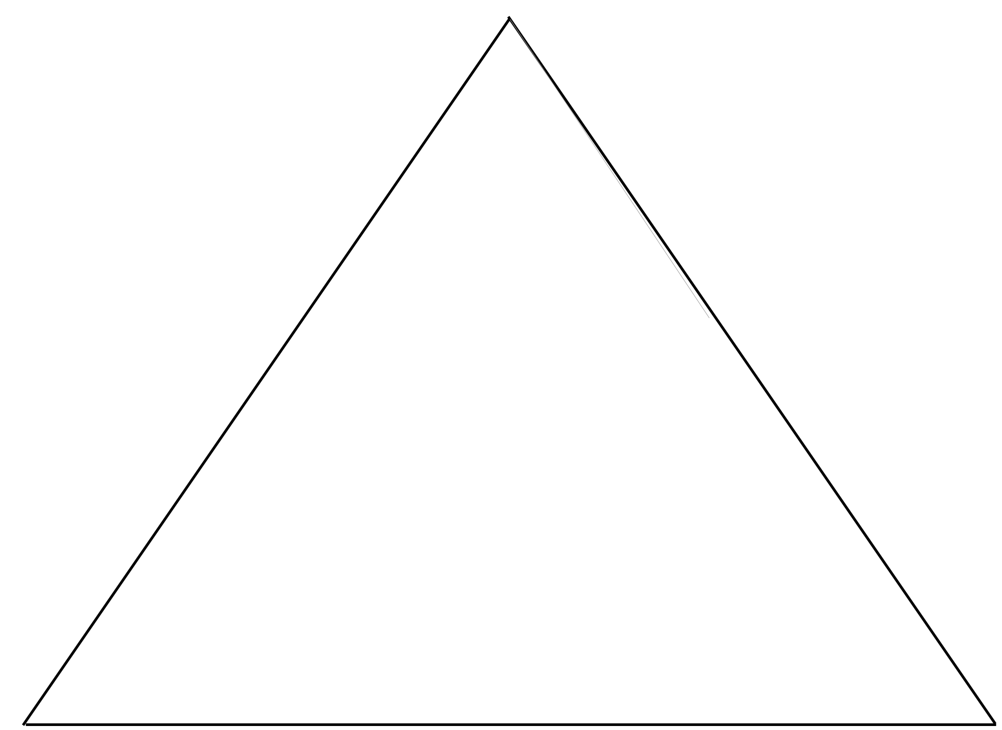
$$l_1 \leq l_2 \leq \cdots \leq l_m.$$

2. Consider all the D-ary sequences of lengths less than or equal to l_m and regard them as the nodes of the full D-ary tree of depth l_m. We will refer to a sequence of length l as a node of order l.

3. There are $D^{l_1} > 1$ (since $l_1 \geq 1$) nodes of order l_1 which can be chosen as the first codeword. Thus choosing the first codeword is always possible.

4. Assume that the first i codewords have been chosen successfully, where $1 \leq i \leq m-1$, and we want to choose a node of order l_{i+1} as the $(i+1)$st codeword such that it is not prefixed by any of the previously chosen codewords.

5. Since all the previously chosen codewords are not prefixes of each other, their descendants of order l_{i+1} do not overlap. The $(i+1)$st node to be chosen cannot be a descendant of any of the previously chosen codewords.



Theorem 4.11 There exists a D-ary prefix code with codeword lengths $l_1, l_2, \cdots, l_m$ if and only if the Kraft inequality

$$\sum_{k=1}^{m} D^{-l_k} \leq 1$$

is satisfied.

Proof (Converse)

1. We need to prove the existence of a D-ary prefix code with codeword lengths $l_1, l_2, \cdots, l_m$ if these lengths satisfy the Kraft inequality. Without loss of generality, assume that

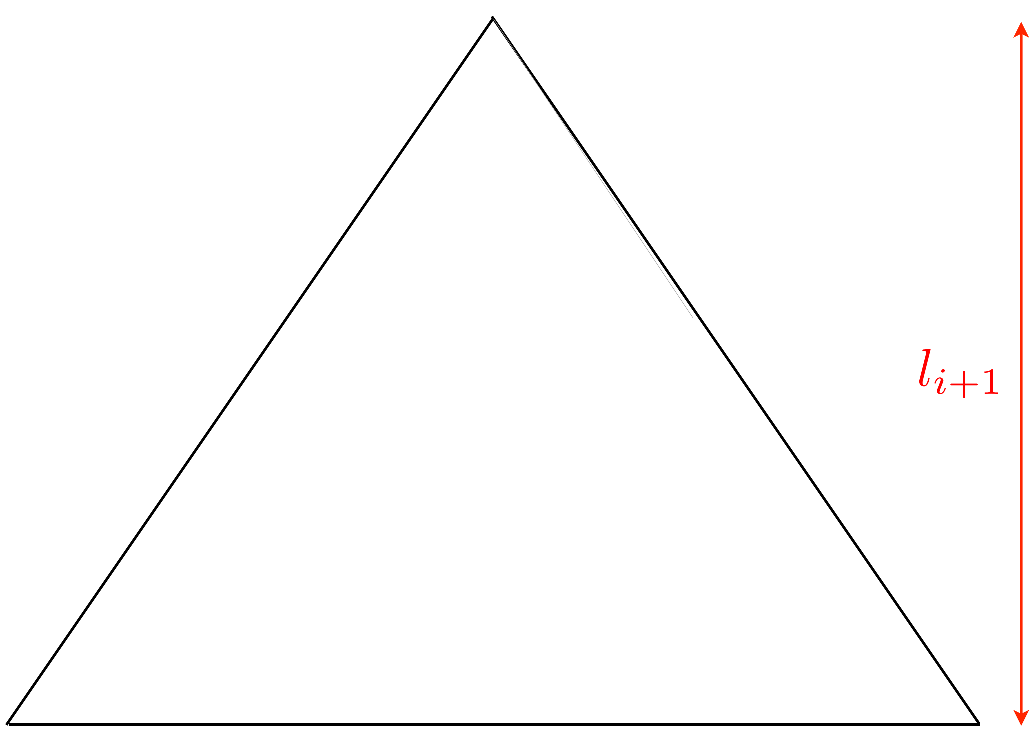
$$l_1 \leq l_2 \leq \cdots \leq l_m.$$

2. Consider all the D-ary sequences of lengths less than or equal to l_m and regard them as the nodes of the full D-tree of depth l_m. We will refer to a sequence of length l as a node of order l.

3. There are $D^{l_1} > 1$ (since $l_1 \geq 1$) nodes of order l_1 which can be chosen as the first codeword. Thus choosing the first codeword is always possible.

4. Assume that the first i codewords have been chosen successfully, where $1 \leq i \leq m-1$, and we want to choose a node of order l_{i+1} as the $(i+1)$st codeword such that it is not prefixed by any of the previously chosen codewords.

5. Since all the previously chosen codewords are not prefixes of each other, their descendants of order l_{i+1} do not overlap. The $(i+1)$st node to be chosen cannot be a descendant of any of the previously chosen codewords.



Theorem 4.11 There exists a D-ary prefix code with codeword lengths $l_1,\ l_2, \cdots, l_m$ if and only if the Kraft inequality

$$\sum_{k=1}^{m} D^{-l_k} \leq 1$$

is satisfied.

Proof (Converse)

1. We need to prove the existence of a D-ary prefix code with codeword lengths $l_1, l_2, \cdots, l_m$ if these lengths satisfy the Kraft inequality. Without loss of generality, assume that

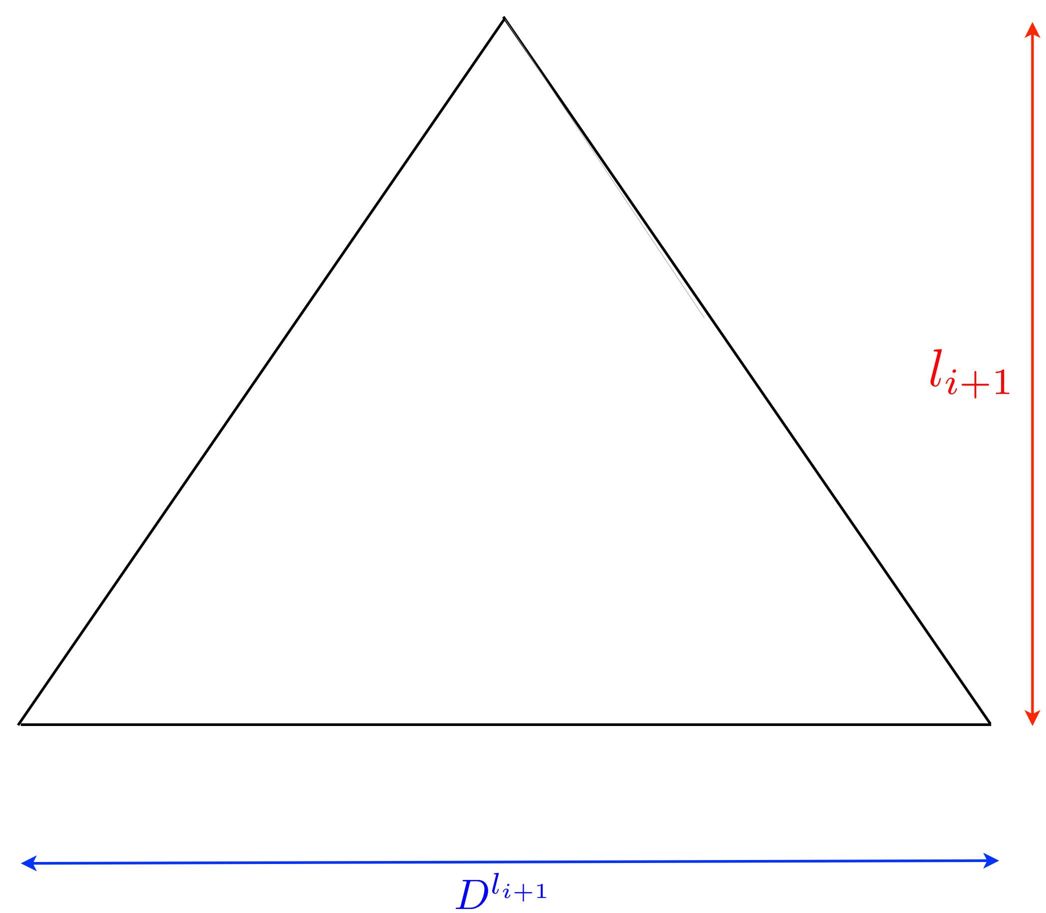
$$l_1 \leq l_2 \leq \cdots \leq l_m.$$

2. Consider all the D-ary sequences of lengths less than or equal to l_m and regard them as the nodes of the full D-ary tree of depth l_m. We will refer to a sequence of length l as a node of order l.

3. There are $D^{l_1} > 1$ (since $l_1 \geq 1$) nodes of order l_1 which can be chosen as the first codeword. Thus choosing the first codeword is always possible.

4. Assume that the first i codewords have been chosen successfully, where $1 \leq i \leq m-1$, and we want to choose a node of order l_{i+1} as the $(i+1)$st codeword such that it is not prefixed by any of the previously chosen codewords.

5. Since all the previously chosen codewords are not prefixes of each other, their descendants of order l_{i+1} do not overlap. The $(i+1)$st node to be chosen cannot be a descendant of any of the previously chosen codewords.



Theorem 4.11 There exists a D-ary prefix code with codeword lengths $l_1, l_2, \cdots, l_m$ if and only if the Kraft inequality

$$\sum_{k=1}^{m} D^{-l_k} \leq 1$$

is satisfied.

Proof (Converse)

1. We need to prove the existence of a D-ary prefix code with codeword lengths $l_1, l_2, \cdots, l_m$ if these lengths satisfy the Kraft inequality. Without loss of generality, assume that

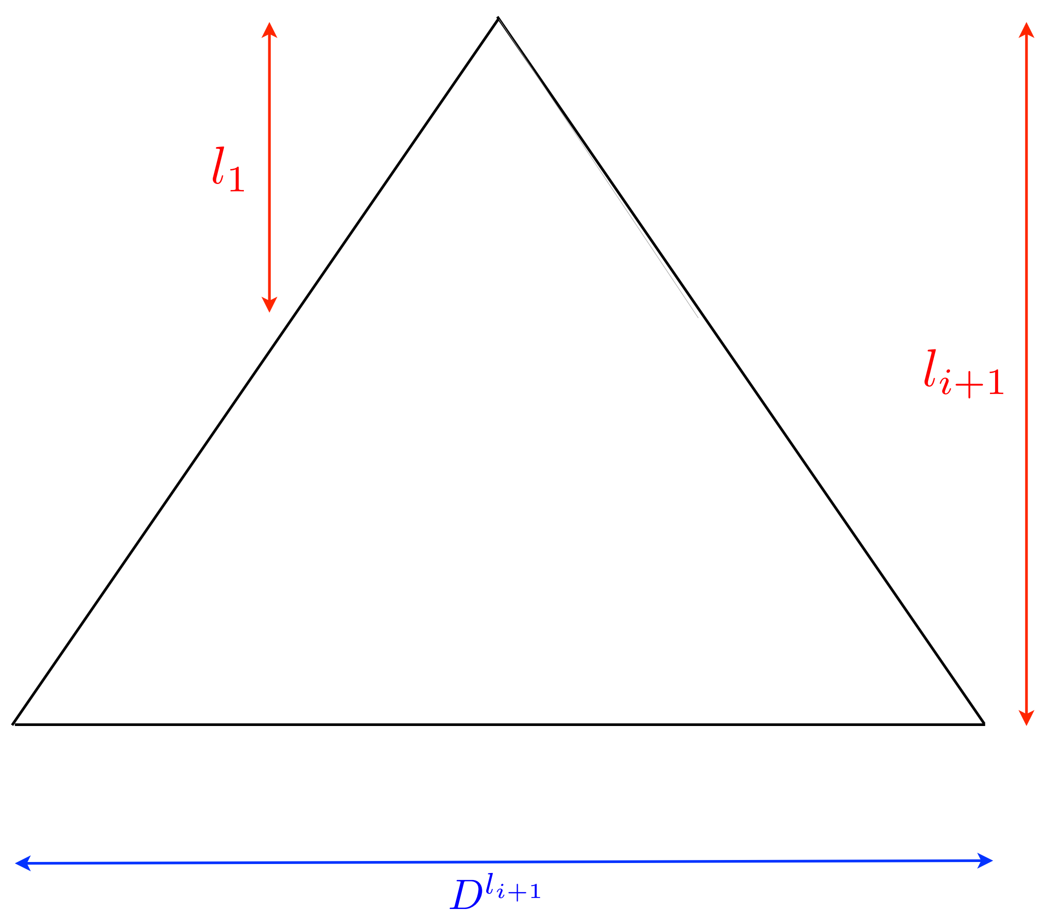
$$l_1 \leq l_2 \leq \cdots \leq l_m.$$

2. Consider all the D-ary sequences of lengths less than or equal to l_m and regard them as the nodes of the full D-ary tree of depth l_m. We will refer to a sequence of length l as a node of order l.

3. There are $D^{l_1} > 1$ (since $l_1 \geq 1$) nodes of order l_1 which can be chosen as the first codeword. Thus choosing the first codeword is always possible.

4. Assume that the first i codewords have been chosen successfully, where $1 \leq i \leq m-1$, and we want to choose a node of order l_{i+1} as the $(i+1)$st codeword such that it is not prefixed by any of the previously chosen codewords.

5. Since all the previously chosen codewords are not prefixes of each other, their descendants of order l_{i+1} do not overlap. The $(i+1)$st node to be chosen cannot be a descendant of any of the previously chosen codewords.



Theorem 4.11 There exists a D-ary prefix code with codeword lengths $l_1, l_2, \cdots, l_m$ if and only if the Kraft inequality

$$\sum_{k=1}^{m} D^{-l_k} \leq 1$$

is satisfied.

Proof (Converse)

1. We need to prove the existence of a D-ary prefix code with codeword lengths $l_1, l_2, \cdots, l_m$ if these lengths satisfy the Kraft inequality. Without loss of generality, assume that

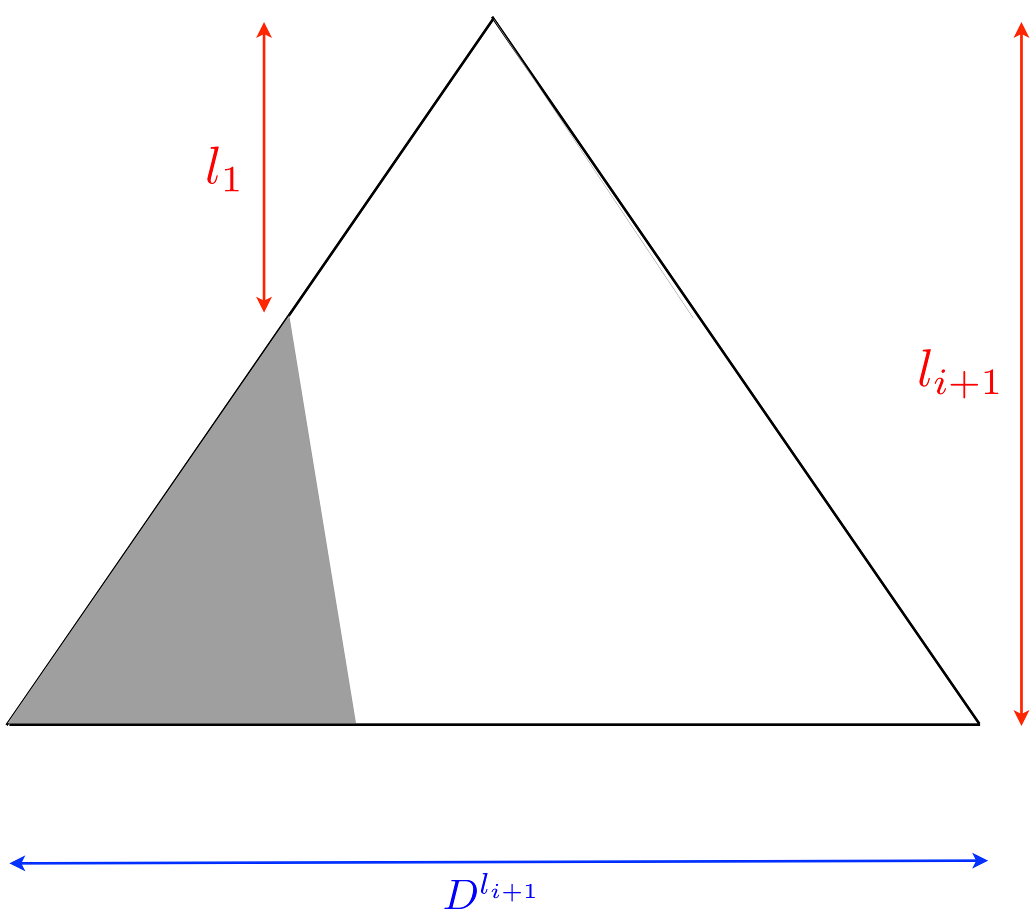
$$l_1 \leq l_2 \leq \cdots \leq l_m.$$

2. Consider all the D-ary sequences of lengths less than or equal to l_m and regard them as the nodes of the full D-ary tree of depth l_m. We will refer to a sequence of length l as a node of order l.

3. There are $D^{l_1} > 1$ (since $l_1 \geq 1$) nodes of order l_1 which can be chosen as the first codeword. Thus choosing the first codeword is always possible.

4. Assume that the first i codewords have been chosen successfully, where $1 \leq i \leq m-1$, and we want to choose a node of order l_{i+1} as the $(i+1)$st codeword such that it is not prefixed by any of the previously chosen codewords.

5. Since all the previously chosen codewords are not prefixes of each other, their descendants of order l_{i+1} do not overlap. The $(i+1)$st node to be chosen cannot be a descendant of any of the previously chosen codewords.



Theorem 4.11 There exists a D-ary prefix code with codeword lengths $l_1, l_2, \cdots, l_m$ if and only if the Kraft inequality

$$\sum_{k=1}^{m} D^{-l_k} \leq 1$$

is satisfied.

Proof (Converse)

1. We need to prove the existence of a D-ary prefix code with codeword lengths $l_1, l_2, \cdots, l_m$ if these lengths satisfy the Kraft inequality. Without loss of generality, assume that

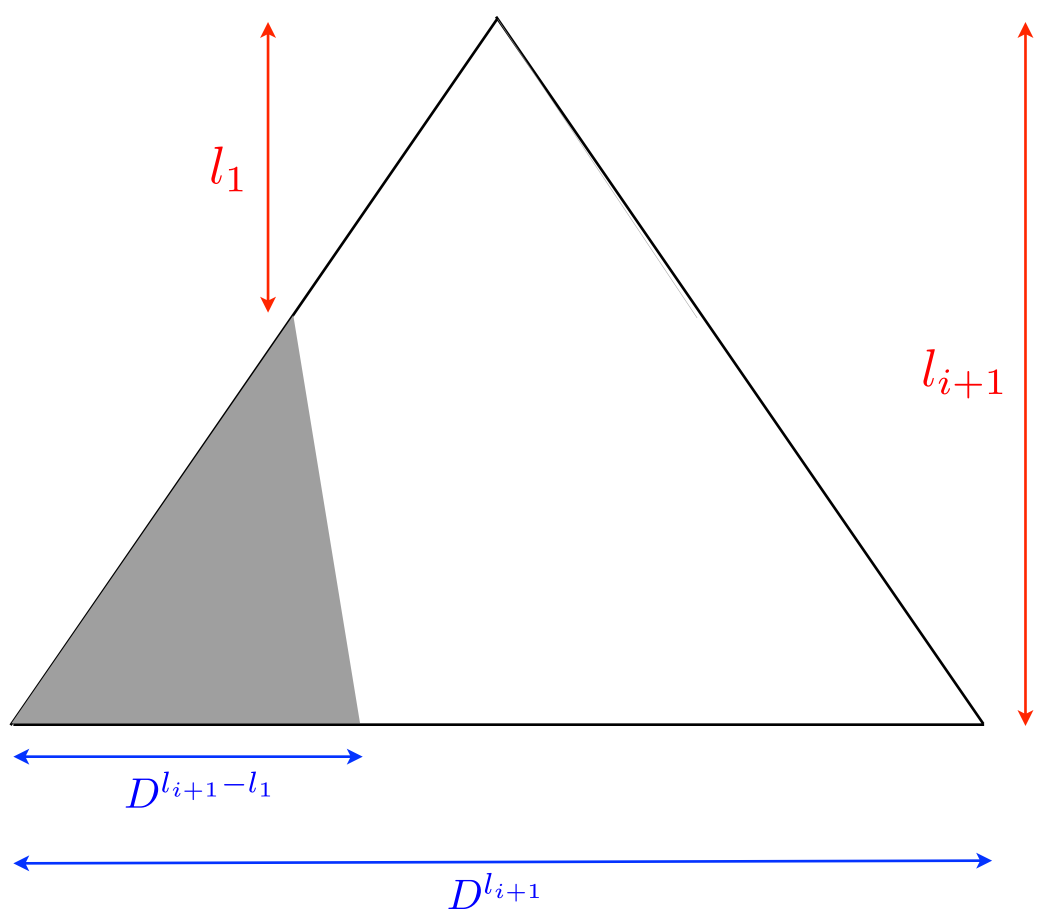
$$l_1 \leq l_2 \leq \cdots \leq l_m.$$

2. Consider all the D-ary sequences of lengths less than or equal to l_m and regard them as the nodes of the full D-ary tree of depth l_m. We will refer to a sequence of length l as a node of order l.

3. There are $D^{l_1} > 1$ (since $l_1 \geq 1$) nodes of order l_1 which can be chosen as the first codeword. Thus choosing the first codeword is always possible.

4. Assume that the first i codewords have been chosen successfully, where $1 \leq i \leq m-1$, and we want to choose a node of order l_{i+1} as the $(i+1)$st codeword such that it is not prefixed by any of the previously chosen codewords.

5. Since all the previously chosen codewords are not prefixes of each other, their descendants of order l_{i+1} do not overlap. The $(i+1)$st node to be chosen cannot be a descendant of any of the previously chosen codewords.



Theorem 4.11 There exists a D-ary prefix code with codeword lengths $l_1, l_2, \cdots, l_m$ if and only if the Kraft inequality

$$\sum_{k=1}^{m} D^{-l_k} \leq 1$$

is satisfied.

Proof (Converse)

1. We need to prove the existence of a D-ary prefix code with codeword lengths $l_1, l_2, \cdots, l_m$ if these lengths satisfy the Kraft inequality. Without loss of generality, assume that

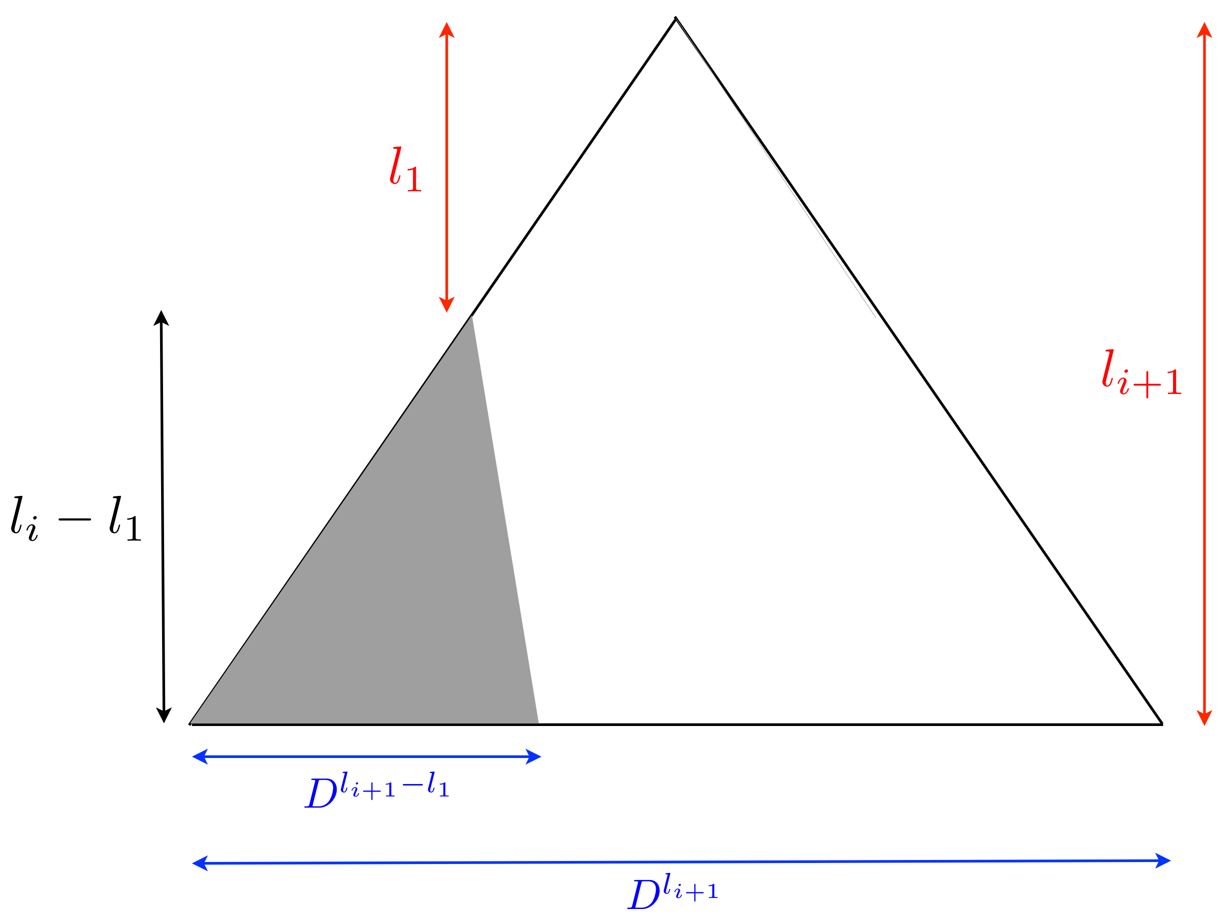
$$l_1 \leq l_2 \leq \cdots \leq l_m.$$

2. Consider all the D-ary sequences of lengths less than or equal to l_m and regard them as the nodes of the full D-ary tree of depth l_m. We will refer to a sequence of length l as a node of order l.

3. There are $D^{l_1} > 1$ (since $l_1 \geq 1$) nodes of order l_1 which can be chosen as the first codeword. Thus choosing the first codeword is always possible.

4. Assume that the first i codewords have been chosen successfully, where $1 \leq i \leq m-1$, and we want to choose a node of order l_{i+1} as the $(i+1)$st codeword such that it is not prefixed by any of the previously chosen codewords.

5. Since all the previously chosen codewords are not prefixes of each other, their descendants of order l_{i+1} do not overlap. The $(i+1)$st node to be chosen cannot be a descendant of any of the previously chosen codewords.



Theorem 4.11 There exists a D-ary prefix code with codeword lengths l_1, $l_2, \cdots, l_m$ if and only if the Kraft inequality

$$\sum_{k=1}^{m} D^{-l_k} \leq 1$$

is satisfied.

Proof (Converse)

1. We need to prove the existence of a D-ary prefix code with codeword lengths $l_1, l_2, \cdots, l_m$ if these lengths satisfy the Kraft inequality. Without loss of generality, assume that

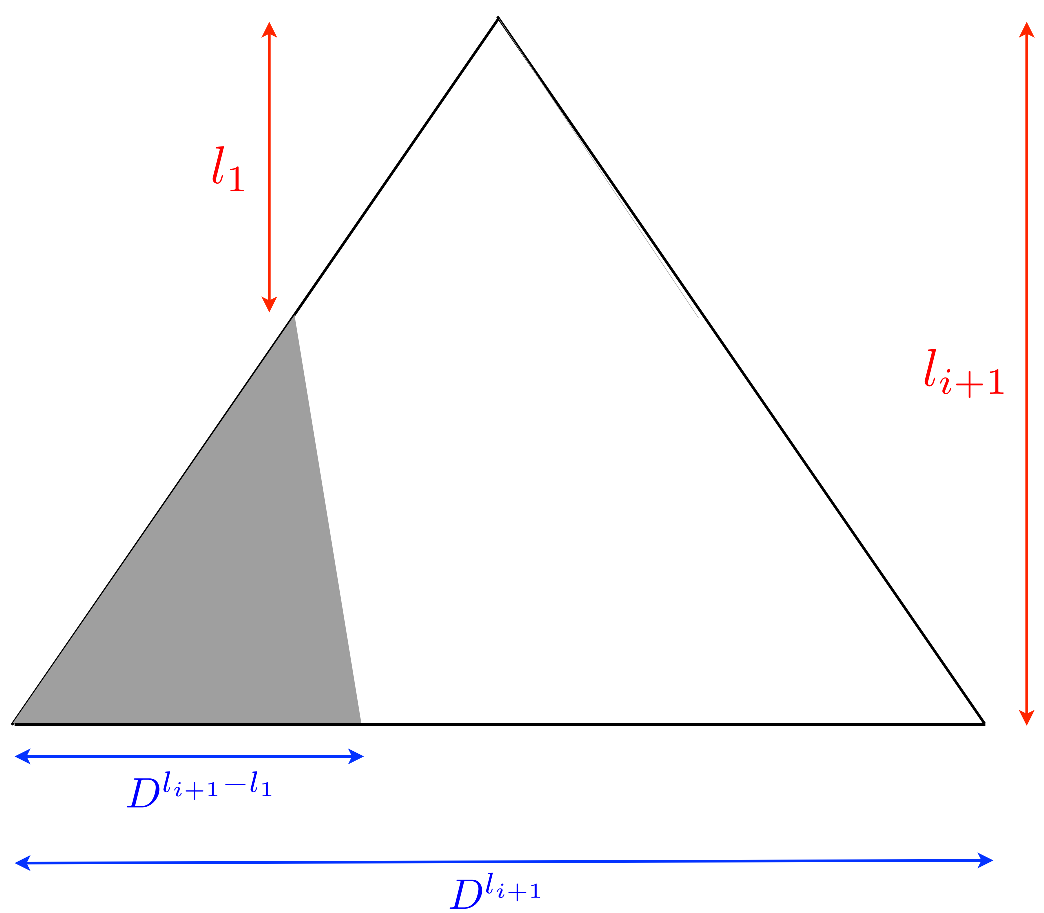
$$l_1 \leq l_2 \leq \cdots \leq l_m.$$

2. Consider all the D-ary sequences of lengths less than or equal to l_m and regard them as the nodes of the full D-ary tree of depth l_m. We will refer to a sequence of length l as a node of order l.

3. There are $D^{l_1} > 1$ (since $l_1 \geq 1$) nodes of order l_1 which can be chosen as the first codeword. Thus choosing the first codeword is always possible.

4. Assume that the first i codewords have been chosen successfully, where $1 \leq i \leq m-1$, and we want to choose a node of order l_{i+1} as the $(i+1)$st codeword such that it is not prefixed by any of the previously chosen codewords.

5. Since all the previously chosen codewords are not prefixes of each other, their descendants of order l_{i+1} do not overlap. The $(i+1)$st node to be chosen cannot be a descendant of any of the previously chosen codewords.



Theorem 4.11 There exists a D-ary prefix code with codeword lengths $l_1, l_2, \cdots, l_m$ if and only if the Kraft inequality

$$\sum_{k=1}^{m} D^{-l_k} \leq 1$$

is satisfied.

Proof (Converse)

1. We need to prove the existence of a D-ary prefix code with codeword lengths $l_1, l_2, \cdots, l_m$ if these lengths satisfy the Kraft inequality. Without loss of generality, assume that

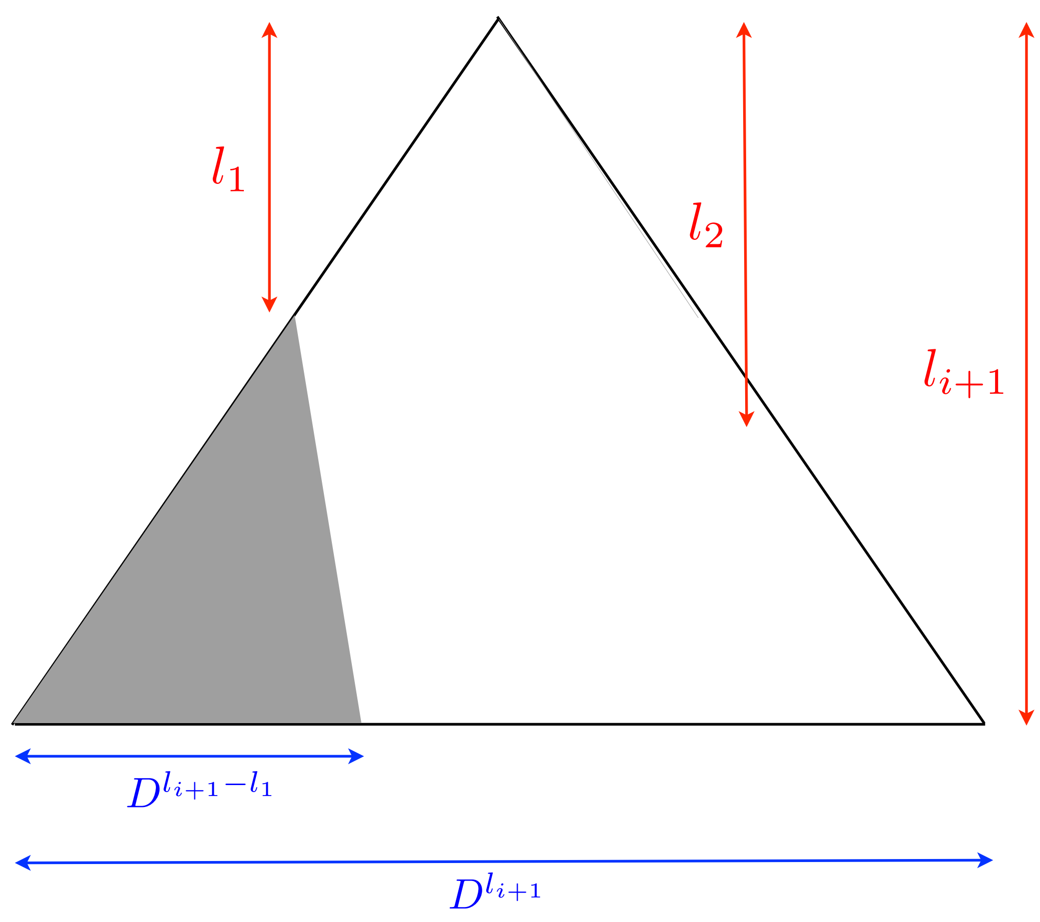
$$l_1 \leq l_2 \leq \cdots \leq l_m.$$

2. Consider all the D-ary sequences of lengths less than or equal to l_m and regard them as the nodes of the full D-ary tree of depth l_m. We will refer to a sequence of length l as a node of order l.

3. There are $D^{l_1} > 1$ (since $l_1 \geq 1$) nodes of order l_1 which can be chosen as the first codeword. Thus choosing the first codeword is always possible.

4. Assume that the first i codewords have been chosen successfully, where $1 \leq i \leq m-1$, and we want to choose a node of order l_{i+1} as the $(i+1)$st codeword such that it is not prefixed by any of the previously chosen codewords.

5. Since all the previously chosen codewords are not prefixes of each other, their descendants of order l_{i+1} do not overlap. The $(i+1)$st node to be chosen cannot be a descendant of any of the previously chosen codewords.



Theorem 4.11 There exists a D-ary prefix code with codeword lengths l_1, $l_2, \cdots, l_m$ if and only if the Kraft inequality

$$\sum_{k=1}^{m} D^{-l_k} \leq 1$$

is satisfied.

Proof (Converse)

1. We need to prove the existence of a D-ary prefix code with codeword lengths $l_1, l_2, \cdots, l_m$ if these lengths satisfy the Kraft inequality. Without loss of generality, assume that

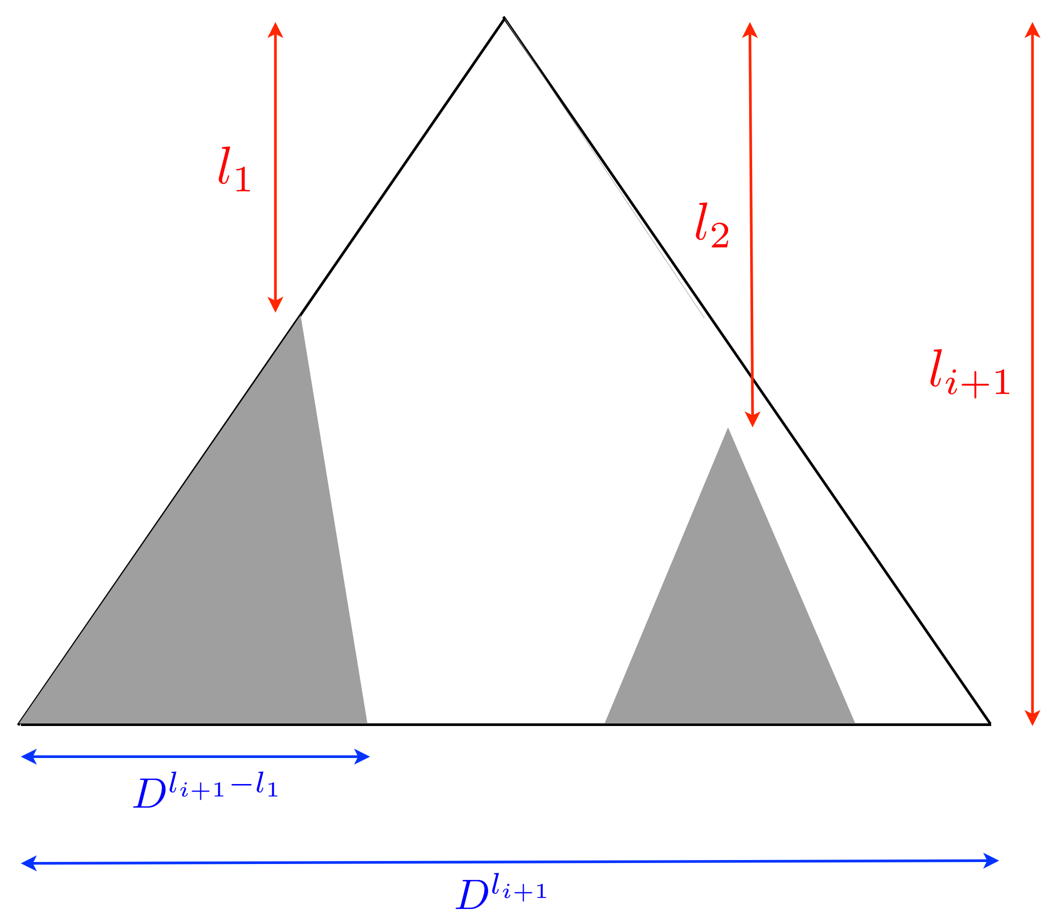
$$l_1 \leq l_2 \leq \cdots \leq l_m.$$

2. Consider all the D-ary sequences of lengths less than or equal to l_m and regard them as the nodes of the full D-ary tree of depth l_m. We will refer to a sequence of length l as a node of order l.

3. There are $D^{l_1} > 1$ (since $l_1 \geq 1$) nodes of order l_1 which can be chosen as the first codeword. Thus choosing the first codeword is always possible.

4. Assume that the first i codewords have been chosen successfully, where $1 \leq i \leq m-1$, and we want to choose a node of order l_{i+1} as the $(i+1)$st codeword such that it is not prefixed by any of the previously chosen codewords.

5. Since all the previously chosen codewords are not prefixes of each other, their descendants of order l_{i+1} do not overlap. The $(i+1)$st node to be chosen cannot be a descendant of any of the previously chosen codewords.



Theorem 4.11 There exists a D-ary prefix code with codeword lengths $l_1, l_2, \cdots, l_m$ if and only if the Kraft inequality

$$\sum_{k=1}^{m} D^{-l_k} \leq 1$$

is satisfied.

Proof (Converse)

1. We need to prove the existence of a D-ary prefix code with codeword lengths $l_1, l_2, \cdots, l_m$ if these lengths satisfy the Kraft inequality. Without loss of generality, assume that

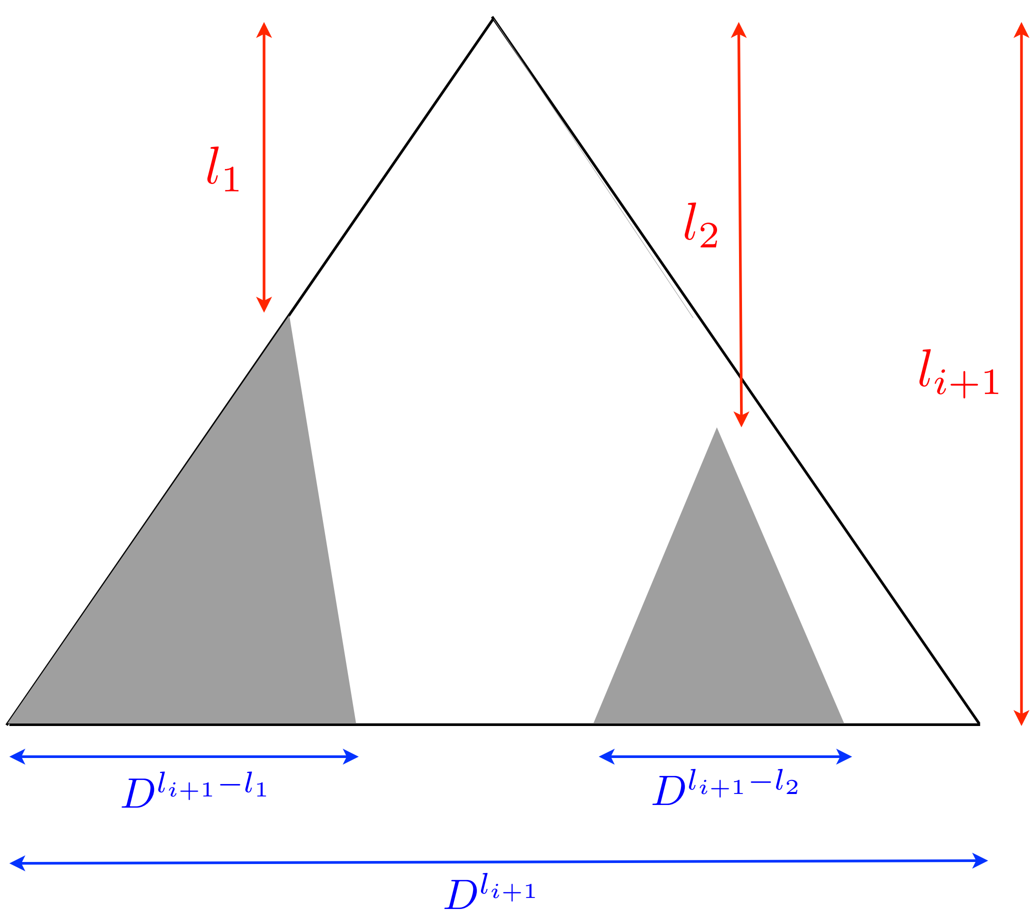
$$l_1 \leq l_2 \leq \cdots \leq l_m.$$

2. Consider all the D-ary sequences of lengths less than or equal to l_m and regard them as the nodes of the full D-ary tree of depth l_m. We will refer to a sequence of length l as a node of order l.

3. There are $D^{l_1} > 1$ (since $l_1 \geq 1$) nodes of order l_1 which can be chosen as the first codeword. Thus choosing the first codeword is always possible.

4. Assume that the first i codewords have been chosen successfully, where $1 \leq i \leq m-1$, and we want to choose a node of order l_{i+1} as the $(i+1)$st codeword such that it is not prefixed by any of the previously chosen codewords.

5. Since all the previously chosen codewords are not prefixes of each other, their descendants of order l_{i+1} do not overlap. The $(i+1)$st node to be chosen cannot be a descendant of any of the previously chosen codewords.



Theorem 4.11 There exists a D-ary prefix code with codeword lengths $l_1,\ l_2, \cdots, l_m$ if and only if the Kraft inequality

$$\sum_{k=1}^{m} D^{-l_k} \leq 1$$

is satisfied.

Proof (Converse)

1. We need to prove the existence of a D-ary prefix code with codeword lengths $l_1, l_2, \cdots, l_m$ if these lengths satisfy the Kraft inequality. Without loss of generality, assume that

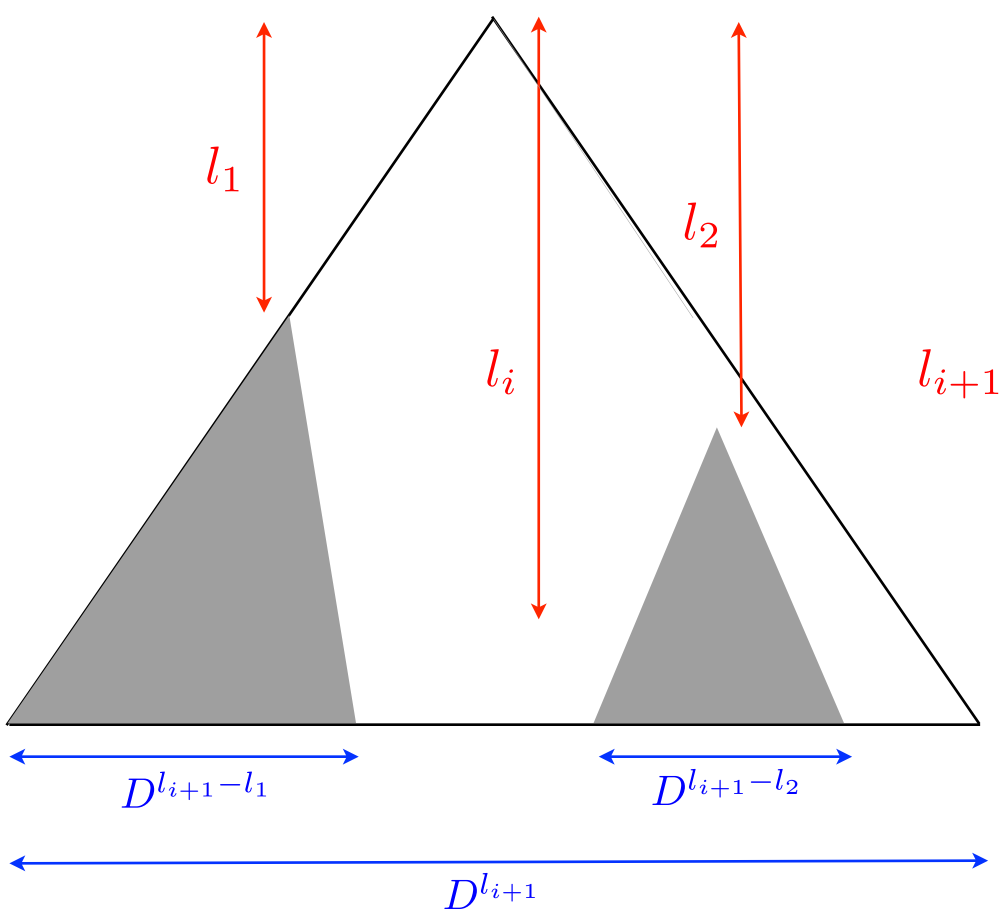
$$l_1 \leq l_2 \leq \cdots \leq l_m.$$

2. Consider all the D-ary sequences of lengths less than or equal to l_m and regard them as the nodes of the full D-ary tree of depth l_m. We will refer to a sequence of length l as a node of order l.

3. There are $D^{l_1} > 1$ (since $l_1 \geq 1$) nodes of order l_1 which can be chosen as the first codeword. Thus choosing the first codeword is always possible.

4. Assume that the first i codewords have been chosen successfully, where $1 \leq i \leq m-1$, and we want to choose a node of order l_{i+1} as the $(i+1)$st codeword such that it is not prefixed by any of the previously chosen codewords.

5. Since all the previously chosen codewords are not prefixes of each other, their descendants of order l_{i+1} do not overlap. The $(i+1)$st node to be chosen cannot be a descendant of any of the previously chosen codewords.



Theorem 4.11 There exists a D-ary prefix code with codeword lengths $l_1, l_2, \cdots, l_m$ if and only if the Kraft inequality

$$\sum_{k=1}^{m} D^{-l_k} \leq 1$$

is satisfied.

Proof (Converse)

1. We need to prove the existence of a D-ary prefix code with codeword lengths $l_1, l_2, \cdots, l_m$ if these lengths satisfy the Kraft inequality. Without loss of generality, assume that

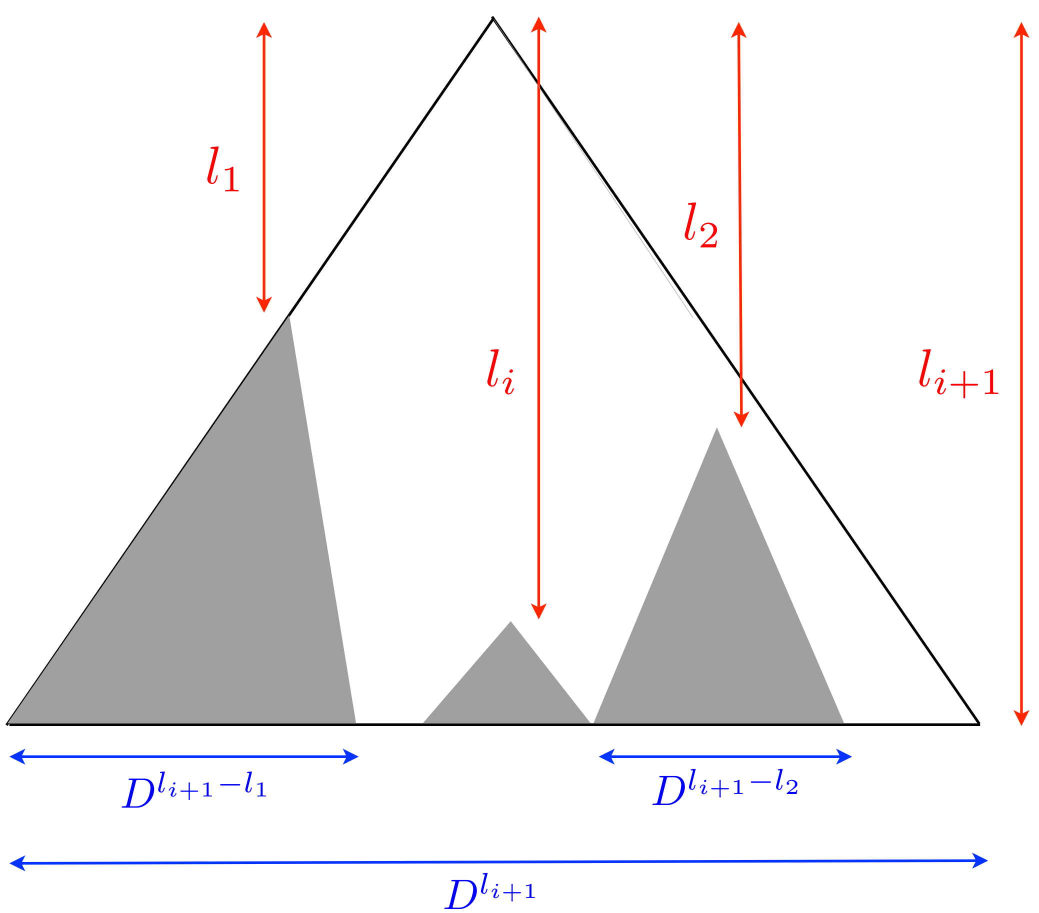
$$l_1 \leq l_2 \leq \cdots \leq l_m.$$

2. Consider all the D-ary sequences of lengths less than or equal to l_m and regard them as the nodes of the full D-ary tree of depth l_m. We will refer to a sequence of length l as a node of order l.

3. There are $D^{l_1} > 1$ (since $l_1 \geq 1$) nodes of order l_1 which can be chosen as the first codeword. Thus choosing the first codeword is always possible.

4. Assume that the first i codewords have been chosen successfully, where $1 \leq i \leq m-1$, and we want to choose a node of order l_{i+1} as the $(i+1)$st codeword such that it is not prefixed by any of the previously chosen codewords.

5. Since all the previously chosen codewords are not prefixes of each other, their descendants of order l_{i+1} do not overlap. The $(i+1)$st node to be chosen cannot be a descendant of any of the previously chosen codewords.



Theorem 4.11 There exists a D-ary prefix code with codeword lengths l_1, $l_2, \cdots, l_m$ if and only if the Kraft inequality

$$\sum_{k=1}^{m} D^{-l_k} \leq 1$$

is satisfied.

Proof (Converse)

1. We need to prove the existence of a D-ary prefix code with codeword lengths $l_1, l_2, \cdots, l_m$ if these lengths satisfy the Kraft inequality. Without loss of generality, assume that

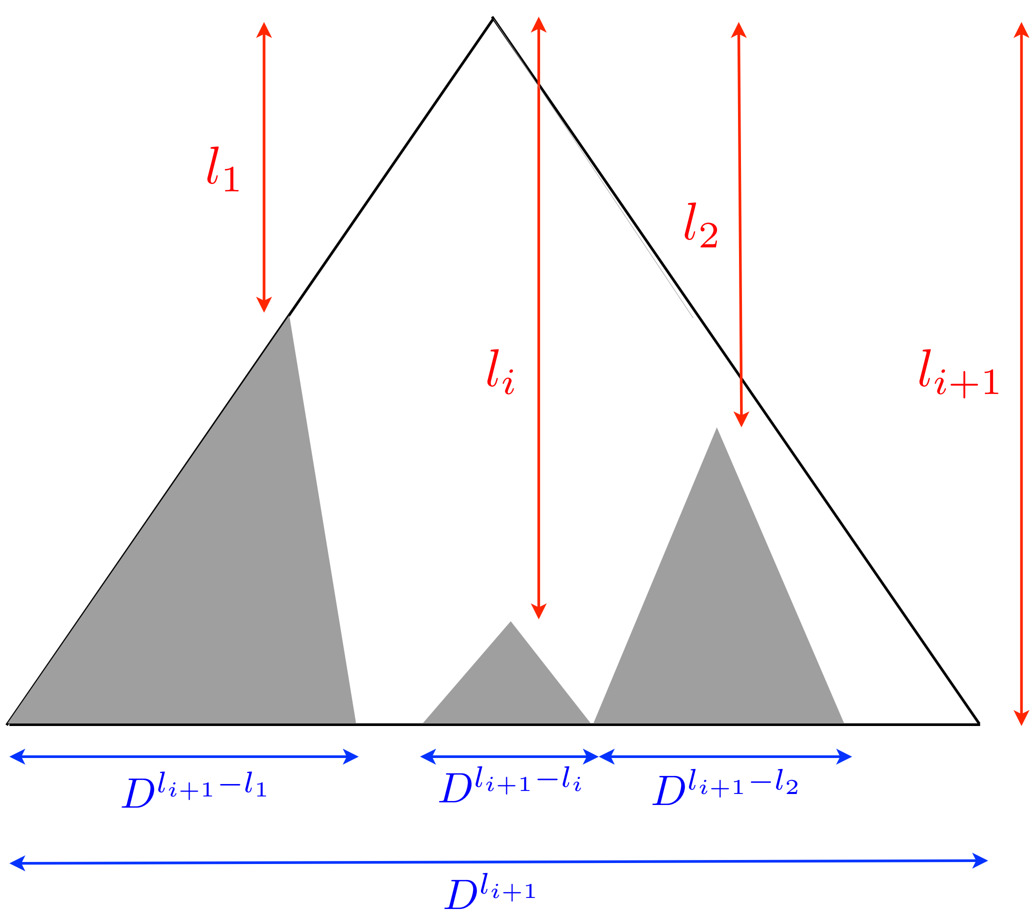
$$l_1 \leq l_2 \leq \cdots \leq l_m.$$

2. Consider all the D-ary sequences of lengths less than or equal to l_m and regard them as the nodes of the full D-ary tree of depth l_m. We will refer to a sequence of length l as a node of order l.

3. There are $D^{l_1} > 1$ (since $l_1 \geq 1$) nodes of order l_1 which can be chosen as the first codeword. Thus choosing the first codeword is always possible.

4. Assume that the first i codewords have been chosen successfully, where $1 \leq i \leq m-1$, and we want to choose a node of order l_{i+1} as the $(i+1)$st codeword such that it is not prefixed by any of the previously chosen codewords.

5. Since all the previously chosen codewords are not prefixes of each other, their descendants of order l_{i+1} do not overlap. The $(i+1)$st node to be chosen cannot be a descendant of any of the previously chosen codewords.



Theorem 4.11 There exists a D-ary prefix code with codeword lengths $l_1, l_2, \cdots, l_m$ if and only if the Kraft inequality

$$\sum_{k=1}^{m} D^{-l_k} \le 1$$

is satisfied.

Proof (Converse)

1. We need to prove the existence of a D-ary prefix code with codeword lengths $l_1, l_2, \cdots, l_m$ if these lengths satisfy the Kraft inequality. Without loss of generality, assume that

$$l_1 \le l_2 \le \cdots \le l_m.$$

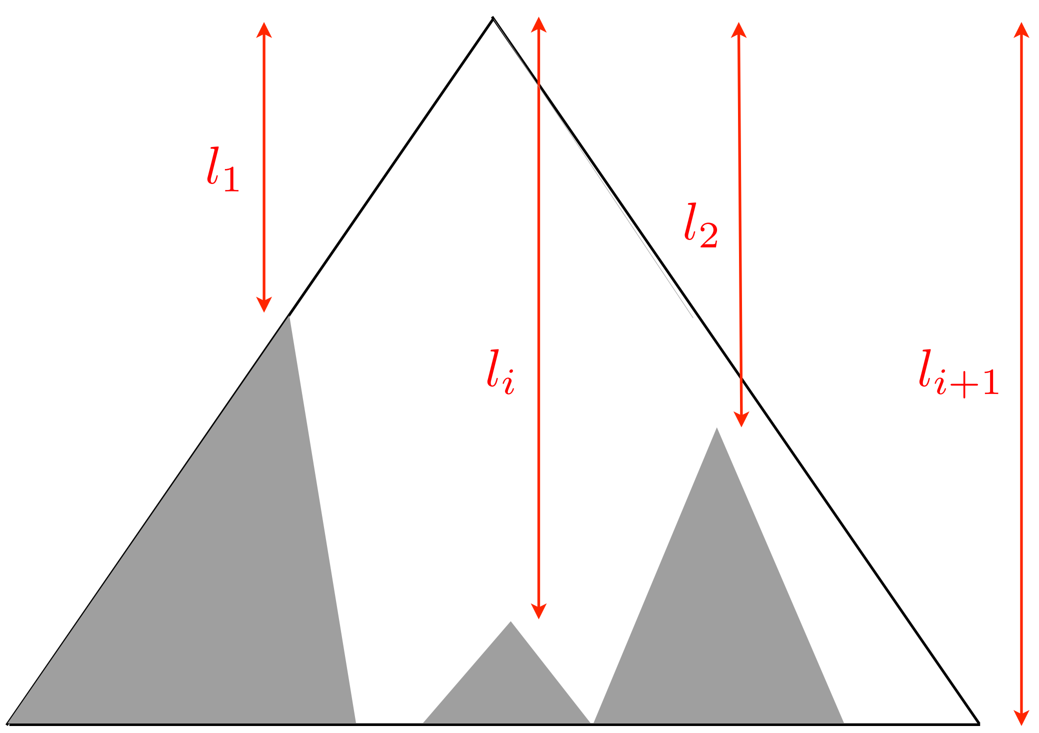
2. Consider all the D-ary sequences of lengths less than or equal to l_m and regard them as the nodes of the full D-ary tree of depth l_m. We will refer to a sequence of length l as a node of order l.

3. There are $D^{l_1} > 1$ (since $l_1 \ge 1$) nodes of order l_1 which can be chosen as the first codeword. Thus choosing the first codeword is always possible.

4. Assume that the first i codewords have been chosen successfully, where $1 \le i \le m-1$, and we want to choose a node of order l_{i+1} as the $(i+1)$st codeword such that it is not prefixed by any of the previously chosen codewords.

5. Since all the previously chosen codewords are not prefixes of each other, their descendants of order l_{i+1} do not overlap. The $(i+1)$st node to be chosen cannot be a descendant of any of the previously chosen codewords. Therefore, the number of nodes which can be chosen as the $(i+1)$st codeword is

$$D^{l_{i+1}} - D^{l_{i+1}-l_1} - D^{l_{i+1}-l_2} - \cdots - D^{l_{i+1}-l_i}.$$



Theorem 4.11 There exists a D-ary prefix code with codeword lengths l_1, $l_2, \cdots, l_m$ if and only if the Kraft inequality

$$\sum_{k=1}^{m} D^{-l_k} \leq 1$$

is satisfied.

Proof (Converse)

1. We need to prove the existence of a D-ary prefix code with codeword lengths $l_1, l_2, \cdots, l_m$ if these lengths satisfy the Kraft inequality. Without loss of generality, assume that

$$l_1 \leq l_2 \leq \cdots \leq l_m.$$

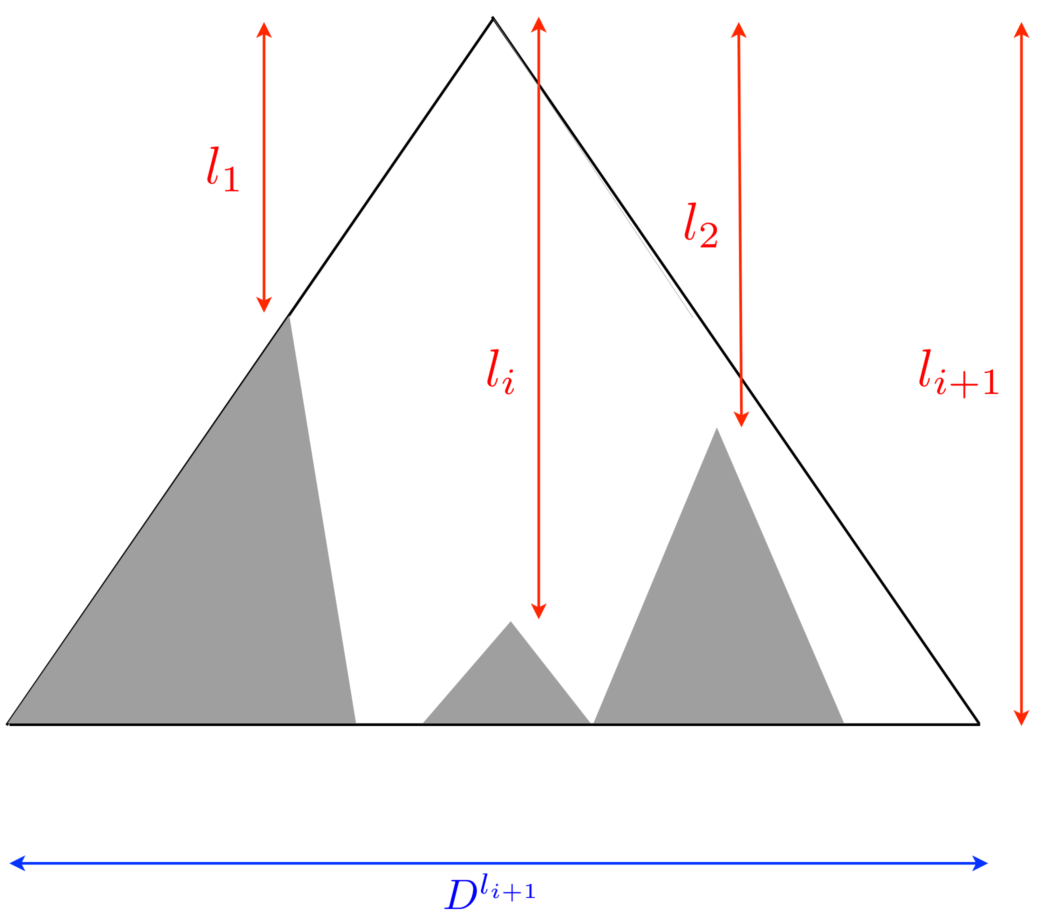
2. Consider all the D-ary sequences of lengths less than or equal to l_m and regard them as the nodes of the full D-ary tree of depth l_m. We will refer to a sequence of length l as a node of order l.

3. There are $D^{l_1} > 1$ (since $l_1 \geq 1$) nodes of order l_1 which can be chosen as the first codeword. Thus choosing the first codeword is always possible.

4. Assume that the first i codewords have been chosen successfully, where $1 \leq i \leq m-1$, and we want to choose a node of order l_{i+1} as the $(i+1)$st codeword such that it is not prefixed by any of the previously chosen codewords.

5. Since all the previously chosen codewords are not prefixes of each other, their descendants of order l_{i+1} do not overlap. The $(i+1)$st node to be chosen cannot be a descendant of any of the previously chosen codewords. Therefore, the number of nodes which can be chosen as the $(i+1)$st codeword is

$$D^{l_{i+1}} - D^{l_{i+1}-l_1} - D^{l_{i+1}-l_2} - \cdots - D^{l_{i+1}-l_i}.$$



Theorem 4.11 There exists a D-ary prefix code with codeword lengths $l_1, l_2, \cdots, l_m$ if and only if the Kraft inequality

$$\sum_{k=1}^{m} D^{-l_k} \leq 1$$

is satisfied.

Proof (Converse)

1. We need to prove the existence of a D-ary prefix code with codeword lengths $l_1, l_2, \cdots, l_m$ if these lengths satisfy the Kraft inequality. Without loss of generality, assume that

$$l_1 \leq l_2 \leq \cdots \leq l_m.$$

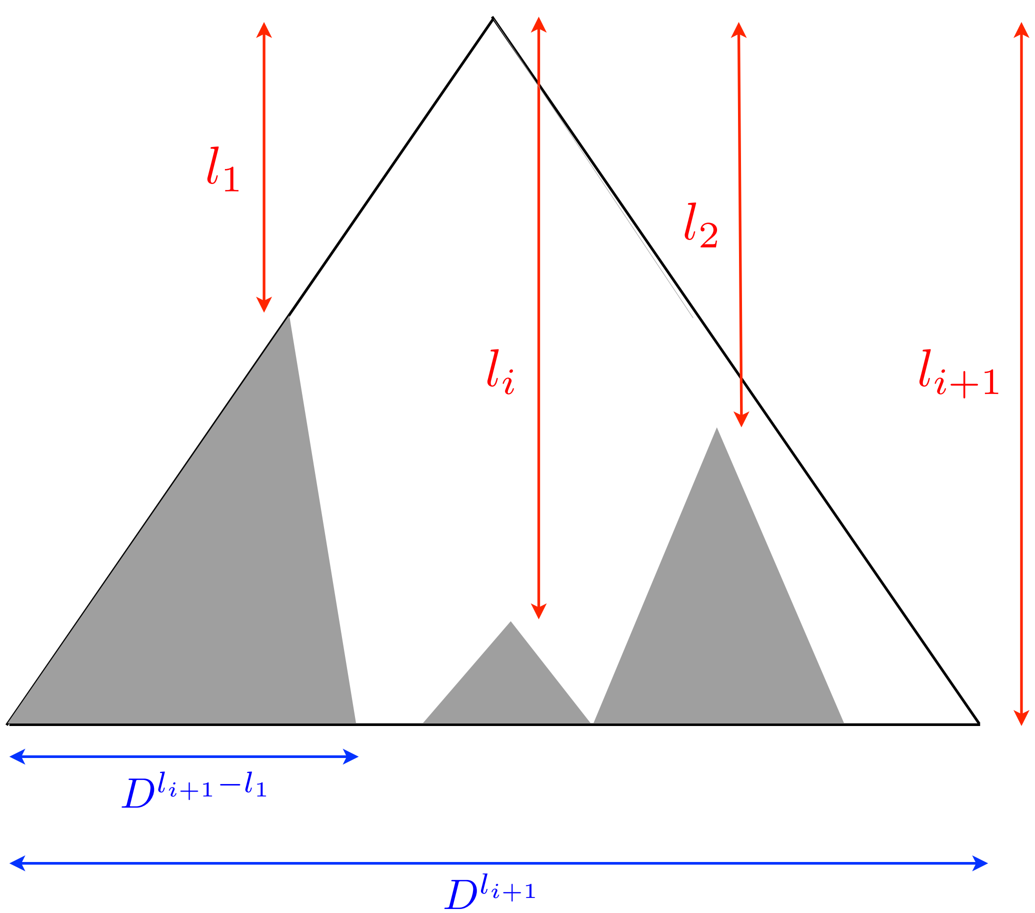
2. Consider all the D-ary sequences of lengths less than or equal to l_m and regard them as the nodes of the full D-ary tree of depth l_m. We will refer to a sequence of length l as a node of order l.

3. There are $D^{l_1} > 1$ (since $l_1 \geq 1$) nodes of order l_1 which can be chosen as the first codeword. Thus choosing the first codeword is always possible.

4. Assume that the first i codewords have been chosen successfully, where $1 \leq i \leq m-1$, and we want to choose a node of order l_{i+1} as the $(i+1)$st codeword such that it is not prefixed by any of the previously chosen codewords.

5. Since all the previously chosen codewords are not prefixes of each other, their descendants of order l_{i+1} do not overlap. The $(i+1)$st node to be chosen cannot be a descendant of any of the previously chosen codewords. Therefore, the number of nodes which can be chosen as the $(i+1)$st codeword is

$$D^{l_{i+1}} - D^{l_{i+1}-l_1} - D^{l_{i+1}-l_2} - \cdots - D^{l_{i+1}-l_i}.$$



Theorem 4.11 There exists a D-ary prefix code with codeword lengths $l_1, l_2, \cdots, l_m$ if and only if the Kraft inequality

$$\sum_{k=1}^{m} D^{-l_k} \leq 1$$

is satisfied.

Proof (Converse)

1. We need to prove the existence of a D-ary prefix code with codeword lengths $l_1, l_2, \cdots, l_m$ if these lengths satisfy the Kraft inequality. Without loss of generality, assume that

$$l_1 \leq l_2 \leq \cdots \leq l_m.$$

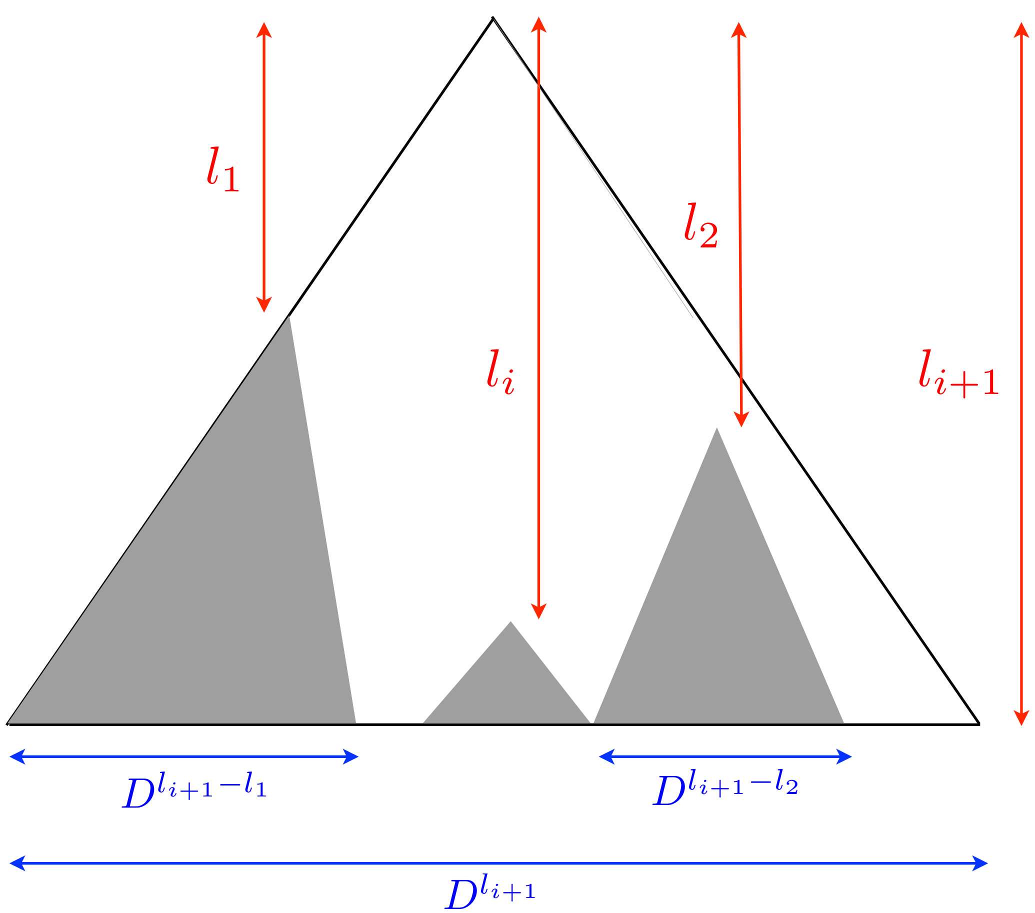
2. Consider all the D-ary sequences of lengths less than or equal to l_m and regard them as the nodes of the full D-tree of depth l_m. We will refer to a sequence of length l as a node of order l.

3. There are $D^{l_1} > 1$ (since $l_1 \geq 1$) nodes of order l_1 which can be chosen as the first codeword. Thus choosing the first codeword is always possible.

4. Assume that the first i codewords have been chosen successfully, where $1 \leq i \leq m-1$, and we want to choose a node of order l_{i+1} as the $(i+1)$st codeword such that it is not prefixed by any of the previously chosen codewords.

5. Since all the previously chosen codewords are not prefixes of each other, their descendants of order l_{i+1} do not overlap. The $(i+1)$st node to be chosen cannot be a descendant of any of the previously chosen codewords. Therefore, the number of nodes which can be chosen as the $(i+1)$st codeword is

$$D^{l_{i+1}} - D^{l_{i+1}-l_1} - D^{l_{i+1}-l_2} - \cdots - D^{l_{i+1}-l_i}.$$



Theorem 4.11 There exists a D-ary prefix code with codeword lengths $l_1, l_2, \cdots, l_m$ if and only if the Kraft inequality

$$\sum_{k=1}^{m} D^{-l_k} \leq 1$$

is satisfied.

Proof (Converse)

1. We need to prove the existence of a D-ary prefix code with codeword lengths $l_1, l_2, \cdots, l_m$ if these lengths satisfy the Kraft inequality. Without loss of generality, assume that

$$l_1 \leq l_2 \leq \cdots \leq l_m.$$

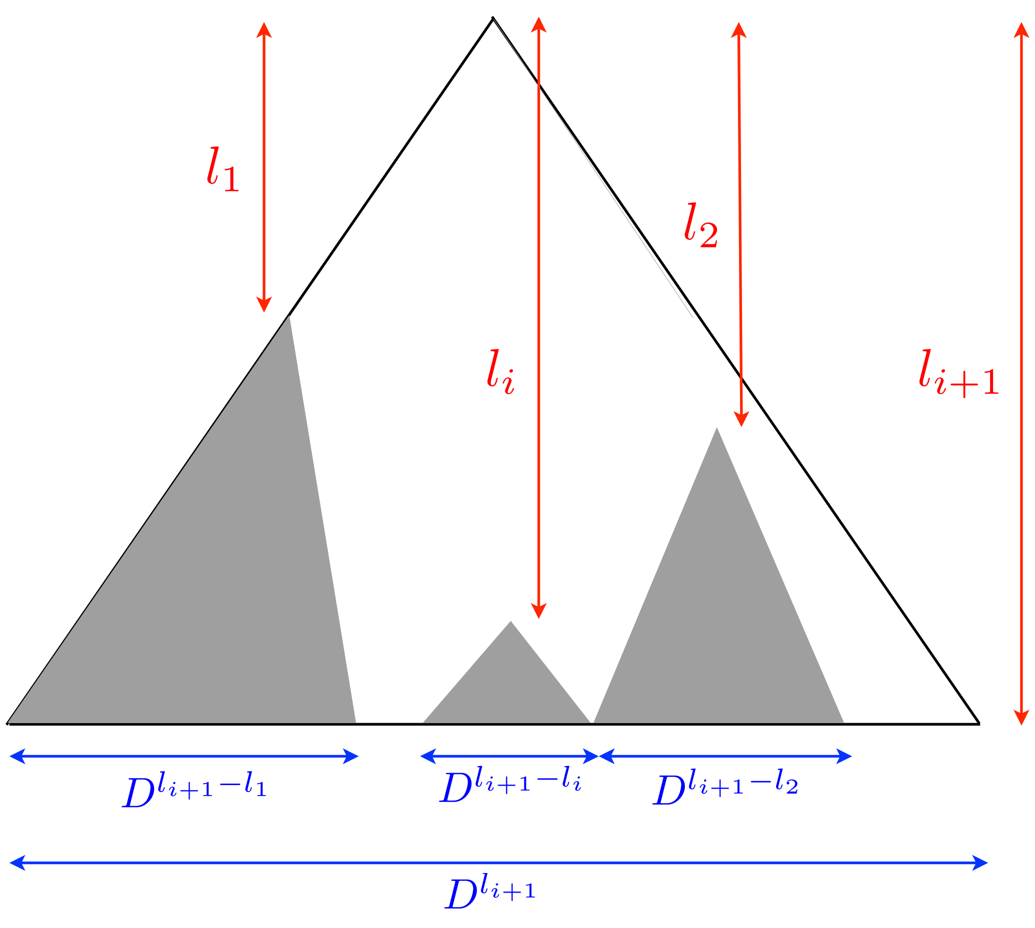
2. Consider all the D-ary sequences of lengths less than or equal to l_m and regard them as the nodes of the full D-ary tree of depth l_m. We will refer to a sequence of length l as a node of order l.

3. There are $D^{l_1} > 1$ (since $l_1 \geq 1$) nodes of order l_1 which can be chosen as the first codeword. Thus choosing the first codeword is always possible.

4. Assume that the first i codewords have been chosen successfully, where $1 \leq i \leq m-1$, and we want to choose a node of order l_{i+1} as the $(i+1)$st codeword such that it is not prefixed by any of the previously chosen codewords.

5. Since all the previously chosen codewords are not prefixes of each other, their descendants of order l_{i+1} do not overlap. The $(i+1)$st node to be chosen cannot be a descendant of any of the previously chosen codewords. Therefore, the number of nodes which can be chosen as the $(i+1)$st codeword is

$$D^{l_{i+1}} - D^{l_{i+1}-l_1} - D^{l_{i+1}-l_2} - \cdots - D^{l_{i+1}-l_i}.$$



Theorem 4.11 There exists a D-ary prefix code with codeword lengths l_1, $l_2, \cdots, l_m$ if and only if the Kraft inequality

$$\sum_{k=1}^{m} D^{-l_k} \leq 1$$

is satisfied.

Proof (Converse)

1. We need to prove the existence of a D-ary prefix code with codeword lengths $l_1, l_2, \cdots, l_m$ if these lengths satisfy the Kraft inequality. Without loss of generality, assume that

$$l_1 \leq l_2 \leq \cdots \leq l_m.$$

2. Consider all the D-ary sequences of lengths less than or equal to l_m and regard them as the nodes of the full D-ary tree of depth l_m. We will refer to a sequence of length l as a node of order l.

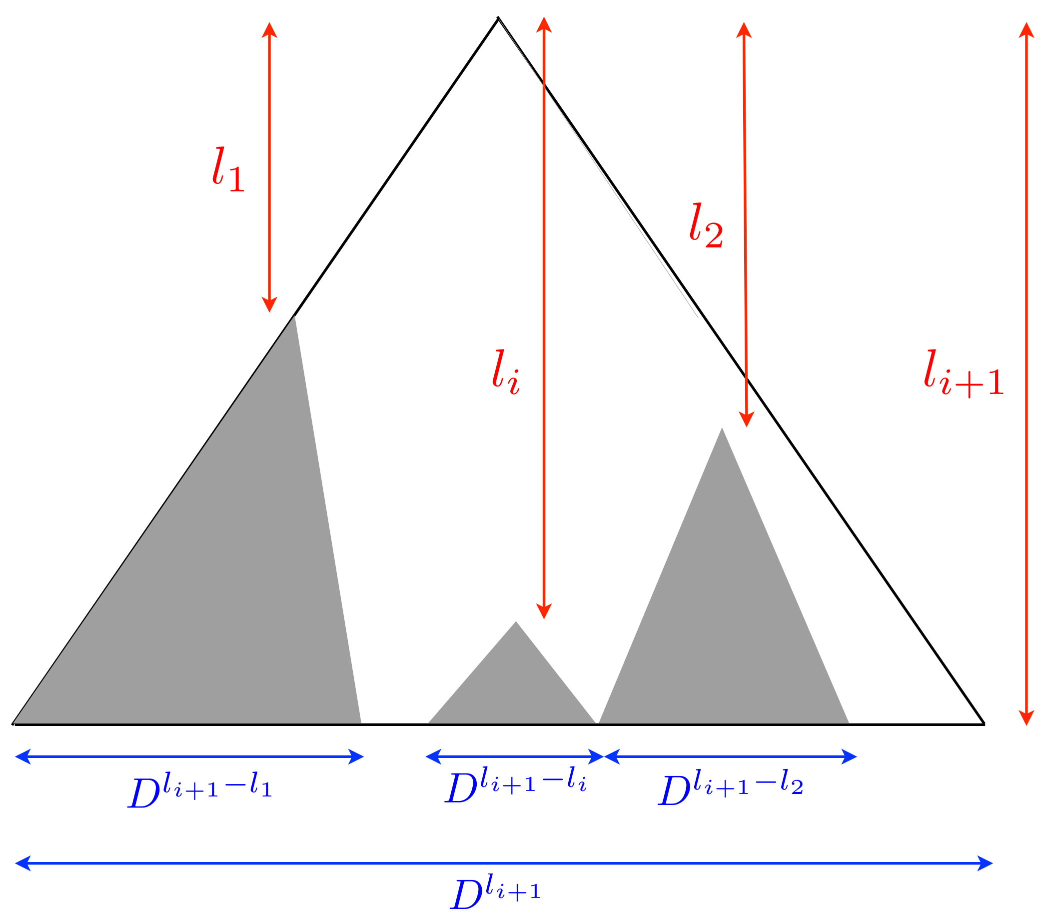
3. There are $D^{l_1} > 1$ (since $l_1 \geq 1$) nodes of order l_1 which can be chosen as the first codeword. Thus choosing the first codeword is always possible.

4. Assume that the first i codewords have been chosen successfully, where $1 \leq i \leq m-1$, and we want to choose a node of order l_{i+1} as the $(i+1)$st codeword such that it is not prefixed by any of the previously chosen codewords.

5. Since all the previously chosen codewords are not prefixes of each other, their descendants of order l_{i+1} do not overlap. The $(i+1)$st node to be chosen cannot be a descendant of any of the previously chosen codewords. Therefore, the number of nodes which can be chosen as the $(i+1)$st codeword is

$$D^{l_{i+1}} - D^{l_{i+1}-l_1} - D^{l_{i+1}-l_2} - \cdots - D^{l_{i+1}-l_i}.$$

6. If $l_1, l_2, \cdots, l_m$ satisfy the Kraft inequality, we have



Theorem 4.11 There exists a D-ary prefix code with codeword lengths $l_1,\ l_2, \cdots, l_m$ if and only if the Kraft inequality

$$\sum_{k=1}^{m} D^{-l_k} \leq 1$$

is satisfied.

Proof (Converse)

1. We need to prove the existence of a D-ary prefix code with codeword lengths $l_1, l_2, \cdots, l_m$ if these lengths satisfy the Kraft inequality. Without loss of generality, assume that

$$l_1 \leq l_2 \leq \cdots \leq l_m.$$

2. Consider all the D-ary sequences of lengths less than or equal to l_m and regard them as the nodes of the full D-ary tree of depth l_m. We will refer to a sequence of length l as a node of order l.

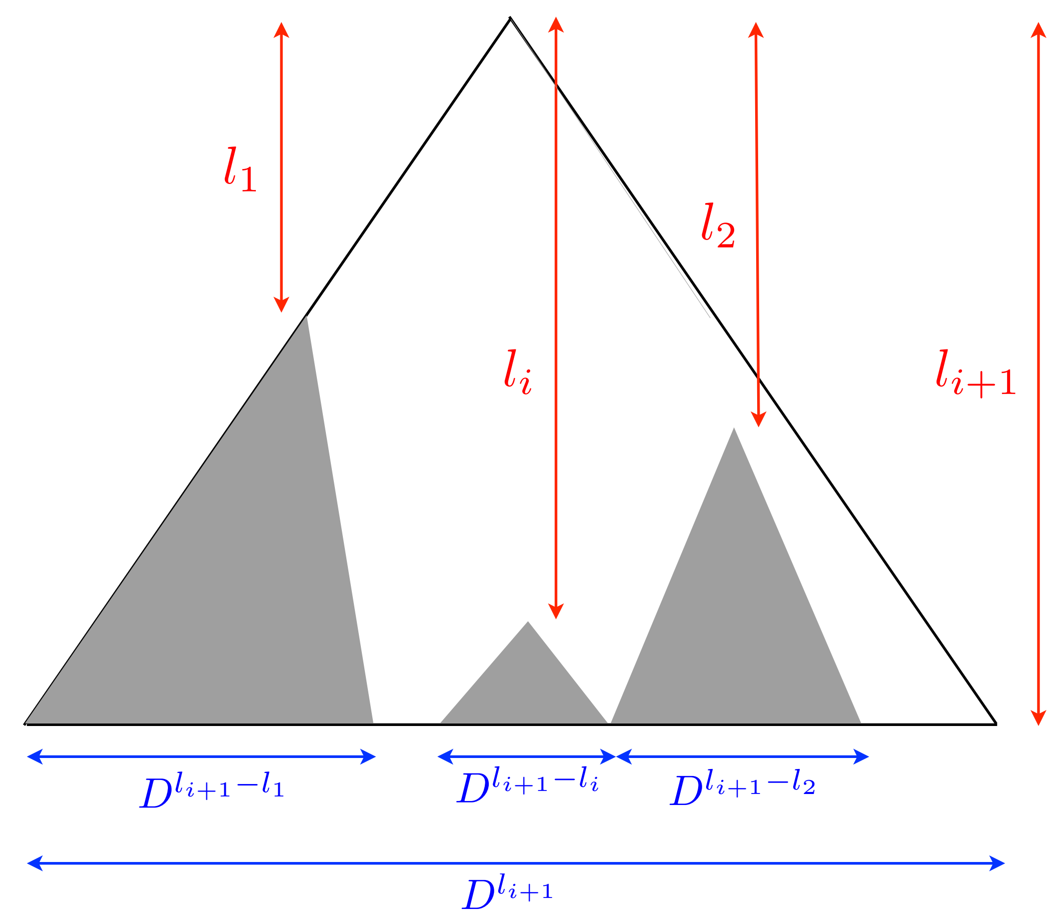
3. There are $D^{l_1} > 1$ (since $l_1 \geq 1$) nodes of order l_1 which can be chosen as the first codeword. Thus choosing the first codeword is always possible.

4. Assume that the first i codewords have been chosen successfully, where $1 \leq i \leq m-1$, and we want to choose a node of order l_{i+1} as the $(i+1)$st codeword such that it is not prefixed by any of the previously chosen codewords.

5. Since all the previously chosen codewords are not prefixes of each other, their descendants of order l_{i+1} do not overlap. The $(i+1)$st node to be chosen cannot be a descendant of any of the previously chosen codewords. Therefore, the number of nodes which can be chosen as the $(i+1)$st codeword is

$$D^{l_{i+1}} - D^{l_{i+1}-l_1} - D^{l_{i+1}-l_2} - \cdots - D^{l_{i+1}-l_i}.$$

6. If $l_1, l_2, \cdots, l_m$ satisfy the Kraft inequality, we have



Theorem 4.11 There exists a D-ary prefix code with codeword lengths $l_1, l_2, \cdots, l_m$ if and only if the Kraft inequality

$$\sum_{k=1}^{m} D^{-l_k} \leq 1$$

is satisfied.

Proof (Converse)

1. We need to prove the existence of a D-ary prefix code with codeword lengths $l_1, l_2, \cdots, l_m$ if these lengths satisfy the Kraft inequality. Without loss of generality, assume that

$$l_1 \leq l_2 \leq \cdots \leq l_m.$$

2. Consider all the D-ary sequences of lengths less than or equal to l_m and regard them as the nodes of the full D-ary tree of depth l_m. We will refer to a sequence of length l as a node of order l.

3. There are $D^{l_1} > 1$ (since $l_1 \geq 1$) nodes of order l_1 which can be chosen as the first codeword. Thus choosing the first codeword is always possible.

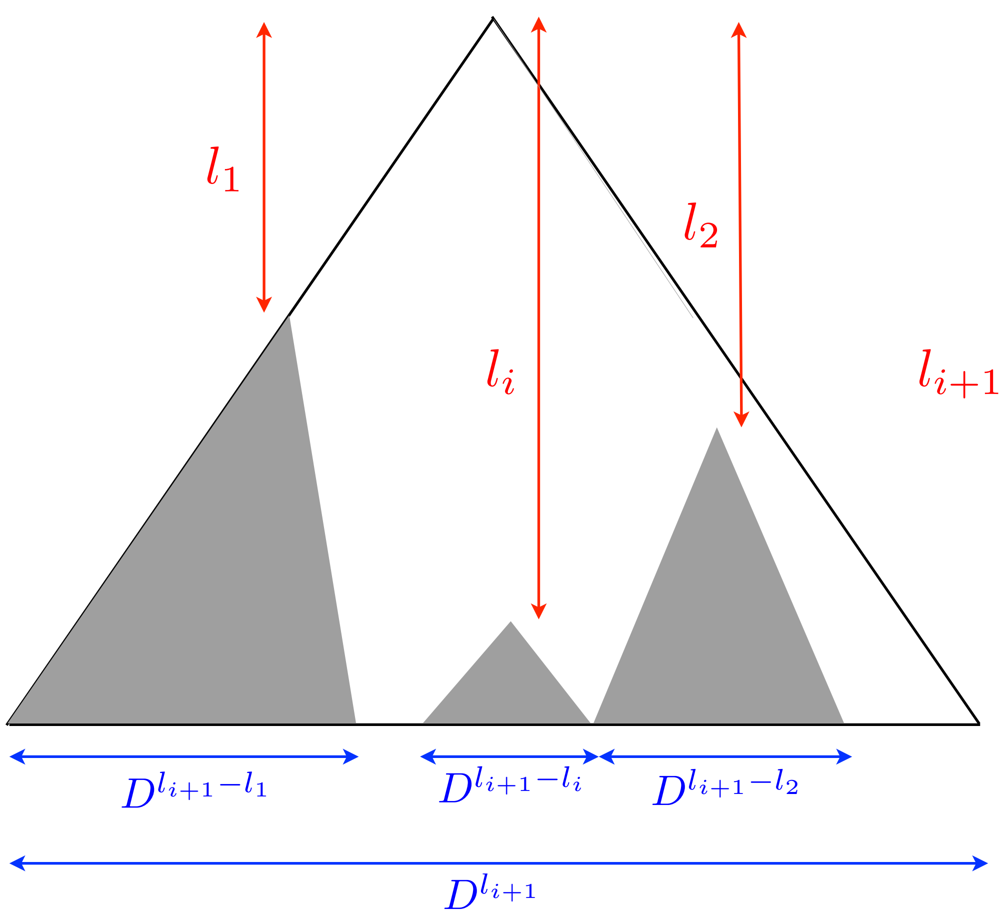
4. Assume that the first i codewords have been chosen successfully, where $1 \leq i \leq m-1$, and we want to choose a node of order l_{i+1} as the $(i+1)$st codeword such that it is not prefixed by any of the previously chosen codewords.

5. Since all the previously chosen codewords are not prefixes of each other, their descendants of order l_{i+1} do not overlap. The $(i+1)$st node to be chosen cannot be a descendant of any of the previously chosen codewords. Therefore, the number of nodes which can be chosen as the $(i+1)$st codeword is

$$D^{l_{i+1}} - D^{l_{i+1}-l_1} - D^{l_{i+1}-l_2} - \cdots - D^{l_{i+1}-l_i}.$$

6. If $l_1, l_2, \cdots, l_m$ satisfy the Kraft inequality, we have

$$D^{-l_1} + \cdots + D^{-l_i} + D^{-l_{i+1}} \leq 1.$$



Theorem 4.11 There exists a D-ary prefix code with codeword lengths $l_1, l_2, \cdots, l_m$ if and only if the Kraft inequality

$$\sum_{k=1}^{m} D^{-l_k} \leq 1$$

is satisfied.

Proof (Converse)

1. We need to prove the existence of a D-ary prefix code with codeword lengths $l_1, l_2, \cdots, l_m$ if these lengths satisfy the Kraft inequality. Without loss of generality, assume that

$$l_1 \leq l_2 \leq \cdots \leq l_m.$$

2. Consider all the D-ary sequences of lengths less than or equal to l_m and regard them as the nodes of the full D-ary tree of depth l_m. We will refer to a sequence of length l as a node of order l.

3. There are $D^{l_1} > 1$ (since $l_1 \geq 1$) nodes of order l_1 which can be chosen as the first codeword. Thus choosing the first codeword is always possible.

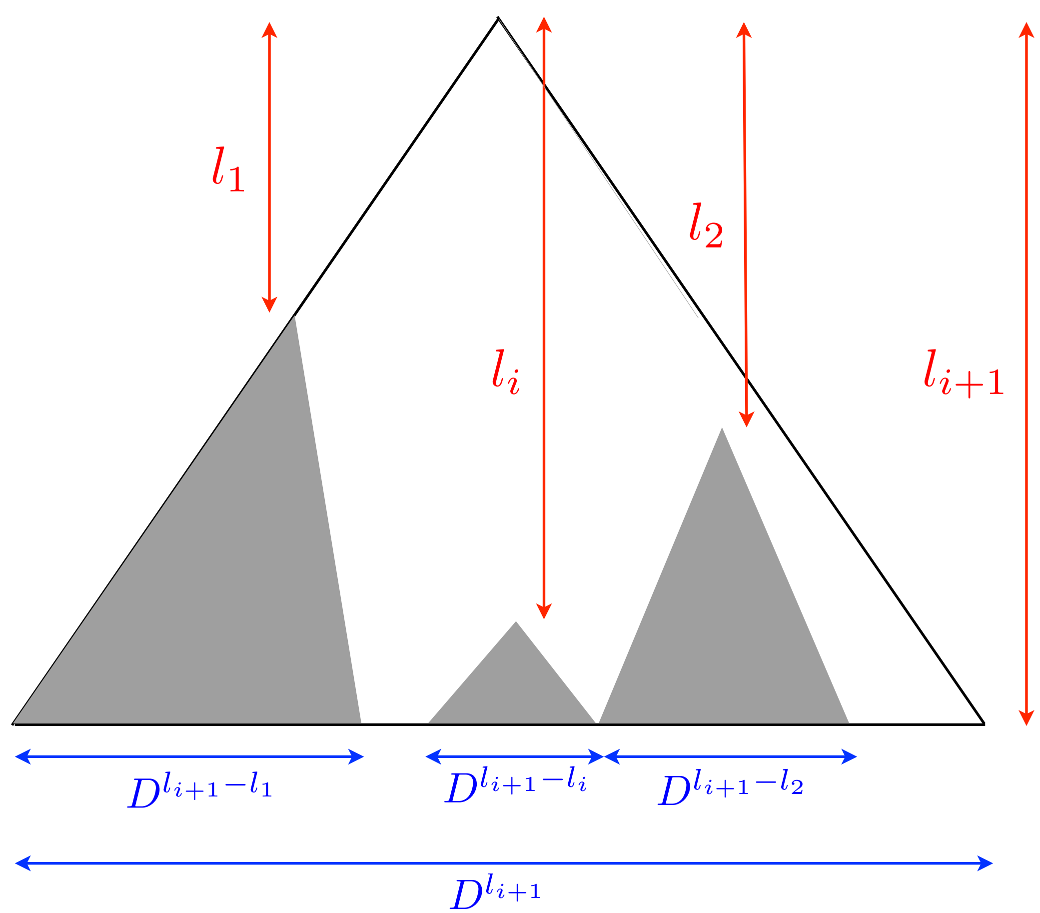
4. Assume that the first i codewords have been chosen successfully, where $1 \leq i \leq m-1$, and we want to choose a node of order l_{i+1} as the $(i+1)$st codeword such that it is not prefixed by any of the previously chosen codewords.

5. Since all the previously chosen codewords are not prefixes of each other, their descendants of order l_{i+1} do not overlap. The $(i+1)$st node to be chosen cannot be a descendant of any of the previously chosen codewords. Therefore, the number of nodes which can be chosen as the $(i+1)$st codeword is

$$D^{l_{i+1}} - D^{l_{i+1}-l_1} - D^{l_{i+1}-l_2} - \cdots - D^{l_{i+1}-l_i}.$$

6. If $l_1, l_2, \cdots, l_m$ satisfy the Kraft inequality, we have

$$D^{-l_1} + \cdots + D^{-l_i} + D^{-l_{i+1}} \leq 1.$$



Theorem 4.11 There exists a D-ary prefix code with codeword lengths $l_1, l_2, \cdots, l_m$ if and only if the Kraft inequality

$$\sum_{k=1}^{m} D^{-l_k} \leq 1$$

is satisfied.

Proof (Converse)

1. We need to prove the existence of a D-ary prefix code with codeword lengths $l_1, l_2, \cdots, l_m$ if these lengths satisfy the Kraft inequality. Without loss of generality, assume that

$$l_1 \leq l_2 \leq \cdots \leq l_m.$$

2. Consider all the D-ary sequences of lengths less than or equal to l_m and regard them as the nodes of the full D-ary tree of depth l_m. We will refer to a sequence of length l as a node of order l.

3. There are $D^{l_1} > 1$ (since $l_1 \geq 1$) nodes of order l_1 which can be chosen as the first codeword. Thus choosing the first codeword is always possible.

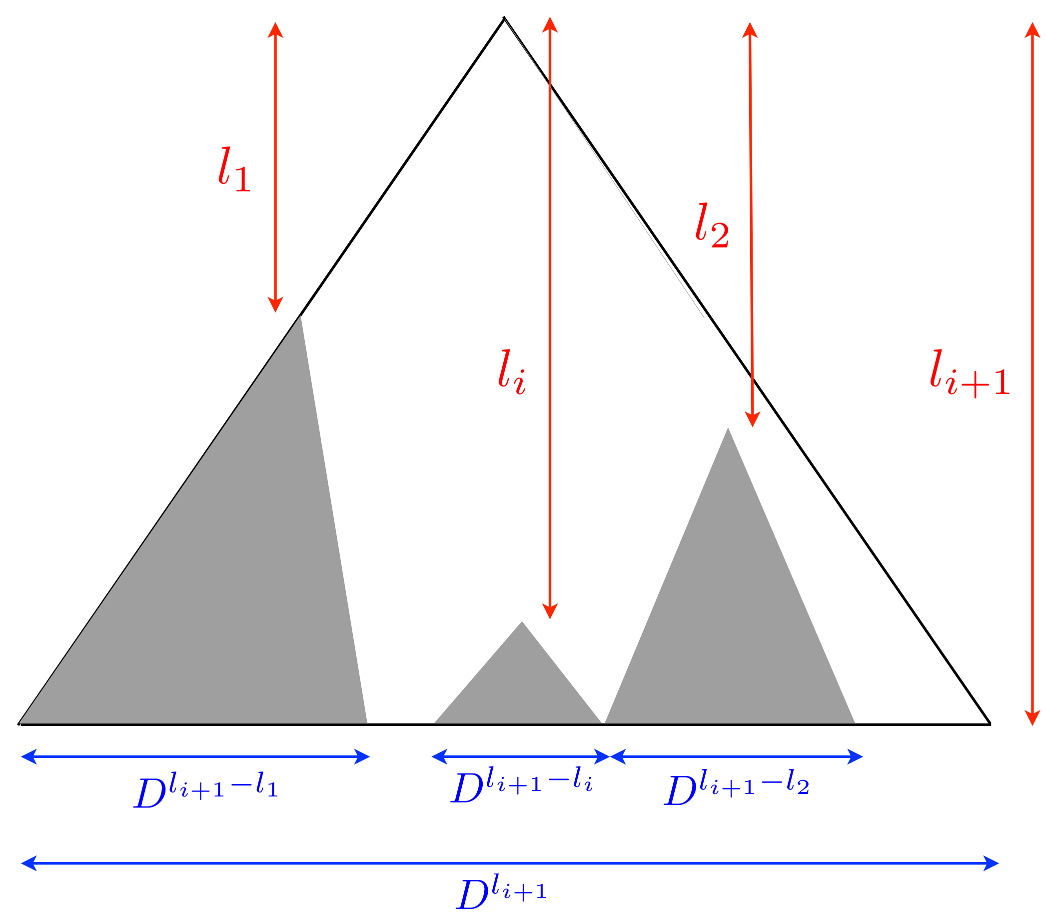
4. Assume that the first i codewords have been chosen successfully, where $1 \leq i \leq m-1$, and we want to choose a node of order l_{i+1} as the $(i+1)$st codeword such that it is not prefixed by any of the previously chosen codewords.

5. Since all the previously chosen codewords are not prefixes of each other, their descendants of order l_{i+1} do not overlap. The $(i+1)$st node to be chosen cannot be a descendant of any of the previously chosen codewords. Therefore, the number of nodes which can be chosen as the $(i+1)$st codeword is

$$D^{l_{i+1}} - D^{l_{i+1}-l_1} - D^{l_{i+1}-l_2} - \cdots - D^{l_{i+1}-l_i}.$$

6. If $l_1, l_2, \cdots, l_m$ satisfy the Kraft inequality, we have

$$D^{-l_1} + \cdots + D^{-l_i} + D^{-l_{i+1}} \leq 1.$$



Theorem 4.11 There exists a D-ary prefix code with codeword lengths $l_1, l_2, \cdots, l_m$ if and only if the Kraft inequality

$$\sum_{k=1}^{m} D^{-l_k} \leq 1$$

is satisfied.

Proof (Converse)

1. We need to prove the existence of a D-ary prefix code with codeword lengths $l_1, l_2, \cdots, l_m$ if these lengths satisfy the Kraft inequality. Without loss of generality, assume that

$$l_1 \leq l_2 \leq \cdots \leq l_m.$$

2. Consider all the D-ary sequences of lengths less than or equal to l_m and regard them as the nodes of the full D-ary tree of depth l_m. We will refer to a sequence of length l as a node of order l.

3. There are $D^{l_1} > 1$ (since $l_1 \geq 1$) nodes of order l_1 which can be chosen as the first codeword. Thus choosing the first codeword is always possible.

4. Assume that the first i codewords have been chosen successfully, where $1 \leq i \leq m-1$, and we want to choose a node of order l_{i+1} as the $(i+1)$st codeword such that it is not prefixed by any of the previously chosen codewords.

5. Since all the previously chosen codewords are not prefixes of each other, their descendants of order l_{i+1} do not overlap. The $(i+1)$st node to be chosen cannot be a descendant of any of the previously chosen codewords. Therefore, the number of nodes which can be chosen as the $(i+1)$st codeword is

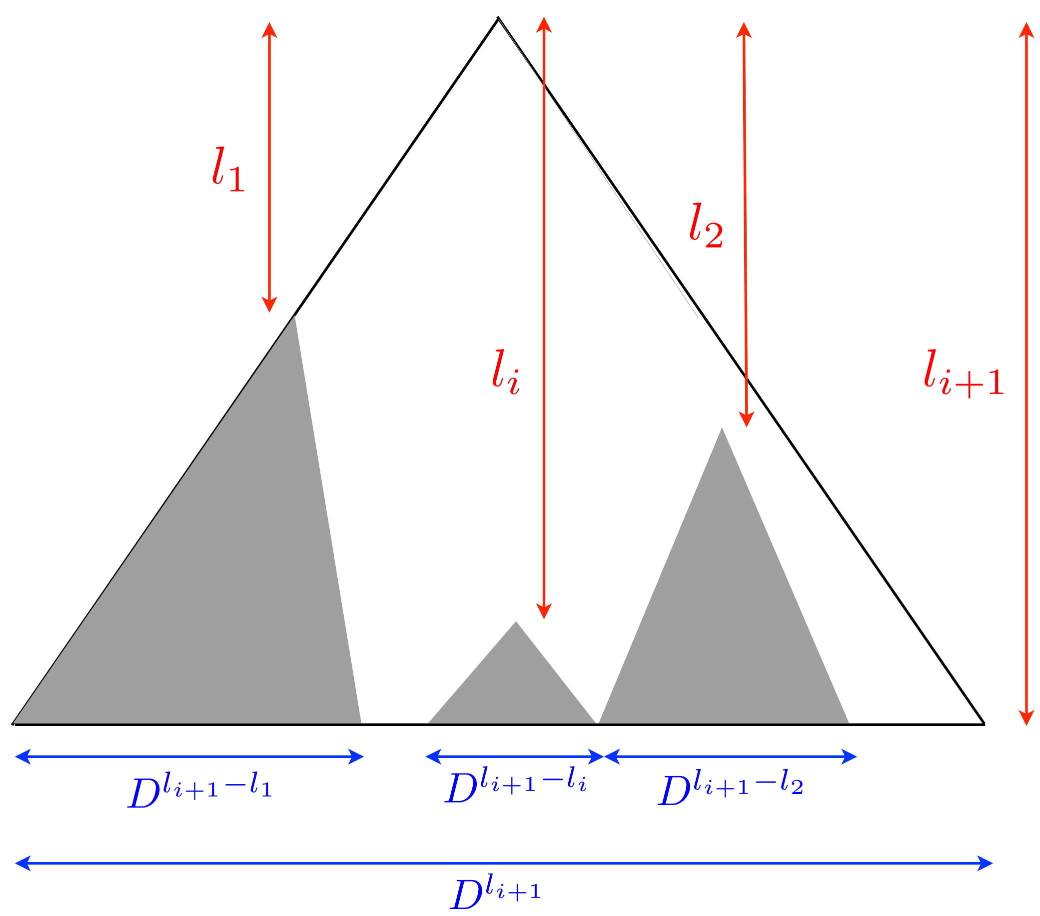
$$D^{l_{i+1}} - D^{l_{i+1}-l_1} - D^{l_{i+1}-l_2} - \cdots - D^{l_{i+1}-l_i}.$$

6. If $l_1, l_2, \cdots, l_m$ satisfy the Kraft inequality, we have

$$D^{-l_1} + \cdots + D^{-l_i} + D^{-l_{i+1}} \leq 1.$$

7. Multiplying by $D^{l_{i+1}}$, we have

$$D^{l_{i+1}-l_1} + \cdots + D^{l_{i+1}-l_i} + D^{l_{i+1}-l_{i+1}} \leq D^{l_{i+1}}$$



Theorem 4.11 There exists a D-ary prefix code with codeword lengths $l_1, l_2, \cdots, l_m$ if and only if the Kraft inequality

$$\sum_{k=1}^{m} D^{-l_k} \leq 1$$

is satisfied.

Proof (Converse)

1. We need to prove the existence of a D-ary prefix code with codeword lengths $l_1, l_2, \cdots, l_m$ if these lengths satisfy the Kraft inequality. Without loss of generality, assume that

$$l_1 \leq l_2 \leq \cdots \leq l_m.$$

2. Consider all the D-ary sequences of lengths less than or equal to l_m and regard them as the nodes of the full D-ary tree of depth l_m. We will refer to a sequence of length l as a node of order l.

3. There are $D^{l_1} > 1$ (since $l_1 \geq 1$) nodes of order l_1 which can be chosen as the first codeword. Thus choosing the first codeword is always possible.

4. Assume that the first i codewords have been chosen successfully, where $1 \leq i \leq m-1$, and we want to choose a node of order l_{i+1} as the $(i+1)$st codeword such that it is not prefixed by any of the previously chosen codewords.

5. Since all the previously chosen codewords are not prefixes of each other, their descendants of order l_{i+1} do not overlap. The $(i+1)$st node to be chosen cannot be a descendant of any of the previously chosen codewords. Therefore, the number of nodes which can be chosen as the $(i+1)$st codeword is

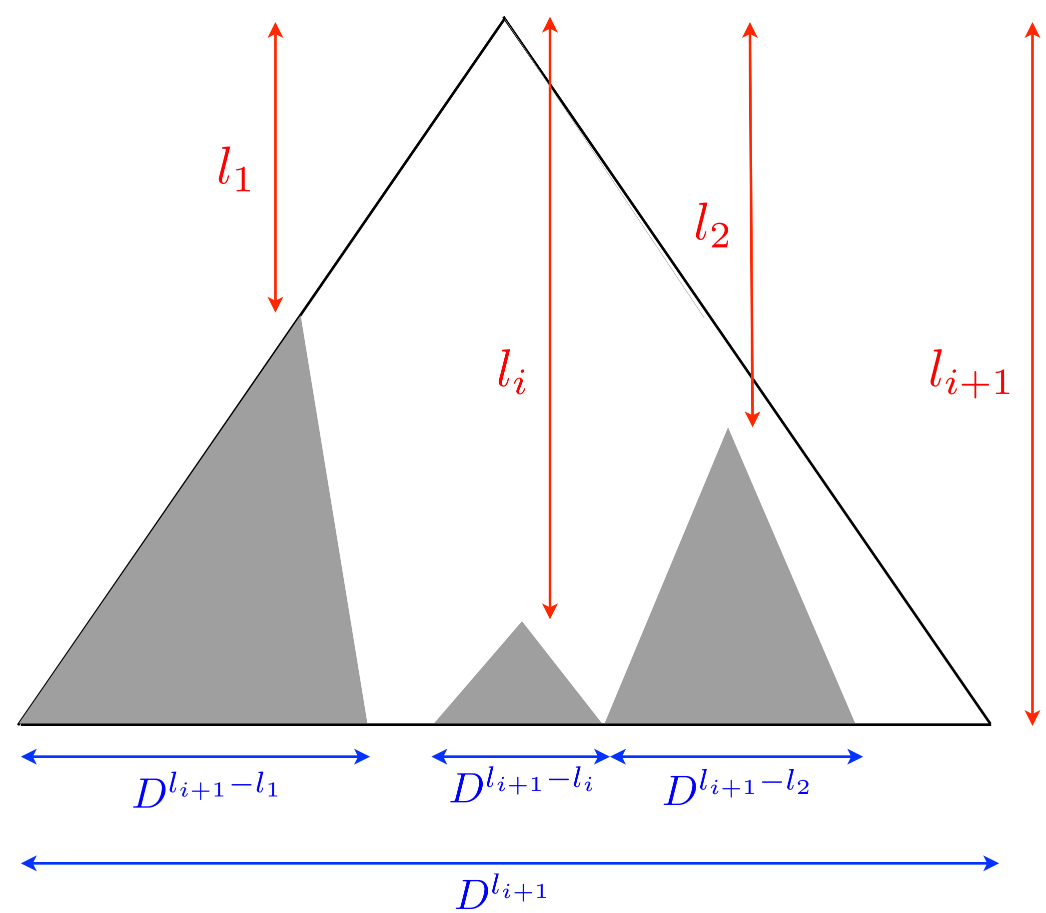
$$D^{l_{i+1}} - D^{l_{i+1}-l_1} - D^{l_{i+1}-l_2} - \cdots - D^{l_{i+1}-l_i}.$$

6. If $l_1, l_2, \cdots, l_m$ satisfy the Kraft inequality, we have

$$D^{-l_1} + \cdots + D^{-l_i} + D^{-l_{i+1}} \leq 1.$$

7. Multiplying by $D^{l_{i+1}}$, we have

$$D^{l_{i+1}-l_1} + \cdots + D^{l_{i+1}-l_i} + D^{l_{i+1}-l_{i+1}} \leq D^{l_{i+1}}$$



Theorem 4.11 There exists a D-ary prefix code with codeword lengths $l_1, l_2, \cdots, l_m$ if and only if the Kraft inequality

$$\sum_{k=1}^{m} D^{-l_k} \leq 1$$

is satisfied.

Proof (Converse)

1. We need to prove the existence of a D-ary prefix code with codeword lengths $l_1, l_2, \cdots, l_m$ if these lengths satisfy the Kraft inequality. Without loss of generality, assume that

$$l_1 \leq l_2 \leq \cdots \leq l_m.$$

2. Consider all the D-ary sequences of lengths less than or equal to l_m and regard them as the nodes of the full D-ary tree of depth l_m. We will refer to a sequence of length l as a node of order l.

3. There are $D^{l_1} > 1$ (since $l_1 \geq 1$) nodes of order l_1 which can be chosen as the first codeword. Thus choosing the first codeword is always possible.

4. Assume that the first i codewords have been chosen successfully, where $1 \leq i \leq m-1$, and we want to choose a node of order l_{i+1} as the $(i+1)$st codeword such that it is not prefixed by any of the previously chosen codewords.

5. Since all the previously chosen codewords are not prefixes of each other, their descendants of order l_{i+1} do not overlap. The $(i+1)$st node to be chosen cannot be a descendant of any of the previously chosen codewords. Therefore, the number of nodes which can be chosen as the $(i+1)$st codeword is

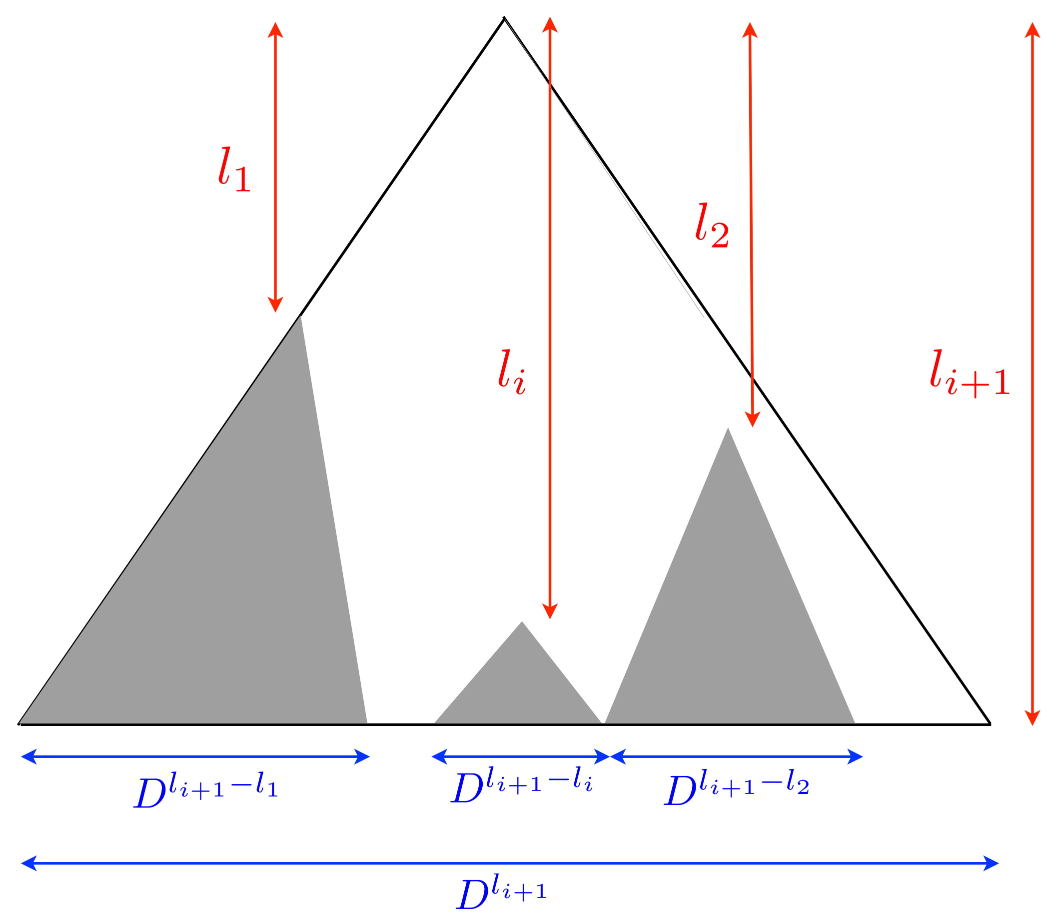
$$D^{l_{i+1}} - D^{l_{i+1}-l_1} - D^{l_{i+1}-l_2} - \cdots - D^{l_{i+1}-l_i}.$$

6. If $l_1, l_2, \cdots, l_m$ satisfy the Kraft inequality, we have

$$\underline{D^{-l_1}} + \cdots + D^{-l_i} + D^{-l_{i+1}} \leq 1.$$

7. Multiplying by $D^{l_{i+1}}$, we have

$$\underline{D^{l_{i+1}-l_1}} + \cdots + D^{l_{i+1}-l_i} + D^{l_{i+1}-l_{i+1}} \leq D^{l_{i+1}}$$



Theorem 4.11 There exists a D-ary prefix code with codeword lengths $l_1, l_2, \cdots, l_m$ if and only if the Kraft inequality

$$\sum_{k=1}^{m} D^{-l_k} \leq 1$$

is satisfied.

Proof (Converse)

1. We need to prove the existence of a D-ary prefix code with codeword lengths $l_1, l_2, \cdots, l_m$ if these lengths satisfy the Kraft inequality. Without loss of generality, assume that

$$l_1 \leq l_2 \leq \cdots \leq l_m.$$

2. Consider all the D-ary sequences of lengths less than or equal to l_m and regard them as the nodes of the full D-ary tree of depth l_m. We will refer to a sequence of length l as a node of order l.

3. There are $D^{l_1} > 1$ (since $l_1 \geq 1$) nodes of order l_1 which can be chosen as the first codeword. Thus choosing the first codeword is always possible.

4. Assume that the first i codewords have been chosen successfully, where $1 \leq i \leq m-1$, and we want to choose a node of order l_{i+1} as the $(i+1)$st codeword such that it is not prefixed by any of the previously chosen codewords.

5. Since all the previously chosen codewords are not prefixes of each other, their descendants of order l_{i+1} do not overlap. The $(i+1)$st node to be chosen cannot be a descendant of any of the previously chosen codewords. Therefore, the number of nodes which can be chosen as the $(i+1)$st codeword is

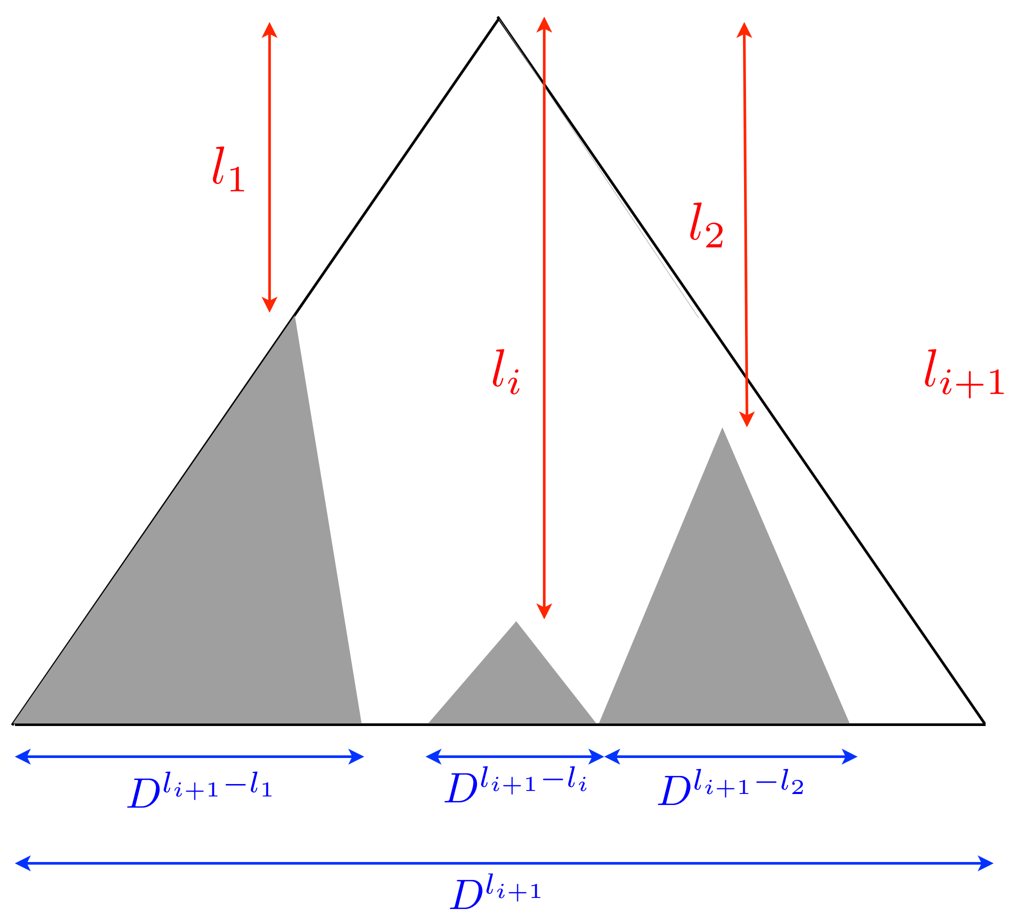
$$D^{l_{i+1}} - D^{l_{i+1}-l_1} - D^{l_{i+1}-l_2} - \cdots - D^{l_{i+1}-l_i}.$$

6. If $l_1, l_2, \cdots, l_m$ satisfy the Kraft inequality, we have

$$D^{-l_1} + \cdots + D^{-l_i} + D^{-l_{i+1}} \leq 1.$$

7. Multiplying by $D^{l_{i+1}}$, we have

$$D^{l_{i+1}-l_1} + \cdots + D^{l_{i+1}-l_i} + D^{l_{i+1}-l_{i+1}} \leq D^{l_{i+1}}$$



Theorem 4.11 There exists a D-ary prefix code with codeword lengths $l_1, l_2, \cdots, l_m$ if and only if the Kraft inequality

$$\sum_{k=1}^{m} D^{-l_k} \leq 1$$

is satisfied.

Proof (Converse)

1. We need to prove the existence of a D-ary prefix code with codeword lengths $l_1, l_2, \cdots, l_m$ if these lengths satisfy the Kraft inequality. Without loss of generality, assume that

$$l_1 \leq l_2 \leq \cdots \leq l_m.$$

2. Consider all the D-ary sequences of lengths less than or equal to l_m and regard them as the nodes of the full D-ary tree of depth l_m. We will refer to a sequence of length l as a node of order l.

3. There are $D^{l_1} > 1$ (since $l_1 \geq 1$) nodes of order l_1 which can be chosen as the first codeword. Thus choosing the first codeword is always possible.

4. Assume that the first i codewords have been chosen successfully, where $1 \leq i \leq m-1$, and we want to choose a node of order l_{i+1} as the $(i+1)$st codeword such that it is not prefixed by any of the previously chosen codewords.

5. Since all the previously chosen codewords are not prefixes of each other, their descendants of order l_{i+1} do not overlap. The $(i+1)$st node to be chosen cannot be a descendant of any of the previously chosen codewords. Therefore, the number of nodes which can be chosen as the $(i+1)$st codeword is

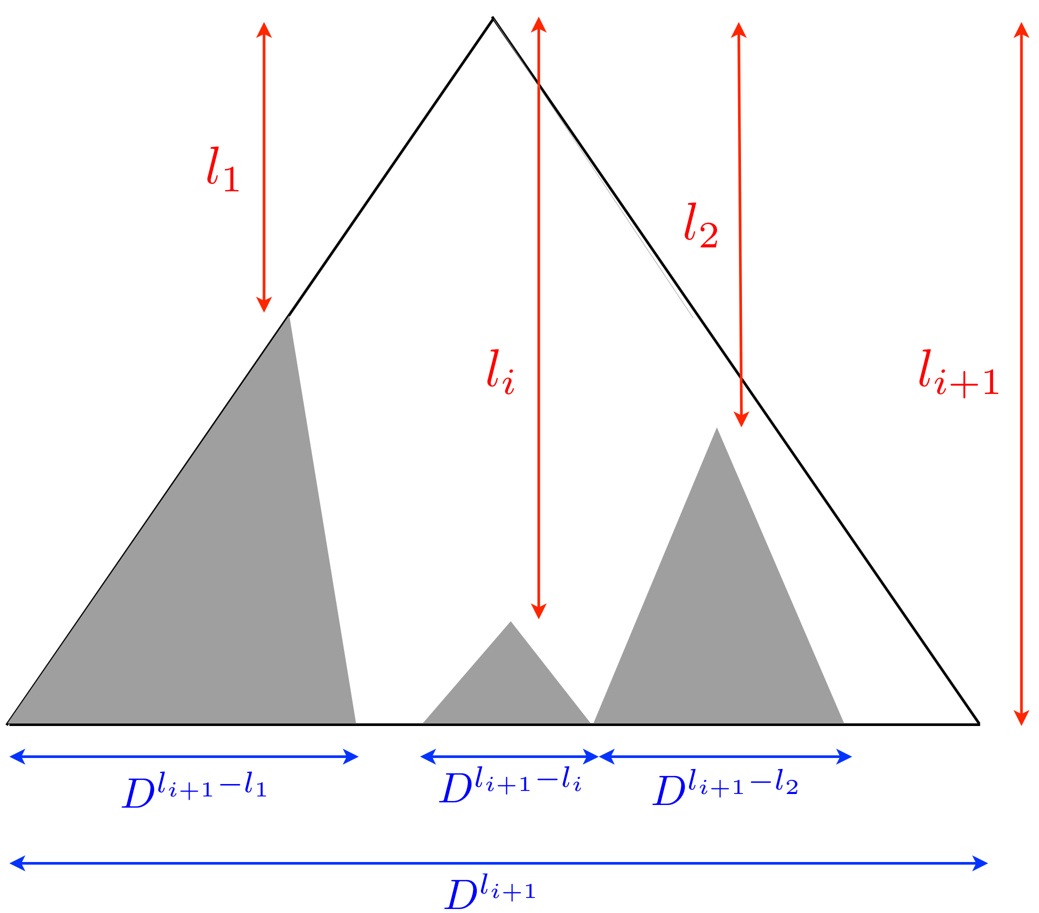
$$D^{l_{i+1}} - D^{l_{i+1}-l_1} - D^{l_{i+1}-l_2} - \cdots - D^{l_{i+1}-l_i}.$$

6. If $l_1, l_2, \cdots, l_m$ satisfy the Kraft inequality, we have

$$D^{-l_1} + \cdots + D^{-l_i} + D^{-l_{i+1}} \leq 1.$$

7. Multiplying by $D^{l_{i+1}}$, we have

$$D^{l_{i+1}-l_1} + \cdots + D^{l_{i+1}-l_i} + D^{l_{i+1}-l_{i+1}} \leq D^{l_{i+1}}$$



Theorem 4.11 There exists a D-ary prefix code with codeword lengths $l_1, l_2, \cdots, l_m$ if and only if the Kraft inequality

$$\sum_{k=1}^{m} D^{-l_k} \leq 1$$

is satisfied.

Proof (Converse)

1. We need to prove the existence of a D-ary prefix code with codeword lengths $l_1, l_2, \cdots, l_m$ if these lengths satisfy the Kraft inequality. Without loss of generality, assume that

$$l_1 \leq l_2 \leq \cdots \leq l_m.$$

2. Consider all the D-ary sequences of lengths less than or equal to l_m and regard them as the nodes of the full D-tree of depth l_m. We will refer to a sequence of length l as a node of order l.

3. There are $D^{l_1} > 1$ (since $l_1 \geq 1$) nodes of order l_1 which can be chosen as the first codeword. Thus choosing the first codeword is always possible.

4. Assume that the first i codewords have been chosen successfully, where $1 \leq i \leq m-1$, and we want to choose a node of order l_{i+1} as the $(i+1)$st codeword such that it is not prefixed by any of the previously chosen codewords.

5. Since all the previously chosen codewords are not prefixes of each other, their descendants of order l_{i+1} do not overlap. The $(i+1)$st node to be chosen cannot be a descendant of any of the previously chosen codewords. Therefore, the number of nodes which can be chosen as the $(i+1)$st codeword is

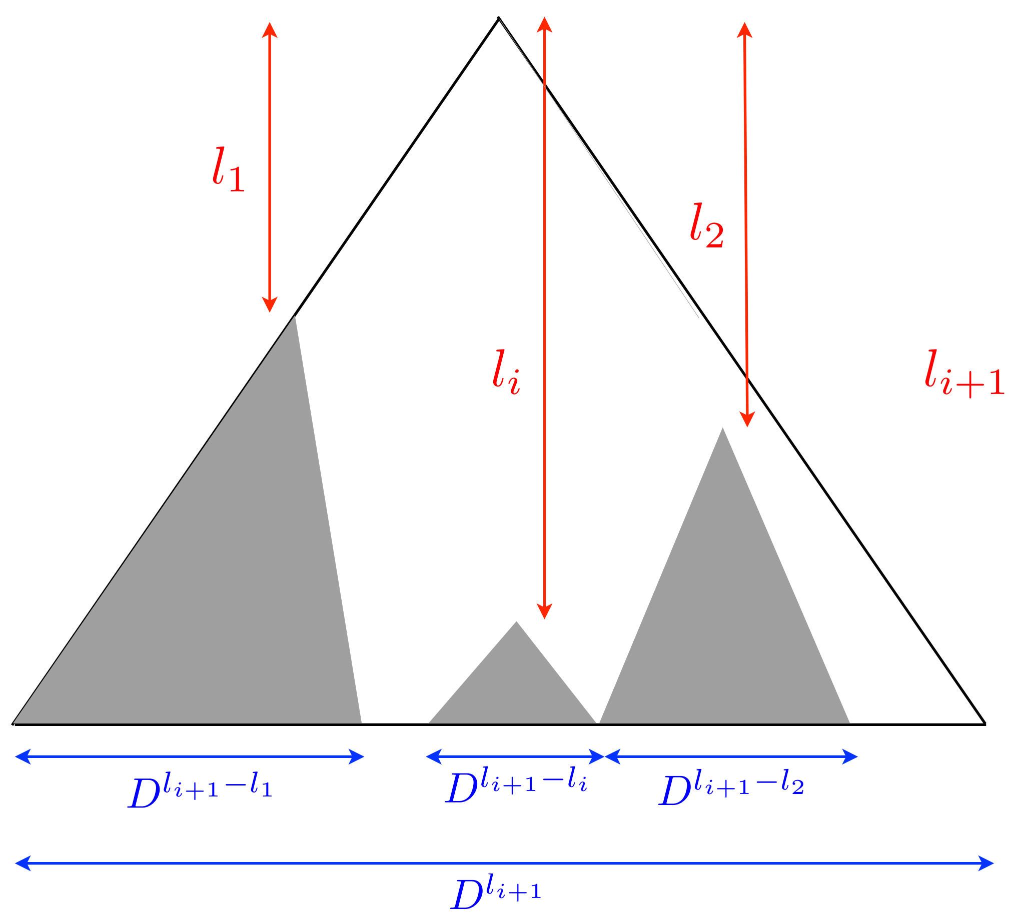
$$D^{l_{i+1}} - D^{l_{i+1}-l_1} - D^{l_{i+1}-l_2} - \cdots - D^{l_{i+1}-l_i}.$$

6. If $l_1, l_2, \cdots, l_m$ satisfy the Kraft inequality, we have

$$D^{-l_1} + \cdots + D^{-l_i} + D^{-l_{i+1}} \leq 1.$$

7. Multiplying by $D^{l_{i+1}}$, we have

$$D^{l_{i+1}-l_1} + \cdots + D^{l_{i+1}-l_i} + D^{l_{i+1}-l_{i+1}} \leq D^{l_{i+1}}$$



Theorem 4.11 There exists a D-ary prefix code with codeword lengths $l_1, l_2, \cdots, l_m$ if and only if the Kraft inequality

$$\sum_{k=1}^{m} D^{-l_k} \leq 1$$

is satisfied.

Proof (Converse)

1. We need to prove the existence of a D-ary prefix code with codeword lengths $l_1, l_2, \cdots, l_m$ if these lengths satisfy the Kraft inequality. Without loss of generality, assume that

$$l_1 \leq l_2 \leq \cdots \leq l_m.$$

2. Consider all the D-ary sequences of lengths less than or equal to l_m and regard them as the nodes of the full D-ary tree of depth l_m. We will refer to a sequence of length l as a node of order l.

3. There are $D^{l_1} > 1$ (since $l_1 \geq 1$) nodes of order l_1 which can be chosen as the first codeword. Thus choosing the first codeword is always possible.

4. Assume that the first i codewords have been chosen successfully, where $1 \leq i \leq m-1$, and we want to choose a node of order l_{i+1} as the $(i+1)$st codeword such that it is not prefixed by any of the previously chosen codewords.

5. Since all the previously chosen codewords are not prefixes of each other, their descendants of order l_{i+1} do not overlap. The $(i+1)$st node to be chosen cannot be a descendant of any of the previously chosen codewords. Therefore, the number of nodes which can be chosen as the $(i+1)$st codeword is

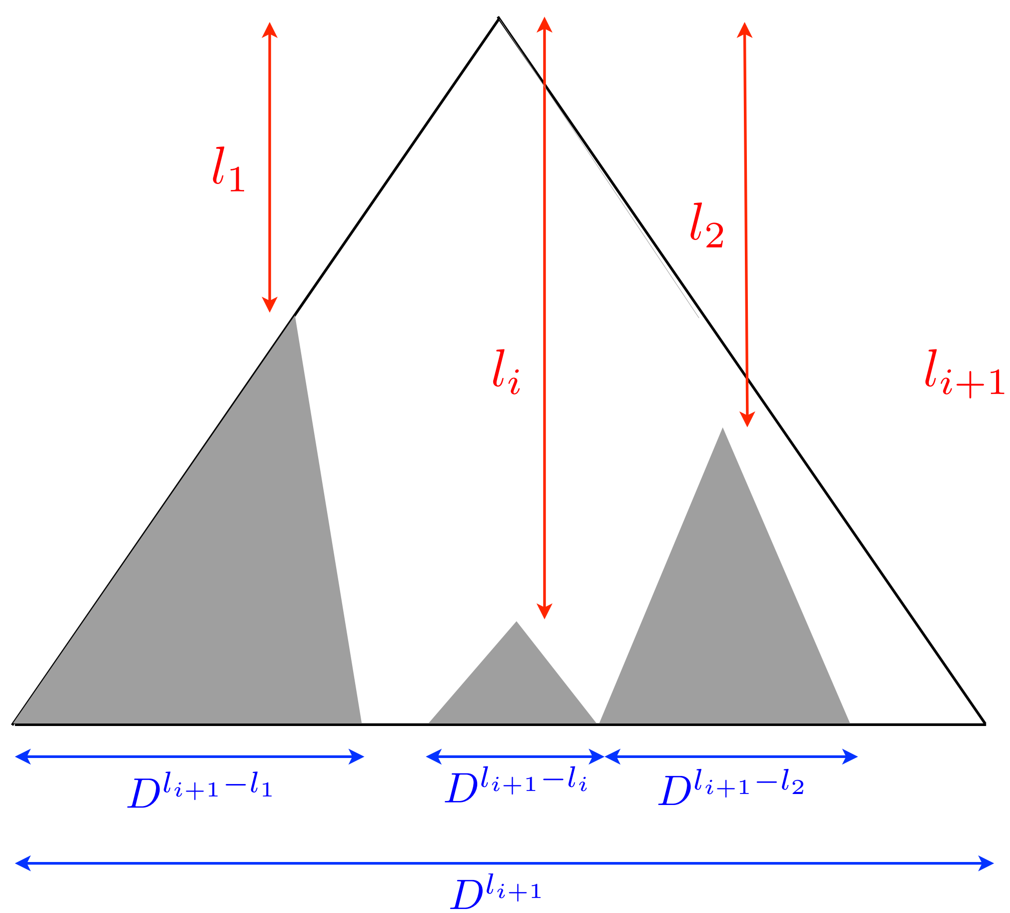
$$D^{l_{i+1}} - D^{l_{i+1}-l_1} - D^{l_{i+1}-l_2} - \cdots - D^{l_{i+1}-l_i}.$$

6. If $l_1, l_2, \cdots, l_m$ satisfy the Kraft inequality, we have

$$D^{-l_1} + \cdots + D^{-l_i} + \underline{D^{-l_{i+1}}} \leq 1.$$

7. Multiplying by $D^{l_{i+1}}$, we have

$$D^{l_{i+1}-l_1} + \cdots + D^{l_{i+1}-l_i} + \underline{D^{l_{i+1}-l_{i+1}}} \leq D^{l_{i+1}}$$



Theorem 4.11 There exists a D-ary prefix code with codeword lengths $l_1,\ l_2, \cdots, l_m$ if and only if the Kraft inequality

$$\sum_{k=1}^{m} D^{-l_k} \leq 1$$

is satisfied.

Proof (Converse)

1. We need to prove the existence of a D-ary prefix code with codeword lengths $l_1, l_2, \cdots, l_m$ if these lengths satisfy the Kraft inequality. Without loss of generality, assume that

$$l_1 \leq l_2 \leq \cdots \leq l_m.$$

2. Consider all the D-ary sequences of lengths less than or equal to l_m and regard them as the nodes of the full D-ary tree of depth l_m. We will refer to a sequence of length l as a node of order l.

3. There are $D^{l_1} > 1$ (since $l_1 \geq 1$) nodes of order l_1 which can be chosen as the first codeword. Thus choosing the first codeword is always possible.

4. Assume that the first i codewords have been chosen successfully, where $1 \leq i \leq m-1$, and we want to choose a node of order l_{i+1} as the $(i+1)$st codeword such that it is not prefixed by any of the previously chosen codewords.

5. Since all the previously chosen codewords are not prefixes of each other, their descendants of order l_{i+1} do not overlap. The $(i+1)$st node to be chosen cannot be a descendant of any of the previously chosen codewords. Therefore, the number of nodes which can be chosen as the $(i+1)$st codeword is

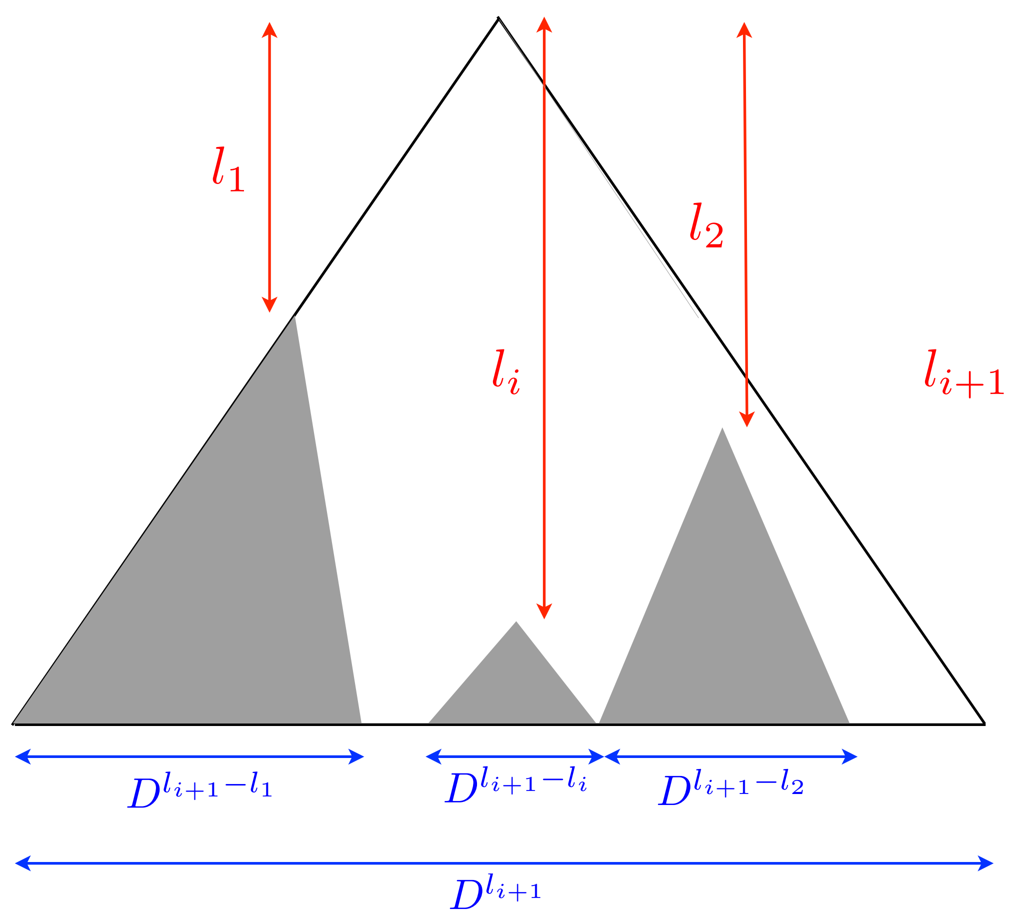
$$D^{l_{i+1}} - D^{l_{i+1}-l_1} - D^{l_{i+1}-l_2} - \cdots - D^{l_{i+1}-l_i}.$$

6. If $l_1, l_2, \cdots, l_m$ satisfy the Kraft inequality, we have

$$D^{-l_1} + \cdots + D^{-l_i} + D^{-l_{i+1}} \leq 1.$$

7. Multiplying by $D^{l_{i+1}}$, we have

$$D^{l_{i+1}-l_1} + \cdots + D^{l_{i+1}-l_i} + D^{l_{i+1}-l_{i+1}} \leq D^{l_{i+1}}$$



Theorem 4.11 There exists a D-ary prefix code with codeword lengths $l_1, l_2, \cdots, l_m$ if and only if the Kraft inequality

$$\sum_{k=1}^{m} D^{-l_k} \leq 1$$

is satisfied.

Proof (Converse)

1. We need to prove the existence of a D-ary prefix code with codeword lengths $l_1, l_2, \cdots, l_m$ if these lengths satisfy the Kraft inequality. Without loss of generality, assume that

$$l_1 \leq l_2 \leq \cdots \leq l_m.$$

2. Consider all the D-ary sequences of lengths less than or equal to l_m and regard them as the nodes of the full D-ary tree of depth l_m. We will refer to a sequence of length l as a node of order l.

3. There are $D^{l_1} > 1$ (since $l_1 \geq 1$) nodes of order l_1 which can be chosen as the first codeword. Thus choosing the first codeword is always possible.

4. Assume that the first i codewords have been chosen successfully, where $1 \leq i \leq m-1$, and we want to choose a node of order l_{i+1} as the $(i+1)$st codeword such that it is not prefixed by any of the previously chosen codewords.

5. Since all the previously chosen codewords are not prefixes of each other, their descendants of order l_{i+1} do not overlap. The $(i+1)$st node to be chosen cannot be a descendant of any of the previously chosen codewords. Therefore, the number of nodes which can be chosen as the $(i+1)$st codeword is

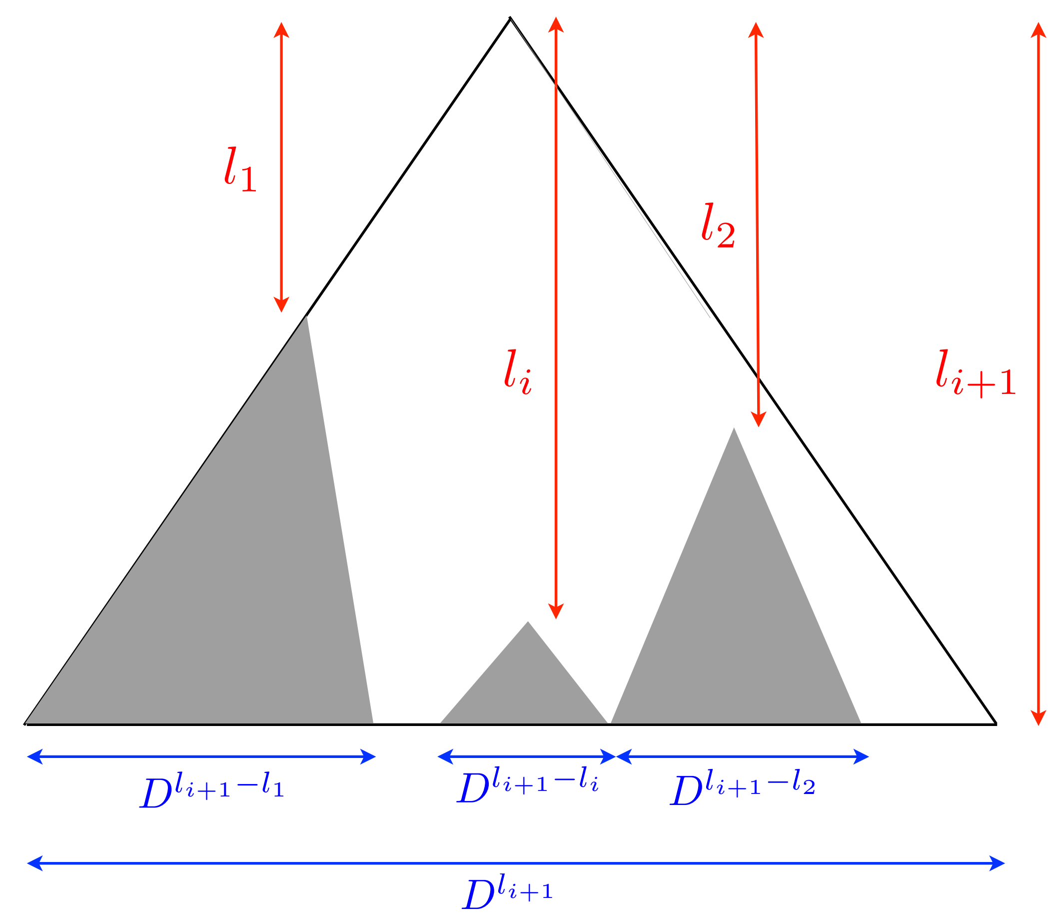
$$D^{l_{i+1}} - D^{l_{i+1}-l_1} - D^{l_{i+1}-l_2} - \cdots - D^{l_{i+1}-l_i}.$$

6. If $l_1, l_2, \cdots, l_m$ satisfy the Kraft inequality, we have

$$D^{-l_1} + \cdots + D^{-l_i} + D^{-l_{i+1}} \leq 1.$$

7. Multiplying by $D^{l_{i+1}}$, we have

$$D^{l_{i+1}-l_1} + \cdots + D^{l_{i+1}-l_i} + D^{l_{i+1}-l_{i+1}} \leq D^{l_{i+1}}$$



Theorem 4.11 There exists a D-ary prefix code with codeword lengths $l_1, l_2, \cdots, l_m$ if and only if the Kraft inequality

$$\sum_{k=1}^{m} D^{-l_k} \leq 1$$

is satisfied.

Proof (Converse)

1. We need to prove the existence of a D-ary prefix code with codeword lengths $l_1, l_2, \cdots, l_m$ if these lengths satisfy the Kraft inequality. Without loss of generality, assume that

$$l_1 \leq l_2 \leq \cdots \leq l_m.$$

2. Consider all the D-ary sequences of lengths less than or equal to l_m and regard them as the nodes of the full D-ary tree of depth l_m. We will refer to a sequence of length l as a node of order l.

3. There are $D^{l_1} > 1$ (since $l_1 \geq 1$) nodes of order l_1 which can be chosen as the first codeword. Thus choosing the first codeword is always possible.

4. Assume that the first i codewords have been chosen successfully, where $1 \leq i \leq m-1$, and we want to choose a node of order l_{i+1} as the $(i+1)$st codeword such that it is not prefixed by any of the previously chosen codewords.

5. Since all the previously chosen codewords are not prefixes of each other, their descendants of order l_{i+1} do not overlap. The $(i+1)$st node to be chosen cannot be a descendant of any of the previously chosen codewords. Therefore, the number of nodes which can be chosen as the $(i+1)$st codeword is

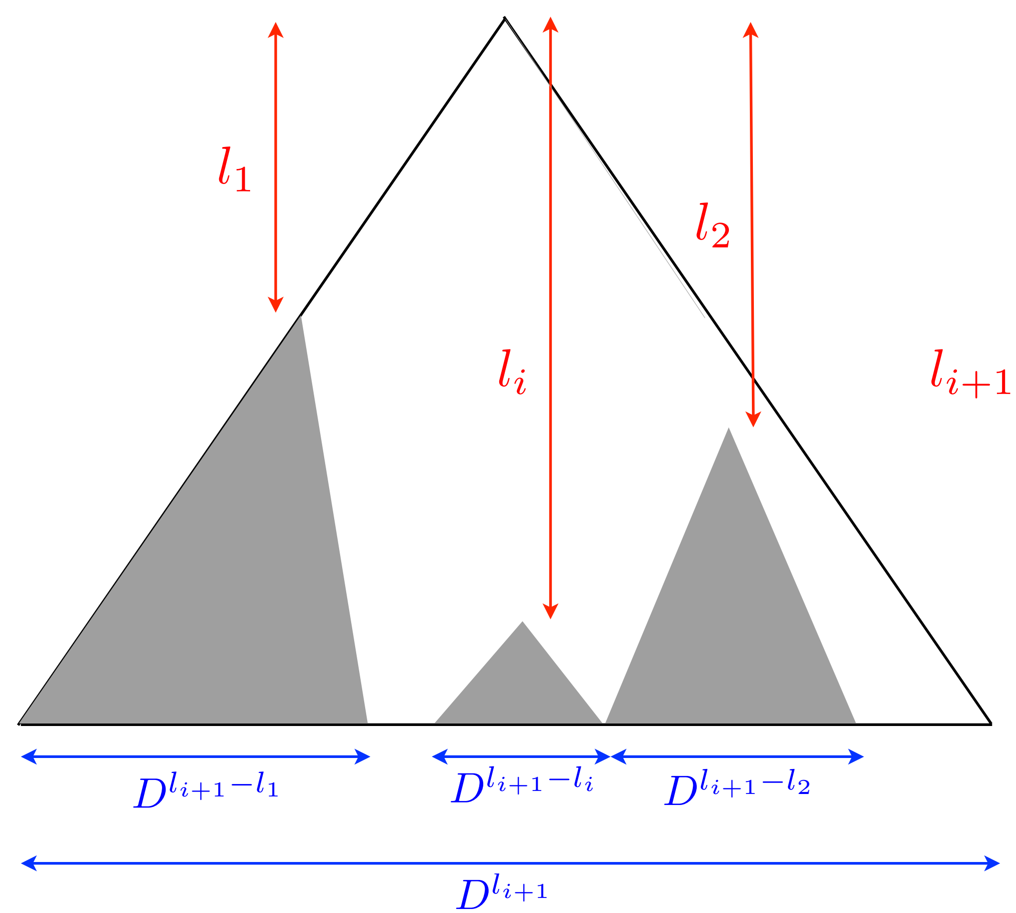
$$D^{l_{i+1}} - D^{l_{i+1}-l_1} - D^{l_{i+1}-l_2} - \cdots - D^{l_{i+1}-l_i}.$$

6. If $l_1, l_2, \cdots, l_m$ satisfy the Kraft inequality, we have

$$D^{-l_1} + \cdots + D^{-l_i} + D^{-l_{i+1}} \leq 1.$$

7. Multiplying by $D^{l_{i+1}}$, we have

$$D^{l_{i+1}-l_1} + \cdots + D^{l_{i+1}-l_i} + \underline{D^{l_{i+1}-l_{i+1}}} \leq D^{l_{i+1}}$$



Theorem 4.11 There exists a D-ary prefix code with codeword lengths $l_1,\ l_2, \cdots, l_m$ if and only if the Kraft inequality

$$\sum_{k=1}^{m} D^{-l_k} \leq 1$$

is satisfied.

Proof (Converse)

1. We need to prove the existence of a D-ary prefix code with codeword lengths $l_1, l_2, \cdots, l_m$ if these lengths satisfy the Kraft inequality. Without loss of generality, assume that

$$l_1 \leq l_2 \leq \cdots \leq l_m.$$

2. Consider all the D-ary sequences of lengths less than or equal to l_m and regard them as the nodes of the full D-tree of depth l_m. We will refer to a sequence of length l as a node of order l.

3. There are $D^{l_1} > 1$ (since $l_1 \geq 1$) nodes of order l_1 which can be chosen as the first codeword. Thus choosing the first codeword is always possible.

4. Assume that the first i codewords have been chosen successfully, where $1 \leq i \leq m-1$, and we want to choose a node of order l_{i+1} as the $(i+1)$st codeword such that it is not prefixed by any of the previously chosen codewords.

5. Since all the previously chosen codewords are not prefixes of each other, their descendants of order l_{i+1} do not overlap. The $(i+1)$st node to be chosen cannot be a descendant of any of the previously chosen codewords. Therefore, the number of nodes which can be chosen as the $(i+1)$st codeword is

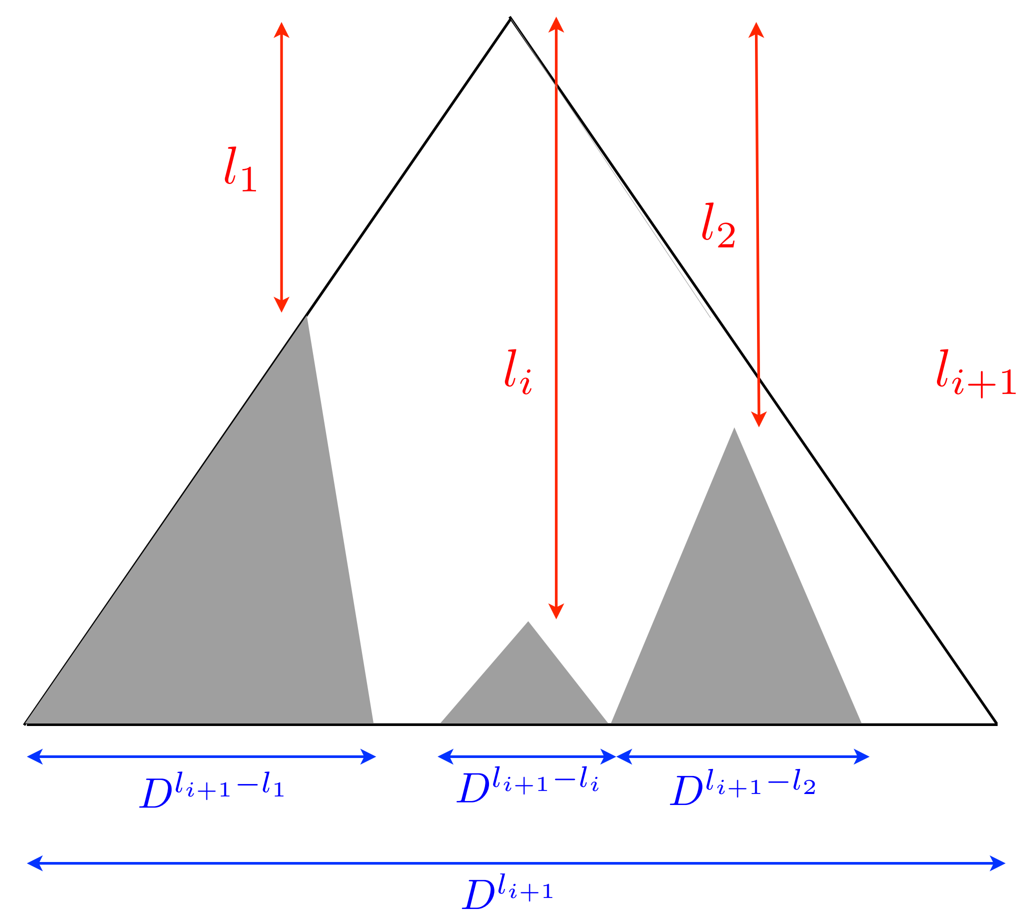
$$D^{l_{i+1}} - D^{l_{i+1}-l_1} - D^{l_{i+1}-l_2} - \cdots - D^{l_{i+1}-l_i}.$$

6. If $l_1, l_2, \cdots, l_m$ satisfy the Kraft inequality, we have

$$D^{-l_1} + \cdots + D^{-l_i} + D^{-l_{i+1}} \leq 1.$$

7. Multiplying by $D^{l_{i+1}}$, we have

$$D^{l_{i+1}-l_1} + \cdots + D^{l_{i+1}-l_i} + \cancelto{1}{D^{l_{i+1}-l_{i+1}}} \leq D^{l_{i+1}}$$



Theorem 4.11 There exists a D-ary prefix code with codeword lengths $l_1, l_2, \cdots, l_m$ if and only if the Kraft inequality

$$\sum_{k=1}^{m} D^{-l_k} \leq 1$$

is satisfied.

Proof (Converse)

1. We need to prove the existence of a D-ary prefix code with codeword lengths $l_1, l_2, \cdots, l_m$ if these lengths satisfy the Kraft inequality. Without loss of generality, assume that

$$l_1 \leq l_2 \leq \cdots \leq l_m.$$

2. Consider all the D-ary sequences of lengths less than or equal to l_m and regard them as the nodes of the full D-ary tree of depth l_m. We will refer to a sequence of length l as a node of order l.

3. There are $D^{l_1} > 1$ (since $l_1 \geq 1$) nodes of order l_1 which can be chosen as the first codeword. Thus choosing the first codeword is always possible.

4. Assume that the first i codewords have been chosen successfully, where $1 \leq i \leq m-1$, and we want to choose a node of order l_{i+1} as the $(i+1)$st codeword such that it is not prefixed by any of the previously chosen codewords.

5. Since all the previously chosen codewords are not prefixes of each other, their descendants of order l_{i+1} do not overlap. The $(i+1)$st node to be chosen cannot be a descendant of any of the previously chosen codewords. Therefore, the number of nodes which can be chosen as the $(i+1)$st codeword is

$$D^{l_{i+1}} - D^{l_{i+1}-l_1} - D^{l_{i+1}-l_2} - \cdots - D^{l_{i+1}-l_i}.$$

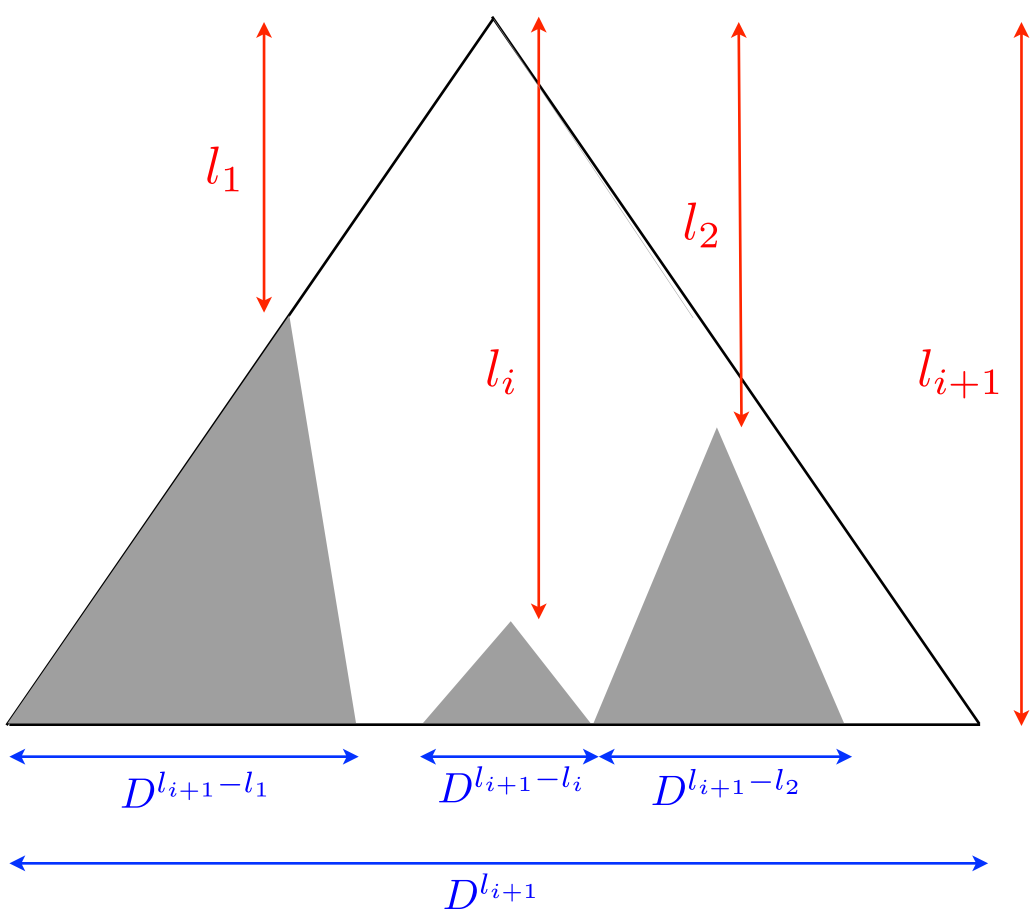
6. If $l_1, l_2, \cdots, l_m$ satisfy the Kraft inequality, we have

$$D^{-l_1} + \cdots + D^{-l_i} + D^{-l_{i+1}} \leq 1.$$

7. Multiplying by $D^{l_{i+1}}$, we have

$$D^{l_{i+1}-l_1} + \cdots + D^{l_{i+1}-l_i} + D^{l_{i+1}-l_{i+1}} \leq D^{l_{i+1}}$$

1



Theorem 4.11 There exists a D-ary prefix code with codeword lengths $l_1, l_2, \cdots, l_m$ if and only if the Kraft inequality

$$\sum_{k=1}^{m} D^{-l_k} \leq 1$$

is satisfied.

Proof (Converse)

1. We need to prove the existence of a D-ary prefix code with codeword lengths $l_1, l_2, \cdots, l_m$ if these lengths satisfy the Kraft inequality. Without loss of generality, assume that

$$l_1 \leq l_2 \leq \cdots \leq l_m.$$

2. Consider all the D-ary sequences of lengths less than or equal to l_m and regard them as the nodes of the full D-ary tree of depth l_m. We will refer to a sequence of length l as a node of order l.

3. There are $D^{l_1} > 1$ (since $l_1 \geq 1$) nodes of order l_1 which can be chosen as the first codeword. Thus choosing the first codeword is always possible.

4. Assume that the first i codewords have been chosen successfully, where $1 \leq i \leq m-1$, and we want to choose a node of order l_{i+1} as the $(i+1)$st codeword such that it is not prefixed by any of the previously chosen codewords.

5. Since all the previously chosen codewords are not prefixes of each other, their descendants of order l_{i+1} do not overlap. The $(i+1)$st node to be chosen cannot be a descendant of any of the previously chosen codewords. Therefore, the number of nodes which can be chosen as the $(i+1)$st codeword is

$$D^{l_{i+1}} - D^{l_{i+1}-l_1} - D^{l_{i+1}-l_2} - \cdots - D^{l_{i+1}-l_i}.$$

6. If $l_1, l_2, \cdots, l_m$ satisfy the Kraft inequality, we have

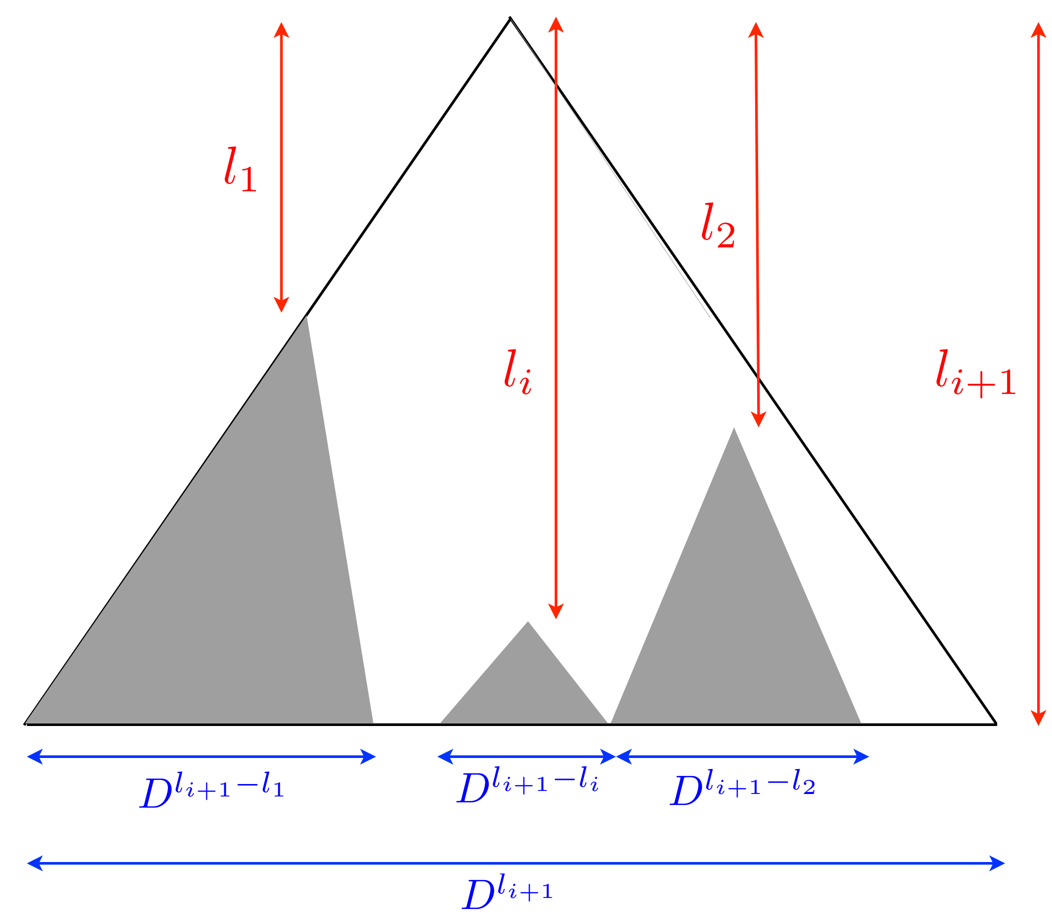
$$D^{-l_1} + \cdots + D^{-l_i} + D^{-l_{i+1}} \leq 1.$$

7. Multiplying by $D^{l_{i+1}}$, we have

$$D^{l_{i+1}-l_1} + \cdots + D^{l_{i+1}-l_i} + \overset{1}{D^{l_{i+1}-l_{i+1}}} \leq D^{l_{i+1}}$$

or

$$D^{l_{i+1}} - D^{l_{i+1}-l_1} - \cdots - D^{l_{i+1}-l_i} \geq 1.$$



Theorem 4.11 There exists a D-ary prefix code with codeword lengths $l_1, l_2, \cdots, l_m$ if and only if the Kraft inequality

$$\sum_{k=1}^{m} D^{-l_k} \leq 1$$

is satisfied.

Proof (Converse)

1. We need to prove the existence of a D-ary prefix code with codeword lengths $l_1, l_2, \cdots, l_m$ if these lengths satisfy the Kraft inequality. Without loss of generality, assume that

$$l_1 \leq l_2 \leq \cdots \leq l_m.$$

2. Consider all the D-ary sequences of lengths less than or equal to l_m and regard them as the nodes of the full D-ary tree of depth l_m. We will refer to a sequence of length l as a node of order l.

3. There are $D^{l_1} > 1$ (since $l_1 \geq 1$) nodes of order l_1 which can be chosen as the first codeword. Thus choosing the first codeword is always possible.

4. Assume that the first i codewords have been chosen successfully, where $1 \leq i \leq m-1$, and we want to choose a node of order l_{i+1} as the $(i+1)$st codeword such that it is not prefixed by any of the previously chosen codewords.

5. Since all the previously chosen codewords are not prefixes of each other, their descendants of order l_{i+1} do not overlap. The $(i+1)$st node to be chosen cannot be a descendant of any of the previously chosen codewords. Therefore, the number of nodes which can be chosen as the $(i+1)$st codeword is

$$D^{l_{i+1}} - D^{l_{i+1}-l_1} - D^{l_{i+1}-l_2} - \cdots - D^{l_{i+1}-l_i}.$$

6. If $l_1, l_2, \cdots, l_m$ satisfy the Kraft inequality, we have

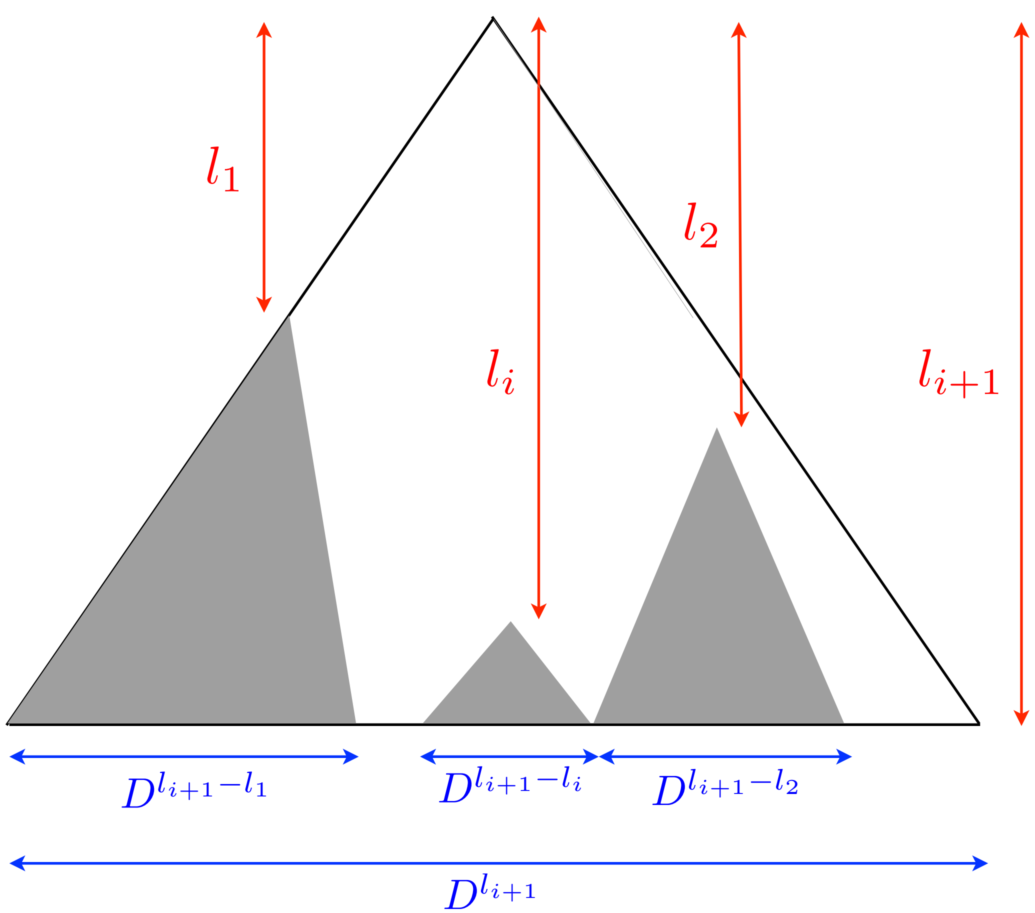
$$D^{-l_1} + \cdots + D^{-l_i} + D^{-l_{i+1}} \leq 1.$$

7. Multiplying by $D^{l_{i+1}}$, we have

$$D^{l_{i+1}-l_1} + \cdots + D^{l_{i+1}-l_i} + D^{l_{i+1}-l_{i+1}} \leq D^{l_{i+1}}$$

or

$$\underline{D^{l_{i+1}} - D^{l_{i+1}-l_1} - \cdots - D^{l_{i+1}-l_i}} \geq 1.$$



Theorem 4.11 There exists a D-ary prefix code with codeword lengths $l_1, l_2, \cdots, l_m$ if and only if the Kraft inequality

$$\sum_{k=1}^{m} D^{-l_k} \leq 1$$

is satisfied.

Proof (Converse)

1. We need to prove the existence of a D-ary prefix code with codeword lengths $l_1, l_2, \cdots, l_m$ if these lengths satisfy the Kraft inequality. Without loss of generality, assume that

$$l_1 \leq l_2 \leq \cdots \leq l_m.$$

2. Consider all the D-ary sequences of lengths less than or equal to l_m and regard them as the nodes of the full D-ary tree of depth l_m. We will refer to a sequence of length l as a node of order l.

3. There are $D^{l_1} > 1$ (since $l_1 \geq 1$) nodes of order l_1 which can be chosen as the first codeword. Thus choosing the first codeword is always possible.

4. Assume that the first i codewords have been chosen successfully, where $1 \leq i \leq m-1$, and we want to choose a node of order l_{i+1} as the $(i+1)$st codeword such that it is not prefixed by any of the previously chosen codewords.

5. Since all the previously chosen codewords are not prefixes of each other, their descendants of order l_{i+1} do not overlap. The $(i+1)$st node to be chosen cannot be a descendant of any of the previously chosen codewords. Therefore, the number of nodes which can be chosen as the $(i+1)$st codeword is

$$D^{l_{i+1}} - D^{l_{i+1}-l_1} - D^{l_{i+1}-l_2} - \cdots - D^{l_{i+1}-l_i}.$$

6. If $l_1, l_2, \cdots, l_m$ satisfy the Kraft inequality, we have

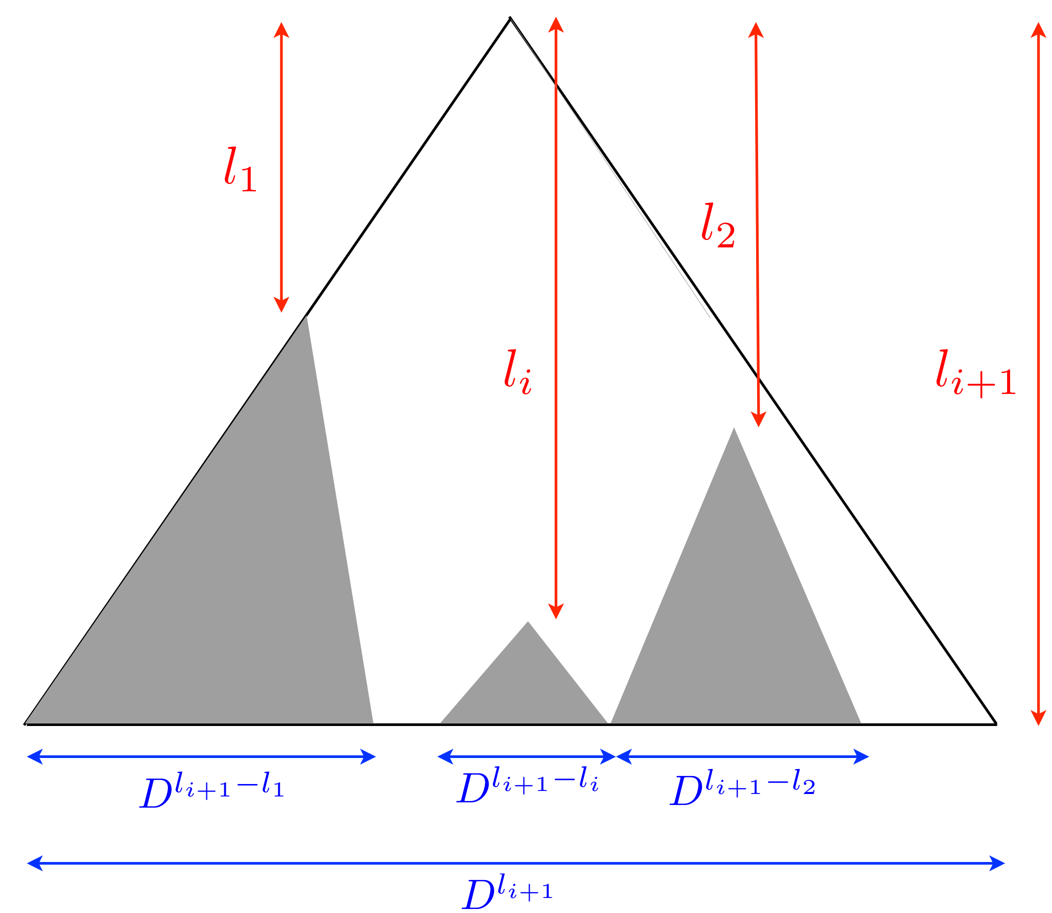
$$D^{-l_1} + \cdots + D^{-l_i} + D^{-l_{i+1}} \leq 1.$$

7. Multiplying by $D^{l_{i+1}}$, we have

$$D^{l_{i+1}-l_1} + \cdots + D^{l_{i+1}-l_i} + D^{l_{i+1}-l_{i+1}} \leq D^{l_{i+1}}$$

or

$$D^{l_{i+1}} - D^{l_{i+1}-l_1} - \cdots - D^{l_{i+1}-l_i} \geq 1.$$



Theorem 4.11 There exists a D-ary prefix code with codeword lengths $l_1,\ l_2, \cdots, l_m$ if and only if the Kraft inequality

$$\sum_{k=1}^{m} D^{-l_k} \leq 1$$

is satisfied.

Proof (Converse)

1. We need to prove the existence of a D-ary prefix code with codeword lengths $l_1, l_2, \cdots, l_m$ if these lengths satisfy the Kraft inequality. Without loss of generality, assume that

$$l_1 \leq l_2 \leq \cdots \leq l_m.$$

2. Consider all the D-ary sequences of lengths less than or equal to l_m and regard them as the nodes of the full D-ary tree of depth l_m. We will refer to a sequence of length l as a node of order l.

3. There are $D^{l_1} > 1$ (since $l_1 \geq 1$) nodes of order l_1 which can be chosen as the first codeword. Thus choosing the first codeword is always possible.

4. Assume that the first i codewords have been chosen successfully, where $1 \leq i \leq m-1$, and we want to choose a node of order l_{i+1} as the $(i+1)$st codeword such that it is not prefixed by any of the previously chosen codewords.

5. Since all the previously chosen codewords are not prefixes of each other, their descendants of order l_{i+1} do not overlap. The $(i+1)$st node to be chosen cannot be a descendant of any of the previously chosen codewords. Therefore, the number of nodes which can be chosen as the $(i+1)$st codeword is

$$D^{l_{i+1}} - D^{l_{i+1}-l_1} - D^{l_{i+1}-l_2} - \cdots - D^{l_{i+1}-l_i}.$$

6. If $l_1, l_2, \cdots, l_m$ satisfy the Kraft inequality, we have

$$D^{-l_1} + \cdots + D^{-l_i} + D^{-l_{i+1}} \leq 1.$$

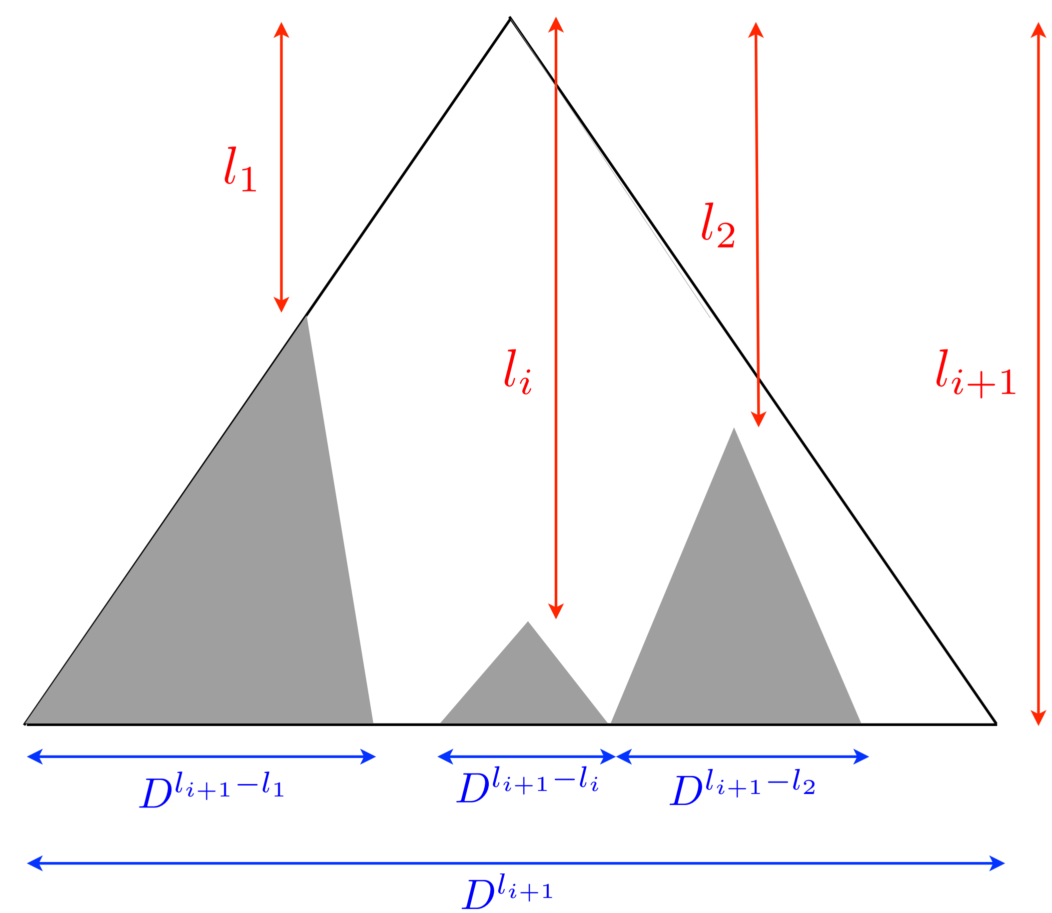
7. Multiplying by $D^{l_{i+1}}$, we have

$$D^{l_{i+1}-l_1} + \cdots + D^{l_{i+1}-l_i} + D^{l_{i+1}-l_{i+1}} \leq D^{l_{i+1}}$$

or

$$D^{l_{i+1}} - D^{l_{i+1}-l_1} - \cdots - D^{l_{i+1}-l_i} \geq 1.$$

Thus we have shown by induction the existence of a prefix code with codeword lengths $l_1, l_2, \cdots, l_m$, completing the proof.



D-adic Distributions

Definition

- $p_i = D^{-t_i}$ for all i, where t_i is integer

D-adic Distributions

Definition

- $p_i = D^{-t_i}$ for all i, where t_i is integer
- *dyadic* when $D = 2$

D-adic Distributions

Definition

- $p_i = D^{-t_i}$ for all i, where t_i is integer
- *dyadic* when $D = 2$

Corollary 4.12 There exists a D-ary prefix code which achieves the entropy bound for a distribution $\{p_i\}$ if and only if $\{p_i\}$ is D-adic.

Corollary 4.12 There exists a D-ary prefix code which achieves the entropy bound for a distribution $\{p_i\}$ if and only if $\{p_i\}$ is D-adic.

Proof

Corollary 4.12 There exists a D-ary prefix code which achieves the entropy bound for a distribution $\{p_i\}$ if and only if $\{p_i\}$ is D-adic.

Proof

A. 'Only if'

Corollary 4.12 There exists a D-ary prefix code which achieves the entropy bound for a distribution $\{p_i\}$ if and only if $\{p_i\}$ is D-adic.

Proof

A. 'Only if'

1. Consider a D-ary prefix code which achieves the entropy bound for a distribution $\{p_i\}$.

Corollary 4.12 There exists a D-ary prefix code which achieves the entropy bound for a distribution $\{p_i\}$ if and only if $\{p_i\}$ is D-adic.

Proof

A. 'Only if'

1. Consider a D-ary prefix code which achieves the entropy bound for a distribution $\{p_i\}$.

2. Let l_i be the length of the codeword assigned to the probability p_i. By Theorem 4.6, for all i,

Corollary 4.12 There exists a D-ary prefix code which achieves the entropy bound for a distribution $\{p_i\}$ if and only if $\{p_i\}$ is D-adic.

Proof

A. 'Only if'

1. Consider a D-ary prefix code which achieves the entropy bound for a distribution $\{p_i\}$.

2. Let l_i be the length of the codeword assigned to the probability p_i. By Theorem 4.6, for all i,

Theorem 4.6 (Entropy Bound)

$$L \geq H_D(X)$$

with equality if and only if $l_i = -\log_D p_i$ for all i.

Corollary 4.12 There exists a D-ary prefix code which achieves the entropy bound for a distribution $\{p_i\}$ if and only if $\{p_i\}$ is D-adic.

Proof

A. 'Only if'

1. Consider a D-ary prefix code which achieves the entropy bound for a distribution $\{p_i\}$.

2. Let l_i be the length of the codeword assigned to the probability p_i. By Theorem 4.6, for all i,

$$l_i = -\log_D p_i,$$

Theorem 4.6 (Entropy Bound)

$$L \geq H_D(X)$$

with equality if and only if $l_i = -\log_D p_i$ for all i.

Corollary 4.12 There exists a D-ary prefix code which achieves the entropy bound for a distribution $\{p_i\}$ if and only if $\{p_i\}$ is D-adic.

Proof

A. 'Only if'

1. Consider a D-ary prefix code which achieves the entropy bound for a distribution $\{p_i\}$.

2. Let l_i be the length of the codeword assigned to the probability p_i. By Theorem 4.6, for all i,

$$l_i = -\log_D p_i,$$

or

$$p_i = D^{-l_i}.$$

> **Theorem 4.6 (Entropy Bound)**
>
> $$L \geq H_D(X)$$
>
> with equality if and only if $l_i = -\log_D p_i$ for all i.

Corollary 4.12 There exists a D-ary prefix code which achieves the entropy bound for a distribution $\{p_i\}$ if and only if $\{p_i\}$ is D-adic.

Proof

A. 'Only if'

1. Consider a D-ary prefix code which achieves the entropy bound for a distribution $\{p_i\}$.

2. Let l_i be the length of the codeword assigned to the probability p_i. By Theorem 4.6, for all i,

$$l_i = -\log_D p_i,$$

or

$$p_i = D^{-l_i}.$$

Thus $\{p_i\}$ is D-adic.

Theorem 4.6 (Entropy Bound)

$$L \geq H_D(X)$$

with equality if and only if $l_i = -\log_D p_i$ for all i.

Corollary 4.12 There exists a D-ary prefix code which achieves the entropy bound for a distribution $\{p_i\}$ if and only if $\{p_i\}$ is D-adic.

Proof

A. 'Only if'

1. Consider a D-ary prefix code which achieves the entropy bound for a distribution $\{p_i\}$.

2. Let l_i be the length of the codeword assigned to the probability p_i. By Theorem 4.6, for all i,

$$l_i = -\log_D p_i,$$

or

$$p_i = D^{-l_i}.$$

Thus $\{p_i\}$ is D-adic.

B. 'If'

Theorem 4.6 (Entropy Bound)

$$L \geq H_D(X)$$

with equality if and only if $l_i = -\log_D p_i$ for all i.

Corollary 4.12 There exists a D-ary prefix code which achieves the entropy bound for a distribution $\{p_i\}$ if and only if $\{p_i\}$ is D-adic.

Proof

A. 'Only if'

1. Consider a D-ary prefix code which achieves the entropy bound for a distribution $\{p_i\}$.

2. Let l_i be the length of the codeword assigned to the probability p_i. By Theorem 4.6, for all i,

$$l_i = -\log_D p_i,$$

or

$$p_i = D^{-l_i}.$$

Thus $\{p_i\}$ is D-adic.

B. 'If'

1. Suppose $\{p_i\}$ is D-adic. Let $p_i = D^{-t_i}$ for all i, where t_i is an integer. Then

Theorem 4.6 (Entropy Bound)

$$L \geq H_D(X)$$

with equality if and only if $l_i = -\log_D p_i$ for all i.

Corollary 4.12 There exists a D-ary prefix code which achieves the entropy bound for a distribution $\{p_i\}$ if and only if $\{p_i\}$ is D-adic.

Proof

A. 'Only if'

1. Consider a D-ary prefix code which achieves the entropy bound for a distribution $\{p_i\}$.

2. Let l_i be the length of the codeword assigned to the probability p_i. By Theorem 4.6, for all i,

$$l_i = -\log_D p_i,$$

or

$$p_i = D^{-l_i}.$$

Thus $\{p_i\}$ is D-adic.

B. 'If'

1. Suppose $\{p_i\}$ is D-adic. Let $p_i = D^{-t_i}$ for all i, where t_i is an integer. Then

$$t_i = -\log_D p_i.$$

Theorem 4.6 (Entropy Bound)

$$L \geq H_D(X)$$

with equality if and only if $l_i = -\log_D p_i$ for all i.

Corollary 4.12 There exists a D-ary prefix code which achieves the entropy bound for a distribution $\{p_i\}$ if and only if $\{p_i\}$ is D-adic.

Proof

A. 'Only if'

1. Consider a D-ary prefix code which achieves the entropy bound for a distribution $\{p_i\}$.

2. Let l_i be the length of the codeword assigned to the probability p_i. By Theorem 4.6, for all i,

$$l_i = -\log_D p_i,$$

or

$$p_i = D^{-l_i}.$$

Thus $\{p_i\}$ is D-adic.

B. 'If'

1. Suppose $\{p_i\}$ is D-adic. Let $p_i = D^{-t_i}$ for all i, where t_i is an integer. Then

$$t_i = -\log_D p_i.$$

2. Let $l_i = t_i$ for all i. Verify that $\{l_i\}$ satisfies the Kraft inequality:

Theorem 4.6 (Entropy Bound)

$$L \geq H_D(X)$$

with equality if and only if $l_i = -\log_D p_i$ for all i.

Corollary 4.12 There exists a D-ary prefix code which achieves the entropy bound for a distribution $\{p_i\}$ if and only if $\{p_i\}$ is D-adic.

Proof

A. 'Only if'

1. Consider a D-ary prefix code which achieves the entropy bound for a distribution $\{p_i\}$.

2. Let l_i be the length of the codeword assigned to the probability p_i. By Theorem 4.6, for all i,

$$l_i = -\log_D p_i,$$

or

$$p_i = D^{-l_i}.$$

Thus $\{p_i\}$ is D-adic.

B. 'If'

1. Suppose $\{p_i\}$ is D-adic. Let $p_i = D^{-t_i}$ for all i, where t_i is an integer. Then

$$t_i = -\log_D p_i.$$

2. Let $l_i = t_i$ for all i. Verify that $\{l_i\}$ satisfies the Kraft inequality:

$$\sum_i D^{-l_i} = \sum_i D^{-t_i} = \sum_i p_i = 1 \leq 1.$$

Theorem 4.6 (Entropy Bound)

$$L \geq H_D(X)$$

with equality if and only if $l_i = -\log_D p_i$ for all i.

Corollary 4.12 There exists a D-ary prefix code which achieves the entropy bound for a distribution $\{p_i\}$ if and only if $\{p_i\}$ is D-adic.

Proof

A. 'Only if'

1. Consider a D-ary prefix code which achieves the entropy bound for a distribution $\{p_i\}$.

2. Let l_i be the length of the codeword assigned to the probability p_i. By Theorem 4.6, for all i,

$$l_i = -\log_D p_i,$$

or

$$p_i = D^{-l_i}.$$

Thus $\{p_i\}$ is D-adic.

B. 'If'

1. Suppose $\{p_i\}$ is D-adic. Let $p_i = D^{-t_i}$ for all i, where t_i is an integer. Then

$$t_i = -\log_D p_i.$$

2. Let $l_i = t_i$ for all i. Verify that $\{l_i\}$ satisfies the Kraft inequality:

$$\sum_i D^{-l_i} = \sum_i D^{-t_i} = \sum_i p_i = 1 \le 1.$$

Theorem 4.6 (Entropy Bound)

$$L \ge H_D(X)$$

with equality if and only if $l_i = -\log_D p_i$ for all i.

Corollary 4.12 There exists a D-ary prefix code which achieves the entropy bound for a distribution $\{p_i\}$ if and only if $\{p_i\}$ is D-adic.

Proof

A. 'Only if'

1. Consider a D-ary prefix code which achieves the entropy bound for a distribution $\{p_i\}$.

2. Let l_i be the length of the codeword assigned to the probability p_i. By Theorem 4.6, for all i,

$$l_i = -\log_D p_i,$$

or

$$p_i = D^{-l_i}.$$

Thus $\{p_i\}$ is D-adic.

B. 'If'

1. Suppose $\{p_i\}$ is D-adic. Let $p_i = D^{-t_i}$ for all i, where t_i is an integer. Then

$$t_i = -\log_D p_i.$$

2. Let $l_i = t_i$ for all i. Verify that $\{l_i\}$ satisfies the Kraft inequality:

$$\sum_i D^{-l_i} = \sum_i D^{-t_i} = \sum_i p_i = 1 \leq 1.$$

Theorem 4.6 (Entropy Bound)

$$L \geq H_D(X)$$

with equality if and only if $l_i = -\log_D p_i$ for all i.

Corollary 4.12 There exists a D-ary prefix code which achieves the entropy bound for a distribution $\{p_i\}$ if and only if $\{p_i\}$ is D-adic.

Proof

A. 'Only if'

1. Consider a D-ary prefix code which achieves the entropy bound for a distribution $\{p_i\}$.

2. Let l_i be the length of the codeword assigned to the probability p_i. By Theorem 4.6, for all i,

$$l_i = -\log_D p_i,$$

or

$$p_i = D^{-l_i}.$$

Thus $\{p_i\}$ is D-adic.

B. 'If'

1. Suppose $\{p_i\}$ is D-adic. Let $p_i = D^{-t_i}$ for all i, where t_i is an integer. Then

$$t_i = -\log_D p_i.$$

2. Let $l_i = t_i$ for all i. Verify that $\{l_i\}$ satisfies the Kraft inequality:

$$\sum_i D^{-l_i} = \sum_i \underline{D^{-t_i}} = \sum_i p_i = 1 \leq 1.$$

Theorem 4.6 (Entropy Bound)

$$L \geq H_D(X)$$

with equality if and only if $l_i = -\log_D p_i$ for all i.

Corollary 4.12 There exists a D-ary prefix code which achieves the entropy bound for a distribution $\{p_i\}$ if and only if $\{p_i\}$ is D-adic.

Proof

A. 'Only if'

1. Consider a D-ary prefix code which achieves the entropy bound for a distribution $\{p_i\}$.

2. Let l_i be the length of the codeword assigned to the probability p_i. By Theorem 4.6, for all i,

$$l_i = -\log_D p_i,$$

or

$$p_i = D^{-l_i}.$$

Thus $\{p_i\}$ is D-adic.

B. 'If'

1. Suppose $\{p_i\}$ is D-adic. Let $p_i = D^{-t_i}$ for all i, where t_i is an integer. Then

$$t_i = -\log_D p_i.$$

2. Let $l_i = t_i$ for all i. Verify that $\{l_i\}$ satisfies the Kraft inequality:

$$\sum_i D^{-l_i} = \sum_i D^{-t_i} = \sum_i p_i = 1 \leq 1.$$

Theorem 4.6 (Entropy Bound)

$$L \geq H_D(X)$$

with equality if and only if $l_i = -\log_D p_i$ for all i.

Corollary 4.12 There exists a D-ary prefix code which achieves the entropy bound for a distribution $\{p_i\}$ if and only if $\{p_i\}$ is D-adic.

Proof

A. 'Only if'

1. Consider a D-ary prefix code which achieves the entropy bound for a distribution $\{p_i\}$.

2. Let l_i be the length of the codeword assigned to the probability p_i. By Theorem 4.6, for all i,

$$l_i = -\log_D p_i,$$

or

$$p_i = D^{-l_i}.$$

Thus $\{p_i\}$ is D-adic.

B. 'If'

1. Suppose $\{p_i\}$ is D-adic. Let $p_i = D^{-t_i}$ for all i, where t_i is an integer. Then

$$t_i = -\log_D p_i.$$

2. Let $l_i = t_i$ for all i. Verify that $\{l_i\}$ satisfies the Kraft inequality:

$$\sum_i D^{-l_i} = \sum_i D^{-t_i} = \sum_i p_i = 1 \le 1.$$

Theorem 4.6 (Entropy Bound)

$$L \ge H_D(X)$$

with equality if and only if $l_i = -\log_D p_i$ for all i.

Corollary 4.12 There exists a D-ary prefix code which achieves the entropy bound for a distribution $\{p_i\}$ if and only if $\{p_i\}$ is D-adic.

Proof

A. 'Only if'

1. Consider a D-ary prefix code which achieves the entropy bound for a distribution $\{p_i\}$.

2. Let l_i be the length of the codeword assigned to the probability p_i. By Theorem 4.6, for all i,

$$l_i = -\log_D p_i,$$

or

$$p_i = D^{-l_i}.$$

Thus $\{p_i\}$ is D-adic.

B. 'If'

1. Suppose $\{p_i\}$ is D-adic. Let $p_i = D^{-t_i}$ for all i, where t_i is an integer. Then

$$t_i = -\log_D p_i.$$

2. Let $l_i = t_i$ for all i. Verify that $\{l_i\}$ satisfies the Kraft inequality:

$$\sum_i D^{-l_i} = \sum_i D^{-t_i} = \sum_i p_i = 1 \leq 1.$$

3. Then there exists a prefix code with codeword lengths $\{l_i\}$. Assign the codeword with length l_i to the probability p_i for all i.

Theorem 4.6 (Entropy Bound)

$$L \geq H_D(X)$$

with equality if and only if $l_i = -\log_D p_i$ for all i.

Corollary 4.12 There exists a D-ary prefix code which achieves the entropy bound for a distribution $\{p_i\}$ if and only if $\{p_i\}$ is D-adic.

Proof

A. 'Only if'

1. Consider a D-ary prefix code which achieves the entropy bound for a distribution $\{p_i\}$.

2. Let l_i be the length of the codeword assigned to the probability p_i. By Theorem 4.6, for all i,

$$l_i = -\log_D p_i,$$

or

$$p_i = D^{-l_i}.$$

Thus $\{p_i\}$ is D-adic.

B. 'If'

1. Suppose $\{p_i\}$ is D-adic. Let $p_i = D^{-t_i}$ for all i, where t_i is an integer. Then

$$t_i = -\log_D p_i.$$

2. Let $l_i = t_i$ for all i. Verify that $\{l_i\}$ satisfies the Kraft inequality:

$$\sum_i D^{-l_i} = \sum_i D^{-t_i} = \sum_i p_i = 1 \leq 1.$$

3. Then there exists a prefix code with codeword lengths $\{l_i\}$. Assign the codeword with length l_i to the probability p_i for all i.

4. Since for all i,

Theorem 4.6 (Entropy Bound)

$$L \geq H_D(X)$$

with equality if and only if $l_i = -\log_D p_i$ for all i.

Corollary 4.12 There exists a D-ary prefix code which achieves the entropy bound for a distribution $\{p_i\}$ if and only if $\{p_i\}$ is D-adic.

Proof

A. 'Only if'

1. Consider a D-ary prefix code which achieves the entropy bound for a distribution $\{p_i\}$.

2. Let l_i be the length of the codeword assigned to the probability p_i. By Theorem 4.6, for all i,

$$l_i = -\log_D p_i,$$

or

$$p_i = D^{-l_i}.$$

Thus $\{p_i\}$ is D-adic.

B. 'If'

1. Suppose $\{p_i\}$ is D-adic. Let $p_i = D^{-t_i}$ for all i, where t_i is an integer. Then

$$t_i = -\log_D p_i.$$

2. Let $l_i = t_i$ for all i. Verify that $\{l_i\}$ satisfies the Kraft inequality:

$$\sum_i D^{-l_i} = \sum_i D^{-t_i} = \sum_i p_i = 1 \leq 1.$$

3. Then there exists a prefix code with codeword lengths $\{l_i\}$. Assign the codeword with length l_i to the probability p_i for all i.

4. Since for all i,

$$l_i = t_i = -\log_D p_i,$$

Theorem 4.6 (Entropy Bound)

$$L \geq H_D(X)$$

with equality if and only if $l_i = -\log_D p_i$ for all i.

Corollary 4.12 There exists a D-ary prefix code which achieves the entropy bound for a distribution $\{p_i\}$ if and only if $\{p_i\}$ is D-adic.

Proof

A. 'Only if'

1. Consider a D-ary prefix code which achieves the entropy bound for a distribution $\{p_i\}$.

2. Let l_i be the length of the codeword assigned to the probability p_i. By Theorem 4.6, for all i,

$$l_i = -\log_D p_i,$$

or

$$p_i = D^{-l_i}.$$

Thus $\{p_i\}$ is D-adic.

B. 'If'

1. Suppose $\{p_i\}$ is D-adic. Let $p_i = D^{-t_i}$ for all i, where t_i is an integer. Then

$$t_i = -\log_D p_i.$$

2. Let $l_i = t_i$ for all i. Verify that $\{l_i\}$ satisfies the Kraft inequality:

$$\sum_i D^{-l_i} = \sum_i D^{-t_i} = \sum_i p_i = 1 \leq 1.$$

3. Then there exists a prefix code with codeword lengths $\{l_i\}$. Assign the codeword with length l_i to the probability p_i for all i.

4. Since for all i,

$$l_i = t_i = -\log_D p_i,$$

Theorem 4.6 (Entropy Bound)

$$L \geq H_D(X)$$

with equality if and only if $l_i = -\log_D p_i$ for all i.

Corollary 4.12 There exists a D-ary prefix code which achieves the entropy bound for a distribution $\{p_i\}$ if and only if $\{p_i\}$ is D-adic.

Proof

A. 'Only if'

1. Consider a D-ary prefix code which achieves the entropy bound for a distribution $\{p_i\}$.

2. Let l_i be the length of the codeword assigned to the probability p_i. By Theorem 4.6, for all i,

$$l_i = -\log_D p_i,$$

or

$$p_i = D^{-l_i}.$$

Thus $\{p_i\}$ is D-adic.

B. 'If'

1. Suppose $\{p_i\}$ is D-adic. Let $p_i = D^{-t_i}$ for all i, where t_i is an integer. Then

$$t_i = -\log_D p_i.$$

2. Let $l_i = t_i$ for all i. Verify that $\{l_i\}$ satisfies the Kraft inequality:

$$\sum_i D^{-l_i} = \sum_i D^{-t_i} = \sum_i p_i = 1 \leq 1.$$

3. Then there exists a prefix code with codeword lengths $\{l_i\}$. Assign the codeword with length l_i to the probability p_i for all i.

4. Since for all i,

$$l_i = t_i = -\log_D p_i,$$

we see from Theorem 4.6 that this code achieves the entropy bound.

Theorem 4.6 (Entropy Bound)

$$L \geq H_D(X)$$

with equality if and only if $l_i = -\log_D p_i$ for all i.

Huffman Codes

- A simple construction of optimal prefix codes.

Huffman Codes

- A simple construction of optimal prefix codes.
- **Binary Case**
  - Keep merging the two smallest probability masses until one probability mass (i.e., 1) is left.

Huffman Codes

- A simple construction of optimal prefix codes.
- **Binary Case**
  - Keep merging the two smallest probability masses until one probability mass (i.e., 1) is left.
- **D-ary Case**
  - Insert zero probability masses until there are $D + k(D-1)$ masses (if necessary).

Huffman Codes

- A simple construction of optimal prefix codes.
- **Binary Case**
  - Keep merging the two smallest probability masses until one probability mass (i.e., 1) is left.
- **D-ary Case**
  - Insert zero probability masses until there are $D + k(D - 1)$ masses (if necessary).
  - Keep merging the D smallest probability masses until one probability mass (i.e., 1) is left.

Huffman Codes

- A simple construction of optimal prefix codes.
- **Binary Case**
  - Keep merging the two smallest probability masses until one probability mass (i.e., 1) is left.
- **D-ary Case**
  - Insert zero probability masses until there are $D + k(D - 1)$ masses (if necessary).
  - Keep merging the D smallest probability masses until one probability mass (i.e., 1) is left.
- In general there can be more than one Huffman code.

Example 4.13:
Huffman Procedure

	p_i	codeword
	0.35	
	0.1	
0 ↑	0.15	
↓ 1	0.2	
	0.2	

Example 4.13:
Huffman Procedure

	p_i	codeword
	0.35	
0	0.1	
$\uparrow$		
	0.15	
$\downarrow$		
1	0.2	
	0.2	

Example 4.13:
Huffman Procedure

	p_i	codeword
	0.35	
	0.1	
0 ↑		
	0.15	
↓ 1		
	0.2	
	0.2	

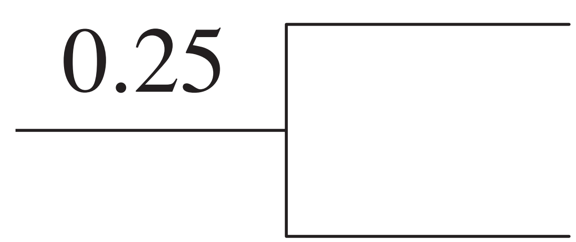
Example 4.13:
Huffman Procedure

	p_i	codeword
	0.35	
	0.1	
0.25		
	0.15	
	0.2	
	0.2	

0 ↑ ↓ 1

Example 4.13:
Huffman Procedure

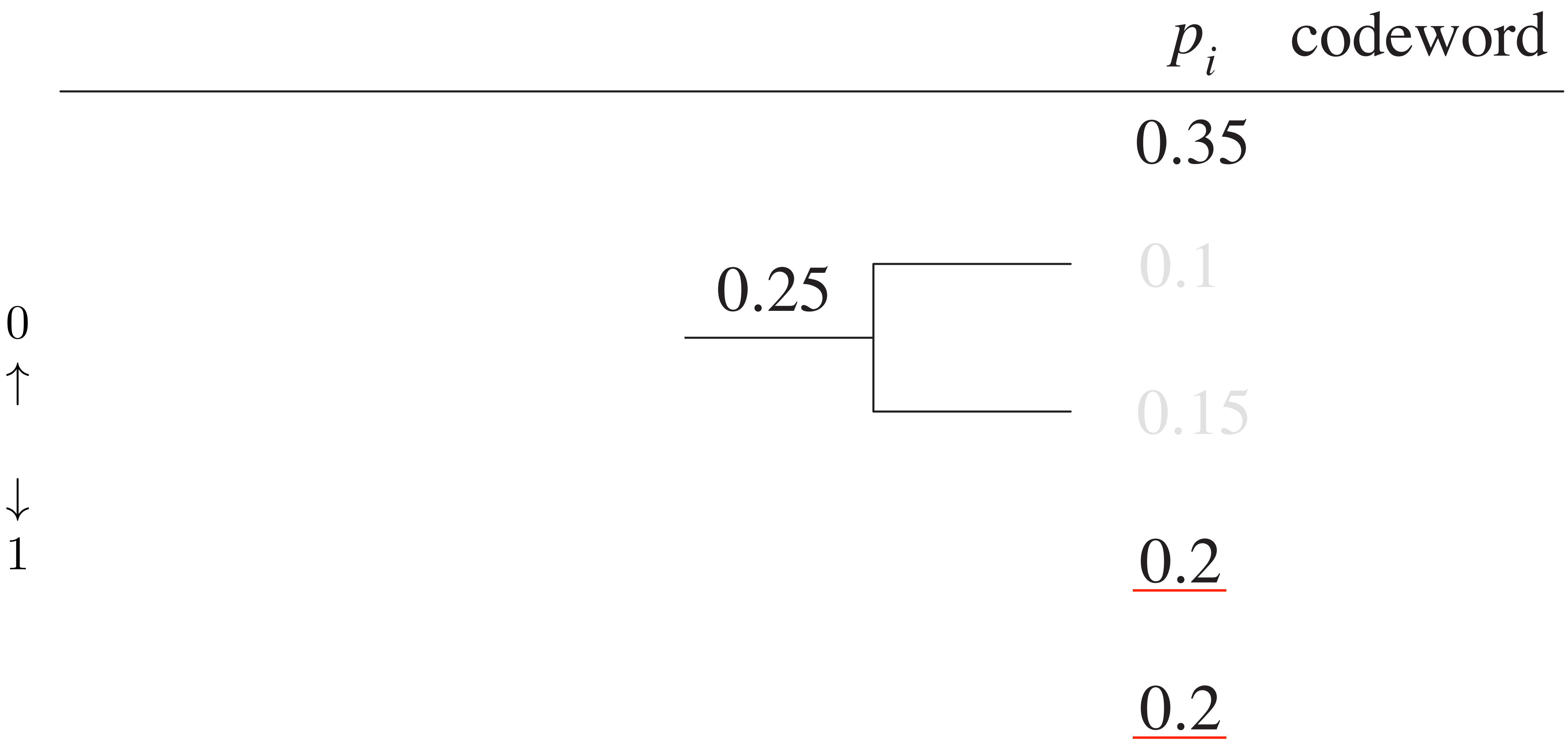
p_i	codeword
0.35	
0.1	
0.15	
0.2	
0.2	



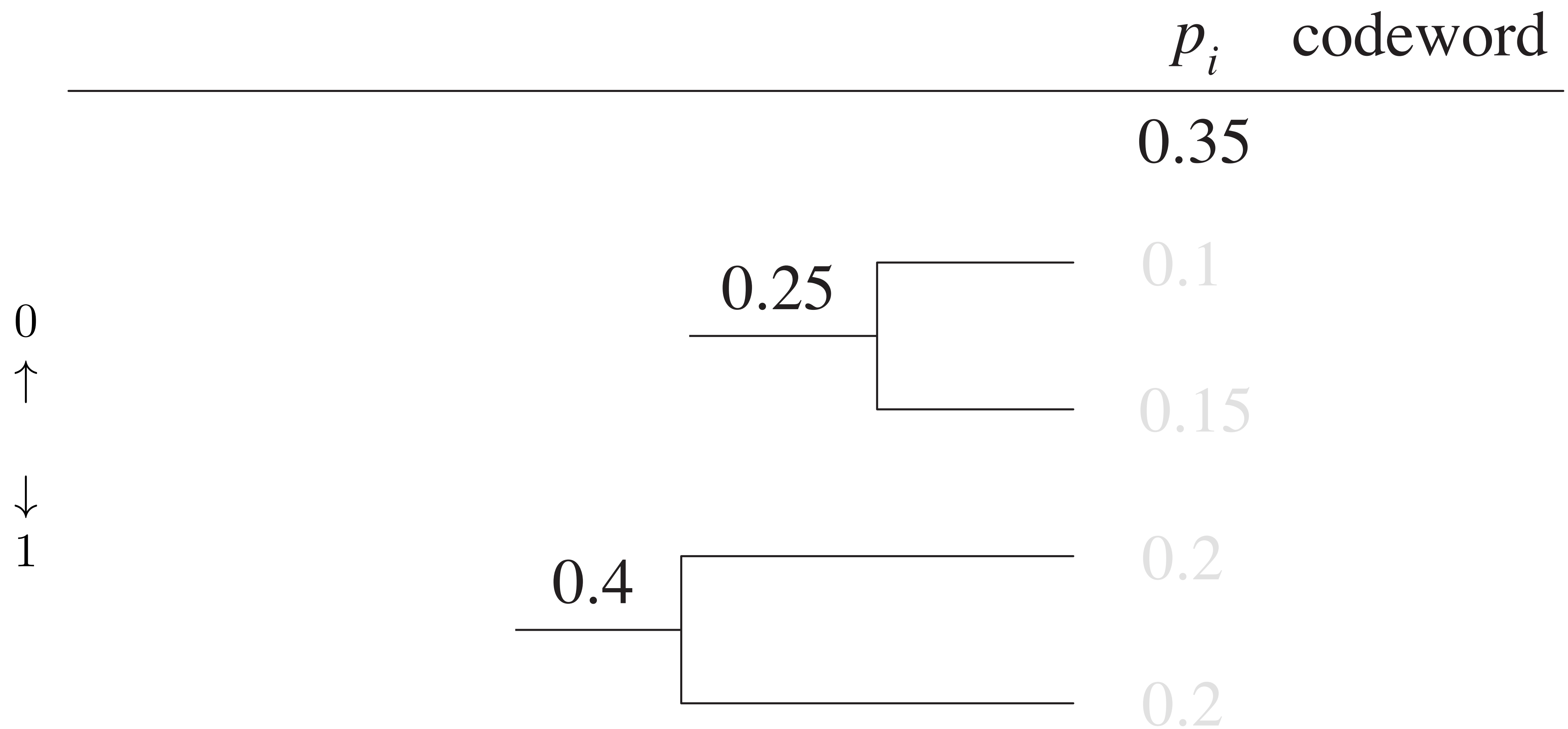
0 ↑

↓ 1

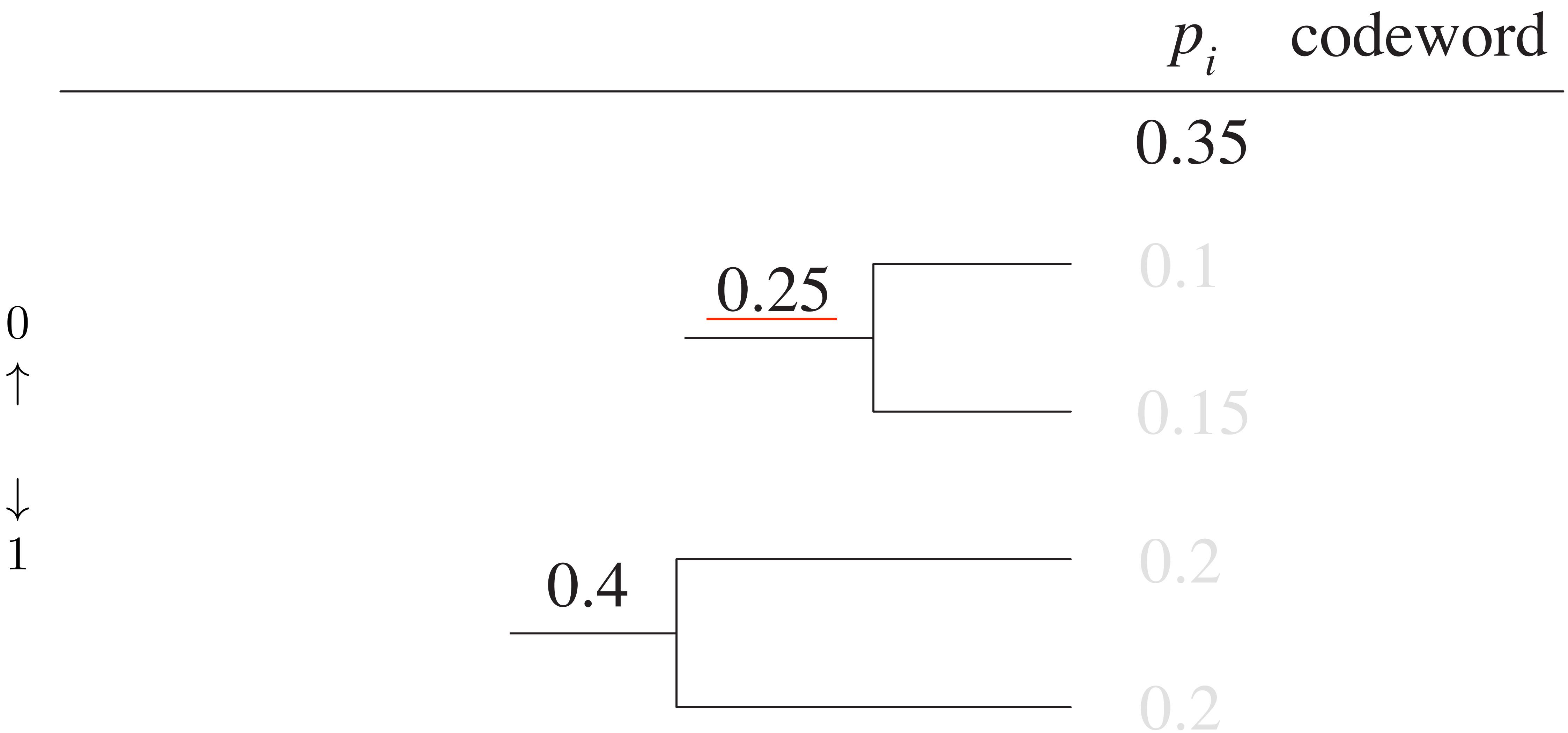
Example 4.13:
Huffman Procedure



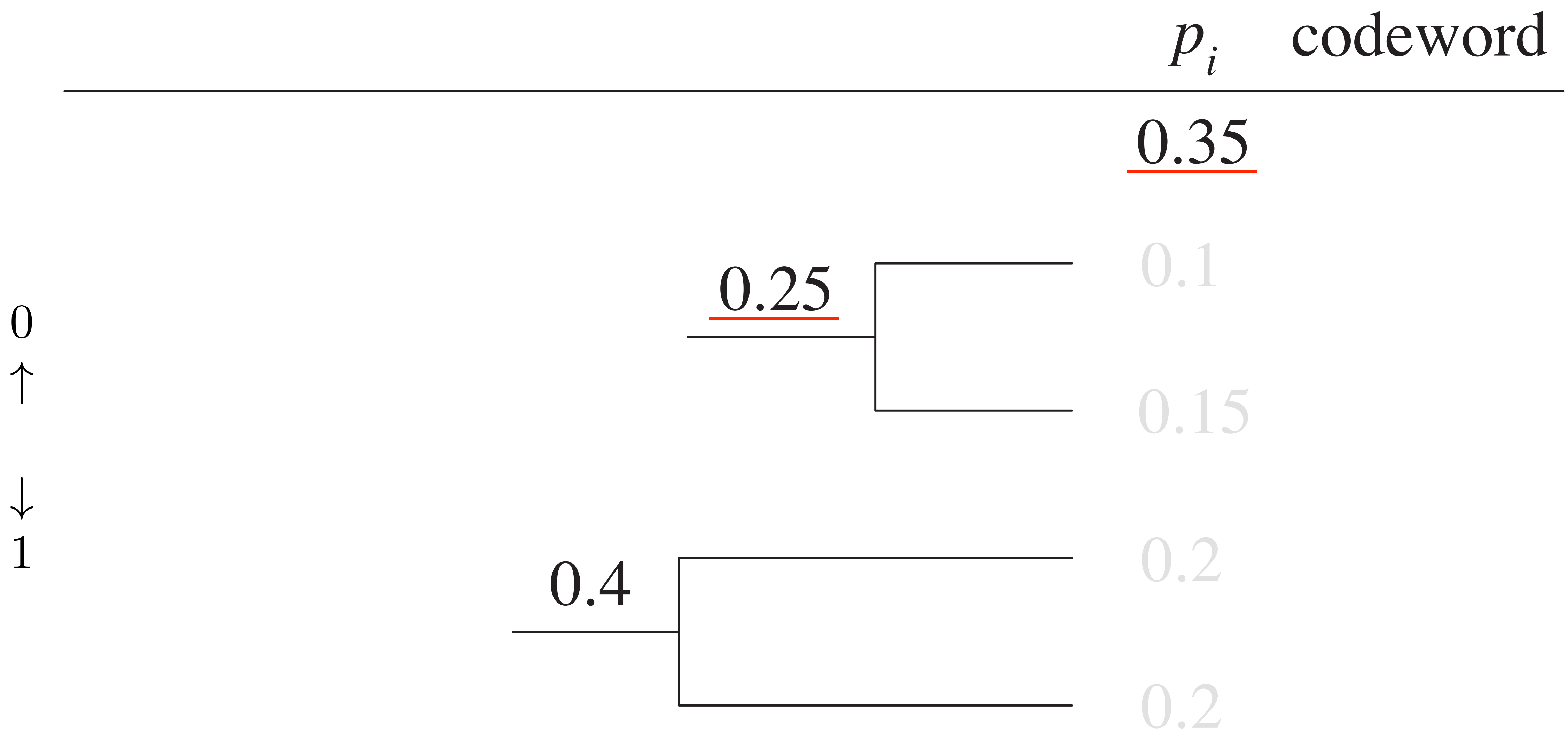
Example 4.13:
Huffman Procedure



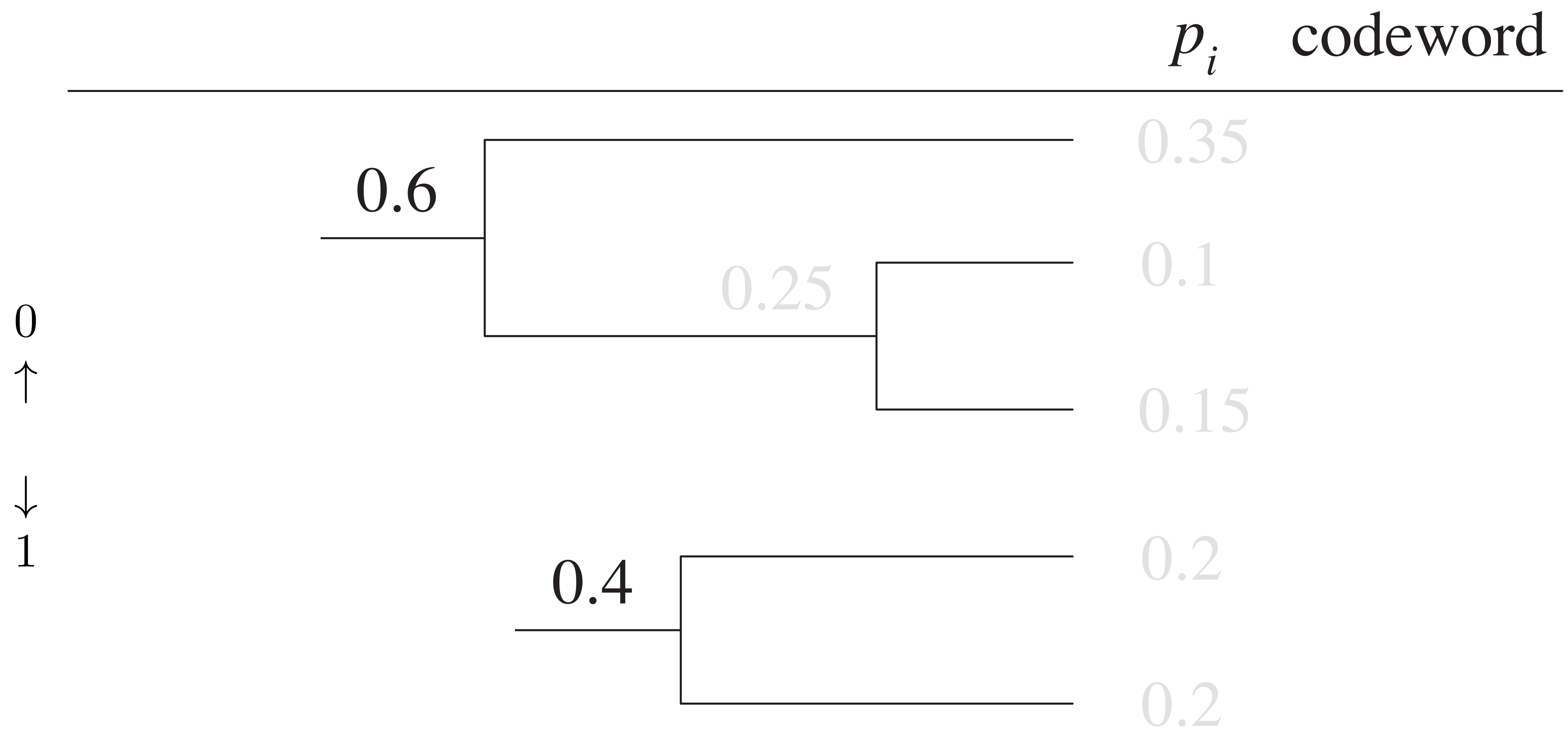
Example 4.13:
Huffman Procedure



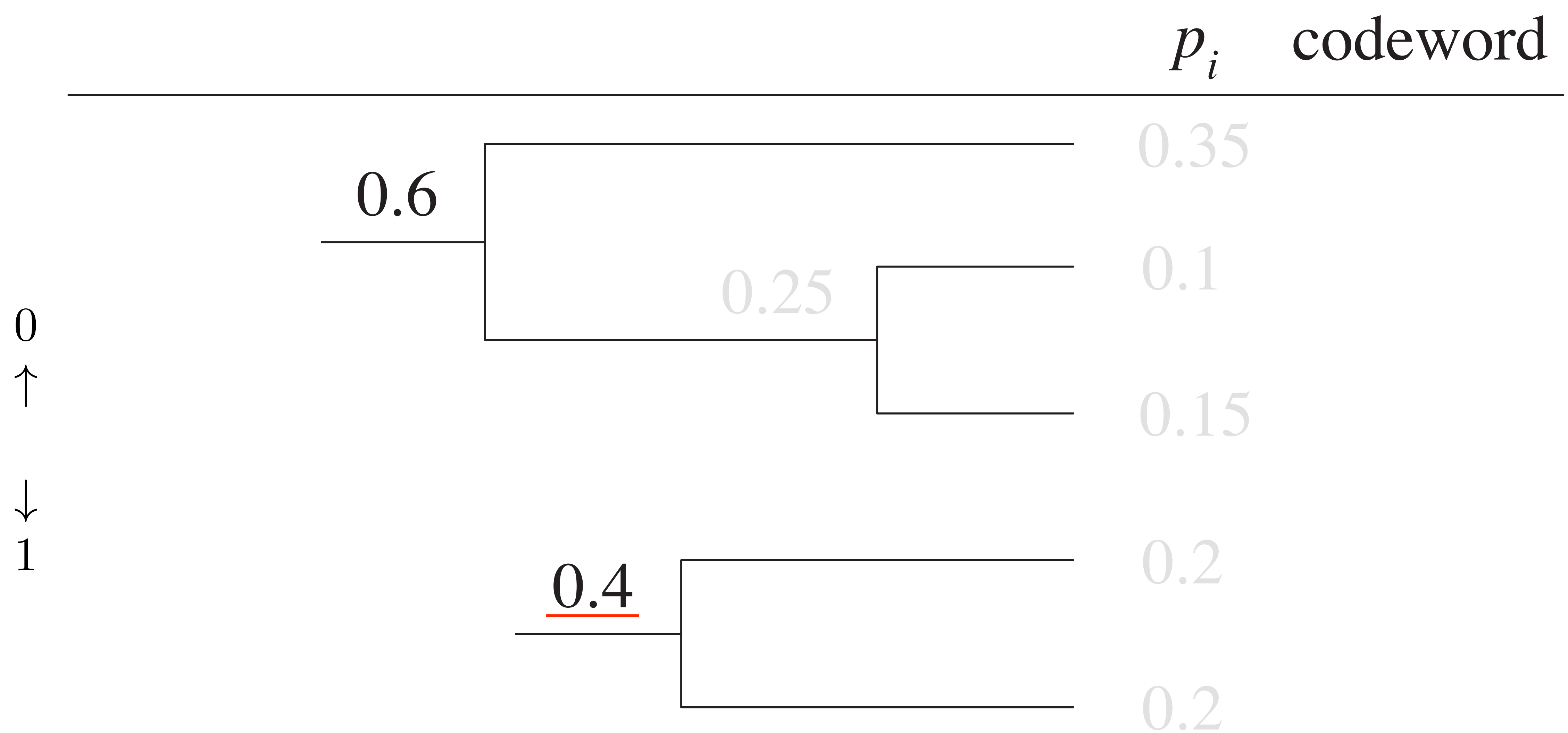
Example 4.13:
Huffman Procedure



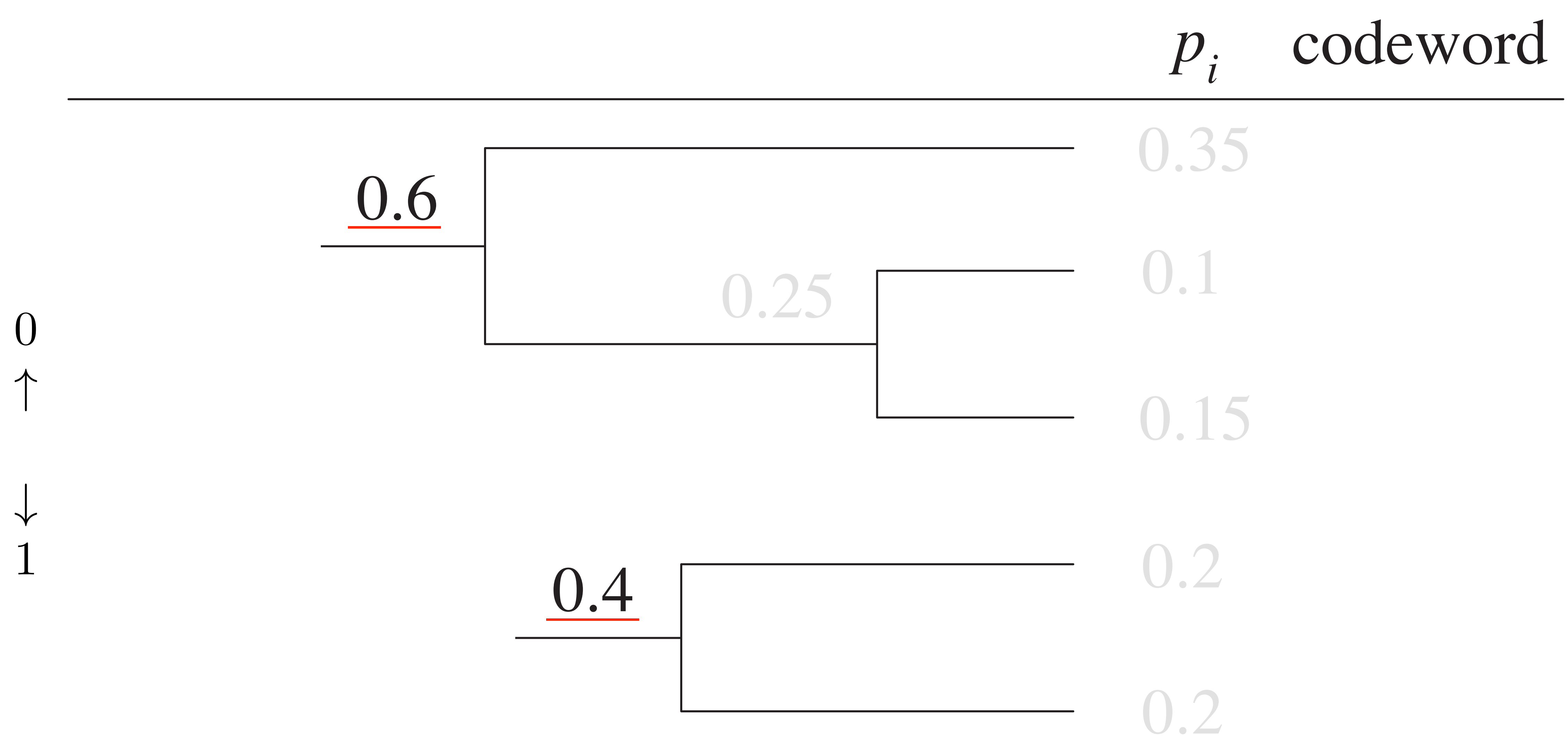
Example 4.13:
Huffman Procedure



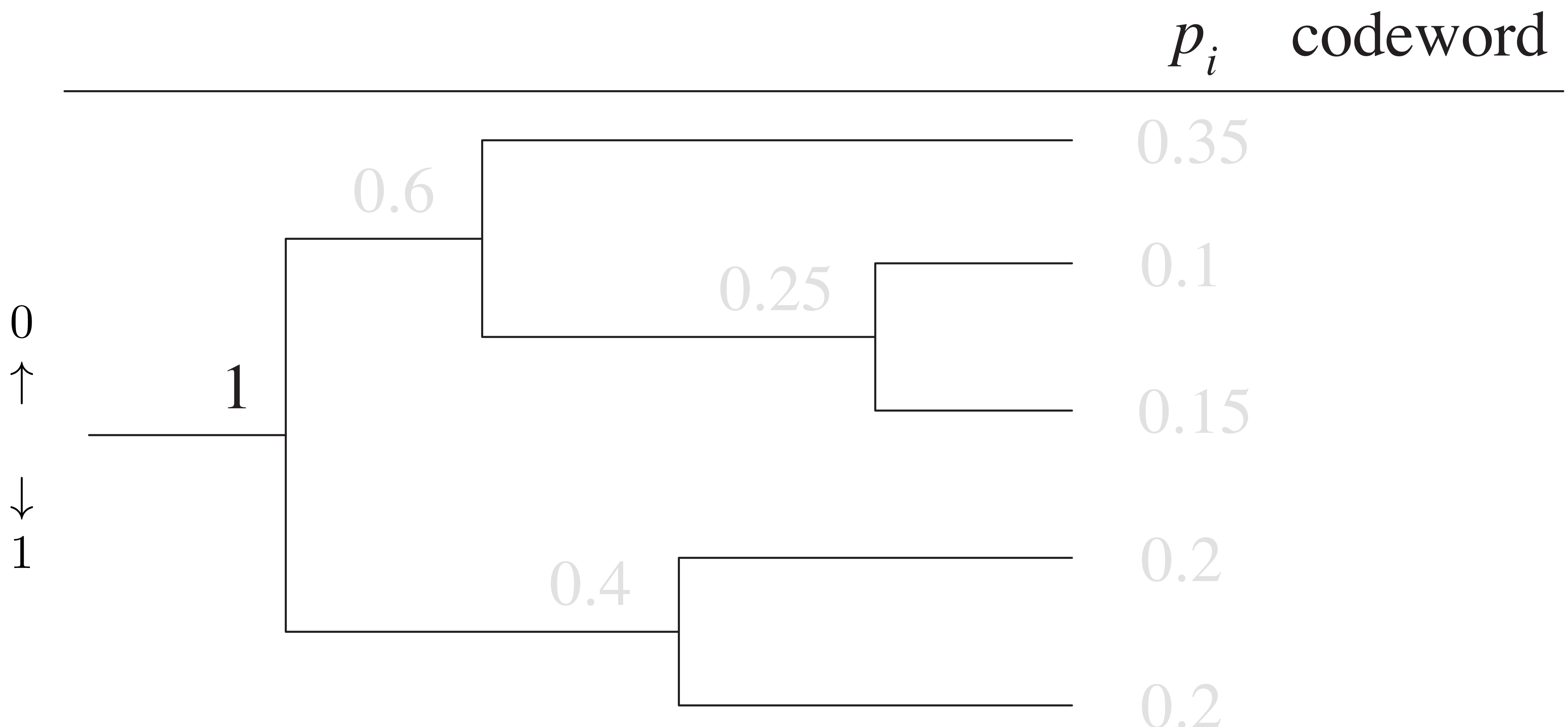
Example 4.13:
Huffman Procedure



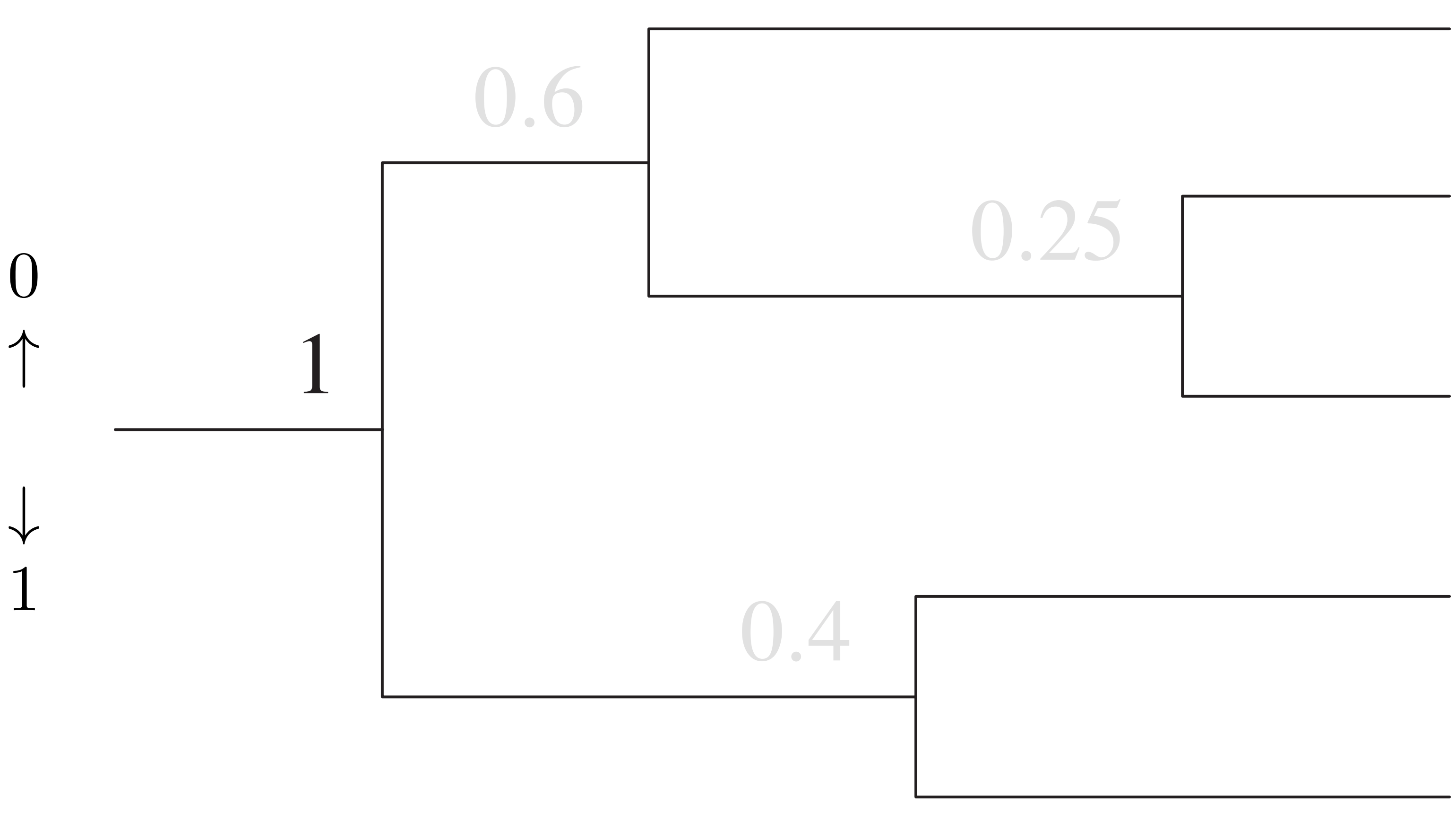
Example 4.13: Huffman Procedure



Example 4.13:
Huffman Procedure



Example 4.13:
Huffman Procedure



p_i	codeword
0.35	00
0.1	010
0.15	011
0.2	10
0.2	11

Optimality of Huffman Codes

- Without loss of generality, assume $p_1 \geq p_2 \geq \cdots \geq p_m$.
- Denote the codeword assigned to p_i by c_i, and its length by l_i.

Optimality of Huffman Codes

- Without loss of generality, assume $p_1 \geq p_2 \geq \cdots \geq p_m$.
- Denote the codeword assigned to p_i by c_i, and its length by l_i.

Theorem 4.17 The Huffman procedure produces an optimal prefix code.

Optimality of Huffman Codes

- Without loss of generality, assume $p_1 \geq p_2 \geq \cdots \geq p_m$.
- Denote the codeword assigned to p_i by c_i, and its length by l_i.

Theorem 4.17 The Huffman procedure produces an optimal prefix code.

Lemma 4.15 In an optimal code, shorter codewords are assigned to larger probabilities, i.e.,

$$l_1 \leq l_2 \leq \cdots \leq l_m.$$

Optimality of Huffman Codes

- Without loss of generality, assume $p_1 \geq p_2 \geq \cdots \geq p_m$.
- Denote the codeword assigned to p_i by c_i, and its length by l_i.

Theorem 4.17 The Huffman procedure produces an optimal prefix code.

Lemma 4.15 In an optimal code, shorter codewords are assigned to larger probabilities, i.e.,

$$l_1 \leq l_2 \leq \cdots \leq l_m.$$

Lemma 4.16 There exists an optimal code in which the codewords assigned to the two smallest probabilities are siblings, i.e., the two codewords have the same length and they differ only in the last symbol.

Lemma 4.15 In an optimal code, shorter codewords are assigned to larger probabilities, i.e.,

$$l_1 \leq l_2 \leq \cdots \leq l_m. \qquad (1)$$

Proof

Lemma 4.15 In an optimal code, shorter codewords are assigned to larger probabilities, i.e.,

$$l_1 \leq l_2 \leq \cdots \leq l_m. \tag{1}$$

Proof

1. Consider a probability distribution

$$\{p_1, \cdots, p_i, \cdots, p_j, \cdots p_m\}$$

such that $p_i > p_j$. Assume that in a particular code, the codewords c_i and c_j are such that $l_i > l_j$, i.e., a shorter codeword is assigned to a smaller probability.

Lemma 4.15 In an optimal code, shorter codewords are assigned to larger probabilities, i.e.,

$$l_1 \leq l_2 \leq \cdots \leq l_m. \tag{1}$$

Proof

1. Consider a probability distribution

$$\{p_1, \cdots, \underline{p_i}, \cdots, p_j, \cdots p_m\}$$

such that $\underline{p_i} > p_j$. Assume that in a particular code, the codewords c_i and c_j are such that $l_i > l_j$, i.e., a shorter codeword is assigned to a smaller probability.

Lemma 4.15 In an optimal code, shorter codewords are assigned to larger probabilities, i.e.,

$$l_1 \leq l_2 \leq \cdots \leq l_m. \tag{1}$$

Proof

1. Consider a probability distribution

$$\{p_1, \cdots, \underline{p_i}, \cdots, \underline{p_j}, \cdots p_m\}$$

such that $\underline{p_i} > \underline{p_j}$. Assume that in a particular code, the codewords c_i and c_j are such that $l_i > l_j$, i.e., a shorter codeword is assigned to a smaller probability.

Lemma 4.15 In an optimal code, shorter codewords are assigned to larger probabilities, i.e.,

$$l_1 \leq l_2 \leq \cdots \leq l_m. \tag{1}$$

Proof

1. Consider a probability distribution

$$\{p_1, \cdots, p_i, \cdots, p_j, \cdots p_m\}$$

such that $p_i > p_j$. Assume that in a particular code, the codewords c_i and c_j are such that $l_i > l_j$, i.e., a shorter codeword is assigned to a smaller probability.

Lemma 4.15 In an optimal code, shorter codewords are assigned to larger probabilities, i.e.,

$$l_1 \le l_2 \le \cdots \le l_m. \quad (1)$$

Proof

1. Consider a probability distribution

$$\{p_1, \cdots, p_i, \cdots, p_j, \cdots p_m\}$$

such that $p_i > p_j$. Assume that in a particular code, the codewords c_i and c_j are such that $l_i > l_j$, i.e., a shorter codeword is assigned to a smaller probability.

Lemma 4.15 In an optimal code, shorter codewords are assigned to larger probabilities, i.e.,

$$l_1 \leq l_2 \leq \cdots \leq l_m. \tag{1}$$

Proof

1. Consider a probability distribution

$$\{p_1, \cdots, p_i, \cdots, p_j, \cdots p_m\}$$

such that $p_i > p_j$. Assume that in a particular code, the codewords c_i and c_j are such that $l_i > l_j$, i.e., a shorter codeword is assigned to a smaller probability.

Lemma 4.15 In an optimal code, shorter codewords are assigned to larger probabilities, i.e.,

$$l_1 \leq l_2 \leq \cdots \leq l_m. \tag{1}$$

Proof

1. Consider a probability distribution

$$\{p_1, \cdots, p_i, \cdots, p_j, \cdots p_m\}$$

such that $p_i > p_j$. Assume that in a particular code, the codewords c_i and c_j are such that $l_i > l_j$, i.e., a shorter codeword is assigned to a smaller probability.

2. Intuitively, by exchanging c_i and c_j, the expected length of the code should be improved.

Lemma 4.15 In an optimal code, shorter codewords are assigned to larger probabilities, i.e.,

$$l_1 \leq l_2 \leq \cdots \leq l_m. \tag{1}$$

Proof

1. Consider a probability distribution

$$\{p_1, \cdots, p_i, \cdots, p_j, \cdots p_m\}$$

such that $p_i > p_j$. Assume that in a particular code, the codewords c_i and c_j are such that $l_i > l_j$, i.e., a shorter codeword is assigned to a smaller probability.

2. Intuitively, by exchanging c_i and c_j, the expected length of the code should be improved.

3. Specifically, let

$$L = \sum_k p_k l_k = \sum_{k \neq i,j} p_k l_k + (p_i l_i + p_j l_j)$$

be the expected length of the code, and

Lemma 4.15 In an optimal code, shorter codewords are assigned to larger probabilities, i.e.,

$$l_1 \leq l_2 \leq \cdots \leq l_m. \tag{1}$$

Proof

1. Consider a probability distribution

$$\{p_1, \cdots, p_i, \cdots, p_j, \cdots p_m\}$$

such that $p_i > p_j$. Assume that in a particular code, the codewords c_i and c_j are such that $l_i > l_j$, i.e., a shorter codeword is assigned to a smaller probability.

2. Intuitively, by exchanging c_i and c_j, the expected length of the code should be improved.

3. Specifically, let

$$L = \sum_k p_k l_k = \sum_{k \neq i,j} p_k l_k + (p_i l_i + p_j l_j)$$

be the expected length of the code, and

$$L' = \sum_{k \neq i,j} p_k l_k + (p_i l_j + p_j l_i)$$

be the expected length of the code obtained by exchanging c_i and c_j.

Lemma 4.15 In an optimal code, shorter codewords are assigned to larger probabilities, i.e.,

$$l_1 \leq l_2 \leq \cdots \leq l_m. \tag{1}$$

Proof

1. Consider a probability distribution

$$\{p_1, \cdots, p_i, \cdots, p_j, \cdots p_m\}$$

such that $p_i > p_j$. Assume that in a particular code, the codewords c_i and c_j are such that $l_i > l_j$, i.e., a shorter codeword is assigned to a smaller probability.

2. Intuitively, by exchanging c_i and c_j, the expected length of the code should be improved.

3. Specifically, let

$$L = \sum_k p_k l_k = \sum_{k \neq i,j} p_k l_k + (p_i \underline{l_i} + p_j l_j)$$

be the expected length of the code, and

$$L' = \sum_{k \neq i,j} p_k l_k + (p_i l_j + p_j l_i)$$

be the expected length of the code obtained by exchanging c_i and c_j.

Lemma 4.15 In an optimal code, shorter codewords are assigned to larger probabilities, i.e.,

$$l_1 \leq l_2 \leq \cdots \leq l_m. \quad (1)$$

Proof

1. Consider a probability distribution

$$\{p_1, \cdots, p_i, \cdots, p_j, \cdots p_m\}$$

such that $p_i > p_j$. Assume that in a particular code, the codewords c_i and c_j are such that $l_i > l_j$, i.e., a shorter codeword is assigned to a smaller probability.

2. Intuitively, by exchanging c_i and c_j, the expected length of the code should be improved.

3. Specifically, let

$$L = \sum_k p_k l_k = \sum_{k \neq i,j} p_k l_k + (p_i \underline{l_i} + p_j l_j)$$

be the expected length of the code, and

$$L' = \sum_{k \neq i,j} p_k l_k + (p_i \underline{l_j} + p_j l_i)$$

be the expected length of the code obtained by exchanging c_i and c_j.

Lemma 4.15 In an optimal code, shorter codewords are assigned to larger probabilities, i.e.,

$$l_1 \leq l_2 \leq \cdots \leq l_m. \tag{1}$$

Proof

1. Consider a probability distribution

$$\{p_1, \cdots, p_i, \cdots, p_j, \cdots p_m\}$$

such that $p_i > p_j$. Assume that in a particular code, the codewords c_i and c_j are such that $l_i > l_j$, i.e., a shorter codeword is assigned to a smaller probability.

2. Intuitively, by exchanging c_i and c_j, the expected length of the code should be improved.

3. Specifically, let

$$L = \sum_k p_k l_k = \sum_{k \neq i,j} p_k l_k + (p_i \underline{l_i} + p_j \underline{l_j})$$

be the expected length of the code, and

$$L' = \sum_{k \neq i,j} p_k l_k + (p_i \underline{l_j} + p_j l_i)$$

be the expected length of the code obtained by exchanging c_i and c_j.

Lemma 4.15 In an optimal code, shorter codewords are assigned to larger probabilities, i.e.,

$$l_1 \le l_2 \le \cdots \le l_m. \tag{1}$$

Proof

1. Consider a probability distribution

$$\{p_1, \cdots, p_i, \cdots, p_j, \cdots p_m\}$$

such that $p_i > p_j$. Assume that in a particular code, the codewords c_i and c_j are such that $l_i > l_j$, i.e., a shorter codeword is assigned to a smaller probability.

2. Intuitively, by exchanging c_i and c_j, the expected length of the code should be improved.

3. Specifically, let

$$L = \sum_k p_k l_k = \sum_{k \neq i,j} p_k l_k + (p_i \underline{l_i} + p_j \underline{l_j})$$

be the expected length of the code, and

$$L' = \sum_{k \neq i,j} p_k l_k + (p_i \underline{l_j} + p_j \underline{l_i})$$

be the expected length of the code obtained by exchanging c_i and c_j.

Lemma 4.15 In an optimal code, shorter codewords are assigned to larger probabilities, i.e.,

$$l_1 \le l_2 \le \cdots \le l_m. \tag{1}$$

Proof

1. Consider a probability distribution

$$\{p_1, \cdots, p_i, \cdots, p_j, \cdots p_m\}$$

such that $p_i > p_j$. Assume that in a particular code, the codewords c_i and c_j are such that $l_i > l_j$, i.e., a shorter codeword is assigned to a smaller probability.

2. Intuitively, by exchanging c_i and c_j, the expected length of the code should be improved.

3. Specifically, let

$$L = \sum_k p_k l_k = \sum_{k \neq i,j} p_k l_k + (p_i l_i + p_j l_j)$$

be the expected length of the code, and

$$L' = \sum_{k \neq i,j} p_k l_k + (p_i l_j + p_j l_i)$$

be the expected length of the code obtained by exchanging c_i and c_j.

4. Comparing L' and L, we see that

Lemma 4.15 In an optimal code, shorter codewords are assigned to larger probabilities, i.e.,

$$l_1 \leq l_2 \leq \cdots \leq l_m. \tag{1}$$

Proof

1. Consider a probability distribution

$$\{p_1, \cdots, p_i, \cdots, p_j, \cdots p_m\}$$

such that $p_i > p_j$. Assume that in a particular code, the codewords c_i and c_j are such that $l_i > l_j$, i.e., a shorter codeword is assigned to a smaller probability.

2. Intuitively, by exchanging c_i and c_j, the expected length of the code should be improved.

3. Specifically, let

$$L = \sum_k p_k l_k = \sum_{k \neq i,j} p_k l_k + (p_i l_i + p_j l_j)$$

be the expected length of the code, and

$$L' = \sum_{k \neq i,j} p_k l_k + (p_i l_j + p_j l_i)$$

be the expected length of the code obtained by exchanging c_i and c_j.

4. Comparing L' and L, we see that

Lemma 4.15 In an optimal code, shorter codewords are assigned to larger probabilities, i.e.,

$$l_1 \leq l_2 \leq \cdots \leq l_m. \qquad (1)$$

Proof

1. Consider a probability distribution

$$\{p_1, \cdots, p_i, \cdots, p_j, \cdots p_m\}$$

such that $p_i > p_j$. Assume that in a particular code, the codewords c_i and c_j are such that $l_i > l_j$, i.e., a shorter codeword is assigned to a smaller probability.

2. Intuitively, by exchanging c_i and c_j, the expected length of the code should be improved.

3. Specifically, let

$$L = \sum_k p_k l_k = \sum_{k \neq i,j} p_k l_k + (p_i l_i + p_j l_j)$$

be the expected length of the code, and

$$L' = \sum_{k \neq i,j} p_k l_k + (p_i l_j + p_j l_i)$$

be the expected length of the code obtained by exchanging c_i and c_j.

4. Comparing L' and L, we see that

$$L' - L \quad = \quad (p_i l_j + p_j l_i) - (p_i l_i + p_j l_j)$$

Lemma 4.15 In an optimal code, shorter codewords are assigned to larger probabilities, i.e.,

$$l_1 \leq l_2 \leq \cdots \leq l_m. \quad (1)$$

Proof

1. Consider a probability distribution

$$\{p_1, \cdots, p_i, \cdots, p_j, \cdots p_m\}$$

such that $p_i > p_j$. Assume that in a particular code, the codewords c_i and c_j are such that $l_i > l_j$, i.e., a shorter codeword is assigned to a smaller probability.

2. Intuitively, by exchanging c_i and c_j, the expected length of the code should be improved.

3. Specifically, let

$$L = \sum_k p_k l_k = \sum_{k \neq i,j} p_k l_k + (p_i l_i + p_j l_j)$$

be the expected length of the code, and

$$L' = \sum_{k \neq i,j} p_k l_k + (p_i l_j + p_j l_i)$$

be the expected length of the code obtained by exchanging c_i and c_j.

4. Comparing L' and L, we see that

$$L' - L = (p_i l_j + p_j l_i) - (p_i l_i + p_j l_j)$$

Lemma 4.15 In an optimal code, shorter codewords are assigned to larger probabilities, i.e.,

$$l_1 \leq l_2 \leq \cdots \leq l_m. \quad (1)$$

Proof

1. Consider a probability distribution

$$\{p_1, \cdots, p_i, \cdots, p_j, \cdots p_m\}$$

such that $p_i > p_j$. Assume that in a particular code, the codewords c_i and c_j are such that $l_i > l_j$, i.e., a shorter codeword is assigned to a smaller probability.

2. Intuitively, by exchanging c_i and c_j, the expected length of the code should be improved.

3. Specifically, let

$$L = \sum_k p_k l_k = \sum_{k \neq i,j} p_k l_k + (p_i l_i + p_j l_j)$$

be the expected length of the code, and

$$L' = \sum_{k \neq i,j} p_k l_k + (p_i l_j + p_j l_i)$$

be the expected length of the code obtained by exchanging c_i and c_j.

4. Comparing L' and L, we see that

$$L' - L = (p_i l_j + p_j l_i) - (p_i l_i + p_j l_j)$$

Lemma 4.15 In an optimal code, shorter codewords are assigned to larger probabilities, i.e.,

$$l_1 \le l_2 \le \cdots \le l_m. \tag{1}$$

Proof

1. Consider a probability distribution

$$\{p_1, \cdots, p_i, \cdots, p_j, \cdots p_m\}$$

such that $p_i > p_j$. Assume that in a particular code, the codewords c_i and c_j are such that $l_i > l_j$, i.e., a shorter codeword is assigned to a smaller probability.

2. Intuitively, by exchanging c_i and c_j, the expected length of the code should be improved.

3. Specifically, let

$$L = \sum_k p_k l_k = \sum_{k \neq i,j} p_k l_k + (p_i l_i + p_j l_j)$$

be the expected length of the code, and

$$L' = \sum_{k \neq i,j} p_k l_k + (p_i l_j + p_j l_i)$$

be the expected length of the code obtained by exchanging c_i and c_j.

4. Comparing L' and L, we see that

$$L' - L = (p_i l_j + p_j l_i) - (p_i l_i + p_j l_j)$$

Lemma 4.15 In an optimal code, shorter codewords are assigned to larger probabilities, i.e.,

$$l_1 \leq l_2 \leq \cdots \leq l_m. \tag{1}$$

Proof

1. Consider a probability distribution

$$\{p_1, \cdots, p_i, \cdots, p_j, \cdots p_m\}$$

such that $p_i > p_j$. Assume that in a particular code, the codewords c_i and c_j are such that $l_i > l_j$, i.e., a shorter codeword is assigned to a smaller probability.

2. Intuitively, by exchanging c_i and c_j, the expected length of the code should be improved.

3. Specifically, let

$$L = \sum_k p_k l_k = \sum_{k \neq i,j} p_k l_k + (p_i l_i + p_j l_j)$$

be the expected length of the code, and

$$L' = \sum_{k \neq i,j} p_k l_k + (p_i l_j + p_j l_i)$$

be the expected length of the code obtained by exchanging c_i and c_j.

4. Comparing L' and L, we see that

$$L' - L = (p_i l_j + p_j l_i) - (p_i l_i + p_j l_j)$$

Lemma 4.15 In an optimal code, shorter codewords are assigned to larger probabilities, i.e.,

$$l_1 \leq l_2 \leq \cdots \leq l_m. \tag{1}$$

Proof

1. Consider a probability distribution

$$\{p_1, \cdots, p_i, \cdots, p_j, \cdots p_m\}$$

such that $p_i > p_j$. Assume that in a particular code, the codewords c_i and c_j are such that $l_i > l_j$, i.e., a shorter codeword is assigned to a smaller probability.

2. Intuitively, by exchanging c_i and c_j, the expected length of the code should be improved.

3. Specifically, let

$$L = \sum_k p_k l_k = \sum_{k \neq i,j} p_k l_k + (p_i l_i + p_j l_j)$$

be the expected length of the code, and

$$L' = \sum_{k \neq i,j} p_k l_k + (p_i l_j + p_j l_i)$$

be the expected length of the code obtained by exchanging c_i and c_j.

4. Comparing L' and L, we see that

$$\begin{aligned} L' - L &= (p_i l_j + p_j l_i) - (p_i l_i + p_j l_j) \\ &= (p_i l_j - p_i l_i) - (p_j l_j - p_j l_i) \end{aligned}$$

Lemma 4.15 In an optimal code, shorter codewords are assigned to larger probabilities, i.e.,

$$l_1 \leq l_2 \leq \cdots \leq l_m. \tag{1}$$

Proof

1. Consider a probability distribution

$$\{p_1, \cdots, p_i, \cdots, p_j, \cdots p_m\}$$

such that $p_i > p_j$. Assume that in a particular code, the codewords c_i and c_j are such that $l_i > l_j$, i.e., a shorter codeword is assigned to a smaller probability.

2. Intuitively, by exchanging c_i and c_j, the expected length of the code should be improved.

3. Specifically, let

$$L = \sum_k p_k l_k = \sum_{k \neq i,j} p_k l_k + (p_i l_i + p_j l_j)$$

be the expected length of the code, and

$$L' = \sum_{k \neq i,j} p_k l_k + (p_i l_j + p_j l_i)$$

be the expected length of the code obtained by exchanging c_i and c_j.

4. Comparing L' and L, we see that

$$\begin{aligned} L' - L &= (\underline{p_i l_j} + p_j l_i) - (p_i l_i + p_j l_j) \\ &= (\underline{p_i l_j} - p_i l_i) - (p_j l_j - p_j l_i) \end{aligned}$$

Lemma 4.15 In an optimal code, shorter codewords are assigned to larger probabilities, i.e.,

$$l_1 \leq l_2 \leq \cdots \leq l_m. \tag{1}$$

Proof

1. Consider a probability distribution

$$\{p_1, \cdots, p_i, \cdots, p_j, \cdots p_m\}$$

such that $p_i > p_j$. Assume that in a particular code, the codewords c_i and c_j are such that $l_i > l_j$, i.e., a shorter codeword is assigned to a smaller probability.

2. Intuitively, by exchanging c_i and c_j, the expected length of the code should be improved.

3. Specifically, let

$$L = \sum_k p_k l_k = \sum_{k \neq i,j} p_k l_k + (p_i l_i + p_j l_j)$$

be the expected length of the code, and

$$L' = \sum_{k \neq i,j} p_k l_k + (p_i l_j + p_j l_i)$$

be the expected length of the code obtained by exchanging c_i and c_j.

4. Comparing L' and L, we see that

$$\begin{aligned} L' - L &= (p_i l_j + p_j l_i) - (\underline{p_i l_i} + p_j l_j) \\ &= (p_i l_j - \underline{p_i l_i}) - (p_j l_j - p_j l_i) \end{aligned}$$

Lemma 4.15 In an optimal code, shorter codewords are assigned to larger probabilities, i.e.,

$$l_1 \leq l_2 \leq \cdots \leq l_m. \tag{1}$$

Proof

1. Consider a probability distribution

$$\{p_1, \cdots, p_i, \cdots, p_j, \cdots p_m\}$$

such that $p_i > p_j$. Assume that in a particular code, the codewords c_i and c_j are such that $l_i > l_j$, i.e., a shorter codeword is assigned to a smaller probability.

2. Intuitively, by exchanging c_i and c_j, the expected length of the code should be improved.

3. Specifically, let

$$L = \sum_k p_k l_k = \sum_{k \neq i,j} p_k l_k + (p_i l_i + p_j l_j)$$

be the expected length of the code, and

$$L' = \sum_{k \neq i,j} p_k l_k + (p_i l_j + p_j l_i)$$

be the expected length of the code obtained by exchanging c_i and c_j.

4. Comparing L' and L, we see that

$$\begin{aligned} L' - L &= (p_i l_j + p_j l_i) - (p_i l_i + \underline{p_j l_j}) \\ &= (p_i l_j - p_i l_i) - (\underline{p_j l_j} - p_j l_i) \end{aligned}$$

Lemma 4.15 In an optimal code, shorter codewords are assigned to larger probabilities, i.e.,

$$l_1 \leq l_2 \leq \cdots \leq l_m. \quad (1)$$

Proof

1. Consider a probability distribution

$$\{p_1, \cdots, p_i, \cdots, p_j, \cdots p_m\}$$

such that $p_i > p_j$. Assume that in a particular code, the codewords c_i and c_j are such that $l_i > l_j$, i.e., a shorter codeword is assigned to a smaller probability.

2. Intuitively, by exchanging c_i and c_j, the expected length of the code should be improved.

3. Specifically, let

$$L = \sum_k p_k l_k = \sum_{k \neq i,j} p_k l_k + (p_i l_i + p_j l_j)$$

be the expected length of the code, and

$$L' = \sum_{k \neq i,j} p_k l_k + (p_i l_j + p_j l_i)$$

be the expected length of the code obtained by exchanging c_i and c_j.

4. Comparing L' and L, we see that

$$\begin{aligned} L' - L &= (p_i l_j + \underline{p_j l_i}) - (p_i l_i + p_j l_j) \\ &= (p_i l_j - p_i l_i) - (p_j l_j - \underline{p_j l_i}) \end{aligned}$$

Lemma 4.15 In an optimal code, shorter codewords are assigned to larger probabilities, i.e.,

$$l_1 \leq l_2 \leq \cdots \leq l_m. \tag{1}$$

Proof

1. Consider a probability distribution

$$\{p_1, \cdots, p_i, \cdots, p_j, \cdots p_m\}$$

such that $p_i > p_j$. Assume that in a particular code, the codewords c_i and c_j are such that $l_i > l_j$, i.e., a shorter codeword is assigned to a smaller probability.

2. Intuitively, by exchanging c_i and c_j, the expected length of the code should be improved.

3. Specifically, let

$$L = \sum_k p_k l_k = \sum_{k \neq i,j} p_k l_k + (p_i l_i + p_j l_j)$$

be the expected length of the code, and

$$L' = \sum_{k \neq i,j} p_k l_k + (p_i l_j + p_j l_i)$$

be the expected length of the code obtained by exchanging c_i and c_j.

4. Comparing L' and L, we see that

$$\begin{aligned} L' - L &= (p_i l_j + p_j l_i) - (p_i l_i + p_j l_j) \\ &= (p_i l_j - p_i l_i) - (p_j l_j - p_j l_i) \end{aligned}$$

Lemma 4.15 In an optimal code, shorter codewords are assigned to larger probabilities, i.e.,

$$l_1 \leq l_2 \leq \cdots \leq l_m. \tag{1}$$

Proof

1. Consider a probability distribution

$$\{p_1, \cdots, p_i, \cdots, p_j, \cdots p_m\}$$

such that $p_i > p_j$. Assume that in a particular code, the codewords c_i and c_j are such that $l_i > l_j$, i.e., a shorter codeword is assigned to a smaller probability.

2. Intuitively, by exchanging c_i and c_j, the expected length of the code should be improved.

3. Specifically, let

$$L = \sum_k p_k l_k = \sum_{k \neq i,j} p_k l_k + (p_i l_i + p_j l_j)$$

be the expected length of the code, and

$$L' = \sum_{k \neq i,j} p_k l_k + (p_i l_j + p_j l_i)$$

be the expected length of the code obtained by exchanging c_i and c_j.

4. Comparing L' and L, we see that

$$\begin{aligned} L' - L &= (p_i l_j + p_j l_i) - (p_i l_i + p_j l_j) \\ &= (\underline{p_i} l_j - \underline{p_i} l_i) - (p_j l_j - p_j l_i) \end{aligned}$$

Lemma 4.15 In an optimal code, shorter codewords are assigned to larger probabilities, i.e.,

$$l_1 \leq l_2 \leq \cdots \leq l_m. \tag{1}$$

Proof

1. Consider a probability distribution

$$\{p_1, \cdots, p_i, \cdots, p_j, \cdots p_m\}$$

such that $p_i > p_j$. Assume that in a particular code, the codewords c_i and c_j are such that $l_i > l_j$, i.e., a shorter codeword is assigned to a smaller probability.

2. Intuitively, by exchanging c_i and c_j, the expected length of the code should be improved.

3. Specifically, let

$$L = \sum_k p_k l_k = \sum_{k \neq i,j} p_k l_k + (p_i l_i + p_j l_j)$$

be the expected length of the code, and

$$L' = \sum_{k \neq i,j} p_k l_k + (p_i l_j + p_j l_i)$$

be the expected length of the code obtained by exchanging c_i and c_j.

4. Comparing L' and L, we see that

$$\begin{aligned} L' - L &= (p_i l_j + p_j l_i) - (p_i l_i + p_j l_j) \\ &= (\underline{p_i} l_j - \underline{p_i} l_i) - (p_j l_j - p_j l_i) \\ &= \underline{p_i}(l_j - l_i) - p_j(l_j - l_i) \end{aligned}$$

Lemma 4.15 In an optimal code, shorter codewords are assigned to larger probabilities, i.e.,

$$l_1 \leq l_2 \leq \cdots \leq l_m. \tag{1}$$

Proof

1. Consider a probability distribution

$$\{p_1, \cdots, p_i, \cdots, p_j, \cdots p_m\}$$

such that $p_i > p_j$. Assume that in a particular code, the codewords c_i and c_j are such that $l_i > l_j$, i.e., a shorter codeword is assigned to a smaller probability.

2. Intuitively, by exchanging c_i and c_j, the expected length of the code should be improved.

3. Specifically, let

$$L = \sum_k p_k l_k = \sum_{k \neq i,j} p_k l_k + (p_i l_i + p_j l_j)$$

be the expected length of the code, and

$$L' = \sum_{k \neq i,j} p_k l_k + (p_i l_j + p_j l_i)$$

be the expected length of the code obtained by exchanging c_i and c_j.

4. Comparing L' and L, we see that

$$\begin{aligned} L' - L &= (p_i l_j + p_j l_i) - (p_i l_i + p_j l_j) \\ &= (p_i l_j - p_i l_i) - (p_j l_j - p_j l_i) \\ &= p_i(l_j - l_i) - p_j(l_j - l_i) \end{aligned}$$

Lemma 4.15 In an optimal code, shorter codewords are assigned to larger probabilities, i.e.,

$$l_1 \leq l_2 \leq \cdots \leq l_m. \tag{1}$$

Proof

1. Consider a probability distribution

$$\{p_1, \cdots, p_i, \cdots, p_j, \cdots p_m\}$$

such that $p_i > p_j$. Assume that in a particular code, the codewords c_i and c_j are such that $l_i > l_j$, i.e., a shorter codeword is assigned to a smaller probability.

2. Intuitively, by exchanging c_i and c_j, the expected length of the code should be improved.

3. Specifically, let

$$L = \sum_k p_k l_k = \sum_{k \neq i,j} p_k l_k + (p_i l_i + p_j l_j)$$

be the expected length of the code, and

$$L' = \sum_{k \neq i,j} p_k l_k + (p_i l_j + p_j l_i)$$

be the expected length of the code obtained by exchanging c_i and c_j.

4. Comparing L' and L, we see that

$$\begin{aligned} L' - L &= (p_i l_j + p_j l_i) - (p_i l_i + p_j l_j) \\ &= (p_i l_j - p_i l_i) - (p_j l_j - p_j l_i) \\ &= p_i (l_j - l_i) - p_j (l_j - l_i) \end{aligned}$$

Lemma 4.15 In an optimal code, shorter codewords are assigned to larger probabilities, i.e.,

$$l_1 \leq l_2 \leq \cdots \leq l_m. \tag{1}$$

Proof

1. Consider a probability distribution

$$\{p_1, \cdots, p_i, \cdots, p_j, \cdots p_m\}$$

such that $p_i > p_j$. Assume that in a particular code, the codewords c_i and c_j are such that $l_i > l_j$, i.e., a shorter codeword is assigned to a smaller probability.

2. Intuitively, by exchanging c_i and c_j, the expected length of the code should be improved.

3. Specifically, let

$$L = \sum_k p_k l_k = \sum_{k \neq i,j} p_k l_k + (p_i l_i + p_j l_j)$$

be the expected length of the code, and

$$L' = \sum_{k \neq i,j} p_k l_k + (p_i l_j + p_j l_i)$$

be the expected length of the code obtained by exchanging c_i and c_j.

4. Comparing L' and L, we see that

$$\begin{aligned} L' - L &= (p_i l_j + p_j l_i) - (p_i l_i + p_j l_j) \\ &= (p_i l_j - p_i l_i) - (\underline{p_j} l_j - \underline{p_j} l_i) \\ &= p_i(l_j - l_i) - \underline{p_j}(l_j - l_i) \end{aligned}$$

Lemma 4.15 In an optimal code, shorter codewords are assigned to larger probabilities, i.e.,

$$l_1 \le l_2 \le \cdots \le l_m. \qquad (1)$$

Proof

1. Consider a probability distribution

$$\{p_1, \cdots, p_i, \cdots, p_j, \cdots p_m\}$$

such that $p_i > p_j$. Assume that in a particular code, the codewords c_i and c_j are such that $l_i > l_j$, i.e., a shorter codeword is assigned to a smaller probability.

2. Intuitively, by exchanging c_i and c_j, the expected length of the code should be improved.

3. Specifically, let

$$L = \sum_k p_k l_k = \sum_{k \ne i,j} p_k l_k + (p_i l_i + p_j l_j)$$

be the expected length of the code, and

$$L' = \sum_{k \ne i,j} p_k l_k + (p_i l_j + p_j l_i)$$

be the expected length of the code obtained by exchanging c_i and c_j.

4. Comparing L' and L, we see that

$$\begin{aligned} L' - L &= (p_i l_j + p_j l_i) - (p_i l_i + p_j l_j) \\ &= (p_i l_j - p_i l_i) - (p_j l_j - p_j l_i) \\ &= p_i (l_j - l_i) - p_j (l_j - l_i) \end{aligned}$$

Lemma 4.15 In an optimal code, shorter codewords are assigned to larger probabilities, i.e.,

$$l_1 \leq l_2 \leq \cdots \leq l_m. \tag{1}$$

Proof

1. Consider a probability distribution

$$\{p_1, \cdots, p_i, \cdots, p_j, \cdots p_m\}$$

such that $p_i > p_j$. Assume that in a particular code, the codewords c_i and c_j are such that $l_i > l_j$, i.e., a shorter codeword is assigned to a smaller probability.

2. Intuitively, by exchanging c_i and c_j, the expected length of the code should be improved.

3. Specifically, let

$$L = \sum_k p_k l_k = \sum_{k \neq i,j} p_k l_k + (p_i l_i + p_j l_j)$$

be the expected length of the code, and

$$L' = \sum_{k \neq i,j} p_k l_k + (p_i l_j + p_j l_i)$$

be the expected length of the code obtained by exchanging c_i and c_j.

4. Comparing L' and L, we see that

$$\begin{aligned} L' - L &= (p_i l_j + p_j l_i) - (p_i l_i + p_j l_j) \\ &= (p_i l_j - p_i l_i) - (p_j l_j - p_j l_i) \\ &= p_i \underline{(l_j - l_i)} - p_j \underline{(l_j - l_i)} \end{aligned}$$

Lemma 4.15 In an optimal code, shorter codewords are assigned to larger probabilities, i.e.,

$$l_1 \leq l_2 \leq \cdots \leq l_m. \tag{1}$$

Proof

1. Consider a probability distribution

$$\{p_1, \cdots, p_i, \cdots, p_j, \cdots p_m\}$$

such that $p_i > p_j$. Assume that in a particular code, the codewords c_i and c_j are such that $l_i > l_j$, i.e., a shorter codeword is assigned to a smaller probability.

2. Intuitively, by exchanging c_i and c_j, the expected length of the code should be improved.

3. Specifically, let

$$L = \sum_k p_k l_k = \sum_{k \neq i,j} p_k l_k + (p_i l_i + p_j l_j)$$

be the expected length of the code, and

$$L' = \sum_{k \neq i,j} p_k l_k + (p_i l_j + p_j l_i)$$

be the expected length of the code obtained by exchanging c_i and c_j.

4. Comparing L' and L, we see that

$$\begin{aligned} L' - L &= (p_i l_j + p_j l_i) - (p_i l_i + p_j l_j) \\ &= (p_i l_j - p_i l_i) - (p_j l_j - p_j l_i) \\ &= \underline{p_i}\underline{(l_j - l_i)} - p_j \underline{(l_j - l_i)} \end{aligned}$$

Lemma 4.15 In an optimal code, shorter codewords are assigned to larger probabilities, i.e.,

$$l_1 \leq l_2 \leq \cdots \leq l_m. \tag{1}$$

Proof

1. Consider a probability distribution

$$\{p_1, \cdots, p_i, \cdots, p_j, \cdots p_m\}$$

such that $p_i > p_j$. Assume that in a particular code, the codewords c_i and c_j are such that $l_i > l_j$, i.e., a shorter codeword is assigned to a smaller probability.

2. Intuitively, by exchanging c_i and c_j, the expected length of the code should be improved.

3. Specifically, let

$$L = \sum_k p_k l_k = \sum_{k \neq i,j} p_k l_k + (p_i l_i + p_j l_j)$$

be the expected length of the code, and

$$L' = \sum_{k \neq i,j} p_k l_k + (p_i l_j + p_j l_i)$$

be the expected length of the code obtained by exchanging c_i and c_j.

4. Comparing L' and L, we see that

$$\begin{aligned} L' - L &= (p_i l_j + p_j l_i) - (p_i l_i + p_j l_j) \\ &= (p_i l_j - p_i l_i) - (p_j l_j - p_j l_i) \\ &= \underline{p_i}\underline{(l_j - l_i)} - \underline{p_j}\underline{(l_j - l_i)} \end{aligned}$$

Lemma 4.15 In an optimal code, shorter codewords are assigned to larger probabilities, i.e.,

$$l_1 \leq l_2 \leq \cdots \leq l_m. \tag{1}$$

Proof

1. Consider a probability distribution

$$\{p_1, \cdots, p_i, \cdots, p_j, \cdots p_m\}$$

such that $p_i > p_j$. Assume that in a particular code, the codewords c_i and c_j are such that $l_i > l_j$, i.e., a shorter codeword is assigned to a smaller probability.

2. Intuitively, by exchanging c_i and c_j, the expected length of the code should be improved.

3. Specifically, let

$$L = \sum_k p_k l_k = \sum_{k \neq i,j} p_k l_k + (p_i l_i + p_j l_j)$$

be the expected length of the code, and

$$L' = \sum_{k \neq i,j} p_k l_k + (p_i l_j + p_j l_i)$$

be the expected length of the code obtained by exchanging c_i and c_j.

4. Comparing L' and L, we see that

$$\begin{aligned} L' - L &= (p_i l_j + p_j l_i) - (p_i l_i + p_j l_j) \\ &= (p_i l_j - p_i l_i) - (p_j l_j - p_j l_i) \\ &= \underline{p_i}\,\underline{(l_j - l_i)} - \underline{p_j}\,\underline{(l_j - l_i)} \\ &= \underline{(p_i - p_j)}(l_j - l_i). \end{aligned}$$

Lemma 4.15 In an optimal code, shorter codewords are assigned to larger probabilities, i.e.,

$$l_1 \leq l_2 \leq \cdots \leq l_m. \tag{1}$$

Proof

1. Consider a probability distribution

$$\{p_1, \cdots, p_i, \cdots, p_j, \cdots p_m\}$$

such that $p_i > p_j$. Assume that in a particular code, the codewords c_i and c_j are such that $l_i > l_j$, i.e., a shorter codeword is assigned to a smaller probability.

2. Intuitively, by exchanging c_i and c_j, the expected length of the code should be improved.

3. Specifically, let

$$L = \sum_k p_k l_k = \sum_{k \neq i,j} p_k l_k + (p_i l_i + p_j l_j)$$

be the expected length of the code, and

$$L' = \sum_{k \neq i,j} p_k l_k + (p_i l_j + p_j l_i)$$

be the expected length of the code obtained by exchanging c_i and c_j.

4. Comparing L' and L, we see that

$$\begin{aligned} L' - L &= (p_i l_j + p_j l_i) - (p_i l_i + p_j l_j) \\ &= (p_i l_j - p_i l_i) - (p_j l_j - p_j l_i) \\ &= \underline{p_i}\,\underline{(l_j - l_i)} - \underline{p_j}\,\underline{(l_j - l_i)} \\ &= \underline{(p_i - p_j)}\,\underline{(l_j - l_i)}. \end{aligned}$$

Lemma 4.15 In an optimal code, shorter codewords are assigned to larger probabilities, i.e.,

$$l_1 \le l_2 \le \cdots \le l_m. \tag{1}$$

Proof

1. Consider a probability distribution

$$\{p_1, \cdots, p_i, \cdots, p_j, \cdots p_m\}$$

such that $p_i > p_j$. Assume that in a particular code, the codewords c_i and c_j are such that $l_i > l_j$, i.e., a shorter codeword is assigned to a smaller probability.

2. Intuitively, by exchanging c_i and c_j, the expected length of the code should be improved.

3. Specifically, let

$$L = \sum_k p_k l_k = \sum_{k \ne i,j} p_k l_k + (p_i l_i + p_j l_j)$$

be the expected length of the code, and

$$L' = \sum_{k \ne i,j} p_k l_k + (p_i l_j + p_j l_i)$$

be the expected length of the code obtained by exchanging c_i and c_j.

4. Comparing L' and L, we see that

$$\begin{aligned} L' - L &= (p_i l_j + p_j l_i) - (p_i l_i + p_j l_j) \\ &= (p_i l_j - p_i l_i) - (p_j l_j - p_j l_i) \\ &= p_i(l_j - l_i) - p_j(l_j - l_i) \\ &= (p_i - p_j)(l_j - l_i). \end{aligned}$$

This is negative because $p_i > p_j$ and $l_i > l_j$. Therefore, $L' < L$.

Lemma 4.15 In an optimal code, shorter codewords are assigned to larger probabilities, i.e.,

$$l_1 \leq l_2 \leq \cdots \leq l_m. \tag{1}$$

Proof

1. Consider a probability distribution

$$\{p_1, \cdots, p_i, \cdots, p_j, \cdots p_m\}$$

such that $p_i > p_j$. Assume that in a particular code, the codewords c_i and c_j are such that $l_i > l_j$, i.e., a shorter codeword is assigned to a smaller probability.

2. Intuitively, by exchanging c_i and c_j, the expected length of the code should be improved.

3. Specifically, let

$$L = \sum_k p_k l_k = \sum_{k \neq i,j} p_k l_k + (p_i l_i + p_j l_j)$$

be the expected length of the code, and

$$L' = \sum_{k \neq i,j} p_k l_k + (p_i l_j + p_j l_i)$$

be the expected length of the code obtained by exchanging c_i and c_j.

4. Comparing L' and L, we see that

$$\begin{aligned} L' - L &= (p_i l_j + p_j l_i) - (p_i l_i + p_j l_j) \\ &= (p_i l_j - p_i l_i) - (p_j l_j - p_j l_i) \\ &= p_i (l_j - l_i) - p_j (l_j - l_i) \\ &= \underbrace{(p_i - p_j)}_{>0}(l_j - l_i). \end{aligned}$$

This is negative because $p_i > p_j$ and $l_i > l_j$. Therefore, $L' < L$.

Lemma 4.15 In an optimal code, shorter codewords are assigned to larger probabilities, i.e.,

$$l_1 \le l_2 \le \cdots \le l_m. \tag{1}$$

Proof

1. Consider a probability distribution

$$\{p_1, \cdots, p_i, \cdots, p_j, \cdots p_m\}$$

such that $p_i > p_j$. Assume that in a particular code, the codewords c_i and c_j are such that $l_i > l_j$, i.e., a shorter codeword is assigned to a smaller probability.

2. Intuitively, by exchanging c_i and c_j, the expected length of the code should be improved.

3. Specifically, let

$$L = \sum_k p_k l_k = \sum_{k \ne i,j} p_k l_k + (p_i l_i + p_j l_j)$$

be the expected length of the code, and

$$L' = \sum_{k \ne i,j} p_k l_k + (p_i l_j + p_j l_i)$$

be the expected length of the code obtained by exchanging c_i and c_j.

4. Comparing L' and L, we see that

$$\begin{aligned} L' - L &= (p_i l_j + p_j l_i) - (p_i l_i + p_j l_j) \\ &= (p_i l_j - p_i l_i) - (p_j l_j - p_j l_i) \\ &= p_i (l_j - l_i) - p_j (l_j - l_i) \\ &= \underbrace{(p_i - p_j)}_{>0} \underbrace{(l_j - l_i)}_{<0}. \end{aligned}$$

This is negative because $p_i > p_j$ and $l_i > l_j$. Therefore, $L' < L$.

Lemma 4.15 In an optimal code, shorter codewords are assigned to larger probabilities, i.e.,

$$l_1 \leq l_2 \leq \cdots \leq l_m. \tag{1}$$

Proof

1. Consider a probability distribution

$$\{p_1, \cdots, p_i, \cdots, p_j, \cdots p_m\}$$

such that $p_i > p_j$. Assume that in a particular code, the codewords c_i and c_j are such that $l_i > l_j$, i.e., a shorter codeword is assigned to a smaller probability.

2. Intuitively, by exchanging c_i and c_j, the expected length of the code should be improved.

3. Specifically, let

$$L = \sum_k p_k l_k = \sum_{k \neq i,j} p_k l_k + (p_i l_i + p_j l_j)$$

be the expected length of the code, and

$$L' = \sum_{k \neq i,j} p_k l_k + (p_i l_j + p_j l_i)$$

be the expected length of the code obtained by exchanging c_i and c_j.

4. Comparing L' and L, we see that

$$\begin{aligned} L' - L &= (p_i l_j + p_j l_i) - (p_i l_i + p_j l_j) \\ &= (p_i l_j - p_i l_i) - (p_j l_j - p_j l_i) \\ &= p_i (l_j - l_i) - p_j (l_j - l_i) \\ &= (p_i - p_j)(l_j - l_i). \end{aligned}$$

This is negative because $p_i > p_j$ and $l_i > l_j$. Therefore, $L' < L$.

5. Since the original code can be improved, it is not an optimal code.

Lemma 4.15 In an optimal code, shorter codewords are assigned to larger probabilities, i.e.,

$$l_1 \leq l_2 \leq \cdots \leq l_m. \tag{1}$$

Proof

1. Consider a probability distribution

$$\{p_1, \cdots, p_i, \cdots, p_j, \cdots p_m\}$$

such that $p_i > p_j$. Assume that in a particular code, the codewords c_i and c_j are such that $l_i > l_j$, i.e., a shorter codeword is assigned to a smaller probability.

2. Intuitively, by exchanging c_i and c_j, the expected length of the code should be improved.

3. Specifically, let

$$L = \sum_k p_k l_k = \sum_{k \neq i,j} p_k l_k + (p_i l_i + p_j l_j)$$

be the expected length of the code, and

$$L' = \sum_{k \neq i,j} p_k l_k + (p_i l_j + p_j l_i)$$

be the expected length of the code obtained by exchanging c_i and c_j.

4. Comparing L' and L, we see that

$$\begin{aligned} L' - L &= (p_i l_j + p_j l_i) - (p_i l_i + p_j l_j) \\ &= (p_i l_j - p_i l_i) - (p_j l_j - p_j l_i) \\ &= p_i (l_j - l_i) - p_j (l_j - l_i) \\ &= (p_i - p_j)(l_j - l_i). \end{aligned}$$

This is negative because $p_i > p_j$ and $l_i > l_j$. Therefore, $L' < L$.

5. Since the original code can be improved, it is not an optimal code.

6. Therefore, for an optimal code, shorter codewords are assigned to larger probabilities, i.e., (1). The lemma is proved.

Lemma 4.15 In an optimal code, shorter codewords are assigned to larger probabilities, i.e.,

$$l_1 \leq l_2 \leq \cdots \leq l_m. \tag{1}$$

Proof

1. Consider a probability distribution

$$\{p_1, \cdots, p_i, \cdots, p_j, \cdots p_m\}$$

such that $p_i > p_j$. Assume that in a particular code, the codewords c_i and c_j are such that $l_i > l_j$, i.e., a shorter codeword is assigned to a smaller probability.

2. Intuitively, by exchanging c_i and c_j, the expected length of the code should be improved.

3. Specifically, let

$$L = \sum_k p_k l_k = \sum_{k \neq i,j} p_k l_k + (p_i l_i + p_j l_j)$$

be the expected length of the code, and

$$L' = \sum_{k \neq i,j} p_k l_k + (p_i l_j + p_j l_i)$$

be the expected length of the code obtained by exchanging c_i and c_j.

4. Comparing L' and L, we see that

$$\begin{aligned} L' - L &= (p_i l_j + p_j l_i) - (p_i l_i + p_j l_j) \\ &= (p_i l_j - p_i l_i) - (p_j l_j - p_j l_i) \\ &= p_i (l_j - l_i) - p_j (l_j - l_i) \\ &= (p_i - p_j)(l_j - l_i). \end{aligned}$$

This is negative because $p_i > p_j$ and $l_i > l_j$. Therefore, $L' < L$.

5. Since the original code can be improved, it is not an optimal code.

6. Therefore, for an optimal code, shorter codewords are assigned to larger probabilities, i.e., (1). The lemma is proved.

Lemma 4.16 There exists an optimal code in which the codewords assigned to the two smallest probabilities are siblings, i.e., the two codewords have the same length and they differ only in the last symbol.

Proof

Lemma 4.16 There exists an optimal code in which the codewords assigned to the two smallest probabilities are siblings, i.e., the two codewords have the same length and they differ only in the last symbol.

Proof

1. Consider any optimal code. From Lemma 4.15, the codeword c_m has the longest length. Then the sibling of c_m cannot be the prefix of another codeword.

Lemma 4.16 There exists an optimal code in which the codewords assigned to the two smallest probabilities are siblings, i.e., the two codewords have the same length and they differ only in the last symbol.

Proof

1. Consider any optimal code. From Lemma 4.15, the codeword c_m has the longest length. Then the sibling of c_m cannot be the prefix of another codeword.

Lemma 4.15 In an optimal code,

$$l_1 \leq l_2 \leq \cdots \leq l_m.$$

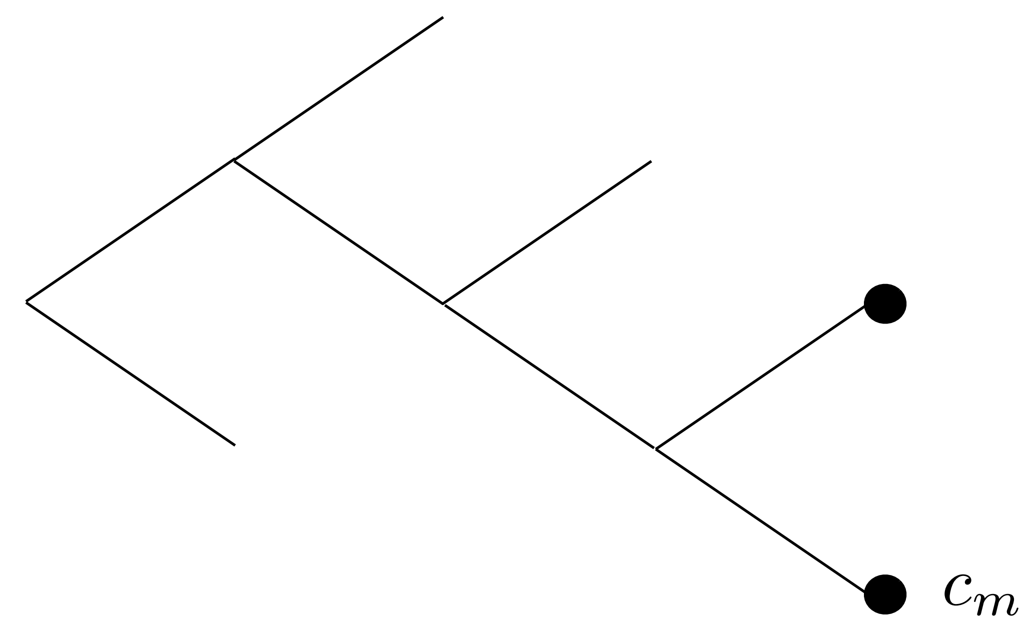
Lemma 4.16 There exists an optimal code in which the codewords assigned to the two smallest probabilities are siblings, i.e., the two codewords have the same length and they differ only in the last symbol.

Proof

1. Consider any optimal code. From Lemma 4.15, the codeword c_m has the longest length. Then the sibling of c_m cannot be the prefix of another codeword.

Lemma 4.15 In an optimal code,

$$l_1 \leq l_2 \leq \cdots \leq l_m.$$



Lemma 4.16 There exists an optimal code in which the codewords assigned to the two smallest probabilities are siblings, i.e., the two codewords have the same length and they differ only in the last symbol.

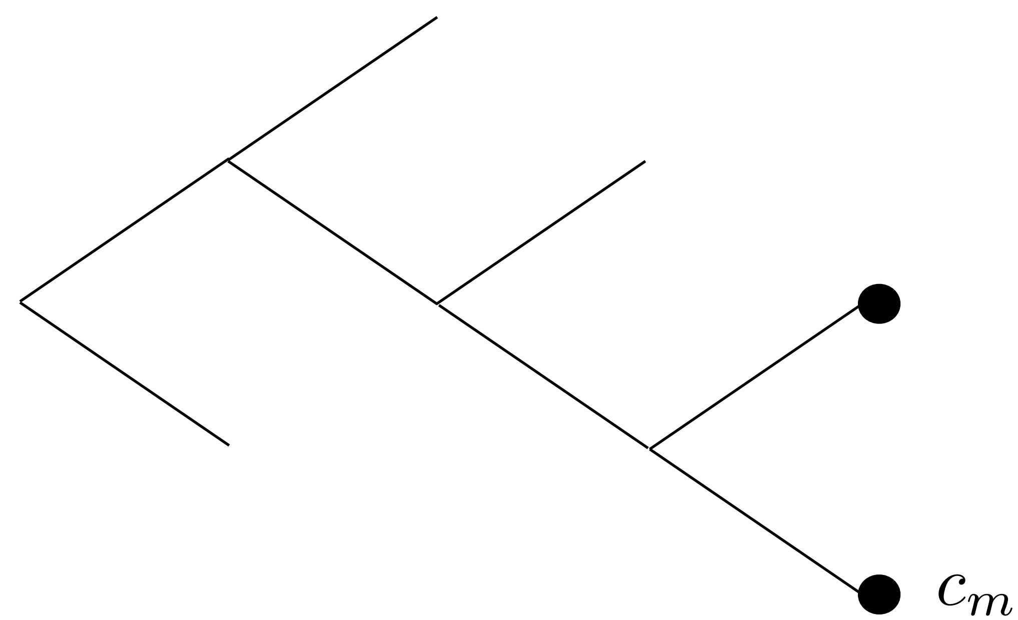
Proof

1. Consider any optimal code. From Lemma 4.15, the codeword c_m has the longest length. Then the sibling of c_m cannot be the prefix of another codeword.

2. We claim that the sibling of c_m must be a codeword. To see this, assume that it is not a codeword (and it is not the prefix of another codeword). Then we can replace c_m by its parent to improve the code because the length of the codeword assigned to p_m is reduced by 1, while all the other codewords remain unchanged.

Lemma 4.15 In an optimal code,

$$l_1 \le l_2 \le \cdots \le l_m.$$



Lemma 4.16 There exists an optimal code in which the codewords assigned to the two smallest probabilities are siblings, i.e., the two codewords have the same length and they differ only in the last symbol.

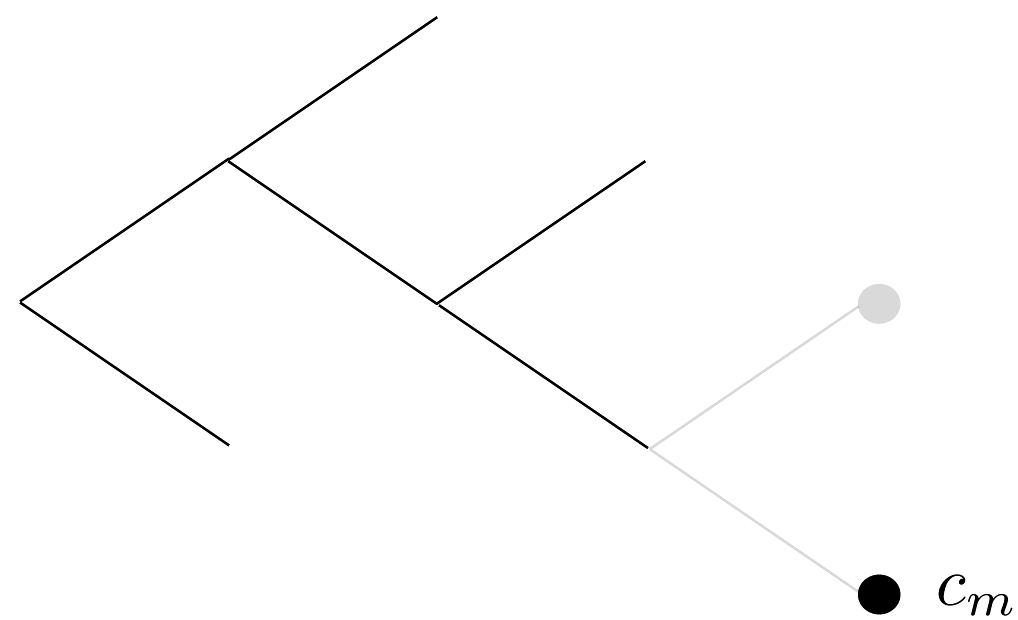
Proof

1. Consider any optimal code. From Lemma 4.15, the codeword c_m has the longest length. Then the sibling of c_m cannot be the prefix of another codeword.

2. We claim that the sibling of c_m must be a codeword. To see this, assume that it is not a codeword (and it is not the prefix of another codeword). Then we can replace c_m by its parent to improve the code because the length of the codeword assigned to p_m is reduced by 1, while all the other codewords remain unchanged.

Lemma 4.15 In an optimal code,

$$l_1 \leq l_2 \leq \cdots \leq l_m.$$



Lemma 4.16 There exists an optimal code in which the codewords assigned to the two smallest probabilities are siblings, i.e., the two codewords have the same length and they differ only in the last symbol.

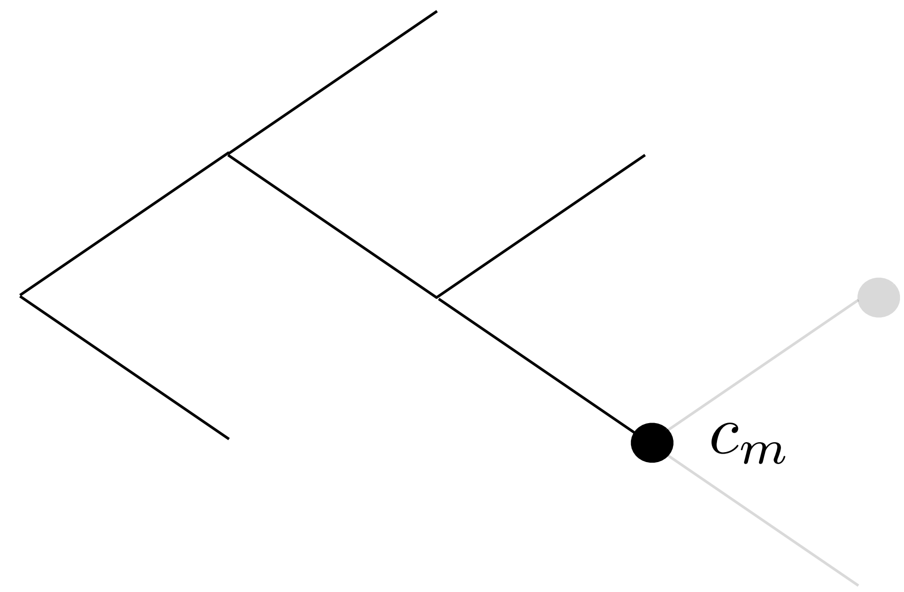
Proof

1. Consider any optimal code. From Lemma 4.15, the codeword c_m has the longest length. Then the sibling of c_m cannot be the prefix of another codeword.

2. We claim that the sibling of c_m must be a codeword. To see this, assume that it is not a codeword (and it is not the prefix of another codeword). Then we can replace c_m by its parent to improve the code because the length of the codeword assigned to p_m is reduced by 1, while all the other codewords remain unchanged.

Lemma 4.15 In an optimal code,

$$l_1 \leq l_2 \leq \cdots \leq l_m.$$



Lemma 4.16 There exists an optimal code in which the codewords assigned to the two smallest probabilities are siblings, i.e., the two codewords have the same length and they differ only in the last symbol.

Proof

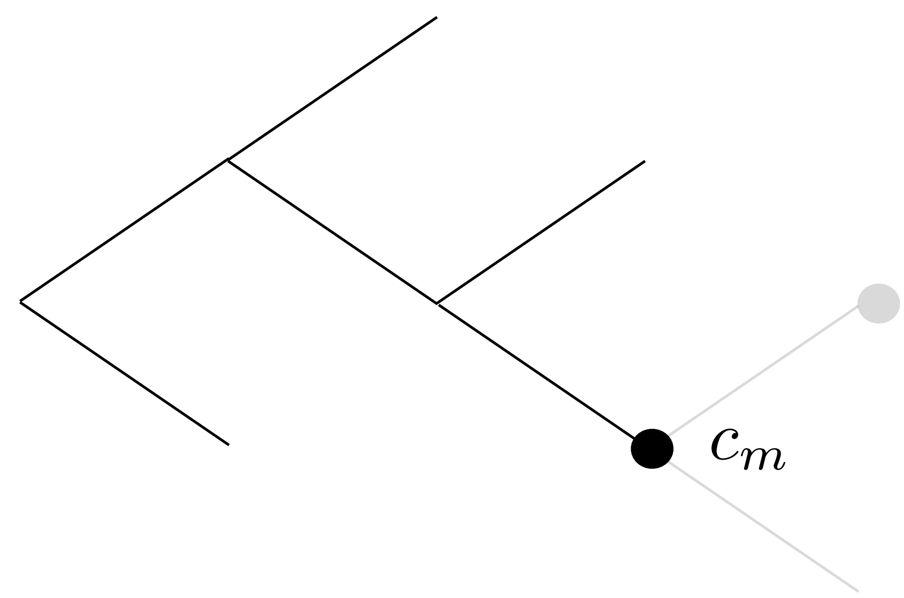
1. Consider any optimal code. From Lemma 4.15, the codeword c_m has the longest length. Then the sibling of c_m cannot be the prefix of another codeword.

2. We claim that the sibling of c_m must be a codeword. To see this, assume that it is not a codeword (and it is not the prefix of another codeword). Then we can replace c_m by its parent to improve the code because the length of the codeword assigned to p_m is reduced by 1, while all the other codewords remain unchanged.

3. This is a contradiction to the assumption that the code is optimal. Therefore, the sibling of c_m must be a codeword.

Lemma 4.15 In an optimal code,

$$l_1 \leq l_2 \leq \cdots \leq l_m.$$



Lemma 4.16 There exists an optimal code in which the codewords assigned to the two smallest probabilities are siblings, i.e., the two codewords have the same length and they differ only in the last symbol.

Lemma 4.15 In an optimal code,

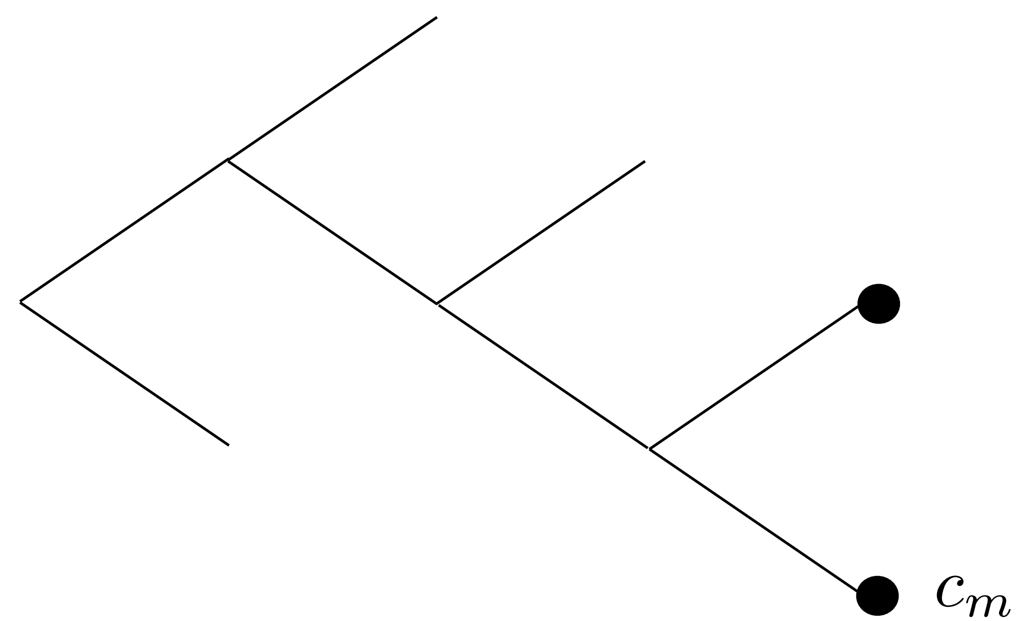
$$l_1 \leq l_2 \leq \cdots \leq l_m.$$

Proof

1. Consider any optimal code. From Lemma 4.15, the codeword c_m has the longest length. Then the sibling of c_m cannot be the prefix of another codeword.

2. We claim that the sibling of c_m must be a codeword. To see this, assume that it is not a codeword (and it is not the prefix of another codeword). Then we can replace c_m by its parent to improve the code because the length of the codeword assigned to p_m is reduced by 1, while all the other codewords remain unchanged.

3. This is a contradiction to the assumption that the code is optimal. Therefore, the sibling of c_m must be a codeword.



Lemma 4.16 There exists an optimal code in which the codewords assigned to the two smallest probabilities are siblings, i.e., the two codewords have the same length and they differ only in the last symbol.

Proof

1. Consider any optimal code. From Lemma 4.15, the codeword c_m has the longest length. Then the sibling of c_m cannot be the prefix of another codeword.

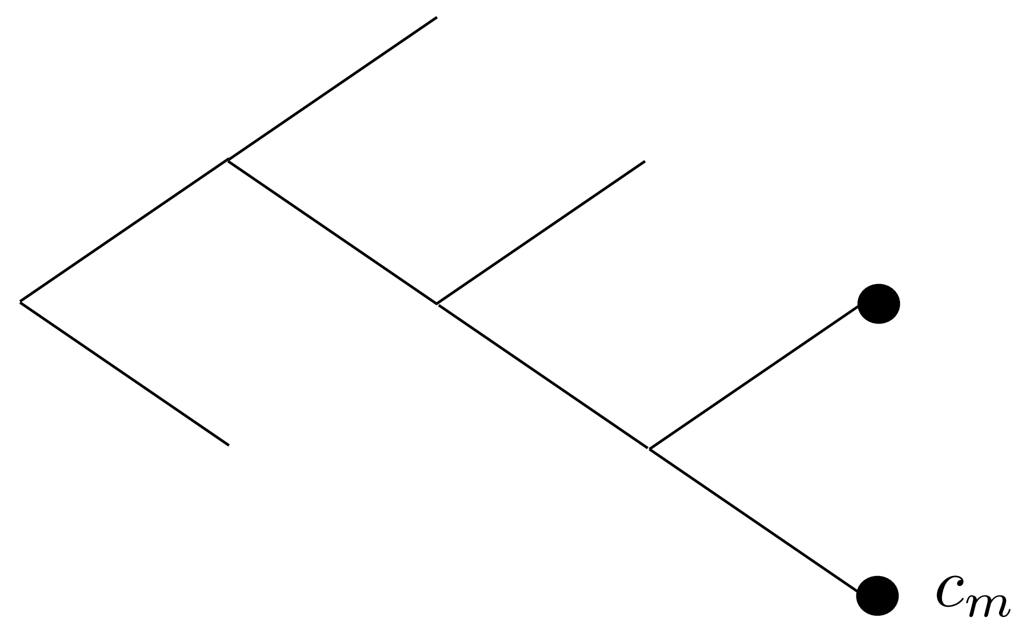
2. We claim that the sibling of c_m must be a codeword. To see this, assume that it is not a codeword (and it is not the prefix of another codeword). Then we can replace c_m by its parent to improve the code because the length of the codeword assigned to p_m is reduced by 1, while all the other codewords remain unchanged.

3. This is a contradiction to the assumption that the code is optimal. Therefore, the sibling of c_m must be a codeword.

4. If the sibling of c_m is c_{m-1}, then the code already has the desired property, i.e., the codewords assigned to the two smallest probabilities are siblings.

Lemma 4.15 In an optimal code,

$$l_1 \leq l_2 \leq \cdots \leq l_m.$$



Lemma 4.16 There exists an optimal code in which the codewords assigned to the two smallest probabilities are siblings, i.e., the two codewords have the same length and they differ only in the last symbol.

Proof

1. Consider any optimal code. From Lemma 4.15, the codeword c_m has the longest length. Then the sibling of c_m cannot be the prefix of another codeword.

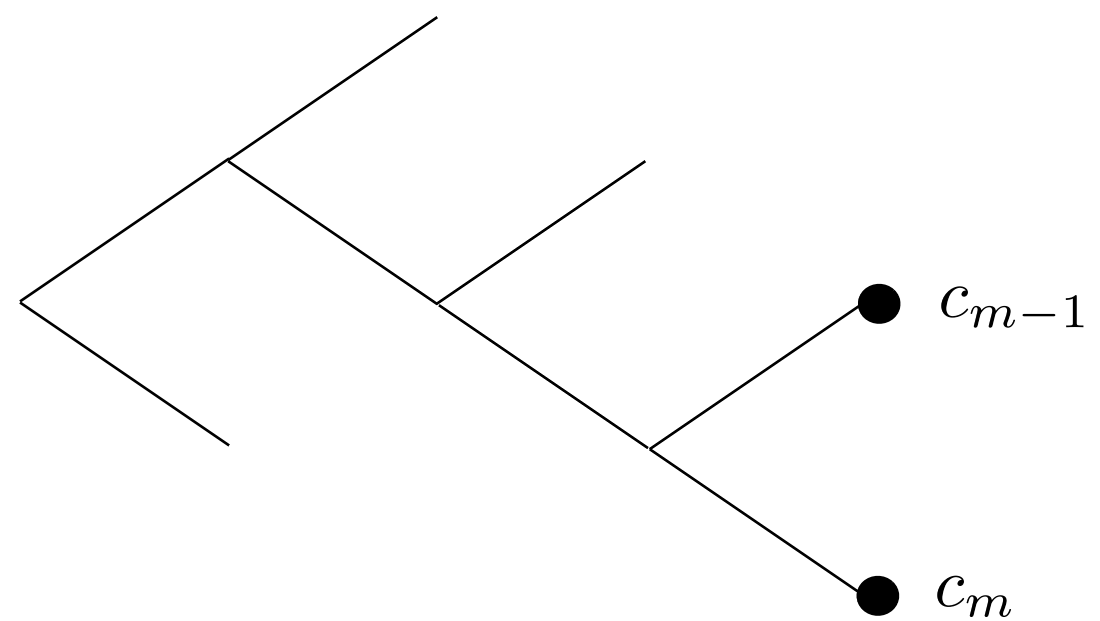
2. We claim that the sibling of c_m must be a codeword. To see this, assume that it is not a codeword (and it is not the prefix of another codeword). Then we can replace c_m by its parent to improve the code because the length of the codeword assigned to p_m is reduced by 1, while all the other codewords remain unchanged.

3. This is a contradiction to the assumption that the code is optimal. Therefore, the sibling of c_m must be a codeword.

4. If the sibling of c_m is c_{m-1}, then the code already has the desired property, i.e., the codewords assigned to the two smallest probabilities are siblings.

Lemma 4.15 In an optimal code,

$$l_1 \leq l_2 \leq \cdots \leq l_m.$$



Lemma 4.16 There exists an optimal code in which the codewords assigned to the two smallest probabilities are siblings, i.e., the two codewords have the same length and they differ only in the last symbol.

Proof

1. Consider any optimal code. From Lemma 4.15, the codeword c_m has the longest length. Then the sibling of c_m cannot be the prefix of another codeword.

2. We claim that the sibling of c_m must be a codeword. To see this, assume that it is not a codeword (and it is not the prefix of another codeword). Then we can replace c_m by its parent to improve the code because the length of the codeword assigned to p_m is reduced by 1, while all the other codewords remain unchanged.

3. This is a contradiction to the assumption that the code is optimal. Therefore, the sibling of c_m must be a codeword.

4. If the sibling of c_m is c_{m-1}, then the code already has the desired property, i.e., the codewords assigned to the two smallest probabilities are siblings.

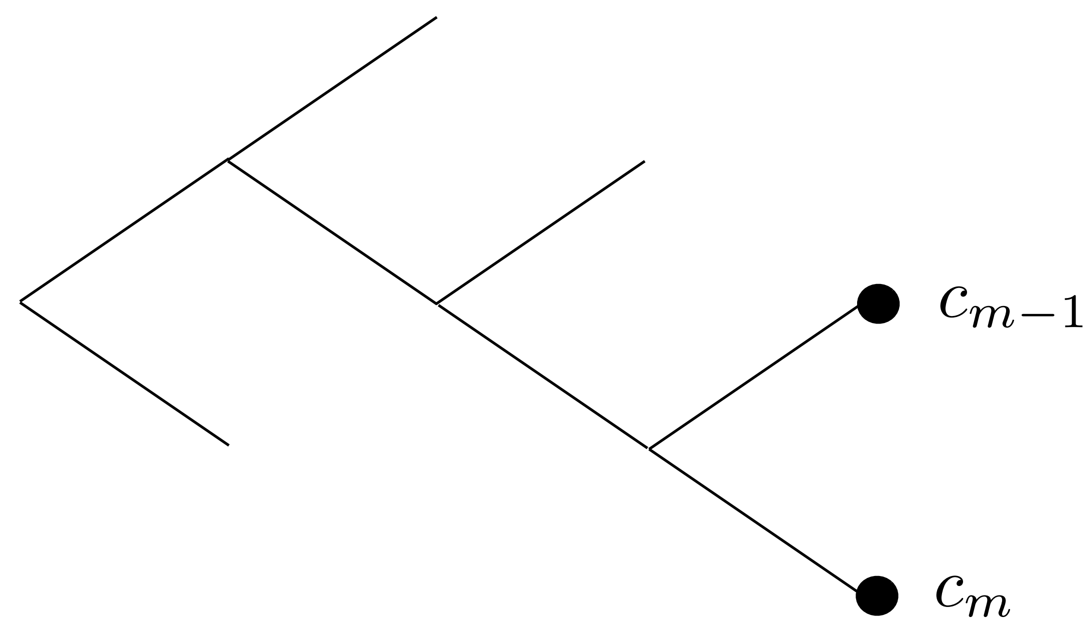
5. If not, assume that the sibling of c_m is c_i, where $i < m - 1$. Since the code is optimal, by Lemma 4.15,

$$l_i \leq l_{m-1} \leq l_m.$$

On the other hand, since c_i and c_m are sibling of each other, they have the same length and so $l_i = l_m$.

Lemma 4.15 In an optimal code,

$$l_1 \leq l_2 \leq \cdots \leq l_m.$$



Lemma 4.16 There exists an optimal code in which the codewords assigned to the two smallest probabilities are siblings, i.e., the two codewords have the same length and they differ only in the last symbol.

Proof

1. Consider any optimal code. From Lemma 4.15, the codeword c_m has the longest length. Then the sibling of c_m cannot be the prefix of another codeword.

2. We claim that the sibling of c_m must be a codeword. To see this, assume that it is not a codeword (and it is not the prefix of another codeword). Then we can replace c_m by its parent to improve the code because the length of the codeword assigned to p_m is reduced by 1, while all the other codewords remain unchanged.

3. This is a contradiction to the assumption that the code is optimal. Therefore, the sibling of c_m must be a codeword.

4. If the sibling of c_m is c_{m-1}, then the code already has the desired property, i.e., the codewords assigned to the two smallest probabilities are siblings.

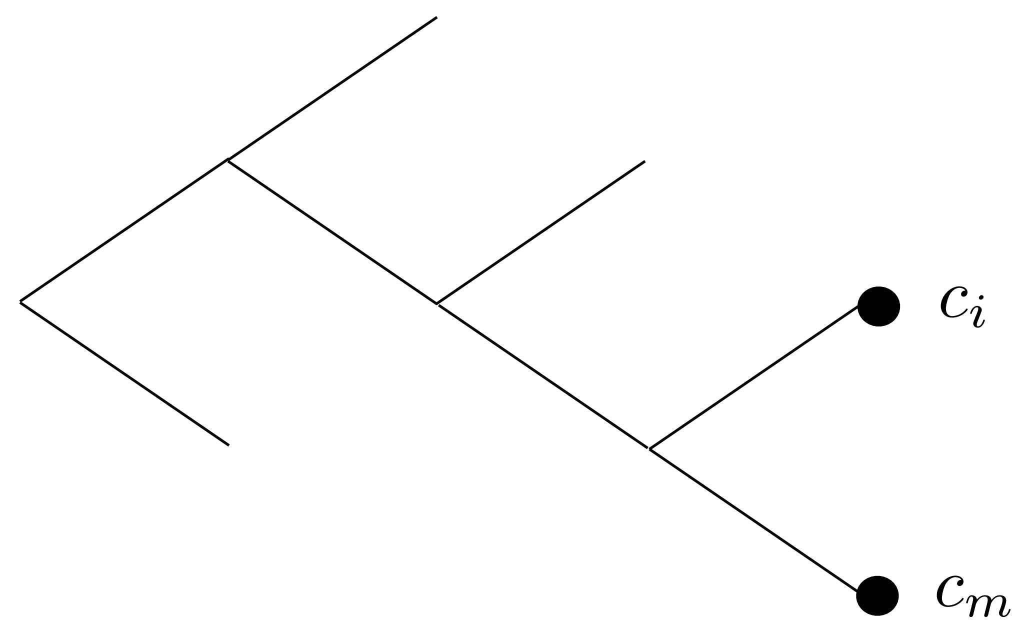
5. If not, assume that the sibling of c_m is c_i, where $i < m - 1$. Since the code is optimal, by Lemma 4.15,

$$l_i \leq l_{m-1} \leq l_m.$$

On the other hand, since c_i and c_m are sibling of each other, they have the same length and so $l_i = l_m$.

Lemma 4.15 In an optimal code,

$$l_1 \leq l_2 \leq \cdots \leq l_m.$$



Lemma 4.16 There exists an optimal code in which the codewords assigned to the two smallest probabilities are siblings, i.e., the two codewords have the same length and they differ only in the last symbol.

Proof

1. Consider any optimal code. From Lemma 4.15, the codeword c_m has the longest length. Then the sibling of c_m cannot be the prefix of another codeword.

2. We claim that the sibling of c_m must be a codeword. To see this, assume that it is not a codeword (and it is not the prefix of another codeword). Then we can replace c_m by its parent to improve the code because the length of the codeword assigned to p_m is reduced by 1, while all the other codewords remain unchanged.

3. This is a contradiction to the assumption that the code is optimal. Therefore, the sibling of c_m must be a codeword.

4. If the sibling of c_m is c_{m-1}, then the code already has the desired property, i.e., the codewords assigned to the two smallest probabilities are siblings.

5. If not, assume that the sibling of c_m is c_i, where $i < m-1$. Since the code is optimal, by Lemma 4.15,

$$l_i \leq l_{m-1} \leq l_m.$$

On the other hand, since c_i and c_m are sibling of each other, they have the same length and so $l_i = l_m$.

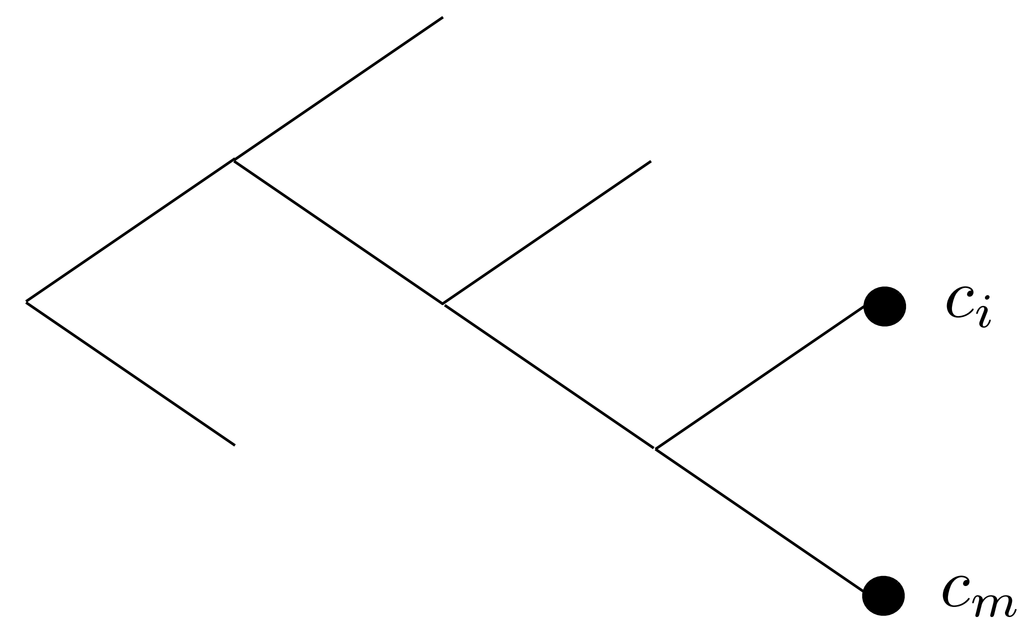
6. Therefore,

$$l_i = l_{m-1} = l_m,$$

i.e., c_i, c_{m-1}, and c_m all have the same order.

Lemma 4.15 In an optimal code,

$$l_1 \leq l_2 \leq \cdots \leq l_m.$$



Lemma 4.16 There exists an optimal code in which the codewords assigned to the two smallest probabilities are siblings, i.e., the two codewords have the same length and they differ only in the last symbol.

Lemma 4.15 In an optimal code,

$$l_1 \leq l_2 \leq \cdots \leq l_m.$$

Proof

1. Consider any optimal code. From Lemma 4.15, the codeword c_m has the longest length. Then the sibling of c_m cannot be the prefix of another codeword.

2. We claim that the sibling of c_m must be a codeword. To see this, assume that it is not a codeword (and it is not the prefix of another codeword). Then we can replace c_m by its parent to improve the code because the length of the codeword assigned to p_m is reduced by 1, while all the other codewords remain unchanged.

3. This is a contradiction to the assumption that the code is optimal. Therefore, the sibling of c_m must be a codeword.

4. If the sibling of c_m is c_{m-1}, then the code already has the desired property, i.e., the codewords assigned to the two smallest probabilities are siblings.

5. If not, assume that the sibling of c_m is c_i, where $i < m - 1$. Since the code is optimal, by Lemma 4.15,

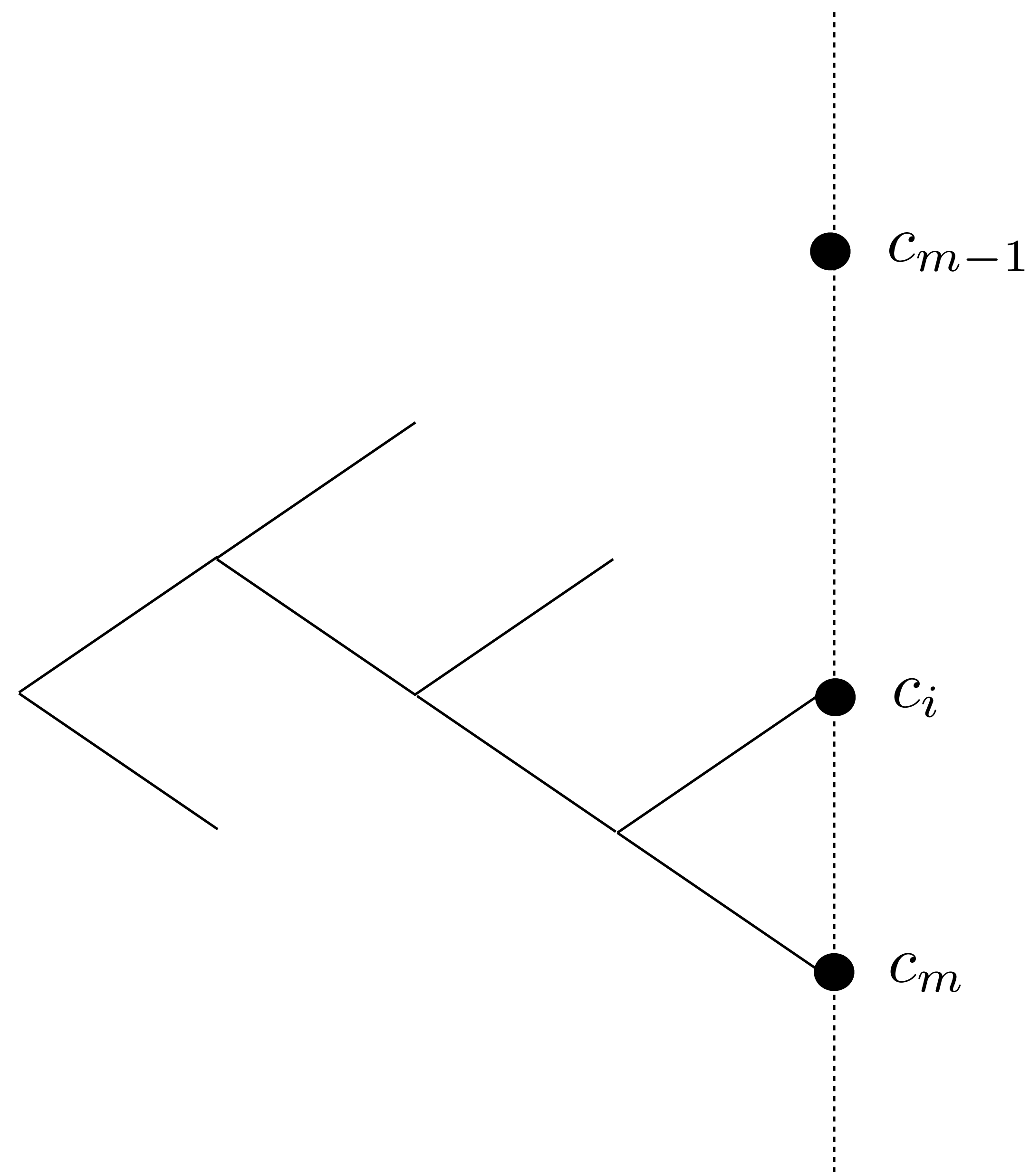
$$l_i \leq l_{m-1} \leq l_m.$$

On the other hand, since c_i and c_m are sibling of each other, they have the same length and so $l_i = l_m$.

6. Therefore,

$$l_i = l_{m-1} = l_m,$$

i.e., c_i, c_{m-1}, and c_m all have the same order.



Lemma 4.16 There exists an optimal code in which the codewords assigned to the two smallest probabilities are siblings, i.e., the two codewords have the same length and they differ only in the last symbol.

Proof

1. Consider any optimal code. From Lemma 4.15, the codeword c_m has the longest length. Then the sibling of c_m cannot be the prefix of another codeword.

2. We claim that the sibling of c_m must be a codeword. To see this, assume that it is not a codeword (and it is not the prefix of another codeword). Then we can replace c_m by its parent to improve the code because the length of the codeword assigned to p_m is reduced by 1, while all the other codewords remain unchanged.

3. This is a contradiction to the assumption that the code is optimal. Therefore, the sibling of c_m must be a codeword.

4. If the sibling of c_m is c_{m-1}, then the code already has the desired property, i.e., the codewords assigned to the two smallest probabilities are siblings.

5. If not, assume that the sibling of c_m is c_i, where $i < m - 1$. Since the code is optimal, by Lemma 4.15,

$$l_i \leq l_{m-1} \leq l_m.$$

On the other hand, since c_i and c_m are sibling of each other, they have the same length and so $l_i = l_m$.

6. Therefore,

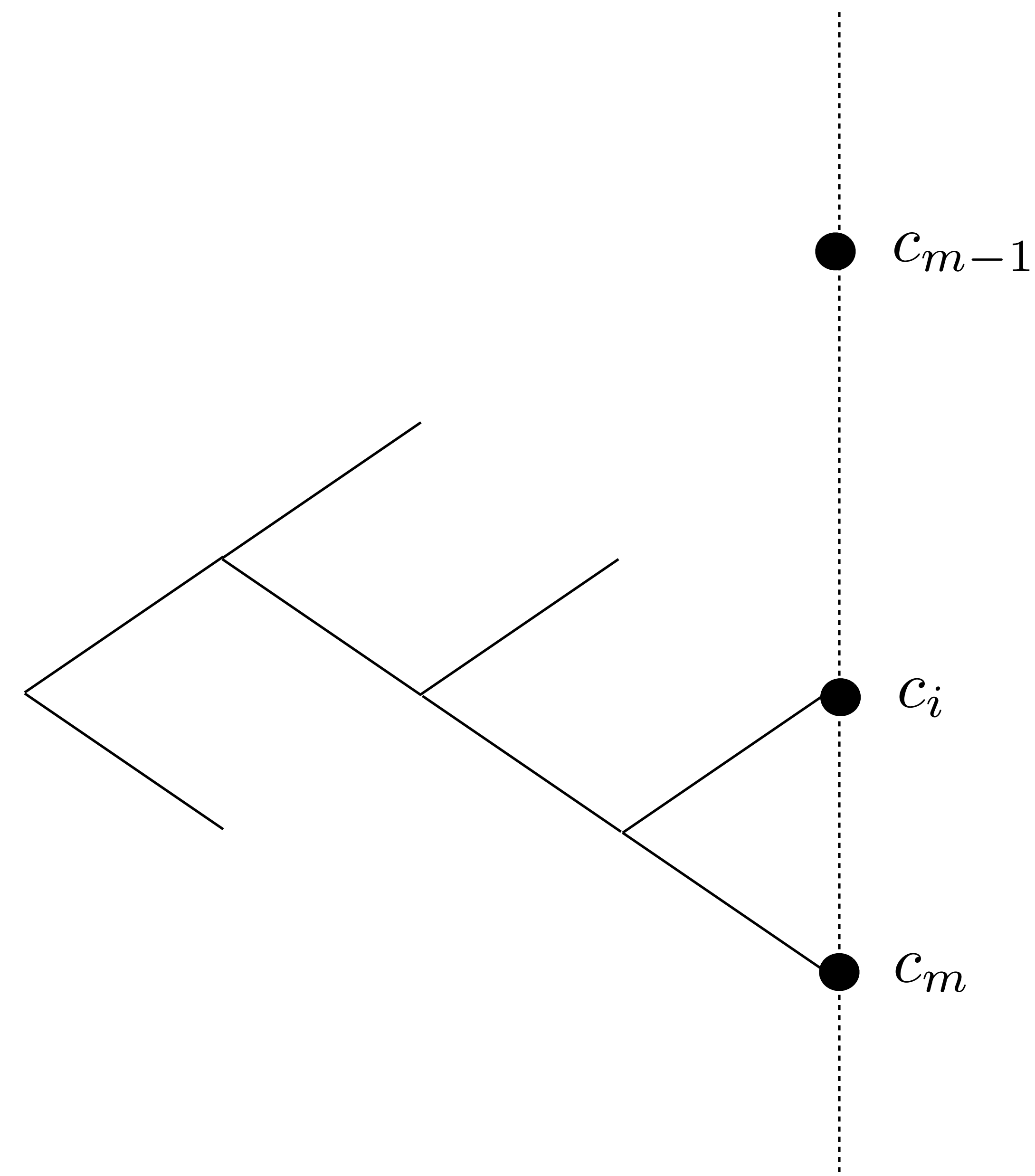
$$l_i = l_{m-1} = l_m,$$

i.e., c_i, c_{m-1}, and c_m all have the same order.

7. Then we can exchange the codewords c_i and c_{m-1} without changing the expected length of the code (i.e., the code remains optimal) to obtain the desired code. The lemma is proved.

Lemma 4.15 In an optimal code,

$$l_1 \leq l_2 \leq \cdots \leq l_m.$$



Lemma 4.15 In an optimal code,

$$l_1 \leq l_2 \leq \cdots \leq l_m.$$

Lemma 4.16 There exists an optimal code in which the codewords assigned to the two smallest probabilities are siblings, i.e., the two codewords have the same length and they differ only in the last symbol.

Proof

1. Consider any optimal code. From Lemma 4.15, the codeword c_m has the longest length. Then the sibling of c_m cannot be the prefix of another codeword.

2. We claim that the sibling of c_m must be a codeword. To see this, assume that it is not a codeword (and it is not the prefix of another codeword). Then we can replace c_m by its parent to improve the code because the length of the codeword assigned to p_m is reduced by 1, while all the other codewords remain unchanged.

3. This is a contradiction to the assumption that the code is optimal. Therefore, the sibling of c_m must be a codeword.

4. If the sibling of c_m is c_{m-1}, then the code already has the desired property, i.e., the codewords assigned to the two smallest probabilities are siblings.

5. If not, assume that the sibling of c_m is c_i, where $i < m - 1$. Since the code is optimal, by Lemma 4.15,

$$l_i \leq l_{m-1} \leq l_m.$$

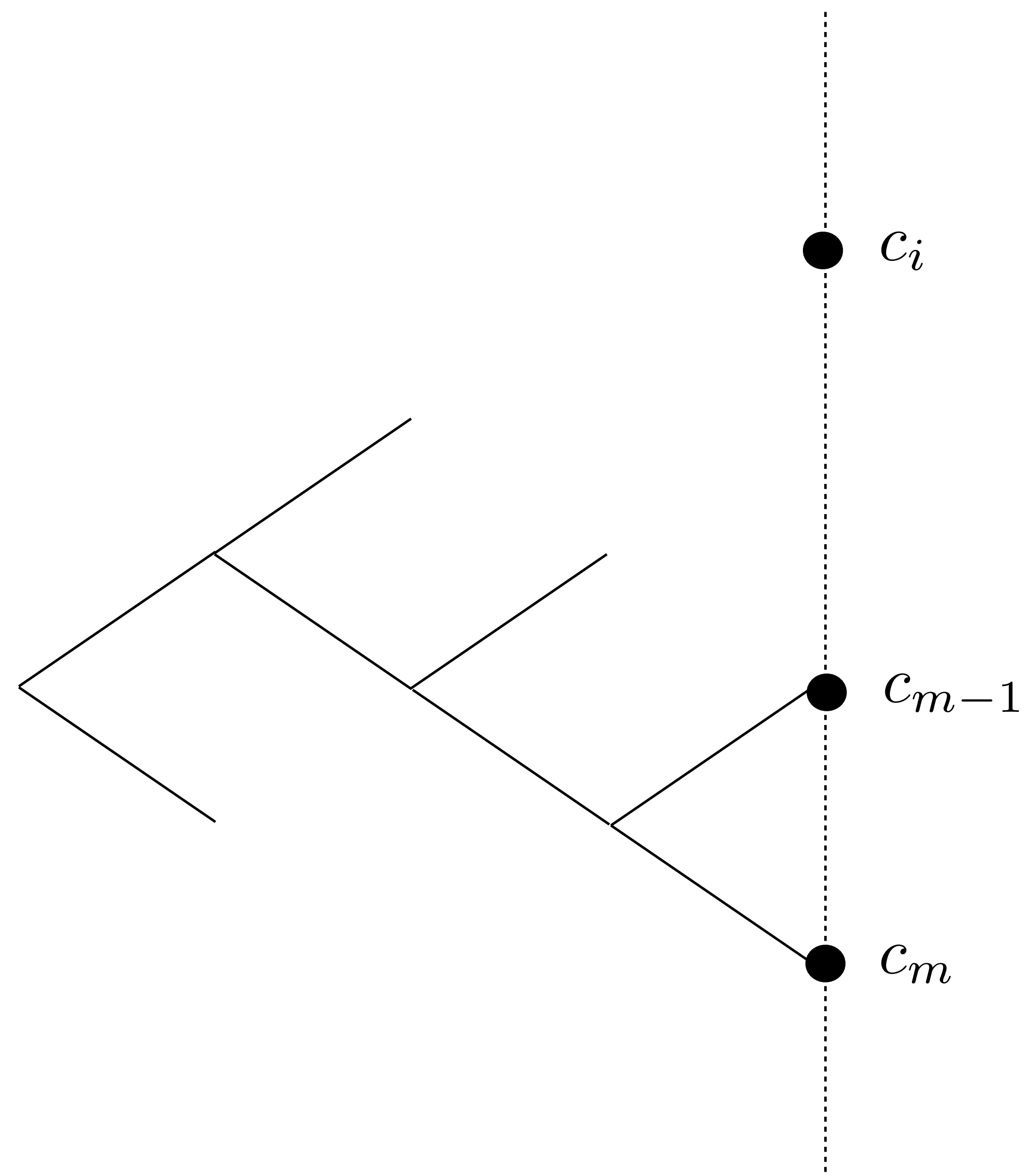
On the other hand, since c_i and c_m are sibling of each other, they have the same length and so $l_i = l_m$.

6. Therefore,

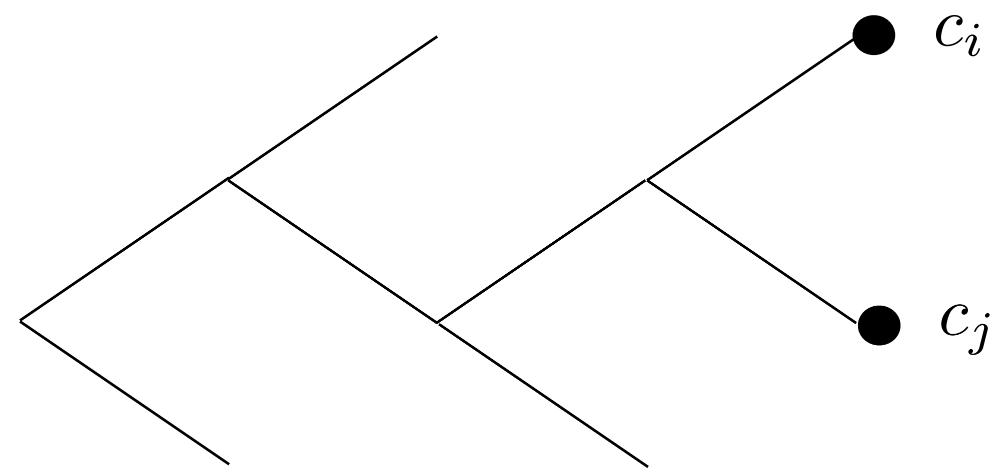
$$l_i = l_{m-1} = l_m,$$

i.e., c_i, c_{m-1}, and c_m all have the same order.

7. Then we can exchange the codewords c_i and c_{m-1} without changing the expected length of the code (i.e., the code remains optimal) to obtain the desired code. The lemma is proved.

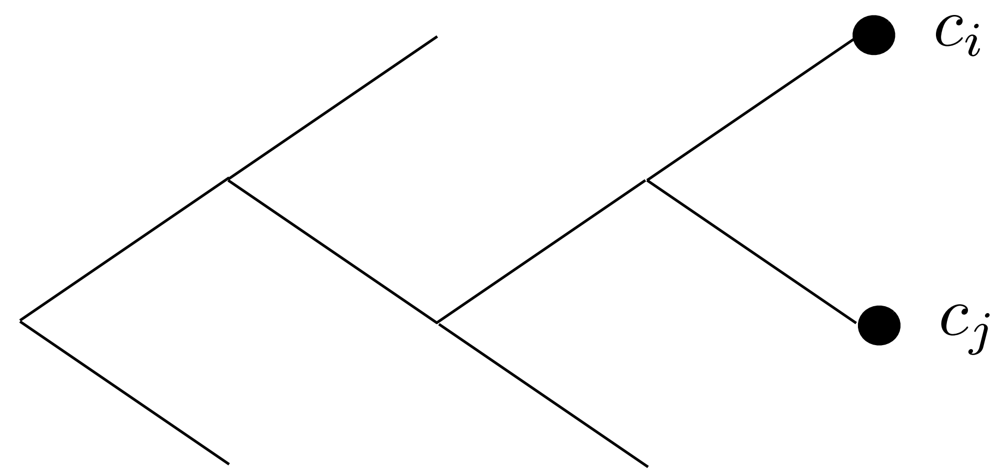


Reduced Code Tree



Reduced Code Tree

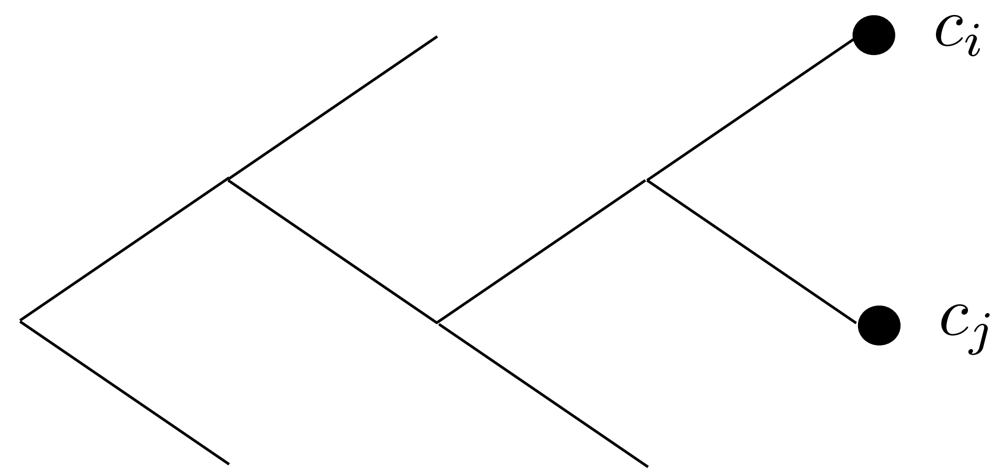
1. Suppose c_i and c_j are siblings in a code tree. Then $l_i = l_j$.



Reduced Code Tree

1. Suppose c_i and c_j are siblings in a code tree. Then $l_i = l_j$.

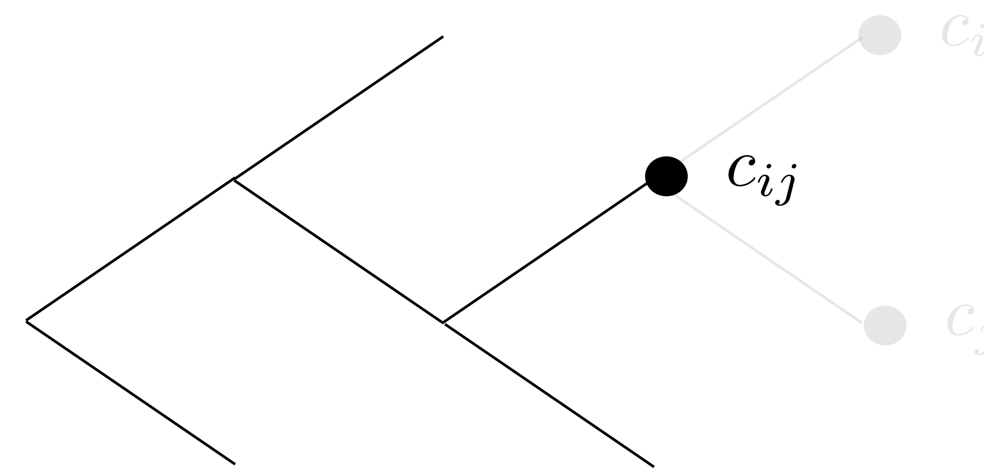
2. If we replace c_i and c_j by a common codeword at their parent, call it c_{ij}, then we obtain a reduced code tree, and the probability of c_{ij} is $p_i + p_j$.



Reduced Code Tree

1. Suppose c_i and c_j are siblings in a code tree. Then $l_i = l_j$.

2. If we replace c_i and c_j by a common codeword at their parent, call it c_{ij}, then we obtain a reduced code tree, and the probability of c_{ij} is $p_i + p_j$.

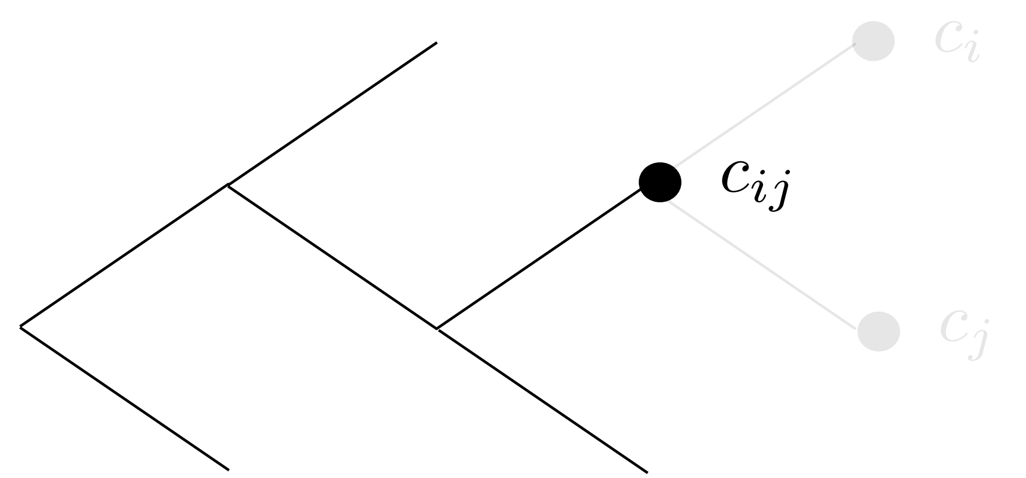


Reduced Code Tree

1. Suppose c_i and c_j are siblings in a code tree. Then $l_i = l_j$.

2. If we replace c_i and c_j by a common codeword at their parent, call it c_{ij}, then we obtain a reduced code tree, and the probability of c_{ij} is $p_i + p_j$.

3. Accordingly, the probability set becomes a reduced probability set with p_i and p_j replaced by a probability $p_i + p_j$.

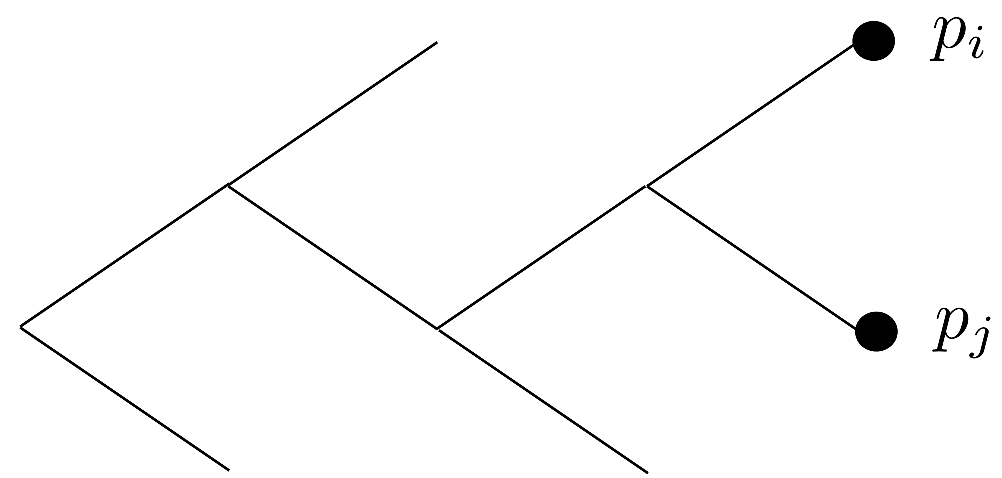


Reduced Code Tree

1. Suppose c_i and c_j are siblings in a code tree. Then $l_i = l_j$.

2. If we replace c_i and c_j by a common codeword at their parent, call it c_{ij}, then we obtain a reduced code tree, and the probability of c_{ij} is $p_i + p_j$.

3. Accordingly, the probability set becomes a reduced probability set with p_i and p_j replaced by a probability $p_i + p_j$.

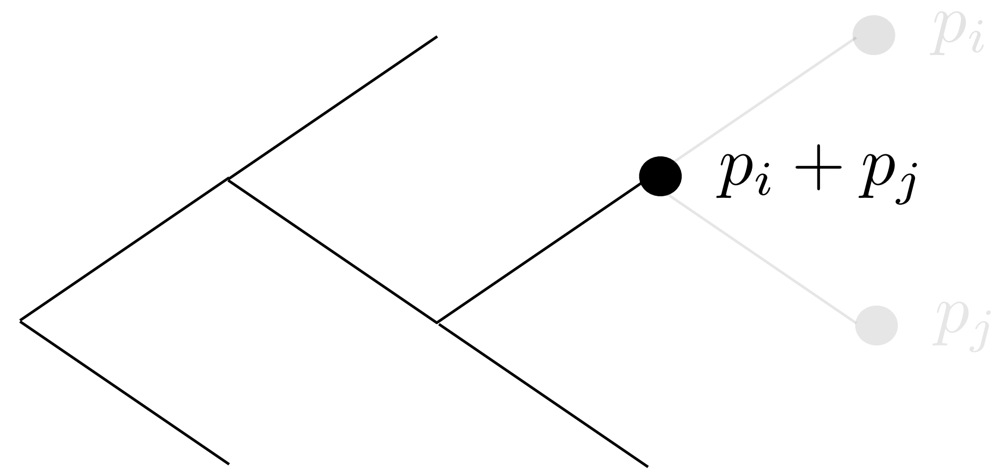


Reduced Code Tree

1. Suppose c_i and c_j are siblings in a code tree. Then $l_i = l_j$.

2. If we replace c_i and c_j by a common codeword at their parent, call it c_{ij}, then we obtain a reduced code tree, and the probability of c_{ij} is $p_i + p_j$.

3. Accordingly, the probability set becomes a reduced probability set with p_i and p_j replaced by a probability $p_i + p_j$.



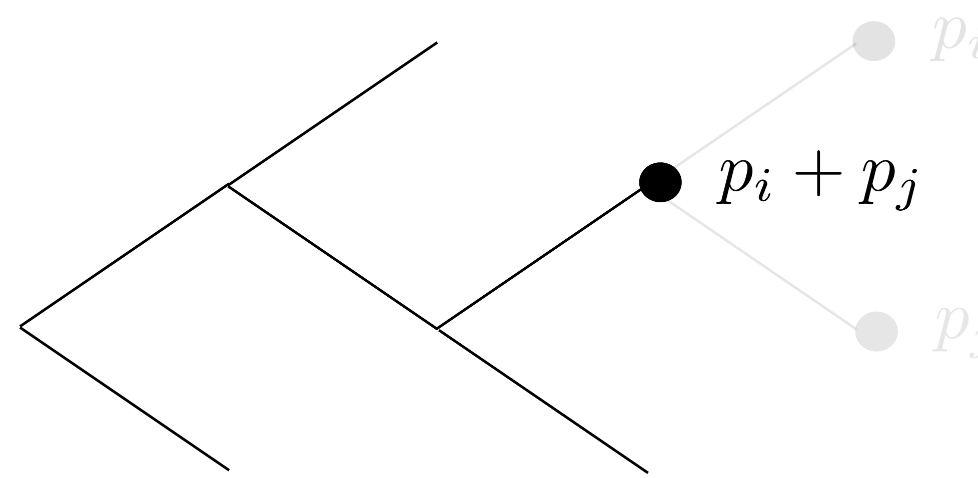
Reduced Code Tree

1. Suppose c_i and c_j are siblings in a code tree. Then $l_i = l_j$.

2. If we replace c_i and c_j by a common codeword at their parent, call it c_{ij}, then we obtain a reduced code tree, and the probability of c_{ij} is $p_i + p_j$.

3. Accordingly, the probability set becomes a reduced probability set with p_i and p_j replaced by a probability $p_i + p_j$.

4. Let L and L' be the expected lengths of the original code and the reduced code, respectively. Then



Reduced Code Tree

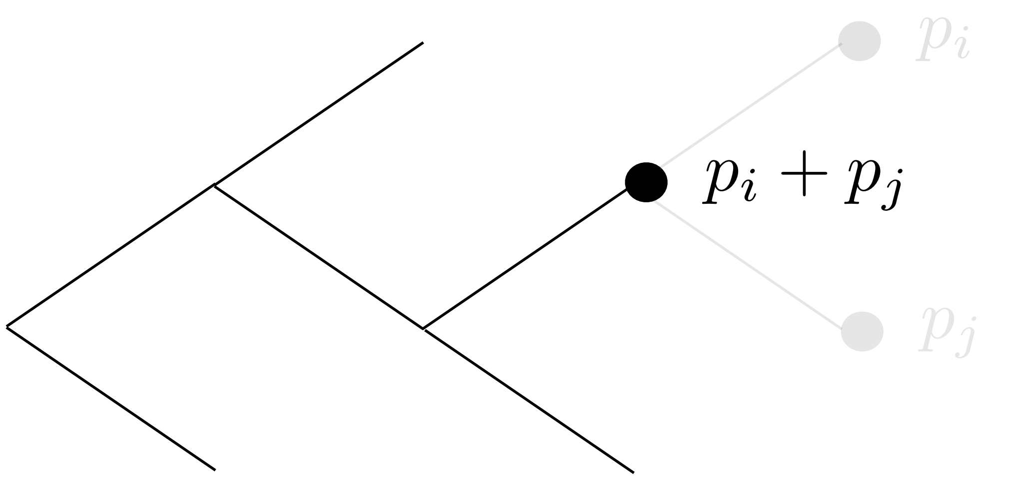
1. Suppose c_i and c_j are siblings in a code tree. Then $l_i = l_j$.

2. If we replace c_i and c_j by a common codeword at their parent, call it c_{ij}, then we obtain a reduced code tree, and the probability of c_{ij} is $p_i + p_j$.

3. Accordingly, the probability set becomes a reduced probability set with p_i and p_j replaced by a probability $p_i + p_j$.

4. Let L and L' be the expected lengths of the original code and the reduced code, respectively. Then

$$L - L' = (p_i l_i + p_j l_j) - (p_i + p_j)(l_i - 1)$$



Reduced Code Tree

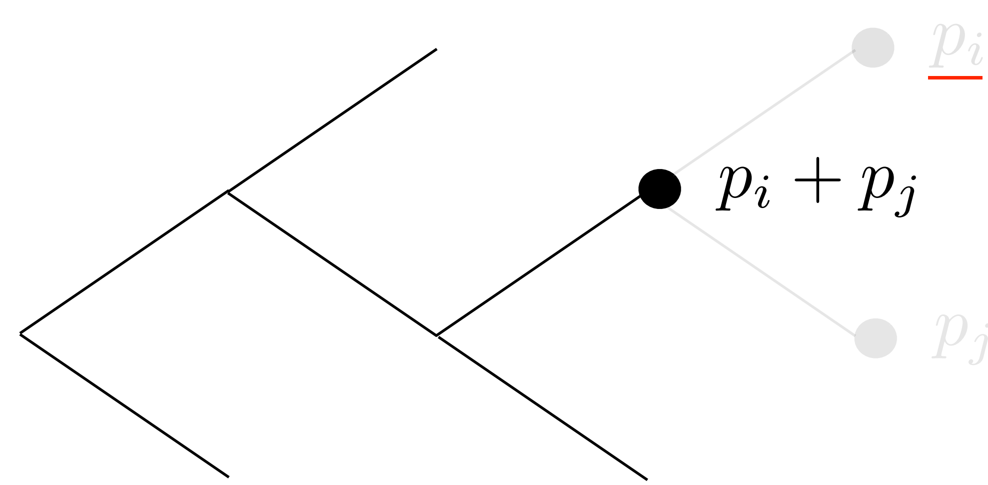
1. Suppose c_i and c_j are siblings in a code tree. Then $l_i = l_j$.

2. If we replace c_i and c_j by a common codeword at their parent, call it c_{ij}, then we obtain a reduced code tree, and the probability of c_{ij} is $p_i + p_j$.

3. Accordingly, the probability set becomes a reduced probability set with p_i and p_j replaced by a probability $p_i + p_j$.

4. Let L and L' be the expected lengths of the original code and the reduced code, respectively. Then

$$L - L' = (p_i l_i + p_j l_j) - (p_i + p_j)(l_i - 1)$$



Reduced Code Tree

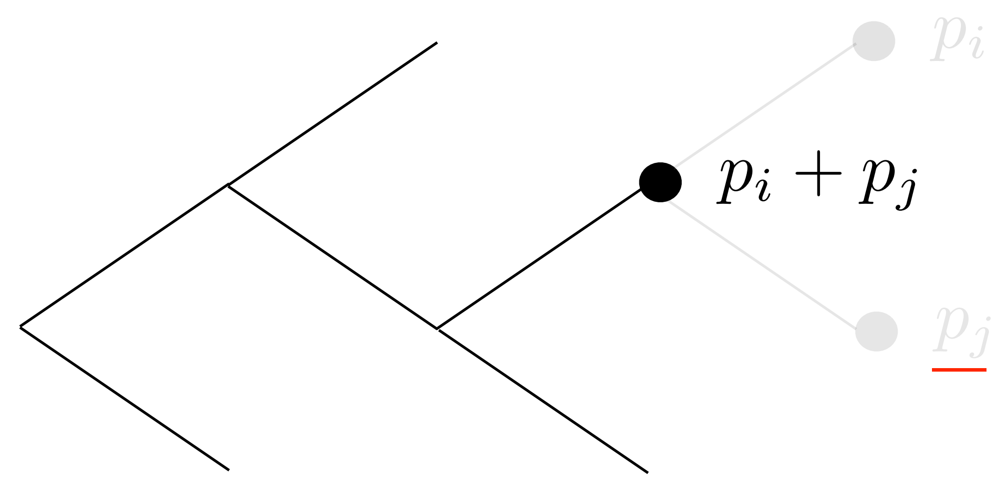
1. Suppose c_i and c_j are siblings in a code tree. Then $l_i = l_j$.

2. If we replace c_i and c_j by a common codeword at their parent, call it c_{ij}, then we obtain a reduced code tree, and the probability of c_{ij} is $p_i + p_j$.

3. Accordingly, the probability set becomes a reduced probability set with p_i and p_j replaced by a probability $p_i + p_j$.

4. Let L and L' be the expected lengths of the original code and the reduced code, respectively. Then

$$L - L' = (p_i l_i + p_j l_j) - (p_i + p_j)(l_i - 1)$$



Reduced Code Tree

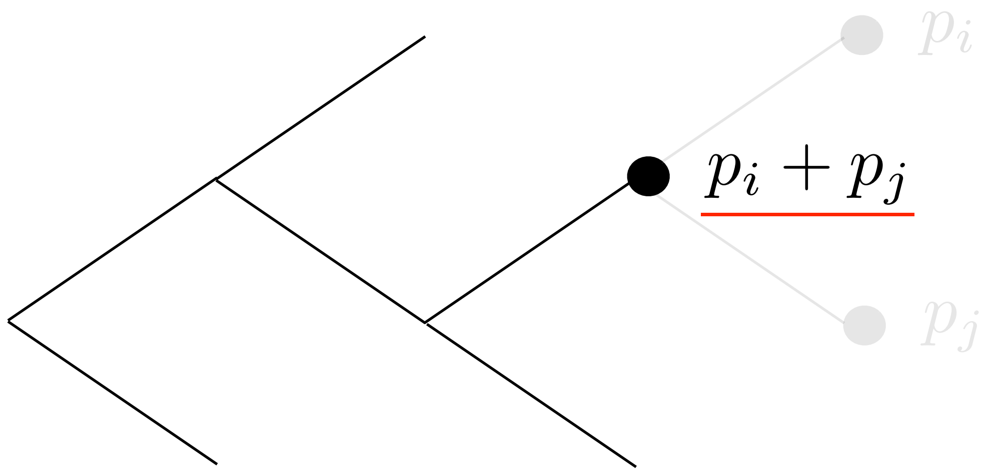
1. Suppose c_i and c_j are siblings in a code tree. Then $l_i = l_j$.

2. If we replace c_i and c_j by a common codeword at their parent, call it c_{ij}, then we obtain a reduced code tree, and the probability of c_{ij} is $p_i + p_j$.

3. Accordingly, the probability set becomes a reduced probability set with p_i and p_j replaced by a probability $p_i + p_j$.

4. Let L and L' be the expected lengths of the original code and the reduced code, respectively. Then

$$L - L' = (p_i l_i + p_j l_j) - (\underline{p_i + p_j})(l_i - 1)$$



Reduced Code Tree

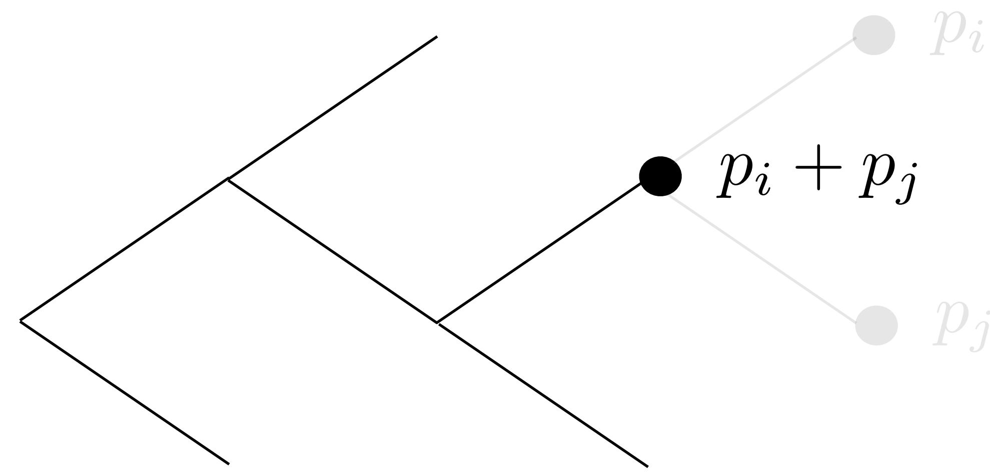
1. Suppose c_i and c_j are siblings in a code tree. Then $l_i = l_j$.

2. If we replace c_i and c_j by a common codeword at their parent, call it c_{ij}, then we obtain a reduced code tree, and the probability of c_{ij} is $p_i + p_j$.

3. Accordingly, the probability set becomes a reduced probability set with p_i and p_j replaced by a probability $p_i + p_j$.

4. Let L and L' be the expected lengths of the original code and the reduced code, respectively. Then

$$L - L' = (p_i l_i + p_j \underline{l_j}) - (p_i + p_j)(l_i - 1)$$



Reduced Code Tree

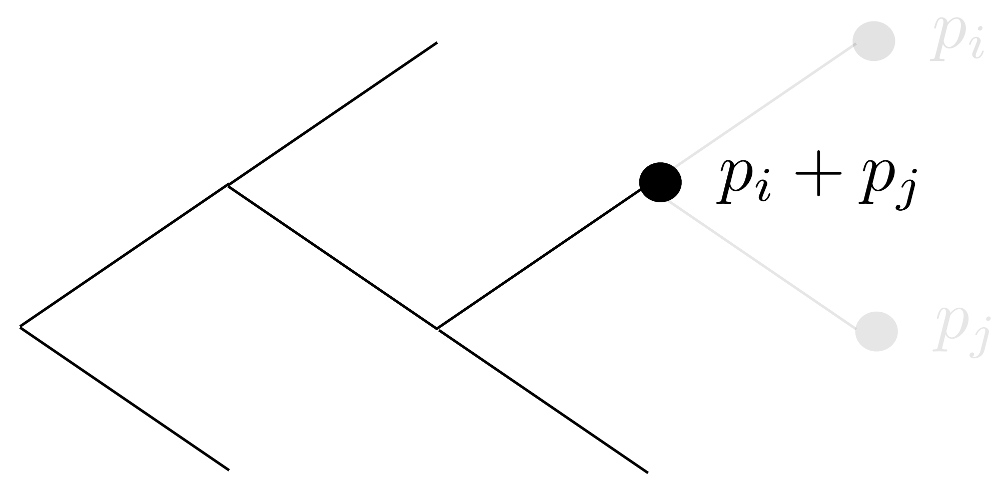
1. Suppose c_i and c_j are siblings in a code tree. Then $l_i = l_j$.

2. If we replace c_i and c_j by a common codeword at their parent, call it c_{ij}, then we obtain a reduced code tree, and the probability of c_{ij} is $p_i + p_j$.

3. Accordingly, the probability set becomes a reduced probability set with p_i and p_j replaced by a probability $p_i + p_j$.

4. Let L and L' be the expected lengths of the original code and the reduced code, respectively. Then

$$\begin{aligned} L - L' &= (p_i l_i + p_j \underline{l_j}) - (p_i + p_j)(l_i - 1) \\ &= (p_i l_i + p_j \underline{l_i}) - (p_i + p_j)(l_i - 1) \end{aligned}$$



Reduced Code Tree

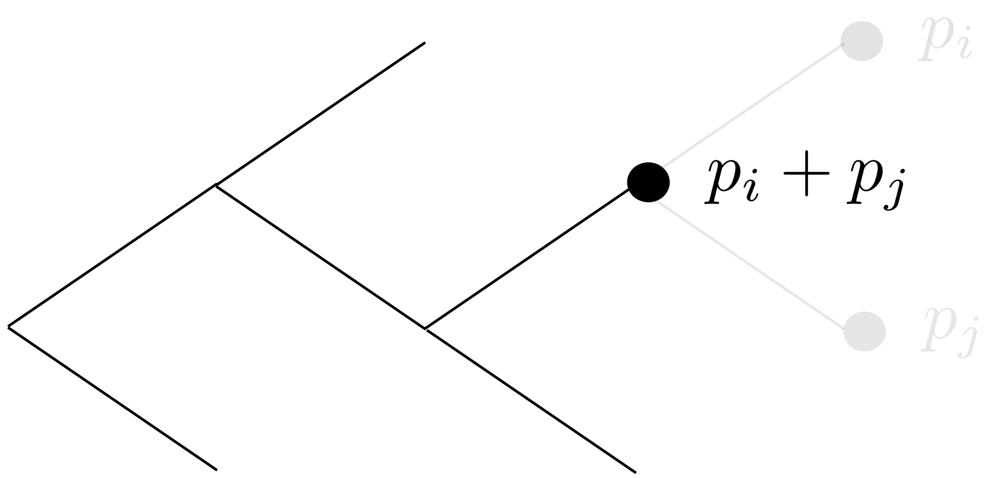
1. Suppose c_i and c_j are siblings in a code tree. Then $l_i = l_j$.

2. If we replace c_i and c_j by a common codeword at their parent, call it c_{ij}, then we obtain a reduced code tree, and the probability of c_{ij} is $p_i + p_j$.

3. Accordingly, the probability set becomes a reduced probability set with p_i and p_j replaced by a probability $p_i + p_j$.

4. Let L and L' be the expected lengths of the original code and the reduced code, respectively. Then

$$
\begin{aligned}
L - L' &= (p_i l_i + p_j l_j) - (p_i + p_j)(l_i - 1) \\
&= (p_i \underline{l_i} + p_j \underline{l_i}) - (p_i + p_j)(l_i - 1)
\end{aligned}
$$



Reduced Code Tree

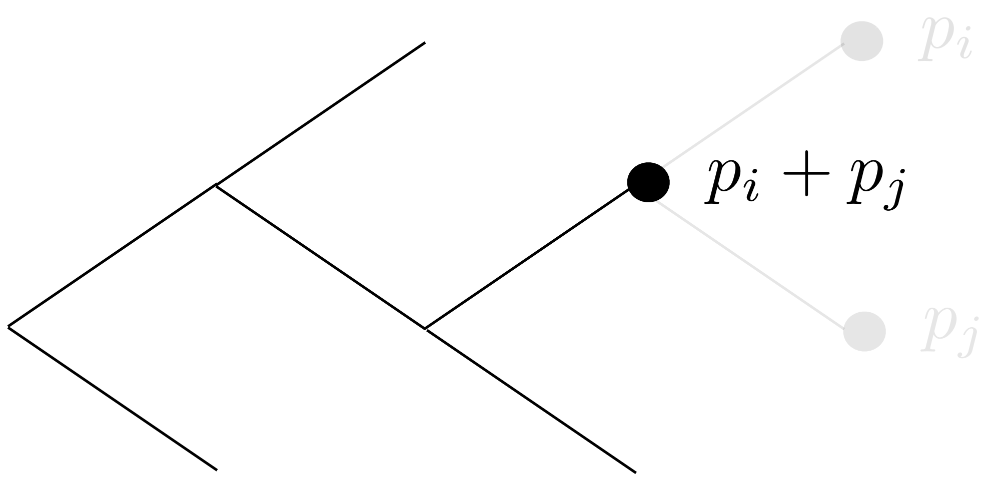
1. Suppose c_i and c_j are siblings in a code tree. Then $l_i = l_j$.

2. If we replace c_i and c_j by a common codeword at their parent, call it c_{ij}, then we obtain a reduced code tree, and the probability of c_{ij} is $p_i + p_j$.

3. Accordingly, the probability set becomes a reduced probability set with p_i and p_j replaced by a probability $p_i + p_j$.

4. Let L and L' be the expected lengths of the original code and the reduced code, respectively. Then

$$\begin{aligned} L - L' &= (p_i l_i + p_j l_j) - (p_i + p_j)(l_i - 1) \\ &= (p_i \underline{l_i} + p_j \underline{l_i}) - (p_i + p_j)(l_i - 1) \\ &= (p_i + p_j)\underline{l_i} - (p_i + p_j)(l_i - 1) \end{aligned}$$



Reduced Code Tree

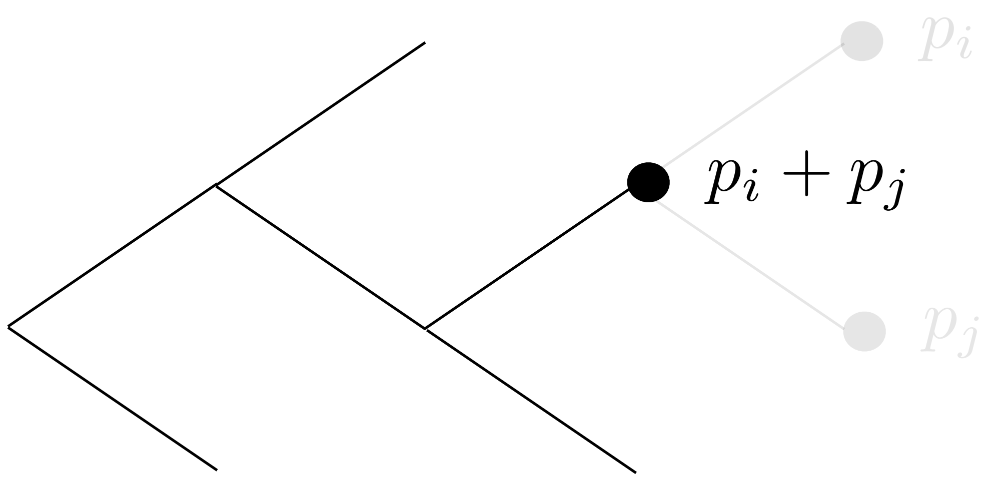
1. Suppose c_i and c_j are siblings in a code tree. Then $l_i = l_j$.

2. If we replace c_i and c_j by a common codeword at their parent, call it c_{ij}, then we obtain a reduced code tree, and the probability of c_{ij} is $p_i + p_j$.

3. Accordingly, the probability set becomes a reduced probability set with p_i and p_j replaced by a probability $p_i + p_j$.

4. Let L and L' be the expected lengths of the original code and the reduced code, respectively. Then

$$
\begin{aligned}
L - L' &= (p_i l_i + p_j l_j) - (p_i + p_j)(l_i - 1) \\
&= (p_i l_i + p_j l_i) - (p_i + p_j)(l_i - 1) \\
&= (p_i + p_j)\underline{l_i} - (p_i + p_j)\underline{(l_i - 1)}
\end{aligned}
$$



Reduced Code Tree

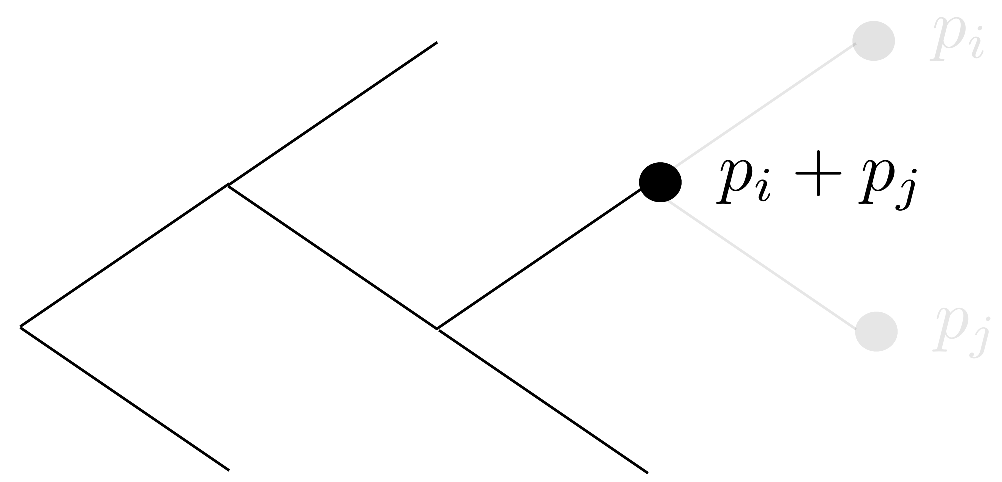
1. Suppose c_i and c_j are siblings in a code tree. Then $l_i = l_j$.

2. If we replace c_i and c_j by a common codeword at their parent, call it c_{ij}, then we obtain a reduced code tree, and the probability of c_{ij} is $p_i + p_j$.

3. Accordingly, the probability set becomes a reduced probability set with p_i and p_j replaced by a probability $p_i + p_j$.

4. Let L and L' be the expected lengths of the original code and the reduced code, respectively. Then

$$\begin{aligned} L - L' &= (p_i l_i + p_j l_j) - (p_i + p_j)(l_i - 1) \\ &= (p_i l_i + p_j l_i) - (p_i + p_j)(l_i - 1) \\ &= (p_i + p_j) l_i - (p_i + p_j)(l_i - 1) \\ &= p_i + p_j, \end{aligned}$$



Reduced Code Tree

1. Suppose c_i and c_j are siblings in a code tree. Then $l_i = l_j$.

2. If we replace c_i and c_j by a common codeword at their parent, call it c_{ij}, then we obtain a reduced code tree, and the probability of c_{ij} is $p_i + p_j$.

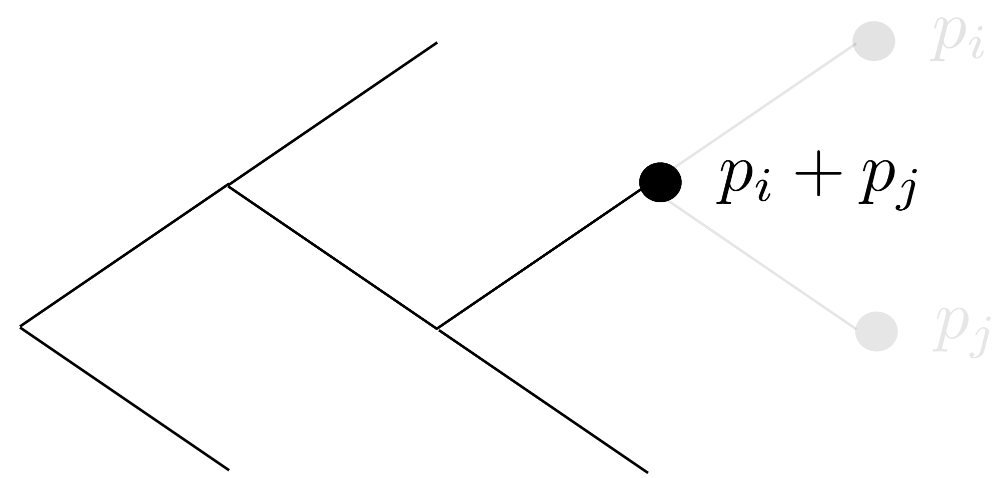
3. Accordingly, the probability set becomes a reduced probability set with p_i and p_j replaced by a probability $p_i + p_j$.

4. Let L and L' be the expected lengths of the original code and the reduced code, respectively. Then

$$\begin{aligned} L - L' &= (p_i l_i + p_j l_j) - (p_i + p_j)(l_i - 1) \\ &= (p_i l_i + p_j l_i) - (p_i + p_j)(l_i - 1) \\ &= (p_i + p_j) l_i - (p_i + p_j)(l_i - 1) \\ &= p_i + p_j, \end{aligned}$$

which implies

$$L = L' + (p_i + p_j).$$



Reduced Code Tree

1. Suppose c_i and c_j are siblings in a code tree. Then $l_i = l_j$.

2. If we replace c_i and c_j by a common codeword at their parent, call it c_{ij}, then we obtain a reduced code tree, and the probability of c_{ij} is $p_i + p_j$.

3. Accordingly, the probability set becomes a reduced probability set with p_i and p_j replaced by a probability $p_i + p_j$.

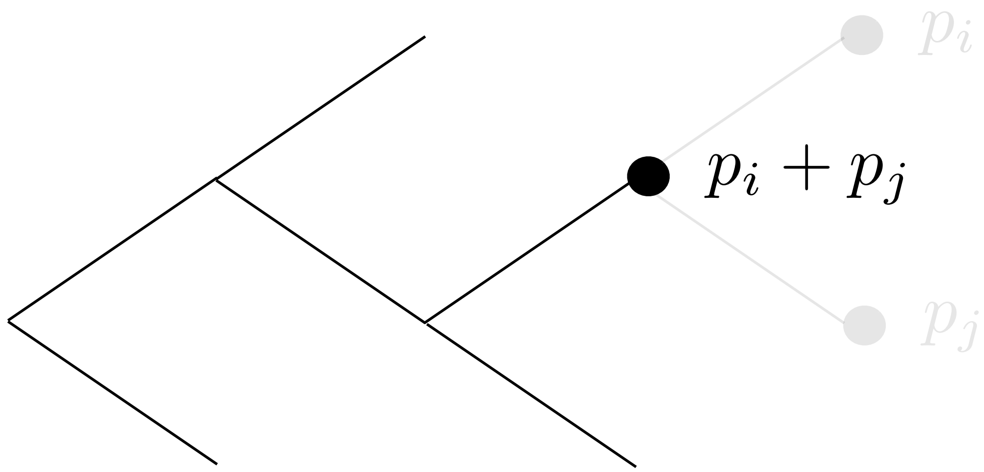
4. Let L and L' be the expected lengths of the original code and the reduced code, respectively. Then

$$\begin{aligned} L - L' &= (p_i l_i + p_j l_j) - (p_i + p_j)(l_i - 1) \\ &= (p_i l_i + p_j l_i) - (p_i + p_j)(l_i - 1) \\ &= (p_i + p_j) l_i - (p_i + p_j)(l_i - 1) \\ &= p_i + p_j, \end{aligned}$$

which implies

$$L = L' + (p_i + p_j).$$

5. This relation says that the difference between the expected length of the original code and the expected length of the reduced code depends only on the values of the two probabilities merged but not on the structure of the reduced code tree.



Theorem 4.17 The Huffman procedure produces an optimal prefix code.

Proof

Theorem 4.17 The Huffman procedure produces an optimal prefix code.

Proof

1. Consider an optimal code in which c_m and c_{m-1} are siblings. Such an optimal code exists by Lemma 4.16.

Theorem 4.17 The Huffman procedure produces an optimal prefix code.

Proof

1. Consider an optimal code in which c_m and c_{m-1} are siblings. Such an optimal code exists by Lemma 4.16.

Lemma 4.16 There exists an optimal code in which the codewords assigned to the two smallest probabilities are siblings.

Theorem 4.17 The Huffman procedure produces an optimal prefix code.

Proof

1. Consider an optimal code in which c_m and c_{m-1} are siblings. Such an optimal code exists by Lemma 4.16.

2. Let $\{p'_i\}$ be the reduced probability set obtained from $\{p_i\}$ by merging p_m and p_{m-1}.

Lemma 4.16 There exists an optimal code in which the codewords assigned to the two smallest probabilities are siblings.

Theorem 4.17 The Huffman procedure produces an optimal prefix code.

Proof

1. Consider an optimal code in which c_m and c_{m-1} are siblings. Such an optimal code exists by Lemma 4.16.

2. Let $\{p'_i\}$ be the reduced probability set obtained from $\{p_i\}$ by merging p_m and p_{m-1}.

3. As before, let L and L' be the expected length of the original code and the reduced code, respectively.

Lemma 4.16 There exists an optimal code in which the codewords assigned to the two smallest probabilities are siblings.

Theorem 4.17 The Huffman procedure produces an optimal prefix code.

Proof

1. Consider an optimal code in which c_m and c_{m-1} are siblings. Such an optimal code exists by Lemma 4.16.

2. Let $\{p'_i\}$ be the reduced probability set obtained from $\{p_i\}$ by merging p_m and p_{m-1}.

3. As before, let L and L' be the expected length of the original code and the reduced code, respectively.

4. Since

$$L = L' + (p_{m-1} + p_m),$$

we see that L is the expected length of an optimal code for $\{p_i\}$ if and only if L' is the expected length of an optimal code for $\{p'_i\}$.

Lemma 4.16 There exists an optimal code in which the codewords assigned to the two smallest probabilities are siblings.

Theorem 4.17 The Huffman procedure produces an optimal prefix code.

Proof

1. Consider an optimal code in which c_m and c_{m-1} are siblings. Such an optimal code exists by Lemma 4.16.

2. Let $\{p'_i\}$ be the reduced probability set obtained from $\{p_i\}$ by merging p_m and p_{m-1}.

3. As before, let L and L' be the expected length of the original code and the reduced code, respectively.

4. Since

$$\underline{L} = L' + (p_{m-1} + p_m),$$

we see that $\underline{L}$ is the expected length of an optimal code for $\{p_i\}$ if and only if L' is the expected length of an optimal code for $\{p'_i\}$.

Lemma 4.16 There exists an optimal code in which the codewords assigned to the two smallest probabilities are siblings.

Theorem 4.17 The Huffman procedure produces an optimal prefix code.

Lemma 4.16 There exists an optimal code in which the codewords assigned to the two smallest probabilities are siblings.

Proof

1. Consider an optimal code in which c_m and c_{m-1} are siblings. Such an optimal code exists by Lemma 4.16.

2. Let $\{p'_i\}$ be the reduced probability set obtained from $\{p_i\}$ by merging p_m and p_{m-1}.

3. As before, let L and L' be the expected length of the original code and the reduced code, respectively.

4. Since

$$\underline{L} = L' + (p_{m-1} + p_m),$$

we see that $\underline{L}$ is the expected length of an optimal code for $\underline{\{p_i\}}$ if and only if L' is the expected length of an optimal code for $\{p'_i\}$.

Theorem 4.17 The Huffman procedure produces an optimal prefix code.

Proof

1. Consider an optimal code in which c_m and c_{m-1} are siblings. Such an optimal code exists by Lemma 4.16.

2. Let $\{p_i'\}$ be the reduced probability set obtained from $\{p_i\}$ by merging p_m and p_{m-1}.

3. As before, let L and L' be the expected length of the original code and the reduced code, respectively.

4. Since

$$\underline{L} = \underline{L'} + (p_{m-1} + p_m),$$

we see that $\underline{L}$ is the expected length of an optimal code for $\underline{\{p_i\}}$ if and only if $\underline{L'}$ is the expected length of an optimal code for $\{p_i'\}$.

Lemma 4.16 There exists an optimal code in which the codewords assigned to the two smallest probabilities are siblings.

Theorem 4.17 The Huffman procedure produces an optimal prefix code.

Proof

1. Consider an optimal code in which c_m and c_{m-1} are siblings. Such an optimal code exists by Lemma 4.16.

2. Let $\{p_i'\}$ be the reduced probability set obtained from $\{p_i\}$ by merging p_m and p_{m-1}.

3. As before, let L and L' be the expected length of the original code and the reduced code, respectively.

4. Since

$$\underline{L} = \underline{L'} + (p_{m-1} + p_m),$$

we see that $\underline{L}$ is the expected length of an optimal code for $\underline{\{p_i\}}$ if and only if $\underline{L'}$ is the expected length of an optimal code for $\underline{\{p_i'\}}$.

Lemma 4.16 There exists an optimal code in which the codewords assigned to the two smallest probabilities are siblings.

Theorem 4.17 The Huffman procedure produces an optimal prefix code.

Proof

1. Consider an optimal code in which c_m and c_{m-1} are siblings. Such an optimal code exists by Lemma 4.16.

2. Let $\{p'_i\}$ be the reduced probability set obtained from $\{p_i\}$ by merging p_m and p_{m-1}.

3. As before, let L and L' be the expected length of the original code and the reduced code, respectively.

4. Since
$$L = L' + (p_{m-1} + p_m),$$
we see that L is the expected length of an optimal code for $\{p_i\}$ if and only if L' is the expected length of an optimal code for $\{p'_i\}$.

5. Therefore, if we can find an optimal code for $\{p'_i\}$, we can use it to construct an optimal code for $\{p_i\}$. Note that by merging p_m and p_{m-1}, the size of the problem, namely the total number of probability masses, is reduced by one.

Lemma 4.16 There exists an optimal code in which the codewords assigned to the two smallest probabilities are siblings.

Theorem 4.17 The Huffman procedure produces an optimal prefix code.

Proof

1. Consider an optimal code in which c_m and c_{m-1} are siblings. Such an optimal code exists by Lemma 4.16.

2. Let $\{p'_i\}$ be the reduced probability set obtained from $\{p_i\}$ by merging p_m and p_{m-1}.

3. As before, let L and L' be the expected length of the original code and the reduced code, respectively.

4. Since

$$L = L' + (p_{m-1} + p_m),$$

we see that L is the expected length of an optimal code for $\{p_i\}$ if and only if L' is the expected length of an optimal code for $\{p'_i\}$.

5. Therefore, if we can find an optimal code for $\{p'_i\}$, we can use it to construct an optimal code for $\{p_i\}$. Note that by merging p_m and p_{m-1}, the size of the problem, namely the total number of probability masses, is reduced by one.

Lemma 4.16 There exists an optimal code in which the codewords assigned to the two smallest probabilities are siblings.

Theorem 4.17 The Huffman procedure produces an optimal prefix code.

Proof

1. Consider an optimal code in which c_m and c_{m-1} are siblings. Such an optimal code exists by Lemma 4.16.

2. Let $\{p'_i\}$ be the reduced probability set obtained from $\{p_i\}$ by merging p_m and p_{m-1}.

3. As before, let L and L' be the expected length of the original code and the reduced code, respectively.

4. Since

$$L = L' + (p_{m-1} + p_m),$$

we see that L is the expected length of an optimal code for $\{p_i\}$ if and only if L' is the expected length of an optimal code for $\{p'_i\}$.

5. Therefore, if we can find an optimal code for $\{p'_i\}$, we can use it to construct an optimal code for $\{p_i\}$. Note that by merging p_m and p_{m-1}, the size of the problem, namely the total number of probability masses, is reduced by one.

Lemma 4.16 There exists an optimal code in which the codewords assigned to the two smallest probabilities are siblings.

Theorem 4.17 The Huffman procedure produces an optimal prefix code.

Proof

1. Consider an optimal code in which c_m and c_{m-1} are siblings. Such an optimal code exists by Lemma 4.16.

2. Let $\{p_i'\}$ be the reduced probability set obtained from $\{p_i\}$ by merging p_m and p_{m-1}.

3. As before, let L and L' be the expected length of the original code and the reduced code, respectively.

4. Since

$$L = L' + (p_{m-1} + p_m),$$

we see that L is the expected length of an optimal code for $\{p_i\}$ if and only if L' is the expected length of an optimal code for $\{p_i'\}$.

5. Therefore, if we can find an optimal code for $\{p_i'\}$, we can use it to construct an optimal code for $\{p_i\}$. Note that by merging p_m and p_{m-1}, the size of the problem, namely the total number of probability masses, is reduced by one.

6. To find an optimal code for $\{p_i'\}$, we again merge the two smallest probability in $\{p_i'\}$. This is repeated until the size of the problem is eventually reduced to 2, which we know that an optimal code has two codewords of length 1.

Lemma 4.16 There exists an optimal code in which the codewords assigned to the two smallest probabilities are siblings.

Theorem 4.17 The Huffman procedure produces an optimal prefix code.

Proof

1. Consider an optimal code in which c_m and c_{m-1} are siblings. Such an optimal code exists by Lemma 4.16.

2. Let $\{p'_i\}$ be the reduced probability set obtained from $\{p_i\}$ by merging p_m and p_{m-1}.

3. As before, let L and L' be the expected length of the original code and the reduced code, respectively.

4. Since

$$L = L' + (p_{m-1} + p_m),$$

we see that L is the expected length of an optimal code for $\{p_i\}$ if and only if L' is the expected length of an optimal code for $\{p'_i\}$.

5. Therefore, if we can find an optimal code for $\{p'_i\}$, we can use it to construct an optimal code for $\{p_i\}$. Note that by merging p_m and p_{m-1}, the size of the problem, namely the total number of probability masses, is reduced by one.

6. To find an optimal code for $\{p'_i\}$, we again merge the two smallest probability in $\{p'_i\}$. This is repeated until the size of the problem is eventually reduced to 2, which we know that an optimal code has two codewords of length 1.

7. In the last step of the Huffman procedure, two probability masses are merged, which corresponds to the formation of a code with two codewords of length 1. Thus the Huffman procedure indeed produces an optimal code.

Lemma 4.16 There exists an optimal code in which the codewords assigned to the two smallest probabilities are siblings.

Theorem 4.17 The Huffman procedure produces an optimal prefix code.

Proof

1. Consider an optimal code in which c_m and c_{m-1} are siblings. Such an optimal code exists by Lemma 4.16.

2. Let $\{p_i'\}$ be the reduced probability set obtained from $\{p_i\}$ by merging p_m and p_{m-1}.

3. As before, let L and L' be the expected length of the original code and the reduced code, respectively.

4. Since

$$L = L' + (p_{m-1} + p_m),$$

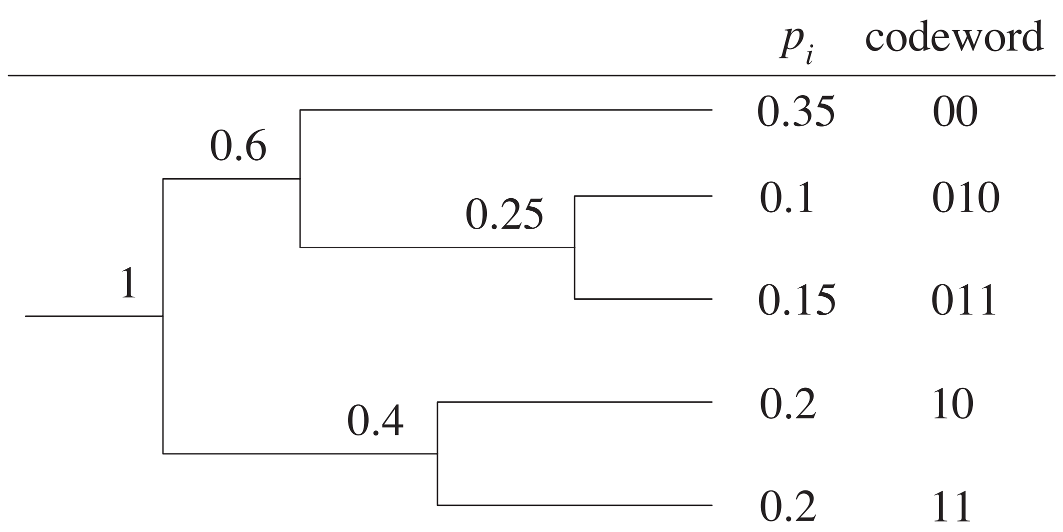
we see that L is the expected length of an optimal code for $\{p_i\}$ if and only if L' is the expected length of an optimal code for $\{p_i'\}$.

5. Therefore, if we can find an optimal code for $\{p_i'\}$, we can use it to construct an optimal code for $\{p_i\}$. Note that by merging p_m and p_{m-1}, the size of the problem, namely the total number of probability masses, is reduced by one.

6. To find an optimal code for $\{p_i'\}$, we again merge the two smallest probability in $\{p_i'\}$. This is repeated until the size of the problem is eventually reduced to 2, which we know that an optimal code has two codewords of length 1.

7. In the last step of the Huffman procedure, two probability masses are merged, which corresponds to the formation of a code with two codewords of length 1. Thus the Huffman procedure indeed produces an optimal code.

Lemma 4.16 There exists an optimal code in which the codewords assigned to the two smallest probabilities are siblings.



Theorem 4.17 The Huffman procedure produces an optimal prefix code.

Proof

1. Consider an optimal code in which c_m and c_{m-1} are siblings. Such an optimal code exists by Lemma 4.16.

2. Let $\{p'_i\}$ be the reduced probability set obtained from $\{p_i\}$ by merging p_m and p_{m-1}.

3. As before, let L and L' be the expected length of the original code and the reduced code, respectively.

4. Since

$$L = L' + (p_{m-1} + p_m),$$

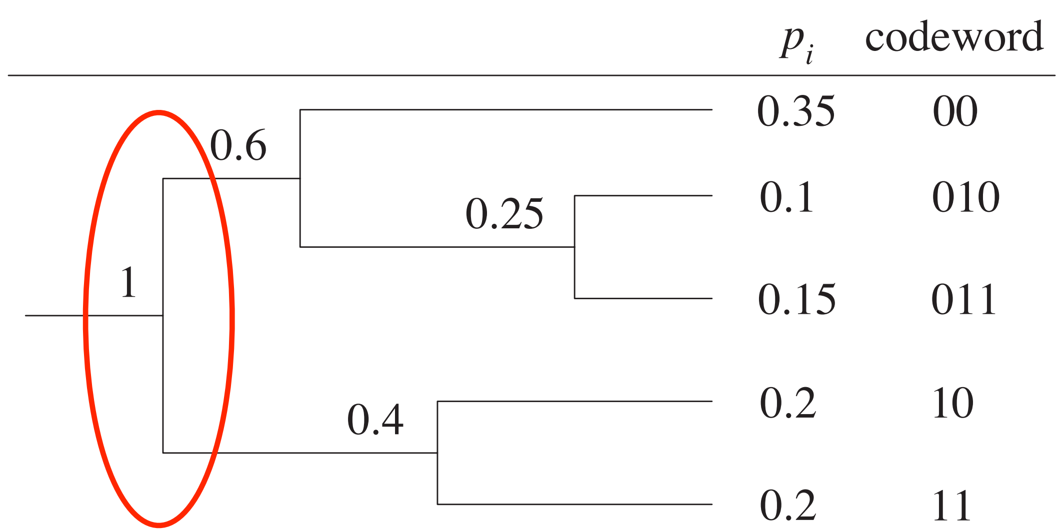
we see that L is the expected length of an optimal code for $\{p_i\}$ if and only if L' is the expected length of an optimal code for $\{p'_i\}$.

5. Therefore, if we can find an optimal code for $\{p'_i\}$, we can use it to construct an optimal code for $\{p_i\}$. Note that by merging p_m and p_{m-1}, the size of the problem, namely the total number of probability masses, is reduced by one.

6. To find an optimal code for $\{p'_i\}$, we again merge the two smallest probability in $\{p'_i\}$. This is repeated until the size of the problem is eventually reduced to 2, which we know that an optimal code has two codewords of length 1.

7. In the last step of the Huffman procedure, two probability masses are merged, which corresponds to the formation of a code with two codewords of length 1. Thus the Huffman procedure indeed produces an optimal code.

Lemma 4.16 There exists an optimal code in which the codewords assigned to the two smallest probabilities are siblings.



D-ary Huffman Procedure

D-ary Huffman Procedure

- The smallest D probability masses are merged in each step, forming an internal node of the resulting code tree.

D-ary Huffman Procedure

- The smallest D probability masses are merged in each step, forming an internal node of the resulting code tree.

- If the resulting code tree is formed in $k + 1$ steps, then there will be $k + 1$ internal nodes and $D + k(D - 1)$ leaves, where each leaf corresponds to a source symbol in the alphabet.

D-ary Huffman Procedure

- The smallest D probability masses are merged in each step, forming an internal node of the resulting code tree.

- If the resulting code tree is formed in $k + 1$ steps, then there will be $k + 1$ internal nodes and $D + k(D - 1)$ leaves, where each leaf corresponds to a source symbol in the alphabet.

- If the alphabet size has the form $D + k(D - 1)$, then apply the Huffman procedure directly.

D-ary Huffman Procedure

- The smallest D probability masses are merged in each step, forming an internal node of the resulting code tree.

- If the resulting code tree is formed in $k + 1$ steps, then there will be $k + 1$ internal nodes and $D + k(D - 1)$ leaves, where each leaf corresponds to a source symbol in the alphabet.

- If the alphabet size has the form $D + k(D - 1)$, then apply the Huffman procedure directly.

- Otherwise, add a few dummy symbols with probability 0 to the alphabet in order to make the total number of symbols have the form $D + k(D - 1)$.

D-ary Huffman Procedure

- The smallest D probability masses are merged in each step, forming an internal node of the resulting code tree.

- If the resulting code tree is formed in $k+1$ steps, then there will be $k+1$ internal nodes and $D + k(D-1)$ leaves, where each leaf corresponds to a source symbol in the alphabet.

- If the alphabet size has the form $D+k(D-1)$, then apply the Huffman procedure directly.

- Otherwise, add a few dummy symbols with probability 0 to the alphabet in order to make the total number of symbols have the form $D + k(D-1)$.

- **Example** $D = 3, k = 2$

D-ary Huffman Procedure

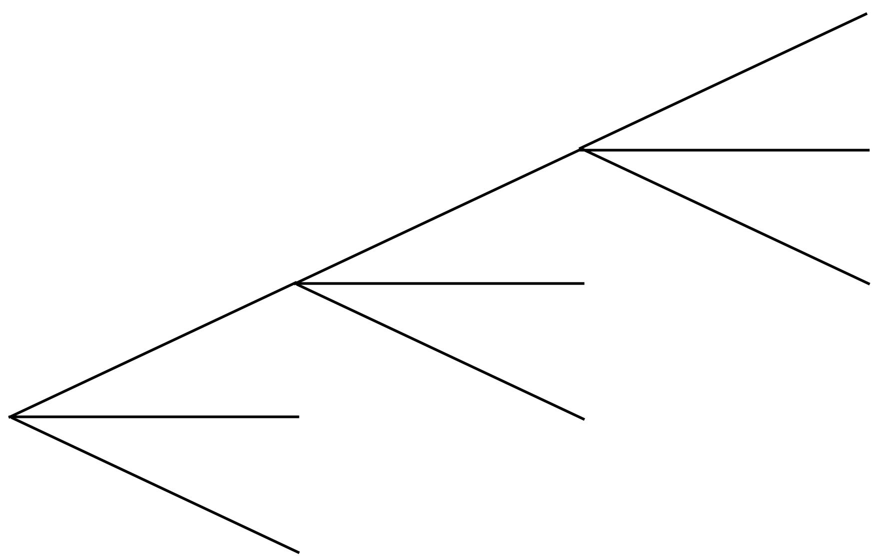
- The smallest D probability masses are merged in each step, forming an internal node of the resulting code tree.

- If the resulting code tree is formed in $k+1$ steps, then there will be $k+1$ internal nodes and $D+k(D-1)$ leaves, where each leaf corresponds to a source symbol in the alphabet.

- If the alphabet size has the form $D+k(D-1)$, then apply the Huffman procedure directly.

- Otherwise, add a few dummy symbols with probability 0 to the alphabet in order to make the total number of symbols have the form $D+k(D-1)$.

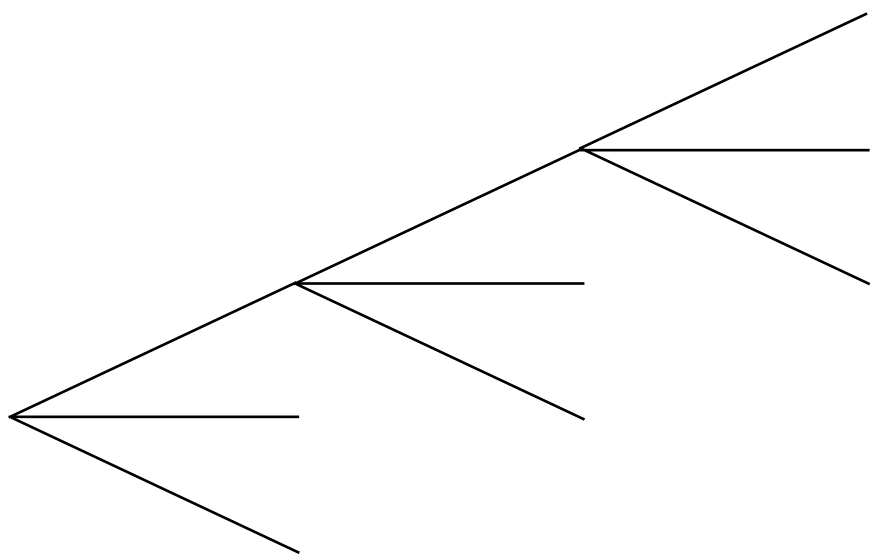
- **Example** $D=3, k=2$



D-ary Huffman Procedure

- The smallest D probability masses are merged in each step, forming an internal node of the resulting code tree.
- If the resulting code tree is formed in $k+1$ steps, then there will be $k+1$ internal nodes and $D+k(D-1)$ leaves, where each leaf corresponds to a source symbol in the alphabet.
- If the alphabet size has the form $D+k(D-1)$, then apply the Huffman procedure directly.
- Otherwise, add a few dummy symbols with probability 0 to the alphabet in order to make the total number of symbols have the form $D+k(D-1)$.
- **Example** $D=3, k=2$

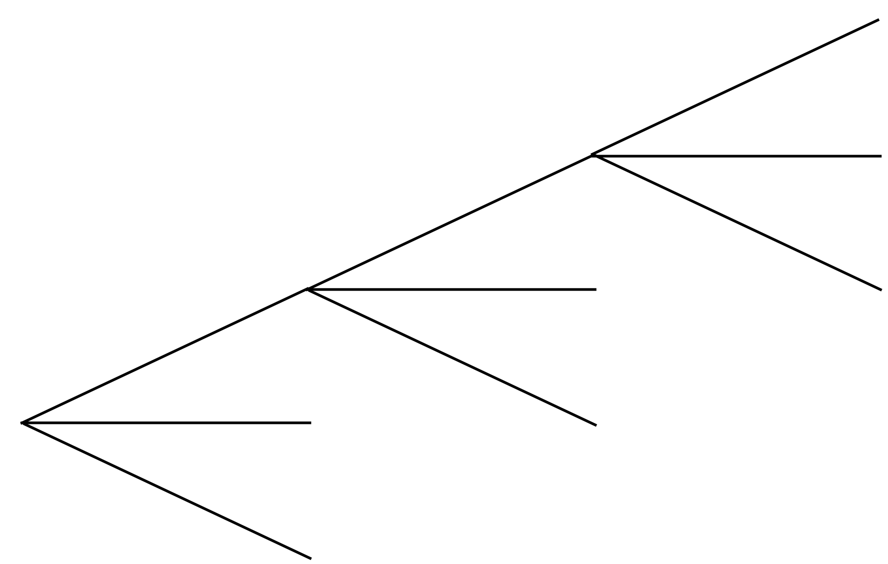
$$\text{No. of steps} \quad = \quad k+1 \quad = \quad 3$$



D-ary Huffman Procedure

- The smallest D probability masses are merged in each step, forming an internal node of the resulting code tree.
- If the resulting code tree is formed in $k + 1$ steps, then there will be $k + 1$ internal nodes and $D + k(D - 1)$ leaves, where each leaf corresponds to a source symbol in the alphabet.
- If the alphabet size has the form $D + k(D - 1)$, then apply the Huffman procedure directly.
- Otherwise, add a few dummy symbols with probability 0 to the alphabet in order to make the total number of symbols have the form $D + k(D - 1)$.
- **Example** $D = 3, k = 2$

$$\begin{aligned} \text{No. of steps} &= k+1 = 3 \\ \text{No. of internal nodes} &= k+1 = 3 \end{aligned}$$



D-ary Huffman Procedure

- The smallest D probability masses are merged in each step, forming an internal node of the resulting code tree.

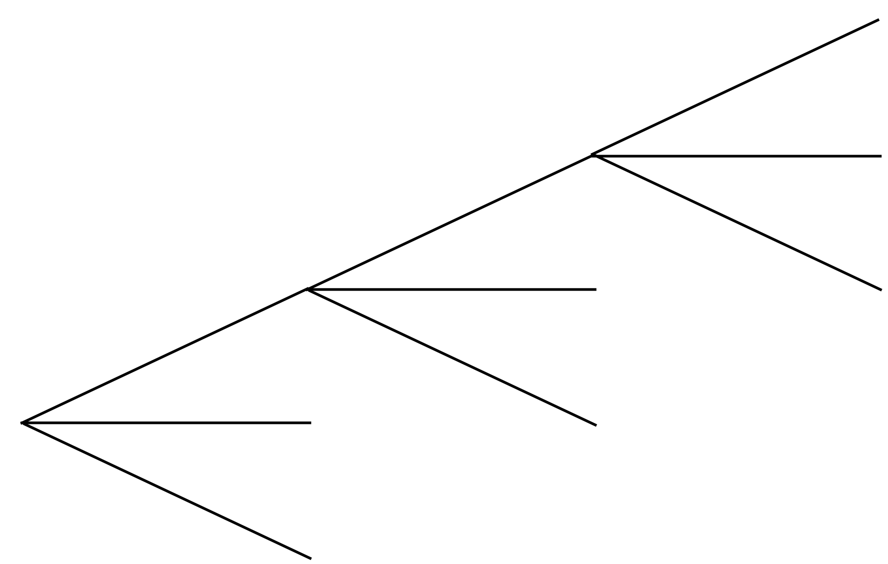
- If the resulting code tree is formed in $k+1$ steps, then there will be $k+1$ internal nodes and $D+k(D-1)$ leaves, where each leaf corresponds to a source symbol in the alphabet.

- If the alphabet size has the form $D+k(D-1)$, then apply the Huffman procedure directly.

- Otherwise, add a few dummy symbols with probability 0 to the alphabet in order to make the total number of symbols have the form $D+k(D-1)$.

- **Example** $D=3, k=2$

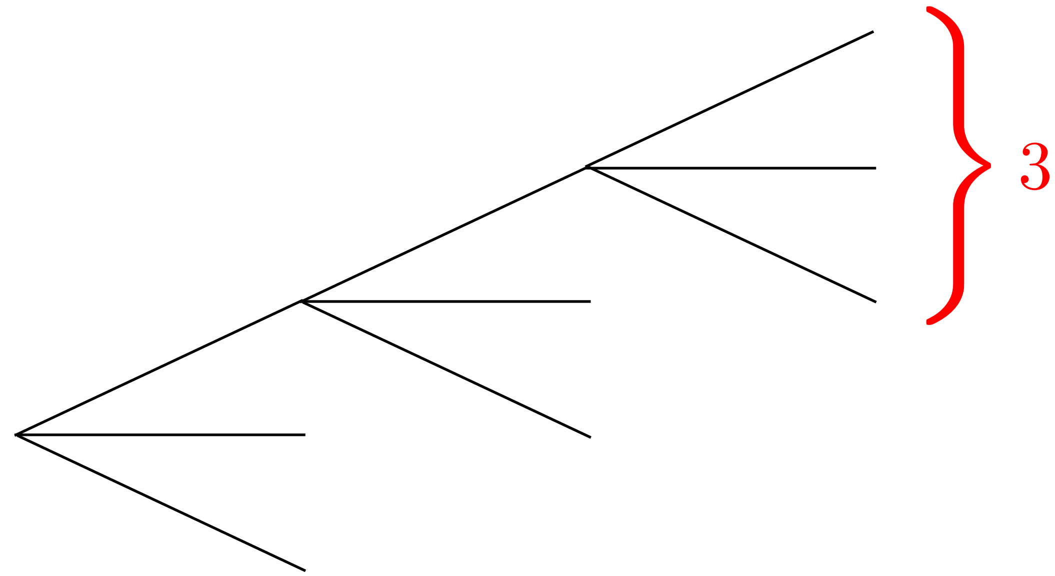
$$\begin{aligned} \text{No. of steps} &= k+1 = 3 \\ \text{No. of internal nodes} &= k+1 = 3 \\ \text{No. of leaves} &= D+k(D-1) \end{aligned}$$



D-ary Huffman Procedure

- The smallest D probability masses are merged in each step, forming an internal node of the resulting code tree.
- If the resulting code tree is formed in $k+1$ steps, then there will be $k+1$ internal nodes and $D+k(D-1)$ leaves, where each leaf corresponds to a source symbol in the alphabet.
- If the alphabet size has the form $D+k(D-1)$, then apply the Huffman procedure directly.
- Otherwise, add a few dummy symbols with probability 0 to the alphabet in order to make the total number of symbols have the form $D+k(D-1)$.
- **Example** $D=3, k=2$

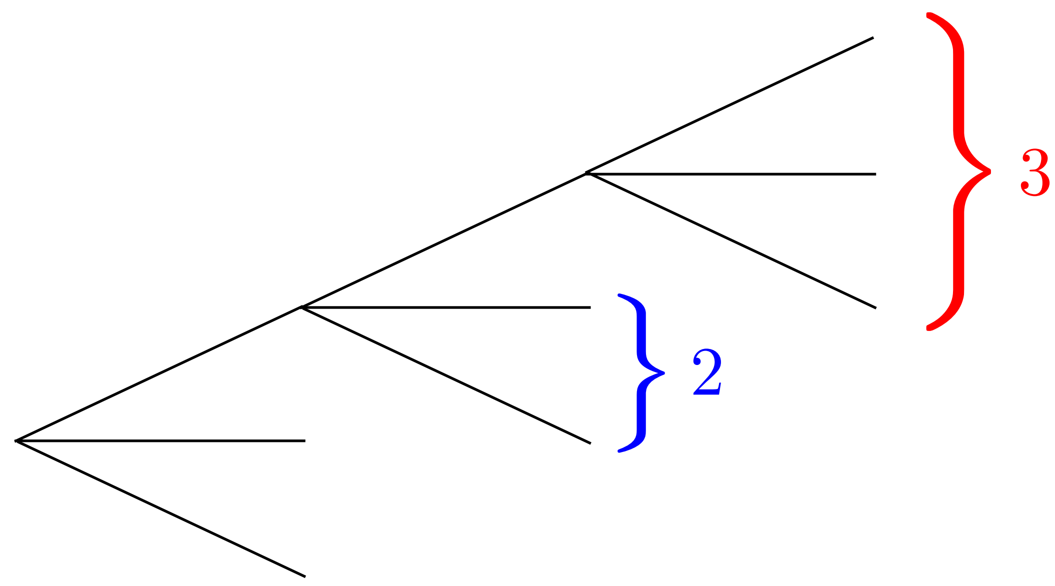
$$\begin{aligned} \text{No. of steps} &= k+1 = 3 \\ \text{No. of internal nodes} &= k+1 = 3 \\ \text{No. of leaves} &= D+k(D-1) \\ &= \underline{3}+2\cdot 2 \end{aligned}$$



D-ary Huffman Procedure

- The smallest D probability masses are merged in each step, forming an internal node of the resulting code tree.
- If the resulting code tree is formed in $k+1$ steps, then there will be $k+1$ internal nodes and $D+k(D-1)$ leaves, where each leaf corresponds to a source symbol in the alphabet.
- If the alphabet size has the form $D+k(D-1)$, then apply the Huffman procedure directly.
- Otherwise, add a few dummy symbols with probability 0 to the alphabet in order to make the total number of symbols have the form $D+k(D-1)$.
- **Example** $D=3, k=2$

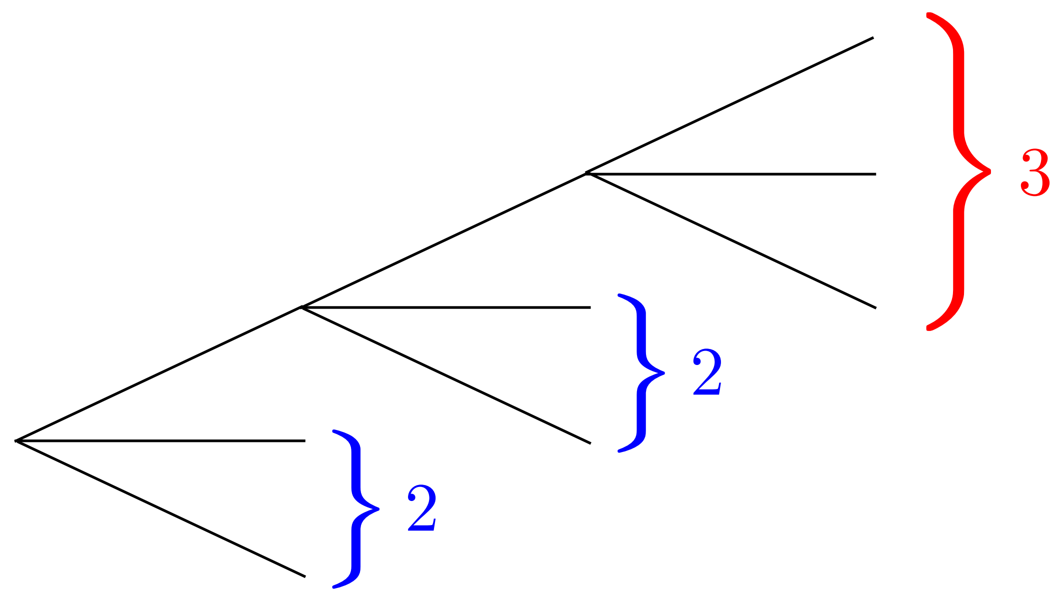
$$
\begin{aligned}
\text{No. of steps} &= k+1 = 3 \\
\text{No. of internal nodes} &= k+1 = 3 \\
\text{No. of leaves} &= D+k(D-1) \\
&= \underline{3} + \underline{2 \cdot 2}
\end{aligned}
$$



D-ary Huffman Procedure

- The smallest D probability masses are merged in each step, forming an internal node of the resulting code tree.
- If the resulting code tree is formed in $k + 1$ steps, then there will be $k + 1$ internal nodes and $D + k(D - 1)$ leaves, where each leaf corresponds to a source symbol in the alphabet.
- If the alphabet size has the form $D + k(D - 1)$, then apply the Huffman procedure directly.
- Otherwise, add a few dummy symbols with probability 0 to the alphabet in order to make the total number of symbols have the form $D + k(D - 1)$.
- **Example** $D = 3, k = 2$

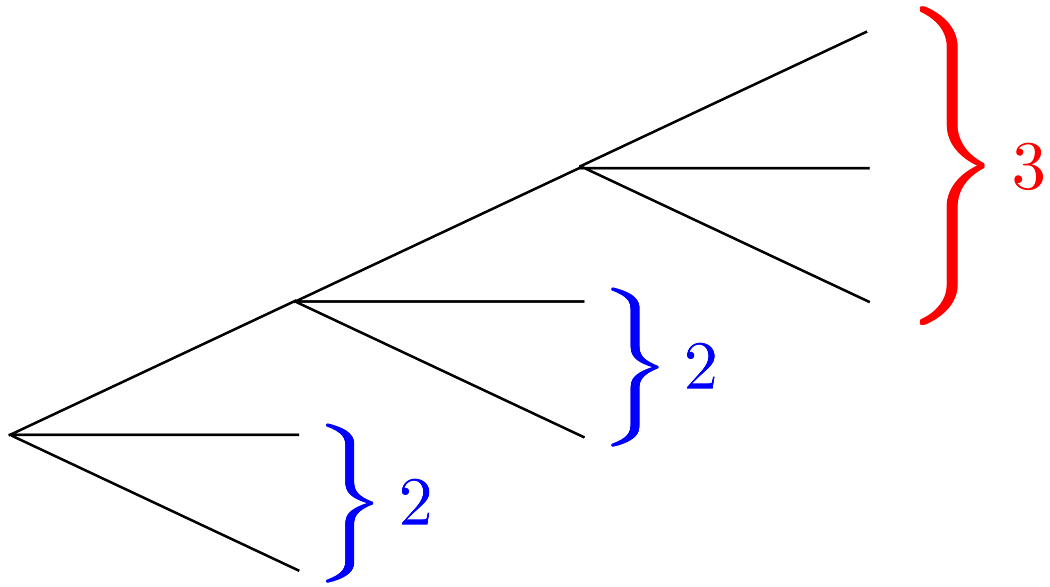
$$
\begin{aligned}
\text{No. of steps} &= k+1 = 3 \\
\text{No. of internal nodes} &= k+1 = 3 \\
\text{No. of leaves} &= D + k(D-1) \\
&= \underline{3} + \underline{2 \cdot 2}
\end{aligned}
$$



D-ary Huffman Procedure

- The smallest D probability masses are merged in each step, forming an internal node of the resulting code tree.
- If the resulting code tree is formed in $k+1$ steps, then there will be $k+1$ internal nodes and $D + k(D-1)$ leaves, where each leaf corresponds to a source symbol in the alphabet.
- If the alphabet size has the form $D + k(D-1)$, then apply the Huffman procedure directly.
- Otherwise, add a few dummy symbols with probability 0 to the alphabet in order to make the total number of symbols have the form $D + k(D-1)$.
- **Example** $D = 3, k = 2$

$$\begin{aligned} \text{No. of steps} &= k+1 = 3 \\ \text{No. of internal nodes} &= k+1 = 3 \\ \text{No. of leaves} &= D + k(D-1) \\ &= 3 + 2 \cdot 2 \\ &= 7 \end{aligned}$$



Upper Bound on L_{Huff}

Theorem 4.18 The expected length of a Huffman code, denoted by L_{Huff}, satisfies

$$L_{\text{Huff}} < H_D(X) + 1.$$

This bound is the tightest among all the upper bounds on L_{Huff} which depend only on the source entropy.

Proof Outline

Upper Bound on L_{Huff}

Theorem 4.18 The expected length of a Huffman code, denoted by L_{Huff}, satisfies

$$L_{\text{Huff}} < H_D(X) + 1.$$

This bound is the tightest among all the upper bounds on L_{Huff} which depend only on the source entropy.

Proof Outline

- Construct a code with codeword lengths $l_i = \lceil -\log_D p_i \rceil$ by showing that the Kraft inequality is satisfied.

Upper Bound on L_{Huff}

Theorem 4.18 The expected length of a Huffman code, denoted by L_{Huff}, satisfies

$$L_{\text{Huff}} < H_D(X) + 1.$$

This bound is the tightest among all the upper bounds on L_{Huff} which depend only on the source entropy.

Proof Outline

- Construct a code with codeword lengths $l_i = \lceil -\log_D p_i \rceil$ by showing that the Kraft inequality is satisfied.
- Show that $L = \sum_i p_i l_i < H(X) + 1$.

Upper Bound on L_{Huff}

Theorem 4.18 The expected length of a Huffman code, denoted by L_{Huff}, satisfies

$$L_{\text{Huff}} < H_D(X) + 1.$$

This bound is the tightest among all the upper bounds on L_{Huff} which depend only on the source entropy.

Proof Outline

- Construct a code with codeword lengths $l_i = \lceil -\log_D p_i \rceil$ by showing that the Kraft inequality is satisfied.
- Show that $L = \sum_i p_i l_i < H(X) + 1$.
- Then $L_{\text{Huff}} \le L < H(X) + 1$.

Theorem 4.18 The expected length of a Huffman code, denoted by L_{Huff}, satisfies

$$L_{\mathrm{Huff}} < H_D(X) + 1.$$

This bound is the tightest among all the upper bounds on L_{Huff} which depend only on the source entropy.

Proof

Theorem 4.18 The expected length of a Huffman code, denoted by L_{Huff}, satisfies

$$L_{\text{Huff}} < H_D(X) + 1.$$

This bound is the tightest among all the upper bounds on L_{Huff} which depend only on the source entropy.

Proof

1. Consider constructing a prefix code with codeword lengths $\{l_i\}$, where $l_i = \lceil -\log_D p_i \rceil$. Then

Theorem 4.18 The expected length of a Huffman code, denoted by L_{Huff}, satisfies

$$L_{\text{Huff}} < H_D(X) + 1.$$

This bound is the tightest among all the upper bounds on L_{Huff} which depend only on the source entropy.

Proof

1. Consider constructing a prefix code with codeword lengths $\{l_i\}$, where $l_i = \lceil -\log_D p_i \rceil$. Then

$$-\log_D p_i \leq l_i < -\log_D p_i + 1$$

Theorem 4.18 The expected length of a Huffman code, denoted by L_{Huff}, satisfies

$$L_{\text{Huff}} < H_D(X) + 1.$$

This bound is the tightest among all the upper bounds on L_{Huff} which depend only on the source entropy.

Proof

1. Consider constructing a prefix code with codeword lengths $\{l_i\}$, where $l_i = \lceil -\log_D p_i \rceil$. Then

$$-\log_D p_i \leq l_i < -\log_D p_i + 1$$

$$\log_D p_i \geq -l_i > \log_D p_i - 1$$

Theorem 4.18 The expected length of a Huffman code, denoted by L_{Huff}, satisfies

$$L_{\text{Huff}} < H_D(X) + 1.$$

This bound is the tightest among all the upper bounds on L_{Huff} which depend only on the source entropy.

Proof

1. Consider constructing a prefix code with codeword lengths $\{l_i\}$, where $l_i = \lceil -\log_D p_i \rceil$. Then

$$-\log_D p_i \leq l_i < -\log_D p_i + 1$$

$$\log_D p_i \geq -l_i > \log_D p_i - 1$$

$$p_i \geq D^{-l_i} > D^{-1} p_i.$$

Theorem 4.18 The expected length of a Huffman code, denoted by L_{Huff}, satisfies

$$L_{\text{Huff}} < H_D(X) + 1.$$

This bound is the tightest among all the upper bounds on L_{Huff} which depend only on the source entropy.

Proof

1. Consider constructing a prefix code with codeword lengths $\{l_i\}$, where $l_i = \lceil -\log_D p_i \rceil$. Then

$$-\log_D p_i \leq l_i < -\log_D p_i + 1$$

$$\log_D p_i \geq -l_i > \log_D p_i - 1$$

$$p_i \geq D^{-l_i} > D^{-1} p_i.$$

Thus

$$\sum_i D^{-l_i} \leq \sum_i p_i = 1,$$

Theorem 4.18 The expected length of a Huffman code, denoted by L_{Huff}, satisfies

$$L_{\text{Huff}} < H_D(X) + 1.$$

This bound is the tightest among all the upper bounds on L_{Huff} which depend only on the source entropy.

Proof

1. Consider constructing a prefix code with codeword lengths $\{l_i\}$, where $l_i = \lceil -\log_D p_i \rceil$. Then

$$-\log_D p_i \leq l_i < -\log_D p_i + 1$$

$$\log_D p_i \geq -l_i > \log_D p_i - 1$$

$$p_i \geq D^{-l_i} > D^{-1} p_i.$$

Thus

$$\sum_i D^{-l_i} \leq \sum_i p_i = 1,$$

i.e., $\{l_i\}$ satisfies the Kraft inequality, which implies that it is possible to construct a prefix code with codeword lengths $\{l_i\}$.

Theorem 4.18 The expected length of a Huffman code, denoted by L_{Huff}, satisfies

$$L_{\text{Huff}} < H_D(X) + 1.$$

This bound is the tightest among all the upper bounds on L_{Huff} which depend only on the source entropy.

Proof

1. Consider constructing a prefix code with codeword lengths $\{l_i\}$, where $l_i = \lceil -\log_D p_i \rceil$. Then

$$-\log_D p_i \le l_i < -\log_D p_i + 1$$

$$\log_D p_i \ge -l_i > \log_D p_i - 1$$

$$p_i \ge D^{-l_i} > D^{-1} p_i.$$

Thus

$$\sum_i D^{-l_i} \le \sum_i p_i = 1,$$

i.e., $\{l_i\}$ satisfies the Kraft inequality, which implies that it is possible to construct a prefix code with codeword lengths $\{l_i\}$.

2. It remains to show that L, the expected length of this code, is less than $H_D(X) + 1$. Consider

Theorem 4.18 The expected length of a Huffman code, denoted by L_{Huff}, satisfies

$$L_{\text{Huff}} < H_D(X) + 1.$$

This bound is the tightest among all the upper bounds on L_{Huff} which depend only on the source entropy.

Proof

1. Consider constructing a prefix code with codeword lengths $\{l_i\}$, where $l_i = \lceil -\log_D p_i \rceil$. Then

$$-\log_D p_i \leq l_i < -\log_D p_i + 1$$

$$\log_D p_i \geq -l_i > \log_D p_i - 1$$

$$p_i \geq D^{-l_i} > D^{-1} p_i.$$

Thus

$$\sum_i D^{-l_i} \leq \sum_i p_i = 1,$$

i.e., $\{l_i\}$ satisfies the Kraft inequality, which implies that it is possible to construct a prefix code with codeword lengths $\{l_i\}$.

2. It remains to show that L, the expected length of this code, is less than $H_D(X) + 1$. Consider

$$L = \sum_i p_i l_i$$

Theorem 4.18 The expected length of a Huffman code, denoted by L_{Huff}, satisfies

$$L_{\text{Huff}} < H_D(X) + 1.$$

This bound is the tightest among all the upper bounds on L_{Huff} which depend only on the source entropy.

Proof

1. Consider constructing a prefix code with codeword lengths $\{l_i\}$, where $l_i = \lceil -\log_D p_i \rceil$. Then

$$-\log_D p_i \leq l_i < -\log_D p_i + 1$$

$$\log_D p_i \geq -l_i > \log_D p_i - 1$$

$$p_i \geq D^{-l_i} > D^{-1} p_i.$$

Thus

$$\sum_i D^{-l_i} \leq \sum_i p_i = 1,$$

i.e., $\{l_i\}$ satisfies the Kraft inequality, which implies that it is possible to construct a prefix code with codeword lengths $\{l_i\}$.

2. It remains to show that L, the expected length of this code, is less than $H_D(X) + 1$. Consider

$$\begin{aligned} L &= \sum_i p_i l_i \\ &< \sum_i p_i(-\log_D p_i + 1) \end{aligned}$$

Theorem 4.18 The expected length of a Huffman code, denoted by L_{Huff}, satisfies

$$L_{\text{Huff}} < H_D(X) + 1.$$

This bound is the tightest among all the upper bounds on L_{Huff} which depend only on the source entropy.

Proof

1. Consider constructing a prefix code with codeword lengths $\{l_i\}$, where $l_i = \lceil -\log_D p_i \rceil$. Then

$$-\log_D p_i \le l_i < -\log_D p_i + 1$$

$$\log_D p_i \ge -l_i > \log_D p_i - 1$$

$$p_i \ge D^{-l_i} > D^{-1} p_i.$$

Thus

$$\sum_i D^{-l_i} \le \sum_i p_i = 1,$$

i.e., $\{l_i\}$ satisfies the Kraft inequality, which implies that it is possible to construct a prefix code with codeword lengths $\{l_i\}$.

2. It remains to show that L, the expected length of this code, is less than $H_D(X) + 1$. Consider

$$\begin{aligned} L &= \sum_i p_i l_i \\ &< \sum_i p_i(-\log_D p_i + 1) \\ &= -\sum_i p_i \log_D p_i + \sum_i p_i \end{aligned}$$

Theorem 4.18 The expected length of a Huffman code, denoted by L_{Huff}, satisfies

$$L_{\text{Huff}} < H_D(X) + 1.$$

This bound is the tightest among all the upper bounds on L_{Huff} which depend only on the source entropy.

Proof

1. Consider constructing a prefix code with codeword lengths $\{l_i\}$, where $l_i = \lceil -\log_D p_i \rceil$. Then

$$-\log_D p_i \leq l_i < -\log_D p_i + 1$$

$$\log_D p_i \geq -l_i > \log_D p_i - 1$$

$$p_i \geq D^{-l_i} > D^{-1} p_i.$$

Thus

$$\sum_i D^{-l_i} \leq \sum_i p_i = 1,$$

i.e., $\{l_i\}$ satisfies the Kraft inequality, which implies that it is possible to construct a prefix code with codeword lengths $\{l_i\}$.

2. It remains to show that L, the expected length of this code, is less than $H_D(X) + 1$. Consider

$$\begin{aligned} L &= \sum_i p_i l_i \\ &< \sum_i p_i(-\log_D p_i + 1) \\ &= -\sum_i p_i \log_D p_i + \sum_i p_i \\ &= H_D(X) + 1. \end{aligned}$$

Theorem 4.18 The expected length of a Huffman code, denoted by L_{Huff}, satisfies

$$L_{\text{Huff}} < H_D(X) + 1.$$

This bound is the tightest among all the upper bounds on L_{Huff} which depend only on the source entropy.

Proof

1. Consider constructing a prefix code with codeword lengths $\{l_i\}$, where $l_i = \lceil -\log_D p_i \rceil$. Then

$$-\log_D p_i \leq l_i < -\log_D p_i + 1$$

$$\log_D p_i \geq -l_i > \log_D p_i - 1$$

$$p_i \geq D^{-l_i} > D^{-1} p_i.$$

Thus

$$\sum_i D^{-l_i} \leq \sum_i p_i = 1,$$

i.e., $\{l_i\}$ satisfies the Kraft inequality, which implies that it is possible to construct a prefix code with codeword lengths $\{l_i\}$.

2. It remains to show that L, the expected length of this code, is less than $H_D(X) + 1$. Consider

$$\begin{aligned} L &= \sum_i p_i l_i \\ &< \sum_i p_i(-\log_D p_i + 1) \\ &= -\sum_i p_i \log_D p_i + \sum_i p_i \\ &= H_D(X) + 1. \end{aligned}$$

Thus we conclude that

$$L_{\text{Huff}} \leq L < H_D(X) + 1.$$

Theorem 4.18 The expected length of a Huffman code, denoted by L_{Huff}, satisfies

$$L_{\text{Huff}} < H_D(X) + 1.$$

This bound is the tightest among all the upper bounds on L_{Huff} which depend only on the source entropy.

Proof

1. Consider constructing a prefix code with codeword lengths $\{l_i\}$, where $l_i = \lceil -\log_D p_i \rceil$. Then

$$-\log_D p_i \le l_i < -\log_D p_i + 1$$

$$\log_D p_i \ge -l_i > \log_D p_i - 1$$

$$p_i \ge D^{-l_i} > D^{-1} p_i.$$

Thus

$$\sum_i D^{-l_i} \le \sum_i p_i = 1,$$

i.e., $\{l_i\}$ satisfies the Kraft inequality, which implies that it is possible to construct a prefix code with codeword lengths $\{l_i\}$.

2. It remains to show that L, the expected length of this code, is less than $H_D(X) + 1$. Consider

$$\begin{aligned} L &= \sum_i p_i l_i \\ &< \sum_i p_i(-\log_D p_i + 1) \\ &= -\sum_i p_i \log_D p_i + \sum_i p_i \\ &= H_D(X) + 1. \end{aligned}$$

Thus we conclude that

$$L_{\text{Huff}} \le L < H_D(X) + 1.$$

Theorem 4.18 The expected length of a Huffman code, denoted by L_{Huff}, satisfies

$$L_{\text{Huff}} < H_D(X) + 1.$$

This bound is the tightest among all the upper bounds on L_{Huff} which depend only on the source entropy.

Proof

1. Consider constructing a prefix code with codeword lengths $\{l_i\}$, where $l_i = \lceil -\log_D p_i \rceil$. Then

$$-\log_D p_i \leq l_i < -\log_D p_i + 1$$

$$\log_D p_i \geq -l_i > \log_D p_i - 1$$

$$p_i \geq D^{-l_i} > D^{-1} p_i.$$

Thus

$$\sum_i D^{-l_i} \leq \sum_i p_i = 1,$$

i.e., $\{l_i\}$ satisfies the Kraft inequality, which implies that it is possible to construct a prefix code with codeword lengths $\{l_i\}$.

2. It remains to show that L, the expected length of this code, is less than $H_D(X) + 1$. Consider

$$\begin{aligned} L &= \sum_i p_i l_i \\ &< \sum_i p_i(-\log_D p_i + 1) \\ &= -\sum_i p_i \log_D p_i + \sum_i p_i \\ &= H_D(X) + 1. \end{aligned}$$

Thus we conclude that

$$L_{\text{Huff}} \leq L < H_D(X) + 1.$$

3. To see that this upper bound is the tightest possible, we have to show that there exists a sequence of distributions P_k such that L_{Huff} approaches $H_D(X) + 1$ as $k \to \infty$.

Theorem 4.18 The expected length of a Huffman code, denoted by L_{Huff}, satisfies

$$L_{\text{Huff}} < H_D(X) + 1.$$

This bound is the tightest among all the upper bounds on L_{Huff} which depend only on the source entropy.

Proof

1. Consider constructing a prefix code with codeword lengths $\{l_i\}$, where $l_i = \lceil -\log_D p_i \rceil$. Then

$$-\log_D p_i \leq l_i < -\log_D p_i + 1$$

$$\log_D p_i \geq -l_i > \log_D p_i - 1$$

$$p_i \geq D^{-l_i} > D^{-1} p_i.$$

Thus

$$\sum_i D^{-l_i} \leq \sum_i p_i = 1,$$

i.e., $\{l_i\}$ satisfies the Kraft inequality, which implies that it is possible to construct a prefix code with codeword lengths $\{l_i\}$.

2. It remains to show that L, the expected length of this code, is less than $H_D(X) + 1$. Consider

$$\begin{aligned} L &= \sum_i p_i l_i \\ &< \sum_i p_i(-\log_D p_i + 1) \\ &= -\sum_i p_i \log_D p_i + \sum_i p_i \\ &= H_D(X) + 1. \end{aligned}$$

Thus we conclude that

$$L_{\text{Huff}} \leq L < H_D(X) + 1.$$

3. To see that this upper bound is the tightest possible, we have to show that there exists a sequence of distributions P_k such that L_{Huff} approaches $H_D(X) + 1$ as $k \to \infty$.

4. This can be done by considering the sequence of D-ary distributions

$$P_k = \left\{1 - \frac{D-1}{k}, \overbrace{\frac{1}{k}, \cdots, \frac{1}{k}}^{D-1}\right\},$$

where $k \geq D$.

Theorem 4.18 The expected length of a Huffman code, denoted by L_{Huff}, satisfies

$$L_{\text{Huff}} < H_D(X) + 1.$$

This bound is the tightest among all the upper bounds on L_{Huff} which depend only on the source entropy.

Proof

1. Consider constructing a prefix code with codeword lengths $\{l_i\}$, where $l_i = \lceil -\log_D p_i \rceil$. Then

$$-\log_D p_i \le l_i < -\log_D p_i + 1$$

$$\log_D p_i \ge -l_i > \log_D p_i - 1$$

$$p_i \ge D^{-l_i} > D^{-1} p_i.$$

Thus

$$\sum_i D^{-l_i} \le \sum_i p_i = 1,$$

i.e., $\{l_i\}$ satisfies the Kraft inequality, which implies that it is possible to construct a prefix code with codeword lengths $\{l_i\}$.

2. It remains to show that L, the expected length of this code, is less than $H_D(X) + 1$. Consider

$$\begin{aligned} L &= \sum_i p_i l_i \\ &< \sum_i p_i(-\log_D p_i + 1) \\ &= -\sum_i p_i \log_D p_i + \sum_i p_i \\ &= H_D(X) + 1. \end{aligned}$$

Thus we conclude that

$$L_{\text{Huff}} \le L < H_D(X) + 1.$$

3. To see that this upper bound is the tightest possible, we have to show that there exists a sequence of distributions P_k such that L_{Huff} approaches $H_D(X) + 1$ as $k \to \infty$.

4. This can be done by considering the sequence of D-ary distributions

$$P_k = \left\{ 1 - \frac{D-1}{k}, \overbrace{\frac{1}{k}, \cdots, \frac{1}{k}}^{D-1} \right\},$$

where $k \ge D$.

5. The Huffman code for each P_k consists of D codewords of length 1. Thus L_{Huff} is equal to 1 for all k. As $k \to \infty$, $H_D(X) \to 0$, and hence L_{Huff} approaches $H_D(X) + 1$. The theorem is proved.

Theorem 4.18 The expected length of a Huffman code, denoted by L_{Huff}, satisfies

$$L_{\text{Huff}} < H_D(X) + 1.$$

This bound is the tightest among all the upper bounds on L_{Huff} which depend only on the source entropy.

Proof

1. Consider constructing a prefix code with codeword lengths $\{l_i\}$, where $l_i = \lceil -\log_D p_i \rceil$. Then

$$-\log_D p_i \le l_i < -\log_D p_i + 1$$

$$\log_D p_i \ge -l_i > \log_D p_i - 1$$

$$p_i \ge D^{-l_i} > D^{-1} p_i.$$

Thus

$$\sum_i D^{-l_i} \le \sum_i p_i = 1,$$

i.e., $\{l_i\}$ satisfies the Kraft inequality, which implies that it is possible to construct a prefix code with codeword lengths $\{l_i\}$.

2. It remains to show that L, the expected length of this code, is less than $H_D(X) + 1$. Consider

$$\begin{aligned} L &= \sum_i p_i l_i \\ &< \sum_i p_i(-\log_D p_i + 1) \\ &= -\sum_i p_i \log_D p_i + \sum_i p_i \\ &= H_D(X) + 1. \end{aligned}$$

Thus we conclude that

$$L_{\text{Huff}} \le L < H_D(X) + 1.$$

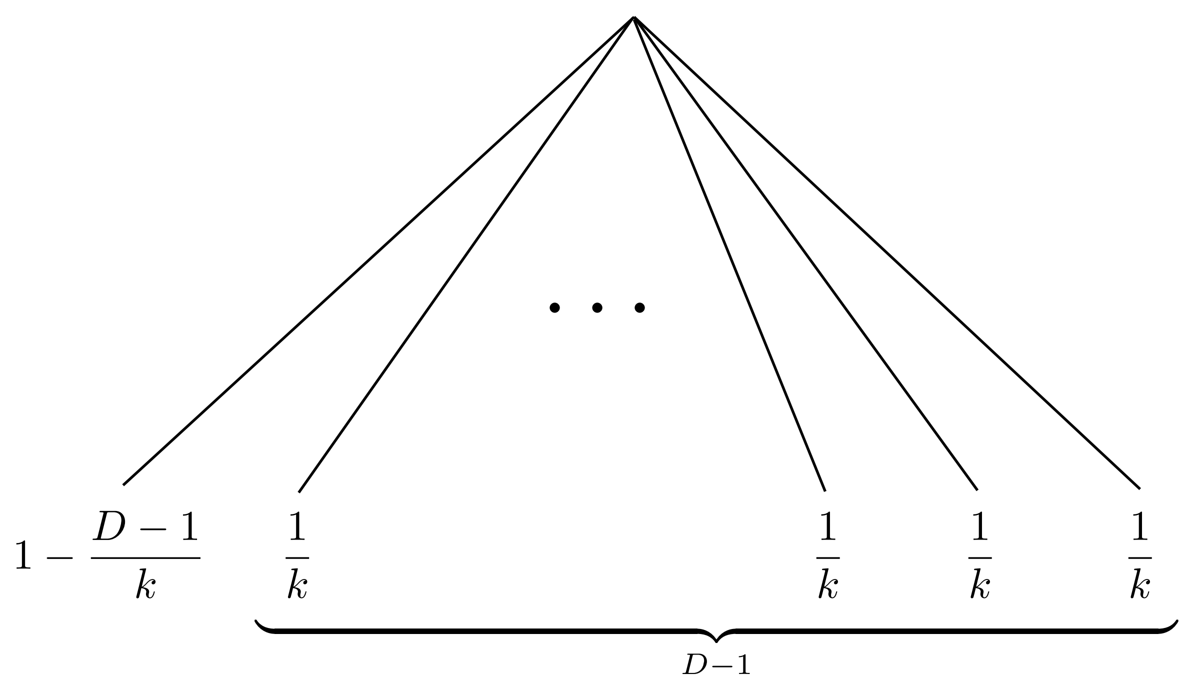
3. To see that this upper bound is the tightest possible, we have to show that there exists a sequence of distributions P_k such that L_{Huff} approaches $H_D(X) + 1$ as $k \to \infty$.

4. This can be done by considering the sequence of D-ary distributions

$$P_k = \left\{ 1 - \frac{D-1}{k}, \overbrace{\frac{1}{k}, \cdots, \frac{1}{k}}^{D-1} \right\},$$

where $k \ge D$.

5. The Huffman code for each P_k consists of D codewords of length 1. Thus L_{Huff} is equal to 1 for all k. As $k \to \infty$, $H_D(X) \to 0$, and hence L_{Huff} approaches $H_D(X) + 1$. The theorem is proved.



Theorem 4.18 The expected length of a Huffman code, denoted by L_{Huff}, satisfies

$$L_{\text{Huff}} < H_D(X) + 1.$$

This bound is the tightest among all the upper bounds on L_{Huff} which depend only on the source entropy.

Proof

1. Consider constructing a prefix code with codeword lengths $\{l_i\}$, where $l_i = \lceil -\log_D p_i \rceil$. Then

$$-\log_D p_i \le l_i < -\log_D p_i + 1$$

$$\log_D p_i \ge -l_i > \log_D p_i - 1$$

$$p_i \ge D^{-l_i} > D^{-1} p_i.$$

Thus

$$\sum_i D^{-l_i} \le \sum_i p_i = 1,$$

i.e., $\{l_i\}$ satisfies the Kraft inequality, which implies that it is possible to construct a prefix code with codeword lengths $\{l_i\}$.

2. It remains to show that L, the expected length of this code, is less than $H_D(X) + 1$. Consider

$$\begin{aligned} L &= \sum_i p_i l_i \\ &< \sum_i p_i(-\log_D p_i + 1) \\ &= -\sum_i p_i \log_D p_i + \sum_i p_i \\ &= H_D(X) + 1. \end{aligned}$$

Thus we conclude that

$$L_{\text{Huff}} \le L < H_D(X) + 1.$$

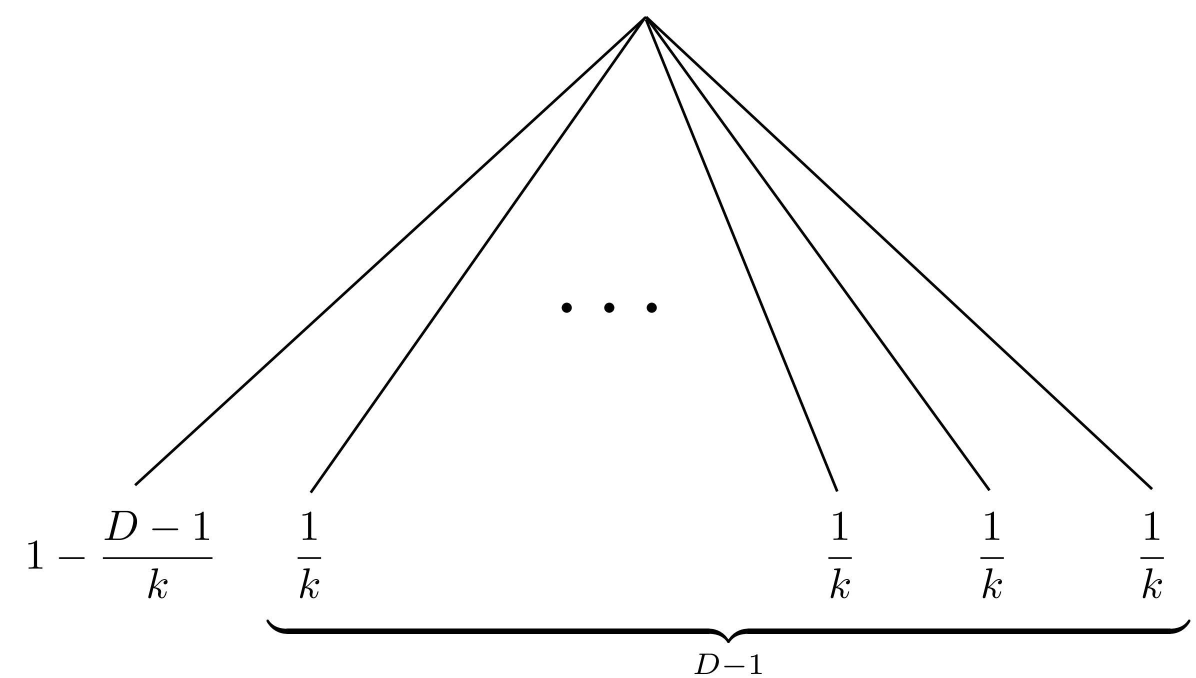
3. To see that this upper bound is the tightest possible, we have to show that there exists a sequence of distributions P_k such that L_{Huff} approaches $H_D(X) + 1$ as $k \to \infty$.

4. This can be done by considering the sequence of D-ary distributions

$$P_k = \left\{1 - \frac{D-1}{k}, \overbrace{\frac{1}{k}, \cdots, \frac{1}{k}}^{D-1}\right\},$$

where $k \ge D$.

5. The Huffman code for each P_k consists of D codewords of length 1. Thus L_{Huff} is equal to 1 for all k. As $k \to \infty$, $H_D(X) \to 0$, and hence L_{Huff} approaches $H_D(X) + 1$. The theorem is proved.



Theorem 4.18 The expected length of a Huffman code, denoted by L_{Huff}, satisfies

$$\overset{1}{L_{\text{Huff}}} < \overset{\to 0}{H_D(X)} + 1.$$

This bound is the tightest among all the upper bounds on L_{Huff} which depend only on the source entropy.

Proof

1. Consider constructing a prefix code with codeword lengths $\{l_i\}$, where $l_i = \lceil -\log_D p_i \rceil$. Then

$$-\log_D p_i \leq l_i < -\log_D p_i + 1$$

$$\log_D p_i \geq -l_i > \log_D p_i - 1$$

$$p_i \geq D^{-l_i} > D^{-1} p_i.$$

Thus

$$\sum_i D^{-l_i} \leq \sum_i p_i = 1,$$

i.e., $\{l_i\}$ satisfies the Kraft inequality, which implies that it is possible to construct a prefix code with codeword lengths $\{l_i\}$.

2. It remains to show that L, the expected length of this code, is less than $H_D(X) + 1$. Consider

$$\begin{aligned} L &= \sum_i p_i l_i \\ &< \sum_i p_i(-\log_D p_i + 1) \\ &= -\sum_i p_i \log_D p_i + \sum_i p_i \\ &= H_D(X) + 1. \end{aligned}$$

Thus we conclude that

$$L_{\text{Huff}} \leq L < H_D(X) + 1.$$

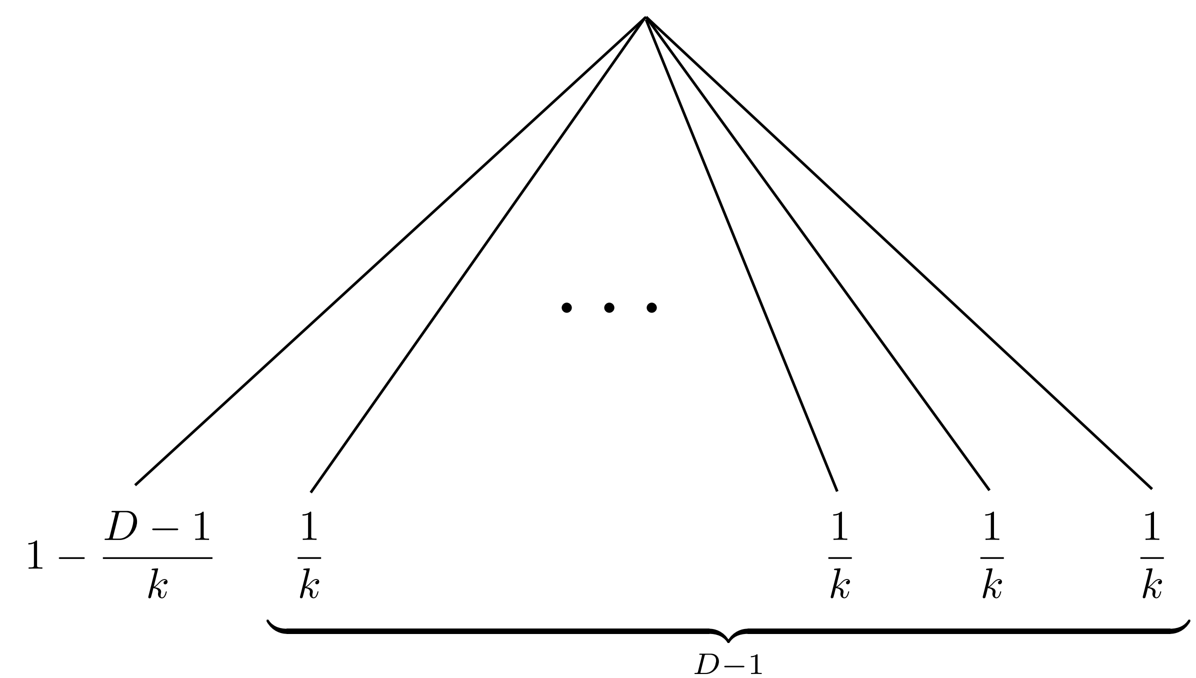
3. To see that this upper bound is the tightest possible, we have to show that there exists a sequence of distributions P_k such that L_{Huff} approaches $H_D(X) + 1$ as $k \to \infty$.

4. This can be done by considering the sequence of D-ary distributions

$$P_k = \left\{1 - \frac{D-1}{k}, \overbrace{\frac{1}{k}, \cdots, \frac{1}{k}}^{D-1}\right\},$$

where $k \geq D$.

5. The Huffman code for each P_k consists of D codewords of length 1. Thus L_{Huff} is equal to 1 for all k. As $k \to \infty$, $H_D(X) \to 0$, and hence L_{Huff} approaches $H_D(X) + 1$. The theorem is proved.



Asymptotic Achievability of H(X)

Asymptotic Achievability of H(X)

- $$H(X) \leq L_{\text{Huff}} < H(X) + 1.$$

Asymptotic Achievability of H(X)

- $$H(X) \le L_{\text{Huff}} < H(X) + 1.$$
- Use a Huffman code to encode $X_1, X_2, \cdots, X_n$, n i.i.d. copies of X. Then

$$nH(X) \le L^n_{\text{Huff}} < nH(X) + 1.$$

Asymptotic Achievability of H(X)

- $$H(X) \le L_{\text{Huff}} < H(X) + 1.$$

- Use a Huffman code to encode $X_1, X_2, \cdots, X_n$, n i.i.d. copies of X. Then
$$nH(X) \le L^n_{\text{Huff}} < nH(X) + 1.$$

- Divide by n to obtain
$$H(X) \le \frac{1}{n} L^n_{\text{Huff}} < H(X) + \frac{1}{n} \quad \to H(X) \text{ as } n \to \infty$$

Asymptotic Achievability of H(X)

- $$H(X) \le L_{\text{Huff}} < H(X) + 1.$$
- Use a Huffman code to encode $X_1, X_2, \cdots, X_n$, n i.i.d. copies of X. Then
$$nH(X) \le L^n_{\text{Huff}} < nH(X) + 1.$$
- Divide by n to obtain
$$H(X) \le \frac{1}{n} L^n_{\text{Huff}} < H(X) + \frac{1}{n} \ \rightarrow H(X) \text{ as } n \rightarrow \infty$$
- $\frac{1}{n} L^n_{\text{Huff}}$ is called the rate of the code, in D-it per source symbol.