The Rod Cutting Problem

Shangqi Lu

Department of Computer Science and Engineering
The Chinese University of Hong Kong

The Rod Cutting Problem

A company buys long steel rods of length n, and cuts them into shorter ones to sell

- integral length only
- cutting is free
- rods of diff. lengths have diff. prices, e.g.

length i	1	2	3	4
price p_i	1	5	8	9

Question: What is the best way to cut the rod of length $n = 4$?

Formalization

Input:

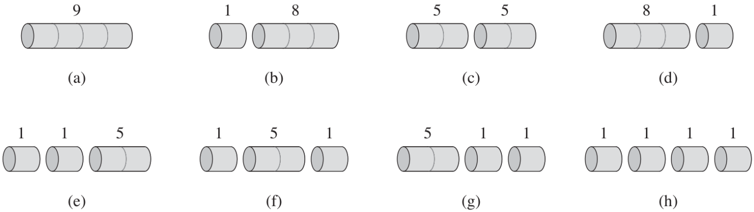
- a rod of length n
- a table of prices p_i for $i \in [1, n]$,
  p_i is the price of a rod of length i.

Goal:

- determine the maximum revenue $opt(n)$ obtained by cutting up the rod and selling all pieces.

Think: How many ways to cut a rod of length n ?

Example 1: $n = 4$



length i	1	2	3	4
price p_i	1	5	8	9

- 8 possible ways

Recall

Three steps to identify the recursive structure:

1. identify all the possible options for the "first" choice;
2. conditioned on the first choice, find the optimal solution;
3. take the first choice that leads to the overall best solution.

Next, we will explain how to do so for the rod cutting problem.

1. Find all the options of the first choice

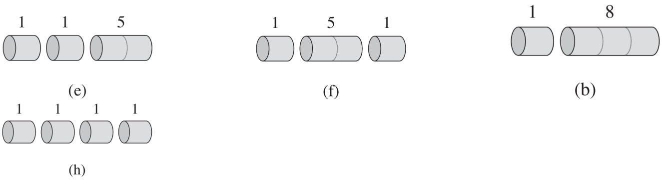
Define $opt(x)$ as the optimal revenue obtained by cutting up a rod of length x. $opt(0) = 0$.

For the rod of length $x > 0$, we have x options for the first piece: its length can range from 1 to x.

Suppose the first piece has length i, the remaining thing is to cut a rod of length $x - i$ into pieces to maximize the total revenue.

Example

In example 1, when $x = 4$ and $i = 1$, the next thing is to find the best way to cut a rod of length 3. There are 3 choices:



2. Conditioned on the first choice, find the optimal solution

Define $opt(x|i)$ as the optimal revenue obtained by cutting up a rod of length x, on condition that the first piece has length i.

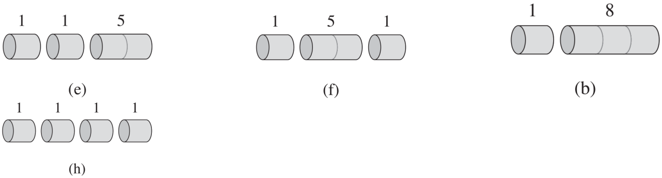
Clearly,

$$opt(x|i) = p_i + opt(x - i)$$

Example

In example 1, when $x = 4$ and conditioned on $i = 1$, there are three choices. The choice shown in "(b)" produces the best solution:

$$opt(4|1) = 1 + 8 = 9$$



3. Selecting the Best First Choice

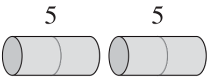
The best choice of i is the one that leads to the largest revenue:

$$\begin{aligned} opt(x) &= \max_{i=1}^{i=x} opt(x|i) \\ &= \max_{i=1}^{i=x} p_i + opt(x-i) \end{aligned}$$

In example 1,

$$opt(4) = opt(4|2) = p_2 + opt(2) = 10$$

and the optimal solution is



(c)

Summary

The recursive structure is as follows.

$$opt(x) = \begin{cases} 0 & \text{if } x = 0 \\ \max_{i=1}^{i=x}(p_i + opt(x-i)) & \text{if } 1 \leq x \leq n \end{cases}$$

The next step is to calculate $opt(n)$ with dynamic programming.

Find the optimal value

We can calculate $opt(x)$ in ascending order of $x = 0, 1, ..., n$.

Algorithm(n)

1. $opt(0) = 0$
2. for $x = 1$ to n
3. $\quad opt(x) = \max_{i=1}^{i=x}(p_i + opt(x - i))$

For each x, $opt(x)$ can be obtained in $O(1 + x)$ time.
The total running time is $\sum_{i=0}^{n} O(1 + x) = O(n^2)$.

Find the optimal solution

The length of each piece in the optimal solution can be constructed in another $O(n)$ time by slightly modifying the algorithm in the previous slide.

For each $x \in [1, n]$, define $bestChoice(x)$ to be the $i \in [1, x]$ that maximizes

$$p_i + opt(x - i).$$

$bestChoice(x)$ can be obtained when computing $opt(x)$ without increasing the time complexity (Think: why?).

Find the optimal solution

After calculating $bestChoice(x)$ for each $x \in [1, n]$.
We can find the optimal solution recursively.

Find(x)

1. $x_1 = bestChoice(x)$
2. output x_1 as a part of the answer
3. if $x_1 \neq x$
4. Find($x - x_1$)

Calling Find(n) will get the optimal solution.
The time complexity of this algorithm is $O(n)$.