Dynamic Programming 2: Rod Cutting

Yufei Tao

Department of Computer Science and Engineering
Chinese University of Hong Kong

The Rod Cutting Problem

Input:

- a rod of length n
- an array P of length n where
  $P[i]$ is the price for a rod of length i, for each $i \in [1, n]$

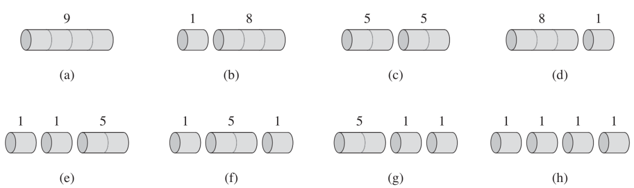
Goal: Cut the rod into segments of integer lengths to maximize the revenue.

Example

Price array P

length i	1	2	3	4
price $P[i]$	1	5	8	9

All possible ways to cut a rod of length 4:



(by courtesy of the textbook)

The optimal cutting method: (c), which has a revenue of 10

The key to solving the problem is to identify its underlying **recursive structure**.

Specifically, how the original problem is related to subproblems.

The recursive structure will then point to an algorithm based on dynamic programming.

Define $opt(n)$ as the optimal revenue from cutting up a rod of length n.

Clearly, $opt(0) = 0$.

Consider now $n \geq 1$.
Let i be the length of the first segment.

- i can be any integer in $[1, x]$.

Conditioned on the first segment having length i, the highest revenue attainable is $P[i] + opt(n - i)$.

Therefore:

$$opt(n) = \max_{i=1}^{n}(P[i] + opt(n - i))$$

We have obtained a recursive structure for the problem.

Given

$$opt(n) = \max_{i=1}^{n}(P[i] + opt(n-i))$$

we can compute $opt(n)$ in $O(n^2)$ time using dynamic programming (this is the problem solved in the last lecture).

Wait! We need to **generate** a cutting method to achieve revenue $opt(n)$.

> This can be done by recording which subproblem yields $opt(n)$.

See the next slide.

Given

$$opt(n) = \max_{i=1}^{n}(P[i] + opt(n-i))$$

define *bestSub*$(n) = k$ if maximization is obtained at $i = k$ (i.e., first segment having length k).

Example

length i	1	2	3	4
price $P[i]$	1	5	8	9
$opt(i)$	1	5	8	10
$bestSub(i)$	1	2	3	2

After we have computed $bestSub(i)$ for every $i \in [1, n]$, the best method for cutting up a rod of length n can be obtained in $O(n)$ time.

(Think: why?)

For each $i \in [1, n]$, computing $bestSub(i)$ is no more expensive than computing $opt(i)$. This is left as a regular exercise.

We conclude that the rod cutting problem can be solved in $O(n^2)$ time.

> The method of using the *bestSub* function to generate an optimal cutting is known as the **piggyback** technique.