Rod cutting

(CLRS 15.1)

The problem: We have a long steel rod and we need to cut it into shorter rods which we then sell. Each cut is free and all our rod lengths are always integers. Assume we know, for each $i = 1, 2, 3, ...,$ the price p_i in dollars that we can sell a rod of length i.

Given a rod of length n inches and a table of prices p_i for $i = 1, 2, 3, ..., n$, determine the maximal revenue r_n obtainable by cutting up the rod and selling the pieces.

Example: Find the maximal revenue r_{10} obtainable with the prices below.

length	1	2	3	4	5	6	7	8	9	10
price p_i	1	5	8	9	19	17	17	20	24	30

Notation. We denote by r_n the maximal revenue obtainable by cutting up the rod.

First steps: Draw all possible ways a rod of length n can be cut for $n = 1, 2, 3, 4$. Write down the revenue of the cut in each case.

$n = 1$

$n = 2$

$n = 3$

$n = 4$

Question: How manny different cuts for a rod of length n?

Answer: You have the choice of $n-1$ cuts —you can cut at distance $1, 2, ..., n-1$ from the beginning of the rod. Can view each cut as a binary variable, with values 0 (no cut) or 1 (cut). There are 2^{n-1} different combinations. Each one corresponds to a different way to cut the rod (but note that two different cuts might result in the same set of rods, and thus have the same cost).

Recursive formulation: Assume someone told us that the first (left-most) cut in the optimal solution was at distance i from the beginning; thus the first piece has length i. Then it has to be that $r_n = p_i + r_{n-i}$. In other words, the optimal revenue consists of p_i plus the optimal revenue for the remaining rod. Basically this says that if we want maximal revenue for a rod of length n, once we determined a cut we want maximal revenue for the remaining piece.

Thus we see that the problem has *optimal substructure*: the optimal solution consists of optimal solutions to sub-problems.

We don't know where the first (leftmost) cut is, so we have to consider all options: the first cut can be at distance 1 from the start, or at distance 2, or 3, ..., or n (in this case, there is no cut). The maximal revenue is the largest revenue obtainable by one of these choices. Thus we get

$$r_n = \max\{p_1 + r_{n-1}, p_2 + r_{n-2}, ..., p_i + r_{n-i}, , , , p_{n-1} + r_1, p_n\}$$

Implementation: Write down pseudo-code that finds the maximal revenue r_n:

```
int cut_rod(p, n)
```

Draw the recursion tree for $n = 4$.

Analysis: Write a recurrence for the running time $T(n)$ of your recursive algorithm $cut\_rod(p, n)$ and show that it is exponential $T(n) = \Omega(2^n)$.

Question: Why is $cut\_rod(p, n)$ so inneficient?
Answer: It solves teh same sub-problems repeatedly..

Improving the solution with Dynamic programming: