Rod Cutting

(CLRS 15.1)

The problem: We are given a long steel rod and we need to cut it into shorter rods which we then sell. Making each cut is free and all rod lengths are always integers. Assume we are given, for each $i = 1, 2, 3, ...$, the price p_i (in dollars) that we can sell a rod of length i. Goal: Given a rod of length n inches and a table of prices p_i for $i = 1, 2, 3, ..., n$, determine the maximal revenue obtainable by cutting up the rod and selling the pieces.

Notation. We denote by r_n the maximal revenue obtainable by cutting up a rod of length n.

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

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 many 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 different cuts might result in the same set of rods, and thus have the same cost).

Optimal substructure: 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.
Claim: Then it has to be that $r_n = p_i + r_{n-i}$. In other words, the optimal revenue consists of the pice of that first piece p_i plus the optimal revenue obtainable 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.
Proof: By contradiction.. Assume $r_n = p_i + x$. If $x < r_{n-i}$ then we found a better revenue for a rod of length $n - i$ which is not possible, because we assumed that r_{n-i} is optimal.

Recursive formulation: Once we established that the problem has optimal substructure (the optimal solution consists of optimal solutions to sub-problems), we can use it in the following way:

If we knew where the first (leftmost) cut was, we'd recurse from there. But we don't know where the first cut is, so, 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 cutRod(p, n)
```

Draw the recursion tree for $n = 4$.

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

Why is `cutRod(p,n)` inefficient?

Rod cutting: DP with memoized recursion

Analysis:

Rod cutting: DP with iterative solution

Analysis:

From optimal revenue to optimal cuts

Describe how to augment your solution for the rod cutting problem so that it computes the actual cuts corresponding to the optimal revenue, instead of just the revenue.
For example, let's say that the optimal revenue for a rod of length 12 was achieved by cutting the rod at $\{5, 5, 2\}$. Show how to compute these cuts.