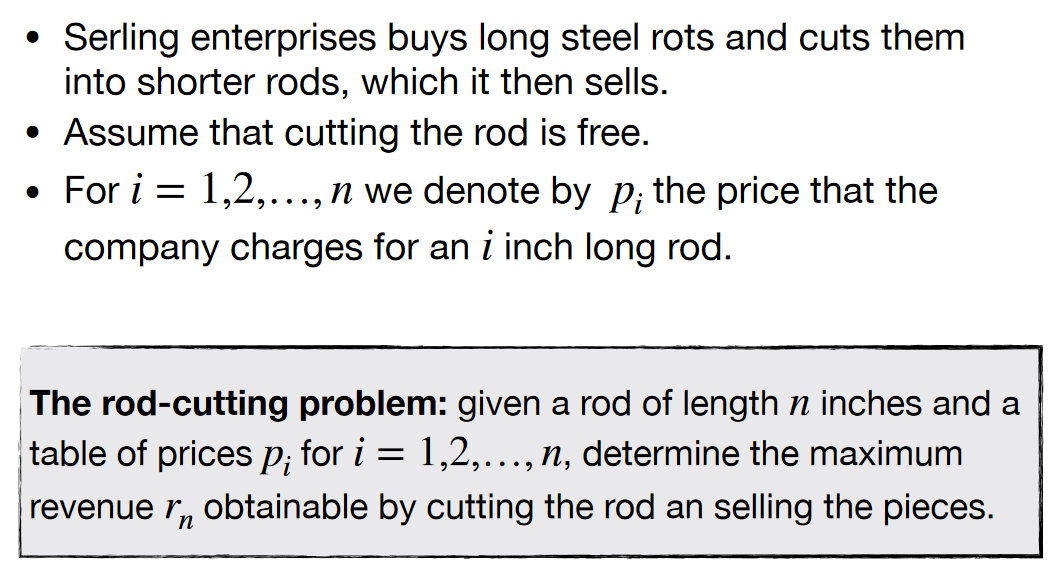
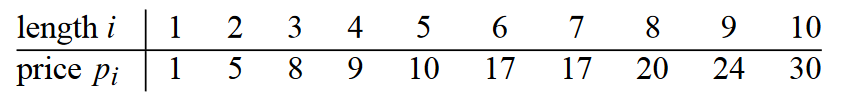
# Rod cutting

Stating the problem:

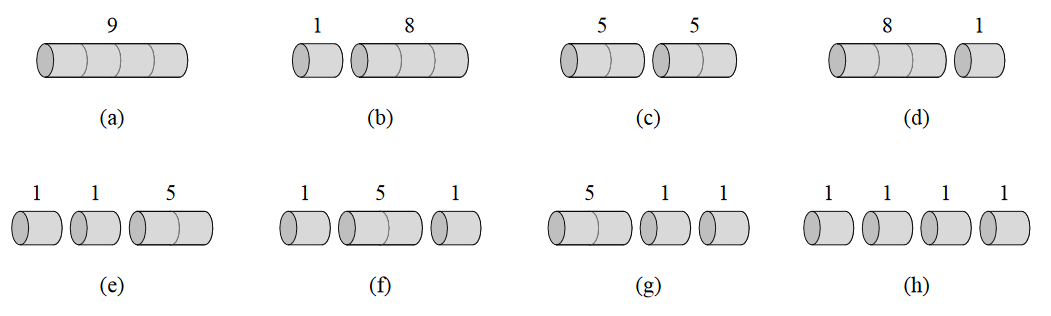


We have to find a maximum profit value for selling the rods. The pricetable looks like this:

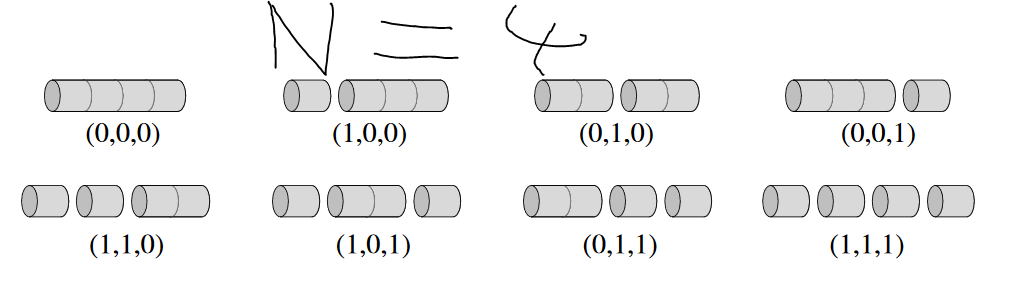


Note that *I* = length and *PI =* Price of *I* length.

Ways to cut the rod could look like this:

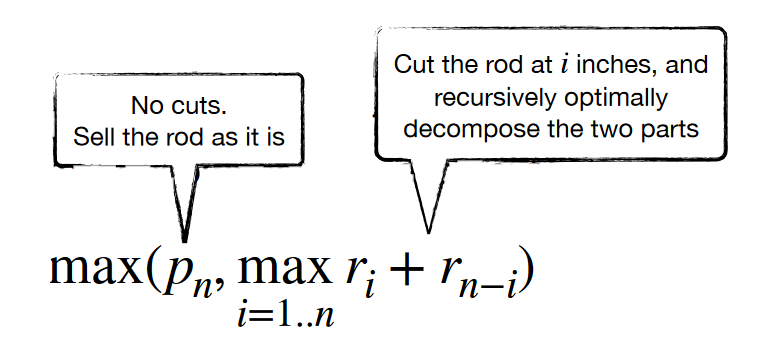


It’s possible to cut a rod of length *N* in 2*N-1* possible ways.

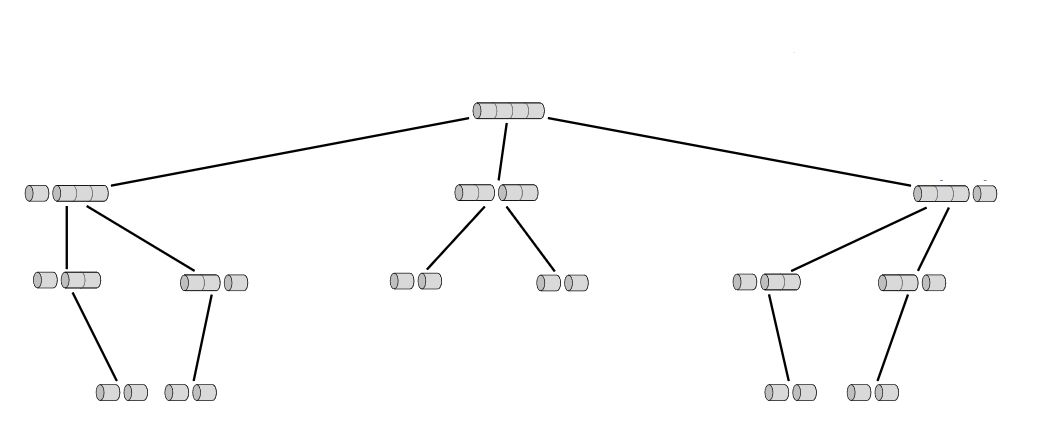


We can cut a rod of length 7 into 2 + 2 + 3 smaller rods. All of the smaller rods(Denoted by *I*) has to add up to the length of the starting rod(Which is denoted by *N)*.   
The price is then denoted by *R* which is all the prices of the smaller rods added together.

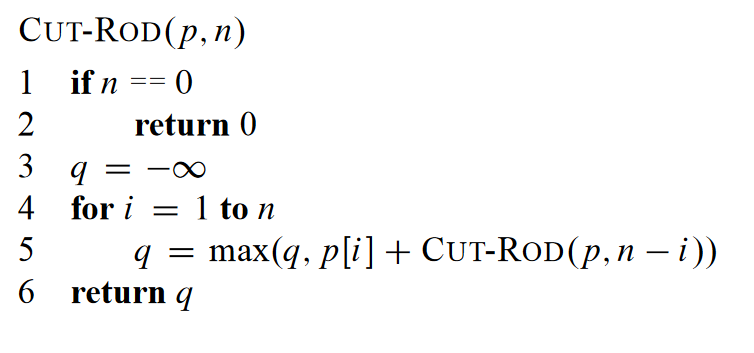
The optimal price can the be found recursively by the following algorithm:



We can the look at the decomposed parts. They look like this:



The code for this implementation looks like this:



*P* = a table of prices.   
*N* = The length of the rod.

Lines 1-2 are our base case. If the length of the rod is 0, then return 0 and our recursion will stop.

On line 3 we set q to -∞ for later use. On line 4 we go from 1 to *N,* which means checking the optimal price for all the lengths up to *N.* If *N* = 7 it will first find the best price for length = 1 and then it will find it for length 1 & 2 and return the best possible value. Then it will do it for 1 & 2 & 3 and return the best possible value for those. But this is not dynamic programming as it recalculates the same problems several times as seen here:

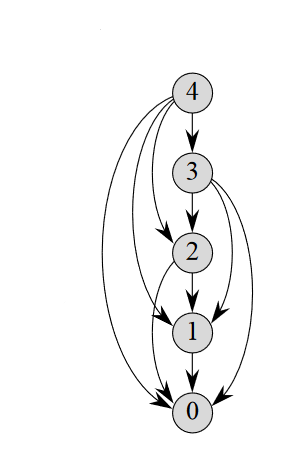
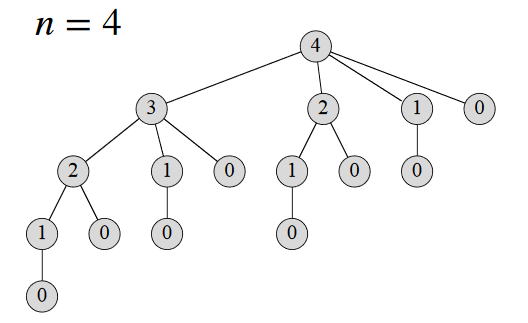


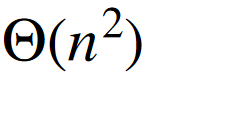
Figure Subproblem graph

As seen in the subproblem graph the subproblems share subproblems. To fix that, we need to ad some memory to our Cut-Rod function, so it can look prior solutions to subproblems up.

This can be down in two ways: Top-down or bottom-up.

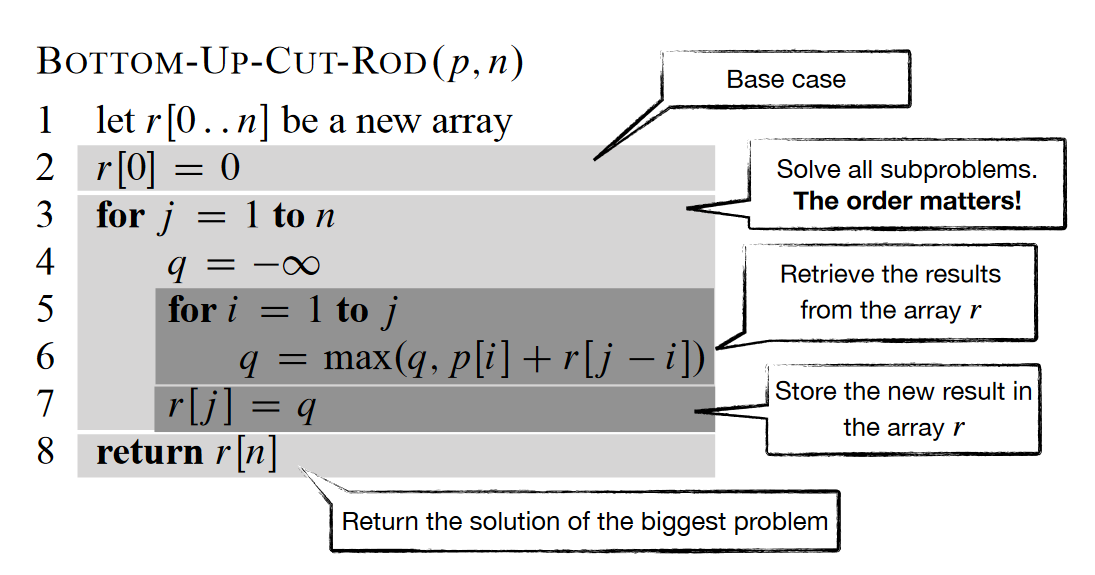
## Top-down

The only difference here is the array called *R* which stores the optimal values. The index values in the array corresponds to the length of the rod.

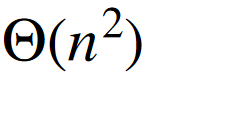
The run-time is: 

## Bottom-up

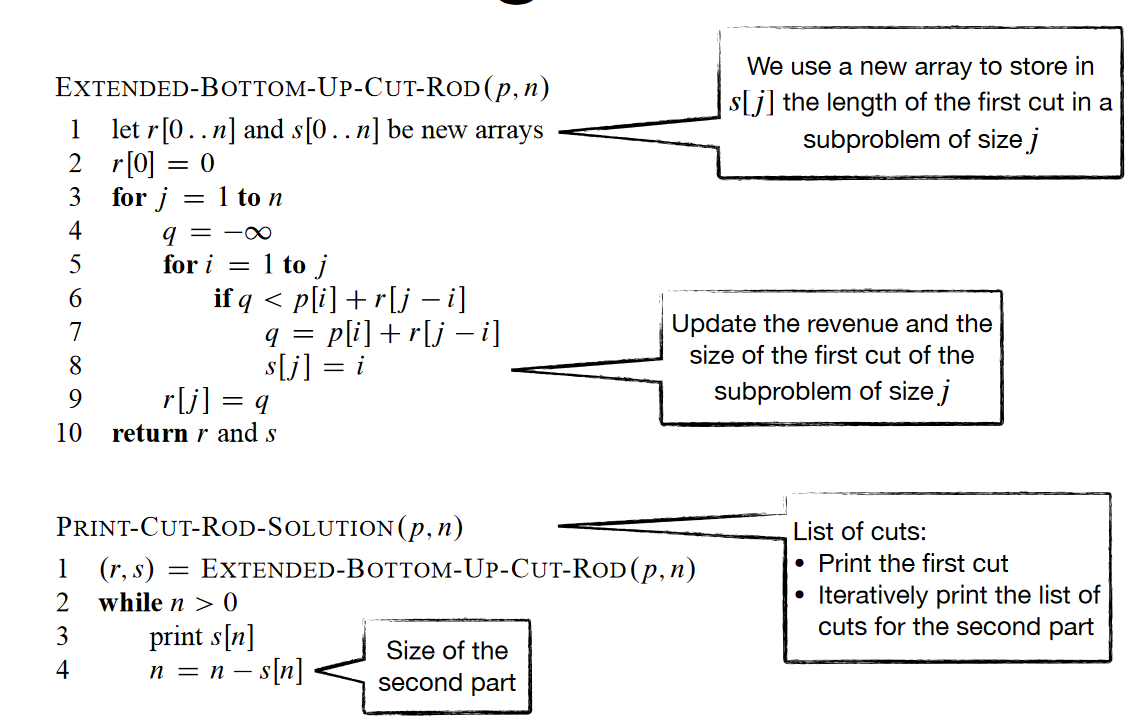
Here we start from the smallest subproblem and work our way up.



The base-case is very important, as it is the smallest subproblem. We again have the array *R* for storing previous results for subproblems.

The run-time is: 

We can now find the best value, but not exactly how to cut it. The extended version for the bottom up method prints the steps for cutting the rod.



The text bobbles explain it all.