

CS375 Small Assignment 7 Solutions

Due Oct 22th 10PM ET

This small assignment will focus on the rod cutting problem. We'll modify it slightly though: this time, we are given a "cut-cost" c , along with a purchase-price array p . Our goal is to determine the maximum payoff obtainable from a length n rod, where any piece of length i will give a payoff of $p[i]$, but it costs c to make each individual cut.

Using dynamic programming, write pseudocode to determine the optimal cuts to make. After, trace through your algorithm (I'm expecting to see an array of values corresponding to whatever your DP alg builds and the tracethrough of the backtrack step to build the actual solution) on the input $p = [0, 3, 10, 20, 22, 24, 28, 37, 37, 40, 49, 57, 58, 58]$ and $c = 3$. Note that in the above array $p[0] = 0$, $p[1] = 3$, etc.

Solution: Suppose we wanted to calculate $\text{OPT}(i)$ = the best value we can get on a length i rod. The optimal solution on such a length either

- makes no cut, in which case the value it gets is $p[i]$, or
- makes at least 1 cut: let that cut length be j . The value it obtains will then be $p[j] - c +$ the optimal value it can obtain on what is left, namely $\text{OPT}(i - j)$.

Note that the value we'd get from a length 0 rod is 0. As such, we get the formula

$$\text{OPT}(i) = \max \left(p[i], \max_{0 < j < i} p[j] - c + \text{OPT}(i - j) \right)$$

In order to calculate this via a dynamic program, consider the following pseudocode:

Initialize A as an empty array of indices 0 to n (inclusive)

$A[0] = 0$ # Base case

for $i = 1, \dots, n + 1$:

$A[i] = \max(p[i], \max_{0 < i < j} (p[j] - c + A[i - j]))$

At this point, $A[i]$ = the best value we

can get on a length i rod for each i

Time to backtrack to get our solution

output = [] # This will hold the lengths of the rod that we cut

at the end of the day, $\text{sum}(\text{output}) = n$

$i = n$

while $i > 0$:

if $A[i] = p[i]$:

add i to output

$i = 0$

else:

find j in $\text{range}(1, i)$ such that $A[i] = p[j] - c + A[i - j]$

By construction of A, such a j must exist

add j to output

return output

On the given input, this gives us the following:

i	0	1	2	3	4	5	6	7	8	9	10	11	12	13
p	0	3	10	20	22	24	28	37	37	40	49	57	58	58
A	0	3	10	20	22	27	37	39	44	54	56	61	71	73

Note that:

$$A[13] = 73 = 56 + 20 - 3 = A[10] + p[3] - c$$

$$A[10] = 56 = 39 + 20 - 3 = A[7] + p[3] - c$$

$$A[7] = 39 = 22 + 20 - 3 = A[4] + p[3] - c$$

$$A[4] = 22 = p[4]$$

which gives us an output of [3, 3, 3, 4].