Advanced Algorithm HW2

Seungho Jang

September 2020

1 Q1

Problem : Create your own length-price table consisting of at least six length-price pairs and with the help of this table explain the rod cutting problem in your own words.

Rod Cutting Problem						
Rod Length	1	2	3	4	5	6
Price	1	4	6	6	8	8

Table 1: Table for rod-cutting problem

As shown the table above, first row represents the rod length, and second row represents the price depending on the first row. The problem statement is following : If I have 6 inch rod, is it better to sell it as it is or break it down so that I can get maximum revenue.

If I have 1 inch rod, then It's better to solve as it is as well as 2. For given rod length = 3, this rod can be divided into 2 and 1 or 1 and 2. The price of rod length 2 is 3 and 1 is 1. If we were to sell 1 and 2 from 3 inch rod, we are getting the revenue as $1 + 4 = 5$. However, if you were to sell 3 inch rod as it is you can get 6. So it's better to sell it as it is.

If I have 4 inch rod, the cases can be, 1+3, 1+1+2, 1+2+1, 1+1+1+1, 2+2, 2+1+1, 3+1, 4. Therefore, we could possibly make an equations for this tendency such as 2^{n-1}. To get the maximum(optimal) revenue, it has to be 2+2 because 2+2 will be 8. In addition to this problem in this case is to know where to make a cut, which is going to be middle.

If I have 5 inch rod, the possible solution can be found in $2^{5-1} = 16$ cases. Then, the maximum revenue can be obtained when the cut is made between 2 and 3 because 4+6 = 10.

If I have 6 inch rod, the possible solution can be found in $2^{6-1} = 32$ cases. Then, the maximum revenue can be obtained when the cut is made between 3 and 3 because 6+6 = 12.

2 Q2

I used bottom-up approach to save where to cut location and optimal revenue in the list.

The code can be found in here : Bottom Up Approach

The output will shows (max-rev, [where-to-cut]). from list in [where-to-cut], the first index will be the first cut will be made.

3 Q3

Problem : Discuss why the recursive solution to the rod-cutting problem is so inefficient?

The reason why recursive solution for the rod-cutting problem is because of calculating repetition in sub-problems. For example, looking at the figure 1, sub-problem(rod length : 2) will be computed twice meaning that those sub-problem will recalculated, which could cause wasting of memory space and computation overhead.

Also, the recursive approach for rod-cutting problem in pseudo code returns q (revenue). We also want to find which place we have to make a cut such that it will return maximum revenue.

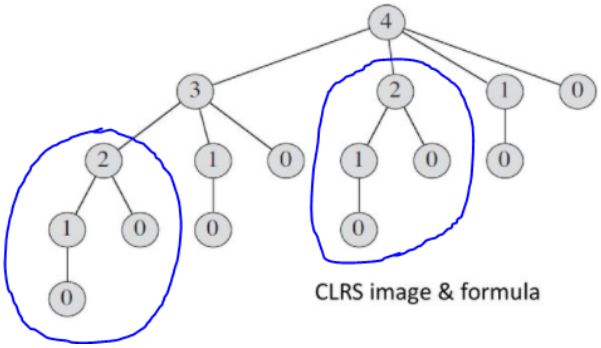

Figure 1: Recursive Tree - Rod-Cutting Problem.

4 Q4

Problem : Discuss the two dynamic programming approaches. If you were asked to implement, which one would you choose, and why?

The two main dynamic programming approaches are memoization(top-down approach) and bottom-up approach. Both of approaches stores the results to sub-problems. However, the way of storing those results is different.

I would choose memoization method because of following:

- Memoization doesn't have to search through the entire search space.

- Memoization doesn't focus on the order of recursive calls.

However, since the difference between memoization and bottom-up approach are subtle. It really depends on the domain of problem. If we were to solve the problem where ordering of the sub-problems matters. For example, if we want to push the values from Fibonacci number in the table, then ordering does matter. Then, the numbers that were used to calculate Fibonacci number can be cached out to utilize the memory space.

However, one of the disadvantage of memoization is that it's incapable for us to utilize the memory. Due to recursive call to sub-problems, it's harder for us to make a decision how to order them.

5 Q5

Problem : Compare and discuss the running times of a recursive solution and a the dynamic programming solution to the rod cutting problem.

The concept of dynamic programming suggests to store the solutions after solving sub-problem such that it does resolve one of the bottleneck of recursive solution; recomputing same sub-problem over again.

By applying this DP approach to rod-cutting problem. There are two ways to store result from sub-problem, which they are top-down(memoization) and bottom up approach.

As the lecture describes, the time complexity for bottom-up and top-down approaches are $\theta(n)$. However, the running time on recursive method is 2^n because it built the recursion tree based on the sub-problems.

To compare the running time on bottom-up approach and recursive method, time library was used in the code. The code link is here : Algorithm Homework2

There might be a trade-off in terms of using dynamical programming approaches again because of putting the sub-problems in lists. This could cause waste of memory. However, the running time is faster as compared to recursive method.