CS 5381 Analysis of Algorithms
Solutions to Homework 2

Fall 2022

1. This is a straightforward verification by inspecting the price table on page 67 of the Lecture Notes.

2.

```
MODIFIED-CUT-ROD($p, n, c$)
  let $r[0..n]$ be a new array
  $r[0] = 0$
  for $j = 1$ to $n$
      $q = p[j]$
      for $i = 1$ to $j - 1$
          $q = \max(q, p[i] + r[j-i] - c)$
      $r[j] = q$
  return $r[n]$
```

The major modification required is in the body of the inner **for** loop, which now reads $q = \max(q, p[i] + r[j-i] - c)$. This change reflects the fixed cost of making the cut, which is deducted from the revenue. We also have to handle the case in which we make no cuts (when i equals j); the total revenue in this case is simply $p[j]$. Thus, we modify the inner **for** loop to run from i to $j-1$ instead of to j. The assignment $q = p[j]$ takes care of the case of no cuts. If we did not make these modifications, then even in the case of no cuts, we would be deducting c from the total revenue.

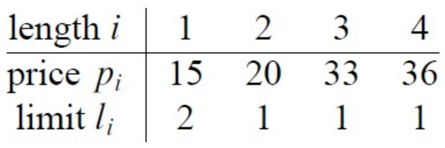
3. The optimal value of $m[2,5]$ is computed as

$$m[2,5] = \min \begin{cases} m[2,2] + m[3,5] + p_1 p_2 p_5 = 0 + 2500 + 35 \cdot 15 \cdot 20 = 13{,}000\,, \\ m[2,3] + m[4,5] + p_1 p_3 p_5 = 2625 + 1000 + 35 \cdot 5 \cdot 20 = 7125\,, \\ m[2,4] + m[5,5] + p_1 p_4 p_5 = 4375 + 0 + 35 \cdot 10 \cdot 20 = 11{,}375 \end{cases}$$
$$= 7125\,.$$

4. Each time the l-loop executes, the i-loop executes $n-l+1$ times. Each time the i-loop executes, the k-loop executes $j-i = l-1$ times, each time referencing m twice. Thus the total number of times that an entry of m is referenced while computing other entries is $\sum_{l=2}^{n}(n-l+1)(l-2)2$. Thus,

$$\begin{aligned} \sum_{i=1}^{n}\sum_{j=i}^{n} R(i,j) &= \sum_{l=2}^{n}(n-l+1)(l-1)2 \\ &= 2\sum_{l=1}^{n-1}(n-l)l \\ &= 2\sum_{l=1}^{n-1} nl - 2\sum_{l=1}^{n-1} l^2 \\ &= 2\frac{n(n-1)n}{2} - 2\frac{(n-1)n(2n-1)}{6} \\ &= n^3 - n^2 - \frac{2n^3 - 3n^2 + n}{3} \\ &= \frac{n^3 - n}{3}\,. \end{aligned}$$

5. We say that a problem exhibits the optimal substructure property when optimal solutions to a problem incorporate optimal solutions to related subproblems, which we may solve independently. When we impose a limit l_i on the number of pieces of size i that we are permitted to produce, the subproblems can no longer be solved independently. For example, consider a rod of length 4 with the following prices and limits:



length i	1	2	3	4
price p_i	15	20	33	36
limit l_i	2	1	1	1

This instance has only three solutions that do not violate the limits: length 4 with price 36; lengths 1 and 3 with price 48; and lengths 1, 1, and 2 with price 50. The optimal solution, therefore is to cut into lengths 1, 1, and 2. When we look at the subproblem for length 2, it has two solutions that do not violate the limits: length 2 with price 20, and lengths 1 and 1 with price 30. The optimal solution for length 2, therefore, is to cut into lengths 1 and 1. But we cannot use this optimal solution for the subproblem in the optimal solution for the original problem, because it would result in using four rods of length 1 to solve the original problem, violating the limit of two length-1 rods.

6. The claimed optimal solution $\{a_2, a_4, a_9, a_{11}\}$ follows directly by the procedure on page 125 of the Lecture Notes when picking up these four non-overlapping activities.

7. The optimal way is to sort A and B into monotonically decreasing (or increasing) order.

Here is a proof that this method yields an optimal solution. Consider any indices i and j such that $i < j$, and consider the terms $a_i^{b_i}$ and $a_j^{b_j}$. We want to show that it is no worse to include these terms in the payoff than to include, i.e., that $a_i^{b_i} a_j^{b_j} \geq a_i^{b_j} a_j^{b_i}$. Since A and B are sorted into monotonically decreasing order and $i < j$, we have $a_i \geq a_j$ and $b_i \geq b_j$. Since a_i and a_j are positive and $b_i - b_j$ is nonnegative, we have $a_i^{b_i - b_j} \geq a_j^{b_i - b_j}$. Multiplying both sides by $a_i^{b_j} a_j^{b_j}$ yields $a_i^{b_i} a_j^{b_j} \geq a_i^{b_j} a_j^{b_i}$.

Since the order of multiplication doesn't matter, sorting A and B into monotonically increasing order works as well.