Assignment 3

Name: Mousumi Akter
SID: 904095347
I Discussed with: Alex Knipper and Md Mahadi Hassan

March 28, 2021

1. Problem 15.2-1 in the text. Show the s and m matrices (like Figure 15.5) and then provide the optimal parenthesization.

Matrix	A1	A2	A3	A4	A5	A6
Dimension	5×10	10×3	3×12	12×5	5×50	50×6

p0 = 5, p1 = 10, p2 = 3, p3 = 12, p4 = 5, p5 = 50, p6 = 6

m(i, i)=0 for all i ∈ [1,6]

m(1,2) = 5×10×3 = 150; m(2,3) = 10×3×12 = 360; m(3,4) = 3×12×5 = 180; m(4,5) = 12×5×50 = 3000; m(5,6) = 5×50×6 = 1500;

Using the below formula we get:

$$m[i,j] = \begin{cases} 0 & \text{if } i = j\ , \\ \min\limits_{i \le k < j} \{m[i,k] + m[k+1,j] + p_{i-1} p_k p_j\} & \text{if } i < j\ . \end{cases}$$

$$\text{m(1,3)} = min \begin{cases} m(1,1) + m(2,3) + p0 * p1 * p3 = 0 + 360 + 5 * 10 * 12 = 0 + 360 + 600 = 960 \\ m(1,2) + m(3,3) + p0 * p2 * p3 = 150 + 0 + 5 * 3 * 12 = 150 + 0 + 180 = 330 \end{cases} = 330$$

$$\text{m(2,4)} = min \begin{cases} m(2,2) + m(3,4) + p1 * p2 * p4 = 0 + 180 + 10 * 3 * 5 = 180 + 150 = 330 \\ m(2,3) + m(4,4) + p1 * p3 * p4 = 360 + 0 + 10 * 12 * 5 = 360 + 600 = 960 \end{cases} = 330$$

$$\text{m(3,5)} = min \begin{cases} m(3,3) + m(4,5) + p2 * p3 * p5 = 0 + 3000 + 3 * 12 * 50 = 4800 \\ m(3,4) + m(5,5) + p2 * p4 * p5 = 180 + 0 + 3 * 5 * 50 = 930 \end{cases} = 930$$

$$\text{m(4,6)} = min \begin{cases} m(4,4) + m(5,6) + p3 * p4 * p6 = 0 + 1500 + 12 * 5 * 6 = 1500 + 360 = 1860 \\ m(4,5) + m(6,6) + p3 * p5 * p6 = 3000 + 0 + 12 * 50 * 6 = 3000 + 3600 = 6600 \end{cases} = 1860$$

$$\text{m(1,4)} = min \begin{cases} m(1,1) + m(2,4) + p0 * p1 * p4 = 0 + 330 + 5 * 10 * 5 = 330 + 250 = 580 \\ m(1,2) + m(3,4) + p0 * p2 * p4 = 150 + 180 + 5 * 3 * 5 = 330 + 75 = 405 = 405 \\ m(1,3) + m(4,4) + p0 * p3 * p4 = 330 + 0 + 5 * 12 * 5 = 330 + 300 = 630 \end{cases}$$

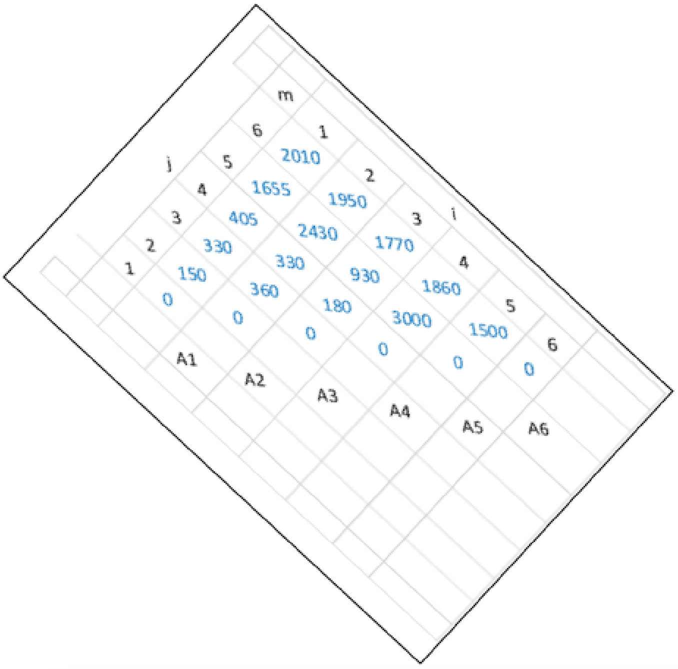
$$\text{m(2,5)} = min \begin{cases} m(2,2) + m(3,5) + p1 * p2 * p5 = 0 + 930 + 10 * 3 * 50 = 930 + 1500 = 2430 \\ m(2,3) + m(4,5) + p1 * p3 * p5 = 360 + 3000 + 10 * 12 * 50 = 3360 + 6000 = 9360 = 2430 \\ m(2,4) + m(5,5) + p1 * p4 * p5 = 330 + 0 + 10 * 5 * 50 = 330 + 2500 = 2830 \end{cases}$$

$$\text{m(3,6)} = min \begin{cases} m(3,3) + m(4,6) + p2 * p3 * p6 = 0 + 1860 + 3 * 12 * 6 = 1860 + 216 = 2076 \\ m(3,4) + m(5,6) + p2 * p4 * p6 = 180 + 1500 + 3 * 5 * 6 = 1770 \qquad = 1770 \\ m(3,5) + m(6,6) + p2 * p5 * p6 = 930 + 0 + 3 * 50 * 6 = 1830 \end{cases}$$

$$\text{m(1,5)} = min \begin{cases} m(1,1) + m(2,5) + p0 * p1 * p5 = 0 + 2430 + 5 * 10 * 50 = 4930 \\ m(1,2) + m(3,5) + p0 * p2 * p5 = 150 + 930 + 5 * 3 * 50 = 1830 \\ m(1,3) + m(4,5) + p0 * p3 * p5 = 330 + 3000 + 5 * 12 * 50 = 6330 \\ m(1,4) + m(5,5) + p0 * p4 * p5 = 405 + 0 + 5 * 5 * 50 = 1655 \end{cases} = 1655$$

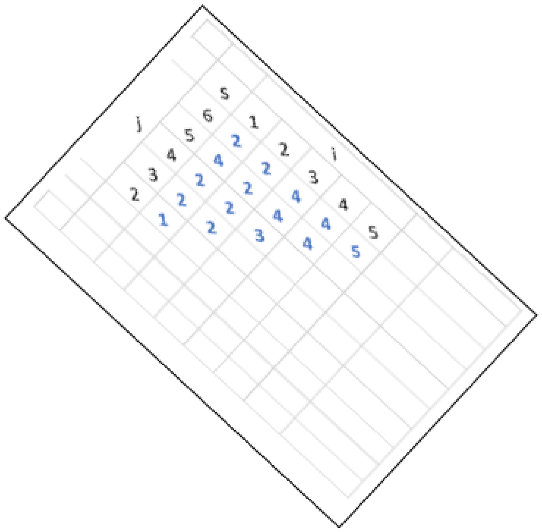
$$\text{m(2,6)} = min \begin{cases} m(2,2) + m(3,6) + p1 * p2 * p6 = 0 + 1770 + 10 * 3 * 6 = 1950 \\ m(2,3) + m(4,6) + p1 * p3 * p6 = 360 + 1860 + 10 * 12 * 6 = 2940 \\ m(2,4) + m(5,6) + p1 * p4 * p6 = 330 + 1500 + 10 * 5 * 6 = 2130 \\ m(2,5) + m(6,6) + p1 * p5 * p6 = 2430 + 0 + 10 * 50 * 6 = 2630 \end{cases} = 1950$$

$$\text{m(1,6)} = min \begin{cases} m(1,1) + m(2,6) + p0 * p1 * p6 = 0 + 1950 + 5 * 10 * 6 = 2250 \\ m(1,2) + m(3,6) + p0 * p2 * p6 = 150 + 1770 + 5 * 3 * 6 = 2010 \\ m(1,3) + m(4,6) + p0 * p3 * p6 = 330 + 1860 + 5 * 12 * 6 = 2550 = 2010 \\ m(1,4) + m(5,6) + p0 * p4 * p6 = 405 + 1500 + 5 * 5 * 6 = 2055 \\ m(1,5) + m(6,6) + p0 * p5 * p6 = 1655 + 0 + 5 * 50 * 6 = 3155 \end{cases}$$



m

j \ i	1	2	3	4	5	6
6	2010	1950	1770	1860	1500	0
5	1655	2430	930	3000	0	
4	405	330	180	0		
3	330	360	0			
2	150	0				
1	0					
	A1	A2	A3	A4	A5	A6



s

j \ i	1	2	3	4	5
6	2	2	4	4	5
5	4	2	4	4	
4	2	2	3		
3	2	2			
2	1				

Let us call PRINT-OPTIMAL-PARENS in the text book as POP in short:

POP(s,1,6)

= (POP(s,1,2)POP(s,3,6))

=((POP(s,1,1)POP(s,2,2))(POP(s,3,4)POP(s,5,6)))

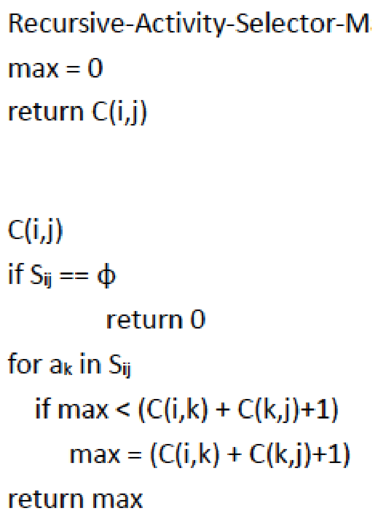
=((A1A2)((POP(s,3,3)POP(s,4,4))(POP(s,5,5)POP(s,6,6))))

= ((A1A2)((A3A4)(A5A6)))

Therefore the optimal parenthesization is (A1A2)((A3A4)(A5A6)).

2. Convert the recursive characterization of equations (16.2) in text into a recursive algorithm and provide the algorithm below.

   S_{ij} is the set of activities that has all the activities that starts after the finish of a_i and ends before the beginning of a_j. Mentioned below, C(i,j) is the recursive algorithm and Recursive-Activity-Selector-MaximumSize is the main algorithm that calls C(i,j).

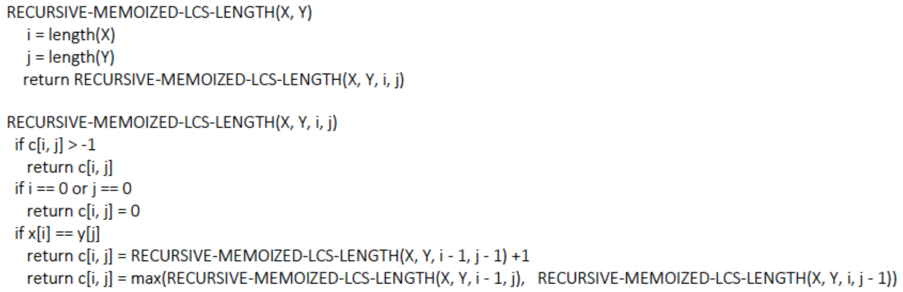


```
Recursive-Activity-Selector-MaximumSize(i,j, $S_{ij}$)
max = 0
return C(i,j)


C(i,j)
if $S_{ij}$ == $\phi$
        return 0
for $a_k$ in $S_{ij}$
   if max < (C(i,k) + C(k,j)+1)
        max = (C(i,k) + C(k,j)+1)
return max
```

3. Write a memorized recursive algorithm RECURSIVE-MEMOIZED-LCS-LENGTH(X,Y) to compute the length of the LCS of X and Y based on equations (15.9), p. 393.

   Let's assume c[i, j] be the length of an LCS of the sequences X_i and Y_j.



```
RECURSIVE-MEMOIZED-LCS-LENGTH(X, Y)
    i = length(X)
    j = length(Y)
   return RECURSIVE-MEMOIZED-LCS-LENGTH(X, Y, i, j)

RECURSIVE-MEMOIZED-LCS-LENGTH(X, Y, i, j)
  if c[i, j] > -1
    return c[i, j]
  if i == 0 or j == 0
    return c[i, j] = 0
  if x[i] == y[j]
    return c[i, j] = RECURSIVE-MEMOIZED-LCS-LENGTH(X, Y, i - 1, j - 1) +1
    return c[i, j] = max(RECURSIVE-MEMOIZED-LCS-LENGTH(X, Y, i - 1, j),   RECURSIVE-MEMOIZED-LCS-LENGTH(X, Y, i, j - 1))
```

Please read 7270-09-DP Part I.pdf, pp 29 for the following two questions, which is also placed here:

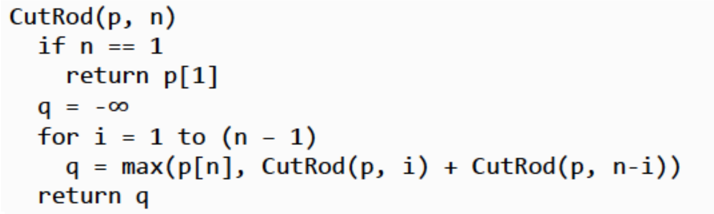
- Another way of characterizing the structure of the optimal solution to this problem recursively is to say: we can first cut the rod into two pieces of length i and n-i inches each, and then recursively calculate the optimal cuts for each of those pieces; if we do that for each possible value of i from 1 to n-1 and calculate the total revenue ri+rn-i for each of those possible cuts, and then take the maximum of those and pn (the revenue if the rod is sold without cutting), that will give us the optimal revenue rn.
- This is formulated as equation 15.1 (p.362). Understand this equation.

1. Develop a corresponding recursive algorithm.
2. Show how this algorithm duplicates work by drawing a recursion tree for a specific input and pointing out duplicate recursive executions.
3. Modify the recursive algorithm to make it more efficient using memoization. Is this more/less/as efficient as Memoized-Cut-Rod?
4. Develop an algorithm to compute the optimal solution in a bottom up fashion using table lookup. Is it more/less/as efficient as Bottom-Up-Cut-Rod?

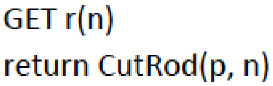
4. Do the Questions 1 & 2. The specific input for which you would draw a recursion Tree should be a rod of length 4 inches.

   Equation 15.1 is: r_n = max $(p_n,\ r_1 + r_{n-1},\ r_2 + r_{n-2},\ ....,\ r_{n-1} + r_1)$.

   Let n be the length of the rod in inches and p be the array p[i] of prices for prices for i=1 through n. Equation 15.1 has been developed into a recursive algorithm as below:



```
CutRod(p, n)
  if n == 1
    return p[1]
  q = -∞
  for i = 1 to (n - 1)
    q = max(p[n], CutRod(p, i) + CutRod(p, n-i))
  return q
```

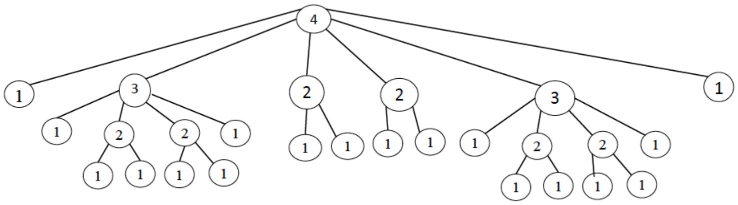


```
GET r(n)
return CutRod(p, n)
```

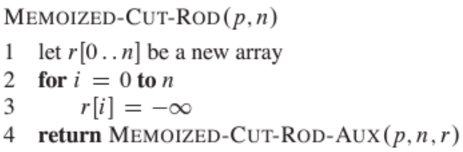
For the second part of the question, for n = 4, we need a recursion tree based on the above algorithm and show that there is duplication of work. It is given that:

Length (i)	1	2	3	4
Price (P[i])	1	5	8	9

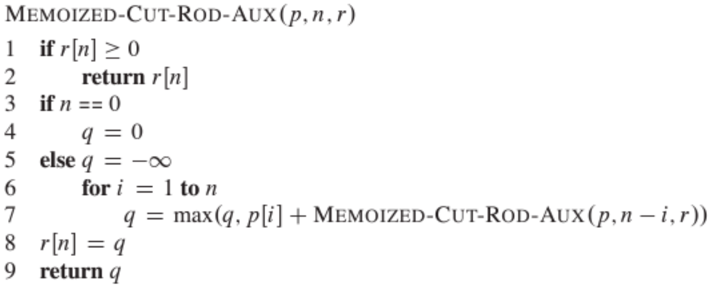
Let us denote CutRod(p, 4) by 4, CutRod(p, 3) by 3, and so on. Thus, we have following recursion tree. From the recursion tree we find that there are many duplication work for CutRod(p, 3), CutRod(p, 2) and CutRod(p, 1).



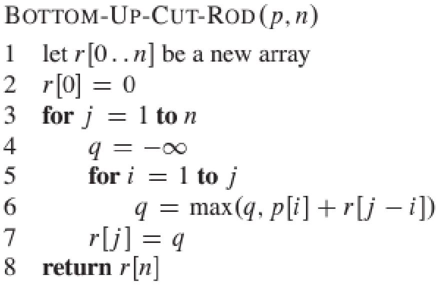
5. Do the Questions 3 & 4. As part of your answer. For Q4, you must explain the lookup table – what it's dimensions are and the order in which its cells will be filled by the algorithm.



```
MEMOIZED-CUT-ROD(p, n)
let r[0..n] be a new array
for i = 0 to n
    r[i] = −∞
return MEMOIZED-CUT-ROD-AUX(p, n, r)
```



```
MEMOIZED-CUT-ROD-AUX(p, n, r)
1  if $r[n] \geq 0$
2      return $r[n]$
3  if $n == 0$
4      $q = 0$
5  else $q = -\infty$
6      for $i = 1$ to $n$
7          $q = \max(q, p[i] + \text{MEMOIZED-CUT-ROD-AUX}(p, n - i, r))$
8  $r[n] = q$
9  return $q$
```



```
BOTTOM-UP-CUT-ROD(p, n)
1  let $r[0..n]$ be a new array
2  $r[0] = 0$
3  for $j = 1$ to $n$
4      $q = -\infty$
5      for $i = 1$ to $j$
6          $q = \max(q, p[i] + r[j - i])$
7      $r[j] = q$
8  return $r[n]$
```

For Bottom-up cut rod,

- Each r[1] to r[n] will give the optimal revenue of length from 1 to n.
- We start from bottom, that is, r[1] is calculated first. Loops 6-7 will be iterated once in this case. Then q will be max(-∞, p[1]). This will return p[1] and will be saved as r[1].
- In the next stage, we have r[2], which will be compared with 2r[1]'s and p[2]. The maximum of the two will be returned as r[2].
- In the same manner for r[i], p[i] will be compared with 2^{i-1} combinations of rod cuts and the maximum value will be returned as r[i].
- Therefore, for the n^{th} iteration, p[n] will be compared with 2^{n-1} combinations of rod cuts and the maximum value will be returned as r[n] finally.