Homework 1

CSCE 411
Sahil Palnitkar

15.1-2)

Assume rod of length 4, n =4 p1 = 1, p2 = 5, p3 = 8, p4 = 9

A greedy algorithm will cut the rod in pieces of 3 and 1 with prices 8+1 = 9

But the optimal solution is cutting the rod in pieces of 2 and 2 with prices 5+5 = 10

Therefore, this proves that a greedy algorithm is not always optimal.

15.1-3)

If the bottom-up-cut rod algorithm is considered, we can add the cost of cutting the rod into the parentheses to take it into account.

The modified algorithm would be q = max(q, p[i] + r[j-i] -c)

15.2-1)

m-table

j/i	1	2	3	4	5	6
1	0					
2	150	0				
3	330	360	0			
4	405	330	180	0		
5	1655	2430	930	930	0	
6	2010	1950	1770	1860	1500	0

s-table

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

According to the s-table, the optimal parenthesization is $(A_1A_2)((A_3A_4)(A_5A_6))$

15.2-6)

We can prove this using induction

Inductive hypothesis: Assuming a full parenthesization of an n-element expression has n-1 pairs of parentheses.

Basis step: Let n = 2, multiplying 2 matrices A_1 and A_2, the unparenthesized product is A_1A_2. With only one multiplication operation, the only parenthesization is (A_1A_2)

Induction step: Considering a product of n = k+1 matrices, A_1, A_2, … , A_k, A_{k+1}. If the original parenthesization splits the product at the jth element, the optimal solution has the recursive structure: $((A_1A_2 \ldots A_j)(A_{j+1}A_{j+2} \ldots A_{k+1}))$. The subproducts are solved optimally recursively. The two subproducts require j-1 and ((k+1) – j+1) -1 parentheses respectively. Therefore, the inductive hypothesis gives us the total number of parentheses = k-1

Adding the outer parenthesis gives us total parentheses = (k-1)+1 = k

Therefore, a full parenthesization of n-element expression has exactly n-1 pairs of parentheses.