CS 570: Analysis of Algorithms – H8

Submitted by: Indronil Bhattacharjee

Exercise 15.2-1

Q- Find an optimal parenthesization of a matrix-chain product whose sequence of dimensions is [5,10,3,12,5,50,6].

Algorithm for computing the optimal costs using Dynamic Programming

```
MATRIX-CHAIN-ORDER(p):
	n = length[p] – 1
	for i = 1 to n
		m[i, i] = 0
	for l = 2 to n
		for i = 1 to n – l + 1
			j = i + l – 1
			m[i, j] = ∞
			for k = i to j – 1
				q = m[i, k] + m[k + 1, j] + p[i-1]*p[k]*p[j]
				if q < m[i, j]
					m[i, j] = q
					s[i, j] = k
	return m and s
```

Algorithm for constructing an optimal solution

```
PRINT-OPTIMAL-PARENS(s, i, j):
	if i = j
		print “A”i
	else
		print “(”
		PRINT-OPTIMAL-PARENS(s, i, s[i, j])
		PRINT-OPTIMAL-PARENS(s, s[i, j] + 1, j)
		print “)”
```

Solution for ⟨5,10,3,12,5,50,6⟩

```
p = [5, 10, 3, 12, 5, 50, 6]
m, s = MATRIX-CHAIN-ORDER(p)
PRINT-OPTIMAL-PARENS(s, 1, 6)
```

m =

	6	5	4	3	2	1
6	.	.	.	.	.	.
5	1500	.	.	.	.	.
4	1860	3000	.	.	.	.
3	1770	930	180	.	.	.
2	1950	2430	330	360	.	.
1	2010	1655	405	330	150	.

Output

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

s =

	6	5	4	3	2	1
6	.	.	.	.	.	.
5	5	.	.	.	.	.
4	4	4	.	.	.	.
3	4	4	3	.	.	.
2	2	2	2	2	.	.
1	2	4	2	2	1	.

Exercise 15.2-2

Q- Give a recursive algorithm MATRIX-CHAIN-MULTIPLY(A, s, i, j) that actually performs the optimal matrix-chain multiplication, given the sequence of matrices ⟨A1, A2, …, An⟩, the s table computed MATRIX-CHAIN-ORDER, and the indices i and j. (The initial call would be MATRIX-CHAIN-MULTIPLY(A, s, 1, n).)

```
MATRIX-CHAIN-MULTIPLY(A, s, i, j):
    if i == j:
        return A[i - 1]
    else:
        A_left = MATRIX-CHAIN-MULTIPLY(A, s, i, s[i][j])
        A_right = MATRIX-CHAIN-MULTIPLY(A, s, s[i][j] + 1, j)
        return MATRIX-MULTIPLY(A_left, A_right)


MATRIX-MULTIPLY(A, B):
    m = number of rows in matrix A
    n = number of columns in matrix A
    p = number of columns in matrix B
    Let C be a new m x p matrix
    for i = 1 to m:
        for j = 1 to p:
            C[i][j] = 0
            for k = 1 to n:
                C[i][j] = C[i][j] + A[i][k] * B[k][j]
    return C


A = [A1, A2, ..., An]
s = computed s table by MATRIX-CHAIN-ORDER
n = length of A
result = MATRIX-CHAIN-MULTIPLY(A, s, 1, n)
```

MATRIX-CHAIN-MULTIPLY(A, s, i, j):

- This function recursively multiplies matrices in the given sequence A between indices i and j.
- If i equals j, it means there's only one matrix left in the chain, so it returns that matrix.
- Otherwise, it splits the chain at index s[i][j], recursively multiplies the left and right subchains, and then multiplies the resulting matrices using the helper function MATRIX-MULTIPLY.

MATRIX-MULTIPLY(A, B):

- This function performs matrix multiplication between matrices A and B.
- It initializes a new matrix C of dimensions m x p. It iterates over each element C[i][j] of the resulting matrix and computes its value by summing up products of corresponding elements from matrices A and B.
- Finally, it returns the resulting matrix C.