EXPT. NO. | NAME | Page No.: | Date: | YOUVA

Q. Write a program to perform matrix chain multiplication. Print optimal order of parenthesis with cost of operations, cost matrix(S) and subproblem matrix(k/m).

ALGORITHM:

Chain Multiplication:

Algorithm MatrixChainMultiplication(P, n)

// Computes the cost matrix(k/m) and subproblem matrix(S)

Input: Size array P, no. of matrices n

Output: Cost matrix m, subproblem matrix S

BEGIN:

1. for i=1 to n
2. m[i, i] = 0
3. for l=2 to n
4. for i=1 to n-l+1
5. j = i+l-1
6. m[i, j] = ∞
7. for k=i to j-1
8. q = m[i, k] + m[k+1, j] + $P_{i-1} P_k P_j$
9. if q < m[i, j]
10. m[i, j] = q, S[i, j] = k
11. return m and S

END.

Teacher's Signature:

OUTPUT:

```
Enter no. of matrices to multiply: 6

Enter rows for matrix 1: 30
Enter columns for matrix 1: 35

Enter rows for matrix 2: 35
Enter columns for matrix 2: 15

Enter rows for matrix 3: 15
Enter rows for matrix 3: 5

Enter rows for matrix 4: 5
Enter columns for matrix 4: 10

Enter rows for matrix 5: 10
Enter columns for matrix 5: 20

Enter rows for matrix 6: 20
Enter columns for matrix 6: 25

P is [30 35 15 5 10 20 25 ]

Starting time: 0.003358
End time: 0.003404
Time taken: 4.6e-05 seconds
```

EXPT. NO. NAME

M T W T F S S

Page No.:

Date:

YOUVA

Optimal Parenthesis:

Algorithm Print OptimalParenthesis (S, i, j)

// Prints the optimal order of matrix from matrix i to j

Input: Subproblem matrix S, starting index i, end index j

Output: Prints optimal set of parenthesis

BEGIN:

1. if i=j
2. print "A;"
3. else
4. print "("
5. print OptimalParenthesis (S, i, S[i, j])
6. Print OptimalParenthesis (S, S[i, j]+1, j)
7. Print ")"

END

SOURCE CODE:

```
#include <iostream>
#include <ctime>
using namespace std;

void matrixChainMul(int *P, int n, int **m, int **S) {
    for (int i = 0; i<n; i++)
        m[i][i] = 0;

    for (int l=1; l<n; l++) {
        for (int i=0; i<n-l; i++) {
            int j= i+l;
```

Teacher's Signature:

m matrix is:

[0	15750	7875	9375	11875	15125]
[0	0	2625	4375	7125	10500]
[0	0	0	750	2500	5375]
[0	0	0	0	1000	3500]
[0	0	0	0	0	5000]
[0	0	0	0	0	0]

S matrix is:

[0	0	0	2	2	2]
[0	0	1	2	2	2]
[0	0	0	2	2	2]
[0	0	0	0	3	4]
[0	0	0	0	0	4]
[0	0	0	0	0	0]

Optimal parenthesis are: ((A1 (A2 A3)) ((A4 A5) A6))
Cost of opercation is 15125

EXPT. NO. NAME

M T W T F S S

Page No.:

Date:

YOUVA

```
            m[i][j] = INT32_MAX;
            for (int k=i; k<j; k++) {
                int q = m[i][k] + m[k+1][j] + p[i] * p[k+1] * p[j+1];
                if (q < m[i][j]) {
                    m[i][j] = q;
                    S[i][j] = k;
                }
            }
        }
    }
}

void printOptimalParen(int **S, int i, int j) {
    if (i == j) cout << "A" << i;
    else {
        cout << "(";
        printOptimalParen(S, i, S[i-1][j-1]+1);
        printOptimalParen(S, S[i-1][j-1]+2, j);
        cout << ")";
    }
}

void printSqMat(int **mat, int n) {
    for (int i=0; i<n; i++) {
        cout << "[";
        for (int j=0; j<n; j++)
            cout << mat[i][j] << " ";
        cout << "]\n";
    }
}
```

Teacher's Signature:

EXPT. NO.

NAME

M T W T F S S

Page No.:

Date:

YOUVA

```
int main() {
    int n;
    cout << "Enter no. of matrices to multiply: ";
    cin >> n;

    int sizes[n][2];
    for (int i = 0; i < n; i++) {
        cout << "\nEnter rows for matrix " << i+1 << ": ";
        cin >> sizes[i][0];
        cout << "Enter columns for matrix: " << i+1 << ": ";
        cin >> sizes[i][1];
    }
    int P[n+1];
    P[0] = sizes[0][0];
    P[1] = sizes[0][1];
    for (int i = 2; i < n; i++) {
        if (sizes[i-2][1] != sizes[i-1][0]) {
            cout << "Error: Cannot multiply matrix A" << i-1 <<
                " and A" << i << endl;
            exit(0);
        }
        P[i] = sizes[i-1][1];
    }
    P[n] = sizes[n-1][1];
```

Teacher's Signature:

```
cout << "\nP is [";
for (int i = 0; i < n+1; i++)
    cout << P[i] << " ";
cout << "]\n";

int **m = new int *[n];
int **S = new int *[n];
for (int i=0; i<n; i++) {
    m[i] = new int[n];
    S[i] = new int[n];
}

double start = (double) clock() / CLOCKS_PER_SEC;
cout << "\nStarting time: " << start << endl;
matrixChainMul(P, n, m, S);
double end = (double) clock() / CLOCKS_PER_SEC;
cout << "End time: " << end << endl;
cout << "Time taken: " << (end - start) << " seconds " << endl;

cout << "\n m matrix is: " << endl;
printSqMat(m, n);

cout << "\n S matrix is: " << endl;
printSqMat(S, n);
```

Teacher's Signature:

```
cout << "\n Optimal parenthesis are: ";
printOptimalParen(S, 1, n);
cout << "\n Cost of operation is " << m[0][n-1] << endl;

delete []S;
delete []m;

return 0;
}
```

ANALYSIS:

Time taken by matrix chain multiplication algorithm is 4.6×10^{-5} seconds

Time Complexity:

Teacher's Signature: _____