Tejas Tripathi

Sub: Algorithm Analysis and Design

Given a sequence of matrices, we want to find the most efficient way to multiply these matrices together to obtain the minimum number of multiplications. The problem is not actually to perform the multiplication of the matrices but to obtain the minimum number of multiplications.

We have many options because matrix multiplication is an associative operation, meaning that the order in which we multiply does not matter. The optimal order depends only on the dimensions of the matrices.

The brute-force algorithm is to consider all possible orders and take the minimum. This is a very inefficient method.

Implement the minimum multiplication algorithm using dynamic programming and determine where to place parentheses to minimize the number of multiplications.

Find an optimal paranthsization of matrix chain product whose sequence of dimension is (5, 10, 3, 12, 5, 50, 6). (use dynamic programming)

CODE:

```
dim=[int(i) for i in input('Enter the dimensions: ').split(' ')]
matr=[]
k=[]
min_of_k=[]
min_of_out=[]
selected_k=[]
k_arr=[]
def functions(mmatr, ii, jj, kk):
    out=mmatr[ii][kk]+mmatr[kk+1][jj]+(dim[ii-1]*dim[kk]*dim[jj])
    return out


def brackets(matrix, initial, end):
    if initial == end:
```

```
            print('A{}'.format(initial), end='')
            return
        k = matrix[initial][end]
        print('(', end='')
        brackets(matrix, initial, k)
        brackets(matrix, k + 1, end)
        print(')', end='')

    for i in range(len(dim)):
        a=[]
        for j in range(len(dim)):
          a.append(0)
        matr.append(a)

    for i in range(len(dim)):
        a=[]
        for j in range(len(dim)):
          a.append(0)
        k_arr.append(a)



    for jj in range(1, len(dim)-1):
        for i in range(1, len(dim)):
            for j in range(1, len(dim)):
                if j-i == jj:

                    for a in range(i, j):
                        k.append(a)
                    for b in k:
                        output=functions(matr, i, j, b)
                        min_of_k.append(b)
```

```
                min_of_out.append(output)
            matr[i][j]=min(min_of_out)
            selected_k.append(min_of_k[min_of_out.index(matr[i][j])])
            k_arr[i][j]=min_of_k[min_of_out.index(matr[i][j])]
        k.clear()
        min_of_out.clear()
        min_of_k.clear()

print('\nFinal counting matrix: ')
for i in matr:
    print(*i)

print('\nK-value matrix: ')
for i in k_arr:
    print(*i)

print('The order of multiplication should be: ', end='')
brackets(k_arr, 1, len(dim)-1)
```

OUTPUT:

Tejas Tripathi

```
In [25]: runfile('C:/Users/Admin/study material/sem5/
Practicals/Algorithms/Practical-8/
matrix_multiplication.py', wdir='C:/Users/Admin/study
material/sem5/Practicals/Algorithms/Practical-8')

Enter the dimensions: 5 10 3 12 5 50 6

Final counting matrix:
0 0 0 0 0 0 0
0 0 150 330 405 1655 2010
0 0 0 360 330 2430 1950
0 0 0 0 180 930 1770
0 0 0 0 0 3000 1860
0 0 0 0 0 0 1500
0 0 0 0 0 0 0

K-value matrix:
0 0 0 0 0 0 0
0 0 1 2 2 4 2
0 0 0 2 2 2 2
0 0 0 0 3 4 4
0 0 0 0 0 4 4
0 0 0 0 0 0 5
0 0 0 0 0 0 0
The Optimal Paranthsization of matrix chain product
will be: ((A1A2)((A3A4)(A5A6)))

In [26]:
```