ASSIGNMENT - 4

ANIKEIT SETHI(190001003)

**Algorithm:** -

* Take the sequence of matrices and separate it into two sub sequences.
* Find the minimum cost of multiplying out each subsequence.
* Add these costs together, and add in the price of multiplying the two result matrices.
* Do this for each possible position at which the sequence of matrices can be split, and take the minimum over all of them.

Code:-

import sys  
  
  
# Function to find the most efficient way to multiply  
# a given sequence of matrices  
def matrixChainMultiplication(dims, i, j):  
 # base case: one matrix  
 if j <= i + 1:  
 return 0  
  
 # stores the minimum number of scalar multiplications (i.e., cost)  
 # needed to compute matrix `M[i+1] … M[j] = M[i…j]`  
 min = sys.maxsize  
  
 # take the minimum over each possible position at which the  
 # sequence of matrices can be split  
  
 '''  
 (M[i+1]) × (M[i+2]………………M[j])  
 (M[i+1]M[i+2]) × (M[i+3…………M[j])  
 …  
 …  
 (M[i+1]M[i+2]…………M[j-1]) × (M[j])  
 '''  
  
 for k in range(i + 1, j):  
  
 # recur for `M[i+1]…M[k]` to get an `i × k` matrix  
 cost = matrixChainMultiplication(dims, i, k)  
  
 # recur for `M[k+1]…M[j]` to get an `k × j` matrix  
 cost += matrixChainMultiplication(dims, k, j)  
  
 # cost to multiply two `i × k` and `k × j` matrix  
 cost += dims[i] \* dims[k] \* dims[j]  
  
 if cost < min:  
 min = cost  
  
 # return the minimum cost to multiply `M[j+1]…M[j]`  
 return min  
  
  
# Matrix Chain Multiplication Problem  
if \_\_name\_\_ == '\_\_main\_\_':  
  
 dims = [40, 60, 20, 10, 5, 80]  
  
 print("The minimum cost is", matrixChainMultiplication(dims, 0, len(dims) - 1))

Output: -

