MATRIX CHAIN ACTIVITY

SYED NAQVI
100590852

```
import numpy as np
```

[2] ✓ 0.0s

```
def matrix_chain_order(p):

    # the actual number of matrices is one less than the number of dimensions
    n = len(p) - 1

    # the first row and column will be redundant, will just contain all zeros
    m = np.zeros((n+1,n+1))
    s = np.zeros((n+1,n+1))

    # this part of the algorithm assigns correct costs to each diagonal of the matrix m as l(chain length) increases
    for l in range(2, n+1):
        for i in range(1, n-l+2):
            j = i + l - 1
            # temporarily assigning max value to m[i,j] so that calculated multiplication amounts will always be less
            m[i][j] = float('inf')
            # finding and storing the optimal k value into the s matrix
            # finding and assigning the minimum multiplication cost for multiplying matrices i through j
            for k in range(i, j):
                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, s

# Test
p = [3, 7, 4, 5, 2]
m, s = matrix_chain_order(p)
print(m[1][len(p)-1])  # Minimum number of multiplications needed
```

[3] ✓ 0.0s

```
138.0
```