**MSCF Financial Computing I**

**Mini 1, 2022**

**Homework 5**

***Due At 11:59 pm Sunday, Oct. 2, 2022***

***You will lose 10 points per hour after that time***

1. **Dynamic Programming (100 points)**

In MSCF Financial Computing I Week 5, Part 2, we described the dynamic programming algorithm for finding the lowest cost ordering for **Matrix Chain Multiplication** of *n* matrices.

The code file **hw5\_1.py** contains test code for this algorithm, based on example matrix (2-dimensional **ndarray**) chains from the lecture slides and on some randomly generated chains. Each chain is represented as a **list** of **ndarray** objects.

For example, this is taken from the example on Lecture slides 4 and 5. The specific values of the matrix cells don’t matter, so we use **np.ones((rows, cols))** to create each matrix in the chain (the *shape* argument is a **tuple**). The **MatrixChainMult()** function takes a matrix chain as its argument, and returns a **tuple** containing the *minimum number of operations* required for the multiplication, and a **string** showing the *ordering of multiplications*.

**A\_chain\_1 = [ np.ones((20,4)), np.ones((4,10)),**

**np.ones((10,5)) ]**

**nops, order = MatrixChainMult(A\_chain\_1)**

**print('Matrix dimensions are:')**

**for m in range(len(A\_chain\_1)):**

**print('A' + str(m+1) + ':', A\_chain\_1[m].shape)**

**# A1: (20, 4)**

**# A2: (4, 10)**

**# A3: (10, 5)**

**print('Chain requires', nops, 'operations.') # 600**

**print('Ordering is:')**

**print(order) # A1 (A2 A3)**

Write the **MatrixChainMult()** function.

Save and test **hw5\_1.py**.

***REMEMBER*** to put all team members’ names and Andrew IDs into your source code files.One team member should put your **hw5\_1.py** file into a **Team\_***N***\_HW5.zip** archive, where *N* is your team number, and upload to Canvas.