Assignment-2

(Total Points: 100)

Instructions:

- Download the attached python file assignment_2.py (make sure to **NOT** change the name of this file).
- Follow the instructions and **replace** all TODO comments in the scaffolding code.
- Do **NOT** change the function signature provided in the scaffolding code, but you can add as many functions as you want to help you complete the task.
- Test your code as much as you can to make certain it is correct.
- Run flake8 in addition to testing your code; I expect professional and clear code with minimal flake8 warnings and having McCabe complexity (<10) from all of you.
- Create a write up with formatted code and screenshots of your output, running the McCabe complexity command and error free console.
- Save the write-up as a PDF and submit it along with your python code (file name assignment_2.py) as separate attachments before the due date on blackboard.

Note: Running flake 8

flake8 path/to/your/file (for warnings and errors)

flake8 --max-complexity 10 path/to/your/file (for complexity)

Problem 1 (Max Points 50):

The Sun is the source of most of the energy that drives the biological and physical processes in the world around us—in oceans and on land it fuels plant growth that forms the base of the food chain, and in the atmosphere, it warms air which drives our weather.

The rate of energy coming from the Sun changes slightly every second. However, NASA scientist want to focus on the **most significant energy increase period** in a day, so that they may get better understanding about the Sun. They have collected the measurement of the energy level in a range of time, starting from 0. Your task is to design a program finding the **most significant energy increases period** from their daily observation.

For example, they observed a sequence of energy level as follows,
100,113,110,85,105,102,86,63,81,101,94,106,101,79,94,90,97
Your program should return (7, 11) since from element at index 7 (63) to element at index 11 (106) gives the most significant energy increase. Note that the indices start from **zero**.

You need to implement a Python program to solve this problem using:

A. The brute-force method $\Theta(n^2)$. (Max Points: 10)
B. The recursive method $\Theta(nlogn)$. (Max Points: 25)
C. The iterative method $\Theta(n)$. (Max Points: 15)

Sample Input

[100,113,110,85,105,102,86,63,81,101,94,106,101,79,94,90,97]

Sample Output

(7, 11)

Note, if their observation data looks like 110 109 107 104 100, in this case your program should return (0, 1).

Problem 2 (Max Points 50):

Given a $n \times n$ matrix A and a positive integer k, compute $S = A^k$.

The input contains the following

- A matrix A containing $n * n$ nonnegative integers (each less than 100). You can assume that $n \leq 32$ and is a power of 2, e.g. 1, 2, 4, 8, 16, 32.
- A positive integer k ($k \leq 10$).

You need to do the following tasks:

A. Implement a function to multiply two matrices using Strassen Matrix Multiplication method $\Theta(n^{\log_2 7})$ (Max Points: 20).

B. Compute S using the function (from **A** above) such that the number of times the above function is called is $\Theta(k)$. (Max Points: 10)

C. Compute S using the function (from **A** above) and the Divide & Conquer Approach such that the number of times the above function is called is $\Theta(\log k)$. (Max Points: 20)

Sample Input

[[0, 1], [1, 1]], 3

Sample Output

[[1, 2], [2, 3]]