CS325: Analysis of Algorithms, Fall 2020

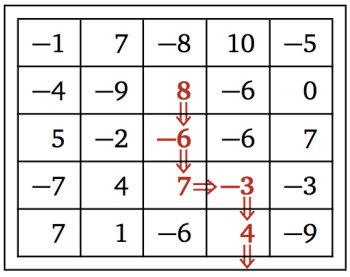
Group Assignment 2*

Due: Tue, 10/26/20

Homework Policy:

1. Students should work on group assignments in groups of preferably three people. Each group submits to CANVAS a zip file that includes their source code and their *typeset* report. Specifically, for this assignment your zipped folder should contain two files named `assignment2.pdf`, and `assignment2.py`. One submission from each group is sufficient.
2. The goal of the homework assignments is for you to learn solving algorithmic problems. So, I recommend spending sufficient time thinking about problems individually before discussing them with your friends.
3. You are allowed to discuss the problems with other groups, and you are allowed to use other resources, but you *must* cite them. Also, you must write everything in your own words, copying verbatim is plagiarism.
4. *I don't know policy:* you may write "I don't know" *and nothing else* to answer a question and receive 25 percent of the total points for that problem whereas a completely wrong answer will receive zero.
5. Algorithms should be explained in plain english. You can use pseudocodes if it helps your explanation, but the grader will not try to understand a complicated pseudocode.
6. More items might be added to this list. ☺

Vankin's Mile is an American solitaire game played on an $n \times n$ square grid. The player starts by placing a token on any square of the grid. Then on each turn, the player moves the token either one square to the right or one square down. The game ends when player moves the token off the edge of the board. Each square of the grid has a numerical value, which could be positive, negative, or zero. The player starts with a score of zero; whenever the token lands on a square, the player adds its value to his score. The object of the game is to score as many points as possible.



*The problem is from Jeff Erickson's lecture notes. Looking into similar problems from his book chapter on dynamic programming is recommended.

For example, given the grid below, the player can score $8-6+7-3+4=10$ points by placing the initial token on the 8 in the second row, and then moving down, down, right, down, down. (This is not the best possible score for these values.)

In this assignment, you describe and analyze an efficient algorithm to compute the maximum possible score for a game of Vankin's Mile, given the $n \times n$ array of values as input.

Report (60%). In your report, include the description of your algorithm, and provide running time analysis and proof of correctness. Algorithms should be explained in plain english. You can use pseudocodes if it helps your explanation, but the grader will not try to understand a complicated pseudocode.

Code (40%). You will complete and submit the `assignment2.py` file to compute the maximum possible score for a given game. The following template is provided. You need to implement the function *vankin_max_score*. This function has two parameters: *input_file_path* and *output_file_path*, specifying the full path of the input and output files, respectively. When called, the function *vankin_max_score* should read the game from the input file, and writes maximum possible score of the game in the output file.

```
'''
    This file contains the template for Assignment2.  For testing it, I will place it
    in a different directory, call the function <vankin_max_score>, and check its output.
    So, you can add/remove  whatever you want to/from this file.  But, don't change the name
    of the file or the name/signature of the following function.

    Also, I will use <python3> to run this code.
'''

def vankin_max_score(input_file_path, output_file_path):
    '''
    This function will ontain your code, it will read the input from the file
    <input_file_path> and write to the file <output_file_path>.

    Params:
        input_file_path (str): full path of the input file.
        output_file_path (str): full path of the output file.
    '''
    pass

# vankin_max_score('input0.in', 'input0.out')

```

Tests Your program will be tested against several test cases, for correctness and efficiency. For each test case, the program will be automatically stopped after 20 seconds if it is not done in that time. In this case, the group will miss the points of that test case. **Note:** it is important that your output is formatted as described below, since your codes will be tested automatically.

Input/Output The input file is formatted as follows. The first line is one integer $1 \leq n \leq 1000$. The following n lines each is a row of the matrix. Each line is composed of n integers, each between -100 and 100, separated by commas.

The output file must contain a single integer: the maximum possible score on the $n \times n$ board of the input.

Sample Input (1):
2
5,-2
-3,1

Sample Output (1):
4

Sample Input (2):
3
1,2,3
2,-10,-20
-20,20,-10

Sample Output (2):
20