CS325: Analysis of Algorithms, Fall 2020

Group Assignment 3*

Due: Tue, 11/17/20

Homework Policy:

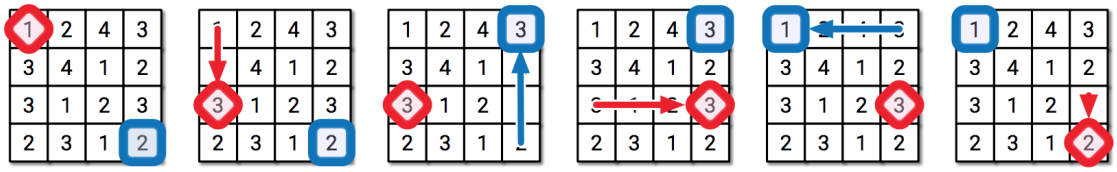
1. Students should work on group assignments in groups of preferably three people. Each group submits to CANVAS a zip file that includes their source code and their *typeset* report. Specifically, for this assignment your zipped folder should contain two files named `assignment3.pdf`, and `assignment3.py`. 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. ☺

The infamous Mongolian puzzle-warrior Vidrach Itky Leda invented the following puzzle in the year 1437. The puzzle consists of an $n \times n$ grid of squares, where each square is labeled with a positive integer, and two tokens, one red and the other blue. The tokens always lie on distinct squares of the grid. The tokens start in the top left and bottom right corners of the grid; the goal of the puzzle is to swap the tokens.

In a single turn, you may move either token up, right, down, or left by a distance determined by the **other** token. For example, if the red token is on a square labeled 3, then you may move the blue token 3 steps up, 3 steps left, 3 steps right, or 3 steps down. However, you may not move either token off the grid, and at the end of a move the two tokens cannot lie on the same square.

Describe and analyze an efficient algorithm that either returns the minimum number of moves required to solve a given Vidrach Itky Leda puzzle, or correctly reports that the puzzle has no solution. For example, given the puzzle in the following figure, your algorithm would return the number 5.

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



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 `assignment3.py` file to compute the minimum number of moves to solve the puzzle if possible, otherwise correctly report that the puzzle cannot be solved. The following template is provided. You need to implement the function *vidrach_itky_leda*. 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 *vidrach_itky_leda* should read the game from the input file, and writes the minimum number of moves to solve the puzzle if the puzzle is solvable. Otherwise, the function *vidrach_itky_leda* should write -1 in the output file.

```
'''
    This file contains the template for Assignment3.  For testing it, I will place it
    in a different directory, call the function <vidrach_itky_leda>, 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 vidrach_itky_leda(input_file_path, output_file_path):
    '''
    This function will contain 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

# vidrach_itky_leda('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 $1 \leq n \leq 50$. The following n lines each is a row of the Vidrach Itky board. Each line is composed of n positive integers, each between 1 and 100, separated by commas.

The output file must contain a single integer: if the puzzle is not solvable this number should be -1, if solvable this number should be the minimum number of moves to solve the puzzle.

Sample Input (1):
2
1,1
1,1

Sample Output (1):
4

Sample Input (2):
3
1,2,3
4,5,6
3,2,1

Sample Output (2):
6

Sample Input (3):
3
1,4,4
1,2,1
4,4,1

Sample Output (3):
6

Practice Problems

Do *not* submit solutions to the following problem. However, I recommend to work on them as good practice problems for graphs and graph algorithms.

Basic Graphs. The following problems from Chapter 5 of the book (`https://jeffe.cs.illinois.edu/teaching/algorithms/book/05-graphs.pdf`): 1, 11, 12, 22