CS 333 - Homework #3

Due To

22 May 2024 Wednesday, by 11.59 pm.

Homework 3

There are 2 independent tasks in this homework.

Task1 (50 Points)

Problem Description

Given a representation of the area to model, your task is to determine how far the water will flow. The land will be represented by topographical map, which is a two-dimensional grid of elevations. Each grid will have r rows and c columns. Each grid location – or grid cell – will have an integer height elevation. Because you will be dealing with multiple grids, each one will have a title as well.

Your task is to figure out the longest sequence of grid locations that water can flow between. Water will flow from a higher elevation to a lower elevation. For the purposes of this problem, water will never flow from a given elevation to the same elevation, nor will it flow uphill. Furthermore, water can only flow from one grid cell to an adjacent cell (adjacent cells are above, below, left, and right; not diagonal!).

As an example, consider the following 5×5 grid. Note that the input in this example is justified to help illustrate the grid; there will only be one space between heights in the actual input.

66 78 41 3 77
4 90 41 8 68
12 11 29 24 53
0 51 58 9 28
97 99 96 58 92

There are many such valid drainage paths in this grid. One starts in the cell $(1, 1)$, and flows through heights 90-78-41-3. Note that the sequennce of heights 90-41-41-3 is not a valid drainage flow, as water is not always flowing downhill (41-41 is not downhill). The longest drainage path in this example is of length 7, and flows from the 99 in cell $(4, 1)$; the full sequence of heights is 99-96-58-29-24-8-3. The equivalent path of locations is (4,1)-(4,2)-(3,2)-(2,2)-(2,3)-(1,3)-(0,3).

You may solve this problem using top-down Dynamic Programming or using bottom-up Dynamic Programming, but a brute-force solution's running time will be too long. The filename for source code will be "drainage.py" or "Drainage.java".

Input Format:

A 2-d array of doubles/floats indicating the altitude of each location. Input will be given as a file by providing the filename as a command line argument. To provide an input grid of size $r \times c$, you will have $r + 1$ lines in your input as follows:

- The first line contains two space-separated ints representing the values of r and c, i.e. or rows and columns in the grid, respectively.
- The remaining r lines contain each contain c space-separated floats/doubles representing the altitudes of each location in the grid.

Output Format:

To give output, your program should print just the length of sequence of grid indices that represent the longest drainage run.For example, for the grid above you should print 7.

Python Sample Run:

python drainage.py test1.txt

Test Cases Summary

There are 5 test cases provided to you. The answers for these tests are listed below:

- test1.txt is length 7
- test2.txt is length 25
- test3.txt is length 1
- test4txt is length 6
- test5.txt is length 9

Task2 (50 Points)

Motivation

In this task, we are trying to help a tofu factory schedule shifts among its workers. The factory has divided its manufacturing process into four steps:

1. Blend: Take soaked soy beans, wash them, add water, then blend until smooth.
2. Cook: Simmer the mixture, stirring constantly.
3. Strain: Strain out the soy bean solids using a cloth.
4. Finish: Add nigari (a coagulant derived from seawater) to the remaining liquid. Add the formed curds to a mold.

Each of these steps requires a specific number of workers per shift, and these workers must be trained in that particular task. For this assignment, you will be given the workers' details and the shift schedule and you must determine whether it is possible for the factory to staff all of its shifts. You solution should operate by using the input to create a flow network, then focus on maxflow to derive the final answer. In other words, your code should perform a reduction from this problem to maxflow.

Problem Description

Your algorithm will be given input in two components. The first is what we will refer to as the "worker details". This input consists of a list of employees, where for each employee you have:

- The stations that this employee is trained on (a list consisting of a subset of "blend", "cook", "strain", and "finish")
- The times the employee is available for working an hour-long shift (a list of times between 0 and 23)

The second part of the input is going to be the "shift schedule". The shift schedule has a list stations ("blend", "cook", "strain", and "finish"). For each station it provides:

- The number of people needed to staff that station per shift (in integer)
- A list of shifts that must be covered (a list of times between 0 and 23)

Your algorithm should determine whether it is possible to fully staff all of the stations for each shift. That is, it must determine whether it is posssible to:

1. Assign employees to shifts so that
2. No employee is assigned to two shifts at the same time,
3. Each employee is only assigned to stations that they have been trained on,

4. Every shift for every station is fully staffed (that is the required number of employees are assigned to each of the shifts)

Your algorithm should print a boolean (True or False) indicating whether or not it is possible to satisfy all of these requirements. Note that this list of requirements is exhaustive. Do not assume any other requirements (e.g. note that we do not require every employee have at least 1 shift, that is intentional, your algorithm should not depend on that or anything else not mentioned). The filename for source code will be "scheduler.py" or "Scheduler.java".

Input Format:

Input will be given as a file by providing the filename as a command line argument. To provide an input with n employees, the input will have $2n + 10$ total lines as follows:

- The first 8 lines will give the shift schedule. in the following way:
  - Line 1 has the word "blend" to indicate that this is the shift schedule for the blend station, then a space, then an integer representing the number of employees who must be assigned to each shift of the blend station.
  - Line 2 contains integers representing the start times of the hour-long shifts for the blend station separated by spaces
  - Lines 3 and 4 are the same but for the cook station
  - Lines 5 and 6 give informtion for the strain station
  - Lines 7 and 8 give information for the finish station
- Line 9 has the word "employees" and then gives the integer n, i.e. the total number of employees
- The next 2n lines give the information for each of the n employees. Each employee's information is provided on two lines:
  - The first line contains space-separated strings representing the stations on which that employee is trained.
  - The second line contains space-separated integers representing the times that this employee is available to work shifts.
- The last line is just the word "done", indicating the end of the input.

Here is an example input for 4 employees. The answer for this input is True.

```
blend 1
0 1 2 3
cook 1
1 3
strain 2
4 5
finish 1
5 6
employees 4
```

blend cook
0 1 2 3
strain
2 4 5
strain finish
4 5 6
blend finish
1 3 5
done

For this input:

- Blending requires 1 employee per shift and shifts occur at 00:00, 01:00, 02:00, and 03:00
- Cooking requires 1 employee per shift and shifts occur at 01:00, and 03:00
- Straining requires 2 employees per shift and shifts occur at 04:00, and 05:00
- Finishing requires 1 employee per shift and shifts occur at 01:00, and 03:00
- Employee 0 is trained in blending and cooking and can shifts at 00:00, 01:00, 02:00, and 03:00
- Employee 1 is trained in straining and can do shifts at 02:00, 04:00, and 05:00
- Employee 2 is trained in finishing and straining and can do shifts at 04:00, 05:00, and 06:00
- Employee 3 is trained in blending and finishing and can do shifts at 01:00, 03:00, and 05:00

Output Format:

Your program should return a single boolean indicating whether or not the given employees can fully-staff the given shift schedule (True/False).

Python Sample Run:

python scheduler.py test1.txt

Submission Rules

You will submit this homework via the LMS system. You should follow the file-naming conventions and guidelines below:

- You should submit your source files as a ZIP archive file (NOT RAR or other formats). The name of the file should be in format **"<USER-ID>_hw<HOMEWORK-NR>.zip"**. For example, if your username is vy1043, then the name of the submitted file should be "vy1043_hw3.zip". Pay attention that all the letters are in lower-case. ZIP archive is supposed to contain just the source files, under two folders corresponding to the two tasks ("Task1" and "Task2" folders).
- Late submissions and files that do not compile are not accepted.
- You can resubmit your homework (until the deadline) if you need to.
- You are not allowed to use any external libraries and/or functions. Everything must be implemented by you from scratch.
- Any type of plagiarism will not be tolerated. Your submitted codes will be compared with other submissions and also the codes available on internet and violations will have a penalty of -100 points. (In case of copying from another student both parties will get -100)