CECS 328 Honors Programming Assignments

Darin Goldstein

Every problem will be sent as an email attachment called input.zip. Unzip this file in the standard way to retrieve the files that you are supposed to have for each assignment. When you submit your file(s), you will zip them into a file called output.zip. In the first assignment, for example, the output file is heights.txt, but when you submit, you will submit the file output.zip (which will be heights.txt, zipped up and renamed if necessary).

If your response files contain extra white space or added characters, they will be counted as incorrect. If you have any questions about the problems, instructions, etc., please feel free to email me[1] or come see during office hours.

Make sure that you refer to the syllabus for my policies on collaboration (*NOT* allowed in any way, shape, or form) and the penalties for doing so.

[1]daring90808@gmail.com

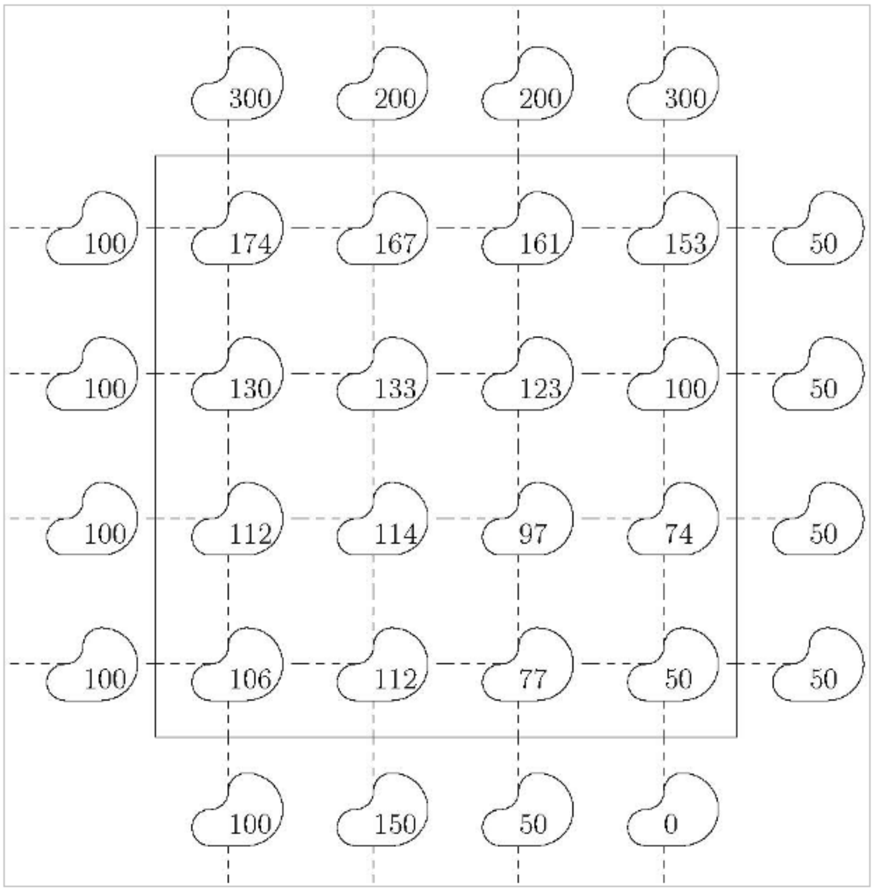
1 Pond Scum

A system of ponds at a casino in Las Vegas is laid out in a checkerboard pattern. At one point, the heights of all the ponds in the system were kept at carefully calibrated heights. They used to be constantly filled or drained as fast as needed to keep them at their constant, pre-set heights. The water flowing through the pipes was an amusing spectacle for children to see.

Due to the crumbling economy in Vegas, the pumps in certain of the ponds no longer function correctly, and the heights of these ponds are no longer directly controlled by the computer system.

Pipes connect each pond to its four neighbors (east, west, north, south). If a pond has a broken pump, then water flows through the pipe between two ponds at a rate that is proportional to the difference in the heights of the water in those ponds. If a pond's pump is still controlled by the system, then you may assume that it's height remains constant.

One day, someone fills up the ponds to a certain initial height and then just lets the system take care of itself. Your problem is to compute an estimate of the heights in all variable-height ponds at equilibrium (that is, when the total rate at which water flows into each variable-height, broken-pump ponds from its higher neighbors is equal to the rate flowing out to its lower neighbors, so that the pond's height remains constant). The equilibrium height does not depend on the constant of proportionality that relates differences in heights to rate of flow.



In the diagram, the interior comprises the ponds inside the square. The numbers inside the ponds indicate the waters equilibrium height above ground level to the nearest centimeter. The dashed lines are pipes. The unconnected pipes leading from the ponds along the exterior are connected to sources (or sinks) that maintain the external ponds at the heights shown.

The input to your program (in a file named ponds.txt) will be the initial heights of the ponds in grid format, separated by comma. Variable-height ponds will be indicated by an exclamation point in front of the number. For example, the input for the square shown above would be as follows.

```
0,300,200,200,300,0
100,!174,!167,!161,!153,50
100,!130,!133,!123,!100,50
100,!112,!114,!97,!74,50
100,!106,!112,!77,!50,50
0,100,150,50,0,0
```

The output, in the file heights.txt, should give all the pond heights at equilibrium, including the external ponds, in the format shown in the example without the exclamation points. All results are to be kept exact. In other words, you may not use decimals to approximate the answer. Fractions should be written as a/b where a and b are both integers.

Consider the following pond configuration.

```
264,450,766
8,!738,144
593,!993,425
848,285,139
```

The correct final configuration is as follows. (Note that all fractions must be in lowest terms so that the greatest common denominator of the numerator and denominator is 1.)

```
264,450,766
8,1237/5,144
593,1938/5,425
848,285,139
```

2 Party

Alice wants to throw a party and is deciding whom to call. She has N people to choose from, and she has made up a list of which pairs of these people know each other. She wants to pick as many people as possible, subject to two constraints: at the party, each person should have at least k_1 other people whom they know and k_2 other people whom they don't know. (Note that you can't know or not know yourself. You don't count in your own computation.)

Input to your program will be a text file called description.txt. The first line will hold the number k_1 and the second line will hold the number k_2. The remainder of the file will be a 0-1 adjacency matrix indicating which people know which others. If there is an edge between two people, then they know each other. If not, they don't. People will be represented numbers from 1 to N. (Part of the problem will be to determine the number N from the adjacency matrix itself.)

Output should consist of a list of integers, one per line, in the file party.txt of the people who should be invited to the party.

For example, assume that the input file description.txt is as follows.

```
1
2
0001110011
0001000010
0001000010
1110111011
1001010010
1001100011
0001000010
0000000010
1111111101
1001010010
```

The file party.txt should be empty. No valid party can be formed from this group of 10 people.

3 Racing Gems

You are playing a racing game. Your character starts at the x axis line ($y = 0$) and proceeds up the racetrack, which has a boundary at the line $x = 0$ and $x = w$. The finish is at $y = h$, and the game ends when you reach that line. You proceed at a fixed vertical velocity v, but you can control your horizontal velocity to be any value between $-v/r$ and v/r, and change it at any time.

There are a set of gems at specific points on the race track. Your job is to maximize the total value of the gems that you collect. Each gems has a value associated with it.

What is the maximum value of the gems that your player can collect? You may start at any horizontal position you want (but your vertical position must be 0 at the start).

Input will be in the file gems.txt. The first line will contain four integers, separated by commas: n (the number of gems), r (the ratio of vertical velocity to maximum horizontal speed), w (the width of the track), and h (the height of the finish line). Following this will be n lines, each containing an x and y coordinate, the coordinates of the gem, and a value v for the gem. All gems will lie within the race track.

Your program will list off the gems that your racer will pick up in chronological order, one per line, in the file race.txt.

For example, we might have a racecourse of length 10, a width of 10, and you had the same potential horizontal velocity as vertical. If there are 5 equally-valued gems on the racetrack, the input file gems.txt might look as follows.

```
5,1.,10.,10.
8.,8.,1
5.,1.,1
4.,6.,1
4.,7.,1
7.,9.,1
```

To maximize our profit, we would only choose three of them in race.txt. The output file looks as follows.

```
5.0,1.0,1
4.0,6.0,1
4.0,7.0,1
```

A slightly more difficult example is as follows.

```
10,17.41488555933587,101.82393555668958,1746.035106821133
61.51644005670237,1526.5427995364948,8
87.3410626634542,467.0492308761329,8
99.64463994648571,1367.0185124455065,1
72.15297059988576,783.0690627936801,9
86.40875144576354,781.190320715566,9
77.42176712320254,1332.772344897496,10
15.99484538428046,852.8673891294774,10
76.98725849972107,151.41099127398022,9
93.67450269524531,985.4944135006884,9
44.55094060734501,305.0735827511105,5
```

The solution is

```
76.98725849972107,151.41099127398022,9
87.3410626634542,467.0492308761329,8
86.40875144576354,781.190320715566,9
93.67450269524531,985.4944135006884,9
77.42176712320254,1332.772344897496,10
```

HINT: It is possible to formulate this problem as the problem of finding the longest path in an appropriately chosen directly acyclic graph. Imagine putting an edge from one gem g_1 to another g_2 if and only if, starting at g_1, it is possible to reach g_2.

4 Mixtures

How to make gold from lead has baffled alchemists for centuries. Scientists recently announced a sensational breakthrough. By mixing N different chemicals in exactly the correct ratio, one can create a mixture that transforms lead into gold. However, these N chemicals are not found in nature individually but rather only in specific ratios in liquid form. So making the correct mixture is not as trivial as it may at first seem.

Consider the following example where $N = 3$ with chemicals A, B, C. Assume that two mixtures are available in liquid form in the ratios of 1:2:3 and 3:7:1, respectively. By mixing those two solutions together in the ratio 1:2, it is possible to obtain a solution of A, B, C with ratio 7:16:5, but there is no way to combine these two mixtures into a new one with ratio 3:4:5. However, if we added a solution solution of A, B, C with ratio 2:1:2:, then a 3:4:5 mixture is possible with eight parts of 1:2:3, one part 3:7:1, and 5 parts of 2:1:2.

So clearly, determining which mixing ratios can be obtained from a given set of solution is not trivial. It is your goal in this assignment to do so.

Your input file will be called solutions.txt. The top line will be a line with a desired solution ratio in the form $a_1 : a_2 : a_3 : \ldots : a_N$. The next M lines will be the base solution ratios.

Your output file (answer.txt) will consists of M lines, each containing a single integer. These will indicate the parts necessary to get the desired solution. If no solution is possible, then return the value -1 on each line.

So, for example above, the input file solutions.txt would look as follows:

```
3:4:5
1:2:3
3:7:1
2:1:2
```

and the output file answer.txt would look as follows:

```
8
1
5
```