CECS 328 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.

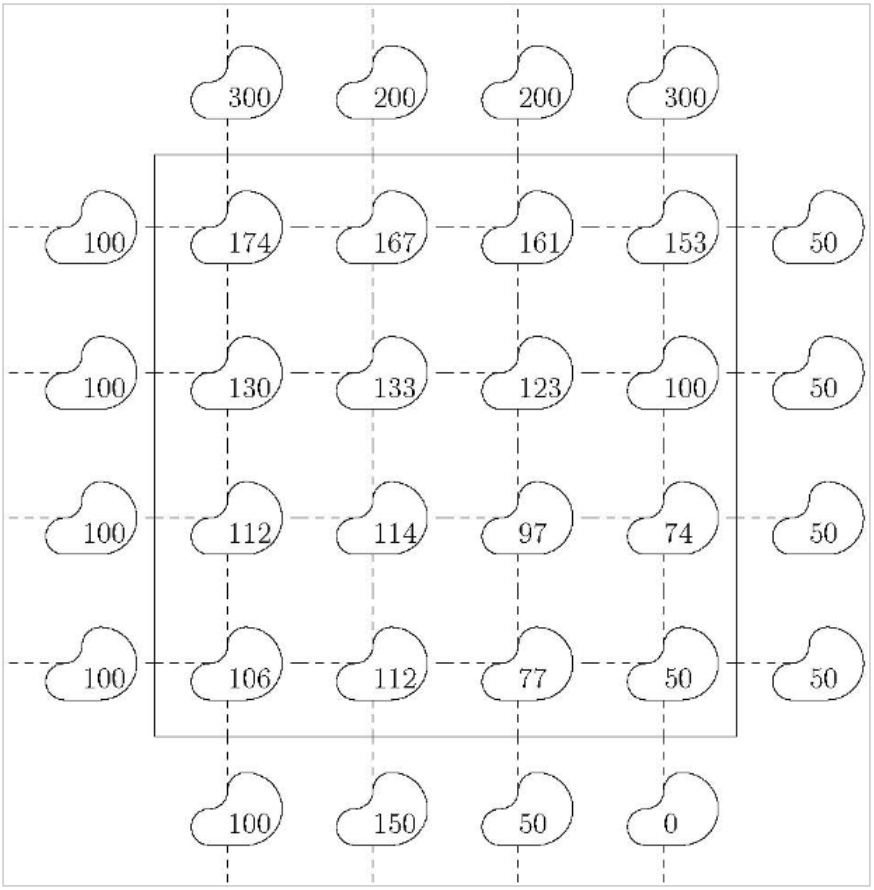
1 Pond Scum

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 Hotlines

It is the middle of the Cold War and America and its western allies are facing down the Union of Soviet Socialist Republics. There is an assassination of a high Party official somewhere in Moscow, and the culprit has been named to be a rogue CIA agent. The countries are now on the brink of a nuclear confrontation.

Luckily, the leadership of America and Russia have connected a series of hot lines between the White House and the Russian Politburo building. However, these hotlines are not direct. They have to pass through numerous intermediate routing stations in various countries. The goal is to get the largest number of connections from one spot to the other. The catch is that even though the switching stations (vertices) can handle a potentially unlimited number of calls, a single line (edge) can only be used for one call at a time.

The goal of your program will be to find the maximum simultaneous number of links that can be made between the White House and Russian Polituro.

Input to your program, the file edges.txt, will be the edge representation of a directed graph with vertices numbered 0 (the White House) through N (the Russian Poltburo). N will be given in the first line. Each directed edge will be two nonnegative integer values, separated by a comma.

Output, in the file paths.txt, should be a list of paths starting at 0 and ending at N, each vertex separated by a comma and each path separated by a newline.

For example, consider the following edges.txt file.

```
10
3,6
3,1
9,4
1,10
5,10
0,2
4,1
0,6
2,4
8,10
7,4
0,3
```

There is only a single path from the White House (0) to the Politburo (10).

```
0,2,4,1,10
```

4 Mining

You are a mining engineer and have the following problem presented to you. There is a rectangular two-dimensional pit of known width and depth. Using various engineering techniques, the compositions of all of the blocks of ore in the pit have been determined. Each block has a value and a cost to extract it from the ground. It's net value is the difference between these two. In order to remove a block from the ground, every block that is strictly above it (in the vertical direction only) must also be removed. Your goal is to determined which ore blocks to remove from the ground so as to maximize the revenue of your company.

Blocks are all exact rectangles of height 1. The input file will be called pit.txt. Each line of the input file will consist of a list of bricks separated by commas. Each brick will be a 3-tuple of nonnegative integers representing in order:

(width of the block, value of the block, cost it would take to remove it)

For example, an input file might look like the following.

(1,13,20),(2,3,7),(3,19,17),(2,0,1),(1,4,17),(1,13,15)

(2,3,10),(4,3,17),(1,1,13),(1,2,5),(1,18,18),(1,0,3)

(3,39,19),(2,10,1),(3,31,12),(1,33,6),(1,9,16)

(1,11,19),(1,0,1),(1,0,2),(2,0,7),(1,8,16),(4,4,10)

(1,0,9),(2,10,4),(2,17,15),(1,0,10),(1,0,5),(3,0,3)

Note that the sum of the widths of all the blocks in any given row is equal to the same number, 10 in this example. Think of the top line, beginning with (1,13,20), as ground level. Note that excavating that block will cost 20 but only yield a profit of 13. However, in order to get to the block beneath, it will need to be taken out.

Output will be in the file blocks.txt. You will indicate which blocks are to be excavated so as to yield the maximum profit for your company. Each block on each line has a number assigned to it, starting from 0 on the left. The optimal solution for the example above is as follows.

0,1,2,3,4

0,1,2,3,4

0,1,2,3

1,2

1

Note that each line corresponds to a line in the input file. For example, at ground level, we excavate every block except for the last one on the far right. At the bottom of the pit, we only remove the second block from the left. The total profit from this excavation is 19.

HINT

Consider the following formulation of undertaking a project: Assume that you are given a set of possible projects P to undertake. Each project $p_i \in P$ has an associated value with completing it v_i. However, any given project might have prerequisites. For example, it might be that project p_i cannot be started until project p_j has been completed.

It is important to notice that the value v_i of some projects might not be positive. In fact, undertaking prerequisites is almost always a negatively valued enterprise. Consider the act of creating a piece of software. Developers need to be hired before the software can be designed, programmed, and finally sold. Committing to paying people for work that is not already completed is a negatively valued prerequisite.

A subset of projects $A \subseteq P$ is considered feasible if and only if, for every project $p \in A$, every prerequisite of p is also in A. The goal is the project selection problem is to find a feasible subset of projects that has the highest associated total value. (Total value is the sum of the values of the projects within the subset.)

It is possible to create a prerequisite graph out of a problem such as this by creating vertices for each project and adding edges (v, w) if and only if w is a prerequisite for v. It is then possible to turn the project selection problem into a network flow problem by performing the following steps.

> Create a source s and sink t. Add edges from the source to each project with a positive value and let the weight of the edge be equal to the value of the project. Add edges from each project with a negative value to the sink and let the weight of the edge be equal to the absolute value of the project. Let the weight of each prerequisite edge be ∞.

The minimum cut of the resulting flow network yields an optimal set of projects by noting that the projects on the source side of the minimum cut comprise a maximum-valued set of feasible projects.

5 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 collect as many gems as possible. 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.

6 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
```

7 Evolutionary Trees

Adapted from a problem in DPV. Reconstructing evolutionary trees by maximum parsimony

Suppose we manage to sequence a particular gene across a whole bunch of different species. For concreteness, say there are n species, and the seqeuences are strings of length k over alphabet $\Sigma = \{A, C, G, T\}$. How can we use this information to reconstruct the evolutionary history of these species?

Evolutionary history is commonly represented by a tree whose leaves are the different species, whose root is their common ancestor, and whose internal branches represent speciation events (that is, moments when a new species broke off from an existing one). Thus we need to find the following:

- a (binary) evolutionary tree with the given species at the leaves
- For each internal node, a string of length k: the gene sequence for that particular ancestor

We will assume that a speciation event occurs when at least one base pair changes from the parent species. (Note that in practice, this is wildly impractical assumption, as there are many possible alterations to a DNA strand including deletions, additions, swaps, exchanges, and other local changes.) By this assumption, we can assume that all species have exactly k base pairs in their DNA.

For each possible tree T, annotated with sequences $s(u) \in \Sigma^k$ at each of its nodes u, we can assign a score based on the principle of parsimony: fewer mutations are more likely.

$$score(T) = \sum_{\{u,v\} \in E(T)} (\text{number of positions on which } s(u) \text{ and } s(v) \text{ agree})$$

Given a list of DNA sequences and a tree, find a labelling of the inner nodes with maximum parsimony.

The input file, called genetics.txt, will be a text file with first line that describes the tree by indicating the steps of a depth-first search from the root. L stands for left, R stands for right, and U stands for up. The next several lines will be the DNA strings that appear on the leaves of the tree from left to right. (HINT: Consider one string position at a time.)

Your output (tree.txt) will be the description of the tree with instructions for how to place DNA strings at every node. For the formatting, please see the example below.

For example, assume that the file genetics looks like the following.

LLLURUURLURUUURLURLURUUU
GCTC
CCAC
ACGA
ACCG
AACC
AGTG
TGGC

If you follow the instructions for creating the tree in the first line (left, left, left, up, right, up, up, right...), the tree that you create should have seven leaves. The DNA strings given should be placed inside those leaves from left to right. Your job will be to fill in the internal nodes of the tree and send the resulting completed tree.

One possible solution for this tree (tree.txt) is as follows.

:AACC
0:ACCC
00:ACAC
000:GCTC
001:CCAC
01:ACCA
010:ACGA
011:ACCG
1:AACC
10:AACC
11:AGCC
110:AGTG
111:TGGC

To determine where a particular DNA strand should be placed in the tree, follow the instructions given on the line. 0 stands for left, and 1 stands for right. The first line is the root (empty instructions) and indicates that the DNA string AACC should be placed inside the root. The second line indicates that the internal node to the left of the root should contain the DNA ACCC. And so on. For this example, the parsimony score is 13.

Every node must be filled with a DNA string.

Note that it is theoretically possible for one or both children to have exactly the same DNA as the parent.

8 Forced-equal

In a satisfiable system of linear equalities

$$a_{11}x_1 + \ldots + a_{1n}x_n \leq b_1$$

$$\ldots$$

$$a_{m1}x_1 + \ldots + a_{mn}x_n \leq b_m$$

we describe the jth inequality as *forced-equal* if it satisfied with equality by *every* solution $x = (x_1, \ldots, x_n)$ of the system. Equivalently, $\sum_i a_{ji}x_i \leq b_j$ is *not* forced-equal if there exists an x that satisfies the whole system and such that $\sum_i a_{ji}x_i < b_j$.

For example, in

$$x_1 + x_2 \leq 2$$
$$-x_1 - x_2 \leq -2$$
$$x_1 \leq 1$$
$$-x_2 \leq 0$$

the first two inequalities are forced-equal, while the third and fourth are not. A solution x to the system is called *characteristic* if, for every inequality I that is not forced-equal, x satisfies I without equality. In the instance above, such a solution is $(x_1, x_2) = (-1, 3)$, for which $x_1 < 1$ and $-x_2 < 0$ while $x_1 + x_2 = 2$ and $-x_1 - x_2 = -2$.

You will be given as input a list of satisfiable linear inequalities. Your goal is to determine which of the inequalities are forced equal and which are not.

The input file will be called inequalities.txt and will contain lines of the form

$$(a_0, a_1, \ldots, a_n)$$

where each a_i is a possibly negative fraction. You should think of the above line as being equivalent to the inequality

$$a_0 + a_1x_1 + a_2x_2 + \ldots + a_nx_n \leq 0$$

Note that a_0 is thought of as the constant in the inequality.

The inequalities are numbered from 1 to m in the order they appear in the file. Your output will simply be a file, forced.txt, that lists the inequalities that are forced, one per line.

For the example above, the input file would be

(-2/1,1/1,1/1)

(2/1,-1/1,-1/1)

(-1/1,1/1,0/1)

(1/1,-1/1,0/1)

The output file should be

1

2