n-ary Gray codes

Darin Goldstein

1 Submission Instructions

Deadline: Fri Mar 22

The programming assignments will be graded as follows: Each assignment is posted online and you may demonstrate the assignment at any time *before the deadline*. Once the deadline arrives and you have not demonstrated, your project will receive no credit. You will generate and submit your assignments via the server as follows.

1. Log into the server and enter your section and the assignment you're interested in.

2. Generate an instance of the problem. Your individual problem will arrive to you via email in an attachment named input.zip. (Make sure that your Beachboard emails are up-to-date. Note that if you add the course after the semester has officially begun, you will need to explicitly inform me via email that you are unable to log into the server so that I can add an account for you.)

3. Solve the problem and generate an answer. Zip up your answer and rename it to output.zip.

4. Log back into the server and submit your answer. Once your answer is checked and verified, you will be notified by email. (Note that you may have to do this step several times.)

5. Once you are satisfied with your grade and have generated and solved as many instances as necessary, you will zip up all of your source files into a single zip file named src.zip. (All that is required is to go into your Java project and literally zip up the src folder.) You will then submit it via the server page. *Do not bother emailing code to me!* Only .java files will be accepted. If your scr folder contains anything but .java files, your submission will be rejected. Any assignment that doesn't have a valid code submission by the deadline will be counted as a 0.

The programming assignments will be graded on a scale of 0 to 4. (You can think of a 4 as an A, 3 as a B, etc.)

2 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.