CISC320 Fall 2016
Programming Assignment 3

You may use any of the following programming languages:

Java
C++ / C
Python
PLT Scheme (Racket subset)

If you would like to use a different one please let me know ahead of time. If you are using a specific library other than ones I list please also let me know.

Recommended graph libraries:
C++: boost, http://www.boost.org/doc/libs/1_62_0/libs/graph/doc/index.html
Python: NetworkX, http://networkx.github.io/index.html
Java: JGraphT, https://github.com/jgrapht/jgrapht
PLT Scheme: Racket Generic Graph Library, https://docs.racket-lang.org/graph/index.html

Testing

You will be given a file as input with the name "scenario_in.txt" where scenario will be different for each scenario and we will expect whatever programming language you use to write a file as output with the same name as the input file replacing "_in.txt" with "_out.txt" (i.e. if the input file is "simple_in.txt" the output file should be "simple_out.txt").

Please check canvas for updated scenarios. There will be some additional hidden scenarios that we will use to check for correctness of implementation.

Grading

Grading will be the total of scores for the 3 parts.
Part 1 = 50% if correct
Part 2 = 30% if correct
Part 3 = 20% if correct

Problem

In a beach town the ice cream stand business is booming. Tourists are overwhelmed with choices, have lots of money, and are incredibly lazy. Most of them will just choose their favorite ice cream vendor located on their street and go no further. You are the regional manager for CrazyCones and have developed a product that is so much better than the competition that tourists on any street covered by your vendors will buy your ice cream product.

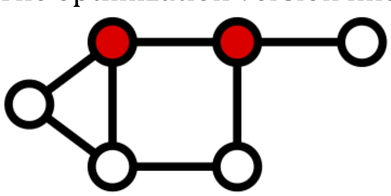
However, it is fairly expensive to staff, supply, and maintain an ice cream stand. You have figured out that the most efficient way to maximize your profit is to place a stand on the corner of a street intersection such that each street joining the intersection is covered by that ice cream stand. Your problem now is to figure out the minimum placement of stands such that every street in town is covered.

Approach

This problem is very similar to the friend party problem from programming assignment 2. In fact, both of these problems are direct instances of two known NP-Complete problems:

Dominating Set (Friend party):

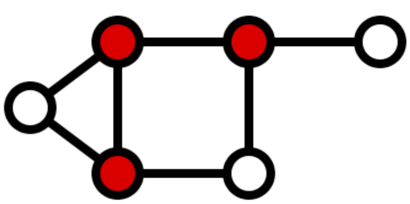
A dominating set for a graph G = (V, E) is a subset D of V such that every vertex not in D is adjacent to at least one member of D.
The decision version of the classic NP-Complete problem asks whether G has a dominating set of size at most K.
The optimization version finds the smallest dominating set for G.



A minimum dominating set

Vertex Cover (Ice cream stand business):

A vertex cover set for a graph G = (V, E) is a subset D of V such that every edge has at least one endpoint in D.
The decision version of the classic NP-Complete problem asks whether G has a vertex cover set of size at most K.
The optimization version finds the smallest vertex cover set for G.



A minimum vertex cover set

Rather than directly solving the ice cream stand business problem, this programming assignment is to apply the NP-Complete reduction from Vertex Cover to Dominating Set, solve the Dominating Set using your code from programming assignment 2, then process the dominating set solution to produce a solution to the ice cream stand business problem.

Part 1:

Given the ice cream stand business problem as an input graph file, output** the transformation graph according to the Vertex Cover to Dominating Set reduction. See your class notes for the details on how to generate this graph. To generate the sample output, new vertices are created from an original edge with ID of 1000 * source vertex + target vertex.

Part 2:

Solve the dominating set problem with your heuristic and program from assignment 2. You will need to change your code to recognize a 0 value for M (the maximum allowed party hosts) as an unknown maximum. For this unknown case, change your heuristic from assignment 2 part 4 to assign as many parties as needed to get the average social awkwardness to 1 (i.e. change your loop from "while assigned hosts < max hosts: assign a new host" to "while average awkwardness > 1: assign a new host"). Your output should be the same as programming assignment 2.

Part 3:

Solve the original vertex cover problem by writing some code that will take the input graph from part 1, the output from part 2 and use them to output a vertex cover for the original problem. Make sure you handle the case where a created vertex (label > 1000) might have been chosen. You must also verify that your output is indeed a vertex cover.

** You are required to generate output for Part 1, Part 2, and Part 3. It is okay if you combine your code from assignment 2 and assignment 3 into one source file / project, but you must generate that output at each step and take the appropriate input. You may not write code that directly solves the vertex cover problem (i.e. do not just read in the input graph from part 1 and output a vertex cover). Note that you may want to create different ways of running your code so that you can test each part.

Sample input/output

The first test case, town1_in.txt, represents the graph in the example on page 2 of this assignment. town1_in.txt:

```
1
6 7
1 2
1 3
2 3
2 4
3 5
4 5
```

```
4 6
```

Expected Part 1 output (which is the input for Part 2):

```
1
13 21 0 0
1 2
1 1002
2 1002
1 3
1 1003
3 1003
2 3
2 2003
3 2003
2 4
2 2004
4 2004
3 5
3 3005
5 3005
4 5
4 4005
5 4005
4 6
4 4006
6 4006
```

One possible correct Part 2 and Part 3 output:

```
Test Case 1.
My heuristic hosts are 2,3,4
Average social awkwardness = 1.00

Test Case 1.
Ice cream stands placed at 2,3,4 are a valid cover
```

Another possible correct Part 2 and Part 3 output for the same problem:

```
Test Case 1.
My heuristic hosts are 2,5,6,1003
Average social awkwardness = 1.00

Test Case 1.
Ice cream stands placed at 2,5,6,1 are a valid cover
```

Another possible correct Part 2 and Part 3 output for the same problem:

```
Test Case 1.
My heuristic hosts are 2,5,6,1003
```

```
Average social awkwardness = 1.00

Test Case 1.
Ice cream stands placed at 2,5,6,3 are a valid cover
```

An *incorrect* Part 2 and Part 3 output for the same problem:

```
Test Case 1.
My heuristic hosts are 2,4
Average social awkwardness = 1.00

Test Case 1.
error: 1->3=1 not in cover
Ice cream stands placed at 2,4 are not a valid cover
```

See Canvas for additional test cases including a hex town and a larger town.