CISC320 Spring 2018
Programming Assignment 4

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 4 parts.
Part 1 = 20% if correct
Part 2 = 20% if correct
Part 3 = 20% if correct
Part 4 = 20% if heuristic is polynomial time and satisfies <= criteria
Part 5 = 20% if correct
Bonus credit = up to 25% which can apply to beyond the Programming Assignment category towards your course grade. 10% of this will be based on internal metrics and 15% rewarded relative to peer submissions (i.e. the top submission will receive 15%; all other solutions will receive an amount based on how close they were to the top submission).

Problem

The new UD president, Dennis Assanis, would like to improve the student experience during Final Exam week and has decided to host a study break party on reading day. He has invited all of the faculty, staff, and student body to his property for snacks and socializing. However, many people find this socially awkward and will choose to stay home because they aren't really friends with the new president.

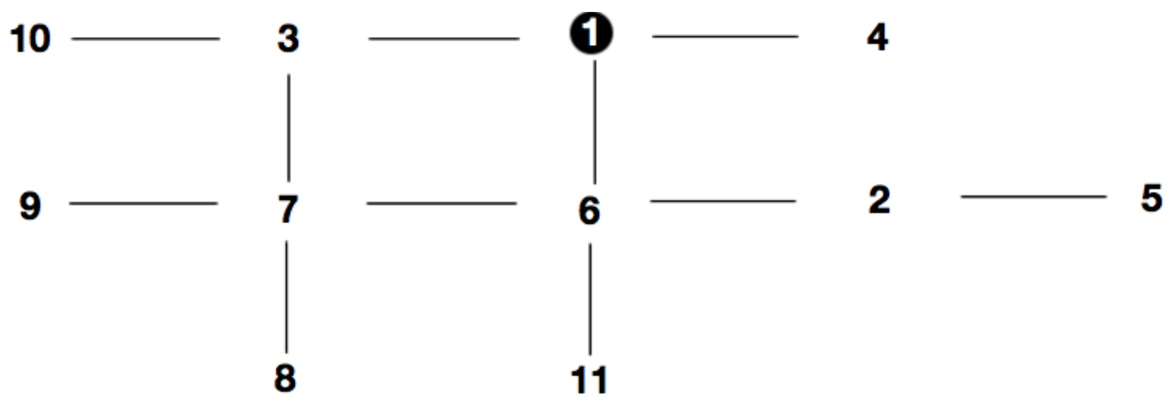

Figure 1 with person 1 marked as a party host

Given a social network of friends we can predict how many people would stay home by computing the average social awkwardness factor. In Figure 1 President Assanis is represented as 1 and 3, 4, and 6 are friends with the president and are considered to have a social awkwardness factor of 1 with the president. 10, 11, 7, and 2 have a social awkwardness factor of 2 with the president and 9, 8, 5 a factor of 3. This metric assumes that the president has a social awkwardness factor of 0 with himself and has no qualms about attending his own study break party.

Part 1:

Given a scenario input with N people and 1 party host compute the average social awkwardness if all people were to attend the party. Do not include the party host in the average! The first scenario in sample_in.txt is the one in Figure 1.

Part 2:

President Assanis is a smart fellow and realizes that hosting multiple study break parties would lower the social awkwardness between party host and attendees. Given a scenario input with N people and M party hosts, compute each person's social awkwardness if they attend the party which they were closest to the host (the party where they would feel the least socially awkward). Report the average social awkwardness value over all people who are not party hosts!

Part 3:

The president realizes he has a fixed budget allowing M party hosts and that he might not be able to choose the party hosts that minimize the average social awkwardness without the help of our CISC320 class. Given a scenario input with N people and a number M (with 0 starting assignments of party hosts), compare the following 2 greedy heuristics by computing their average social awkwardness for each scenario:

1. Find the person with the most friends and make them a party host. Repeat until M party hosts have been assigned. Ties should be resolved by choosing the person with a lower numerical ID.
2. First pick the person with the most friends and make them a party host. On subsequent rounds, find the person with the maximum current social awkwardness and make them a party host. Repeat until M party hosts have been assigned. Ties should be resolved by choosing the person with a lower numerical ID.

Part 4:

Come up with a better heuristic than the ones given in Part 3. Your heuristic must run in polynomial time. To receive credit for Part 4 your heuristic must produce at least one scenario that has a lower average social awkwardness than the given heuristics and must be lower or equal to the given heuristics for all scenarios.

Part 5:

Create a new version of your heuristic that is called when the budget M is 0. In this scenario we interpret the 0 as an unknown maximum of party hosts and want your heuristic 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 have the same format as part 4.

Additional Information about the context of this problem

If we were to come up with a perfect heuristic for all scenarios for Part 4 it would be able to solve a known NP-complete problem in polynomial time. The dominating set problem could be represented as the decision problem: in a perfect allocation with N vertices and M party hosts is the average social awkwardness equal to 1? This might give you a hint as to how to come up with a really good heuristic for part 4.

Input format:

The first line of the input contains the number of test cases. Each test case starts with one line that has four items of information, separated by one or more blank spaces:

- The number of people (N). It will be in the range 3..100. In the Figure 1 example it is 11, which means that the people IDs are 1..11.
- The number of friendship links (F) between people.
- The number of parties (M) allowed by the budget. In the Figure 1 example it is 1.
- The number of assigned party hosts (A). It will be either 0 or M. In the Figure 1 example it is M which is 1.

The next F lines of the test case contain a pair of numbers, separated by one or more blank spaces. Each pair of numbers is a bi-directional friendship link. If the two values were 1 and 2 this would mean that person 1 is a friend with person 2 and that person 2 is a friend with person 1. There are no one-direction friendship links.

The next A lines of the test case contain a single number which is the ID of a party host. In scenarios where A = 0 there will be no lines and your program is expected to compute the party hosts according to the 2 heuristics in part 3 and your heuristic you develop in part 4.
Sample input that would test parts 1,2,3,4:

```
4
11 11 1 1
1 4
1 6
2 5
6 2
3 7
6 7
3 1
7 8
7 9
10 3
11 6
1
11 11 2 2
1 4
1 6
2 5
6 2
3 7
6 7
3 1
7 8
7 9
10 3
11 6
9
1
11 11 5 5
1 4
1 6
2 5
6 2
3 7
6 7
3 1
7 8
7 9
10 3
11 6
1
2
3
6
7
11 11 5 0
1 4
1 6
2 5
```

```
6 2
3 7
6 7
3 1
7 8
7 9
10 3
11 6
```

Sample output. This shows part 1 working (test case 1), part 2 working (test cases 2, 3), and parts 3 and 4 working (test case 4). Note that my sample part 4 heuristic is identical to part 3 heuristic 2 so it would earn no points for part 4. Please round to 2 decimal places of precision.

```
Test Case 1.
Average social awkwardness = 2.00
Test Case 2.
Average social awkwardness = 1.67
Test Case 3.
Average social awkwardness = 1.00
Test Case 4.
Heuristic 1 hosts are 6,7,1,3,2
Average social awkwardness = 1.00
Heuristic 2 hosts are 6,10,4,5,8
Average social awkwardness = 1.17
My heuristic hosts are 6,10,4,5,8
Average social awkwardness = 1.17
```