Approximation Algorithms - Programming Assignment 4

Due: Monday, November 16

Deliverables:

GitHub Classroom: `https://classroom.github.com/a/rc4meoQL`

Required Files: `compile.sh`, `run.sh`

Optional Files: `*.py`, `*.java`, `*.clj`, `*.kt`, `*.js`, `*.sh`

By now you are very familiar with the Vertex Cover problem:

Input: A graph $G = (V, E)$.

Goal: Find a vertex cover of minimum size.

We will implement a $\log(n)$-approximation algorithm, a 2-approximation algorithm for this problem (this differs from the 2-approximation in class), and an exact (but slow) algorithm.

log(n)-Approximation Algorithm

SmartGreedyVertexCover(G)

Input: A graph G.
Output: A set of vertices that form a (not necessarily optimal) vertex cover.

```
1: C ← ∅
2: while G has at least one edge do
3:     v ← vertex in G with maximum degree
4:     G ← G \ v (This also removes all edges adjacent to v)
5:     C ← C ∪ v
6: return C
```

Implement this algorithm.

2-Approximation Algorithm

BasicGreedyVertexCover(G)

Input: A graph G.
Output: A set of vertices that form a (not necessarily optimal) vertex cover.

```
1: C ← ∅
2: while G has at least one edge do
3:     (u, v) ← any edge in G
4:     G ← G \ {u, v} (This also removes all edges adjacent to u and v)
5:     C ← C ∪ {u, v}
6: return C
```

Implement this algorithm.

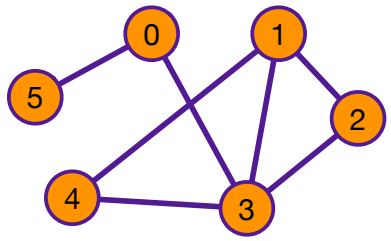
Exact Algorithm

Implement a brute force (exact) algorithm for vertex cover.

Format:

The input graph will be undirected and simple (no self loops or multiple edges). The format will be a simple edge list. There will be n vertices and m edges. Each edge in the graph will be represented by a pair of vertex identifiers. Note that I will not give you graphs with isolated vertices. All edge lists will begin at 0.

For example the following graph is represented by the list:



1 2
1 3
2 3
1 4
0 5
4 3
3 0

What to turn in via GitHub Classroom by 8pm of the due date:

- Accept the following GitHub Classroom assignment: `https://classroom.github.com/a/rc4meoQL`. This will create a GitHub repository which you will use to submit your source code for this assignment. The repository will also include some basic sample inputs and outputs.
- Your algorithm should take an edge list and output the vertices corresponding to three vertex covers.
- Your program must read input from a file (given as the first command line argument) and write output to `stdout`.

Test run: On the date before the due date, the grader will be run after 8pm and provides feedback containing only what your program scores. Only submissions made before 8pm are guaranteed to receive feedback. What your program scores on the official run (the due date) is the final score.

Late submission:

- Late submissions made within 48 hours of the deadline receive no penalties.
- Late submissions made after 48 hours of the deadline receive a .8 multiplier. For example, if your late submission scores 80%, the final score will be 80% * .8 = 64%.
- If you decide to make a late submission, please send an email to vnguy143@calpoly.edu.

Resubmission: No resubmissions will be accepted. What you get on a submission (other than the test run) will be your final score. Please do not include `compile.sh` and `run.sh` in your repository if you don't intend to submit yet.