Approximation Algorithms- Programming Assignment 5

Due: Monday, November 27th and Tuesday, November 28th

Directions: Your code will be due by 8pm, Tuesday, November 28th on PolyLearn. You may use Python 3.4.3, Clojure, or Java 8.

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

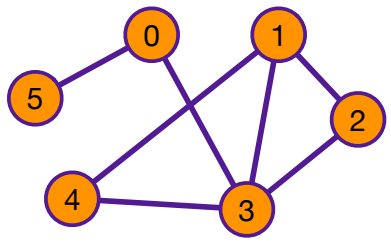
Implement a brute force (exact) algorithm for vertex cover.

Format:

The input graph will be undirected and simple (no self loops or multiple edges). I provide a sample input graph on PolyLearn. I strongly encourage you to create your own test cases.

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 on PolyLearn:

- Submit all of your source code along with a shell script driver named "asgn5.sh". I should be able to run your program by typing, "./asgn5.sh <input file name>". Your program must read input from a file and write output to stdout. Your output will be tested using diff, so it must match exactly. Please don't zip your files.

  You will be provided with a working example with sample input and output.

What to bring with you to lab:

1. Give an example (a graph) for which the 2-approximation algorithm (BasicGreedyVertexCover) returns more vertices than the $\log(n)$-approximation algorithm (SmartGreedyVertexCover).
2. Give an example (a graph) for which the 2-approximation algorithm (BasicGreedyVertexCover) returns precisely 2 times the number of vertices returned by the exact algorithm.
3. Give an example (a graph) in which all three algorithms return the same number of vertices (assuming that BasicGreedyVertexCover chooses the edge that you wish it to choose).