CS 577: Introduction to Algorithms	Program 10 - Graph Coloring
Out: 04/20/21	**Due: 04/27/21**

1 Coding Question:

Reminders

- Must be coded individually in your choice of either Python, Java, C, C++, or C#
- Submitted through Gradescope, and there are hidden test cases
- There is a class-wide runtime leaderboard on Gradescope
- We encourage the use of Piazza for debugging help
- Please do not cheat

Problem

Let $G = (V, E)$ be a graph. A (vertex) k-coloring of G is a function $c : V \to [k]$ for some $k \in \mathbb{N}$. Intuitively, this colors each vertex with one of the colors labeled $\{1, 2, \ldots k\}$. We say a coloring c of G is *proper* if for all $\{u, v\} \in G$, $c(u) \neq c(u)$, that is no pairs of adjacent vertices are the same color. A natural question is whether or not there eixsts a proper k-coloring of a graph for a particular k. This question has many applications. For example, suppose you have n classes to schedule and k time slots in which to schedule them. If a student is enrolled in multiple classes, you can't schedule those classes at the same time. We can reduce this problem to k-colorable by encoding the classes as vertices and adding edges joining all pairs of classes that share a student.

Problem: Given a graph G and a natural number k, determine whether there exists a proper k-coloring of G.

Note This problem is NP-complete, so we don't expect to be able to solve it efficiently in general. You may write a heuristic approach to solve it to quickly enumerate over all possibilities. However, you can also solve it by reducing the problem to SAT and solve it via command-line SAT solver we provide. It uses clever heuristics and is often able to solve instances of SAT encountered in practice quickly. You may run it via the command line by executing the program "minisat" and feeding as input the constraints of the SAT instance in DIMACS CNF format. The last line of the output will be "SATISFIABLE" if the given instance is a YES instance. See this page for details: `https://dwheeler.com/essays/minisat-user-guide.html`.

Input:

- Input should be read in from stdin.
- The first line will contain the number of vertices (n) the number of edges (m) and the parameter k
- Each subsequent line contains two natural numbers u and v, and represents an edge $\{u, v\} \in E$ between the nodes corresponding to those indices
- This input is in a similar format to Program3 and Program6, so you may be able to reuse some code.

Constraint:

- You can expect $n \in [1, 250]$.

Output:

- The output should be written to stdout.
- The output should be a boolean, “True" if there exists a proper k-coloring of G and “False" otherwise.

Examples

Example 1
input:

```
4 6 3
1 2
1 3
1 4
2 3
2 4
3 4
```

output:

```
False
```

Example 2
input:

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

output:

```
False
```

Example 3
input:

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

output:

```
True
```