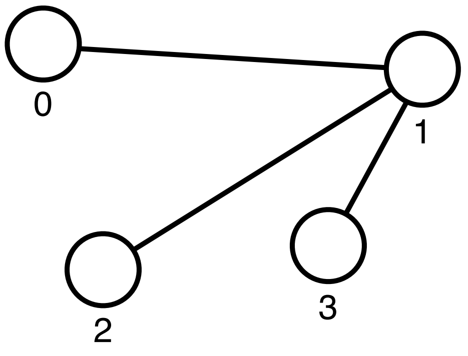
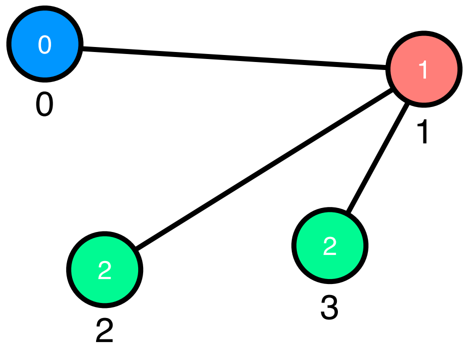
Discrete Optimization Assignment: **Graph Coloring**

1 Problem Statement

In this assignment you will design an algorithm to find the smallest *coloring* of a graph.[1] You are provided with a graph and your task is to label the graph's nodes with as few colors as possible such that all pairs of nodes joined by an edge do not have the same color. Figure 1 illustrates a graph and a three coloring of that graph. The nodes of the graph are labeled with black numbers while the coloring of the graph is labeled with white numbers. You may notice that a three coloring is not a minimal coloring of this graph. In fact, a two coloring is possible.



(a) A graph.



(b) A three-coloring of the graph.

Figure 1: A Graph Coloring Example

2 Assignment

Write an algorithm to minimize the coloring of a graph. The problem is mathematically formulated in the following way. Given a graph $G = \langle N, E \rangle$ with nodes $N = 0 \ldots n-1$ and edges E, let $c_i \in \mathbb{N}$ be a variable denoting the color of node i. Then the graph coloring problem is formalized as the following optimization problem,

$$\begin{array}{ll} \text{minimize:} & \displaystyle\max_{i \in 0 \ldots n-1} c_i \\ \text{subject to:} & \\ & c_i \neq c_j \quad (\langle i, j \rangle \in E) \end{array}$$

3 Data Format Specification

The input consists of $|E| + 1$ lines. The first line contains two numbers $|N|$ and $|E|$. It is followed by $|E|$ lines, each line represents an edge $\langle u_i, v_j \rangle$ where $u_i, v_j \in 0 \ldots |N| - 1$.

[1]See `http://mathworld.wolfram.com/ChromaticNumber.html`

Input Format

```
|N| |E|
u_0 v_0
u_1 v_1
...
u_|E|-1 v_|E|-1
```

The output has two lines. The first line contains two values *obj* and *opt*. *obj* is the numbers of colors used in the coloring (i.e. the objective value). *opt* should be 1 if your algorithm proved optimality and 0 otherwise. The next line is a list of n values in $\mathbb{N}$, one for each of the c_i variables. This line encodes the solution.

Output Format

```
obj opt
c_0 c_1 c_2 ... c_n-1
```

Examples (based on Figure 1)

Input Example

```
4 3
0 1
1 2
1 3
```

Output Example

```
3 0
0 1 2 2
```

4 Instructions

Edit `solver.py` and modify the `solve_it(input_data)` function to solve the optimization problem described above. The function argument, `input_data`, contains the problem data in the format described above. The return value of `solve_it` is a solution to the problem in the output format described above. Your `solve_it` implementation can be tested with the command,

```
python ./solver.py ./data/<inputFileName>
```

You should limit the `solve_it` method to terminate within 5 hours, otherwise the submission will not be eligible for full credit. You may choose to implement your solver directly in python or modify the `solve_it` function to call an external application.

Resources You will find several graph coloring instances in the `data` directory provided with the handout.

Handin Run `submit.py` with the command, `python ./submit.py`. Follow the instructions to apply your `solve_it` method on the various assignment parts. You can submit multiple times and your grade will be the best of all submissions. However, it may take several minutes before your assignment is graded; please be patient. You can track the status of your submission on the *feedback* section of the assignment website.

Grading Infeasible solutions (i.e. those that do not conform to the output format or violate problem constraints) will receive 0 points. Feasible solutions will receive at least 3 points. Feasible solutions passing a low quality bar will receive at least 7 points and solutions meeting a high quality bar will receive all 10 points. The grading feedback indicates how much your solution must improve to receive a higher grade.

Collaboration Rules In all assignments we encourage collaboration and the exchange of ideas on the discussion forums. However, please refrain from the following:

1. Posting code or pseudo-code related to the assignments.
2. Using code which is not your own.
3. Posting or sharing problem solutions.

Discussion of solution quality (i.e. objective value) and algorithm performance (i.e. run time) is allowed and the assignment leader board is designed to encourage such discussions.

Warnings

1. It is recommended you do not modify the `data` directory. Modifying the files in the data directory risks making your assignment submissions incorrect.
2. You cannot rename the `solver.py` file or the `solve_it()` method.
3. Be careful when using global variables in your implementation. The `solve_it()` method will be run repeatedly and it is your job to clear the global data between runs.
4. `solver.py` must remain in the same directory as `submit.py`.

Hint The optimal value for `data/gc_1000_5` is near 85.

5 Technical Requirements

You will need to have python 2.7.9 or 3.5 (at least) installed on your system (installation instructions,
`http://www.python.org/downloads/`).