CMSC 501: Advanced Algorithms
Spring 2018

Class Project: **Super Graph Coloring**

Instructor: Thang Dinh
Virginia Commonwealth University
School of Engineering

Due: Tuesday, May 1, 2018

Consider an undirected graph $G = (V, E)$, in which each node $u \in V$ may be colored with some color between 1 and C. Your task is to write a program that determine the colors for the uncolored nodes in the graph such that

- For all edges $(u, v) \in E$, u and v have different colors.
- The number of additional colors needed is minimum.

Input (Standard input): Includes multiples lines. The first line contains two integers $1 \leq n \leq 1000, 1 \leq m \leq 100000$ that correspond to the number of nodes and edges, respectively.

Each of the following m lines contain two integers u and v, separated by one space, to denote an edge from u to v. Nodes are numbered from 1 to n.

The last line contains n integers that are the colors of the nodes. Uncolored nodes are indicated with color 0.

Output (Standard output): The first line contains an integer $0 \leq D \leq n$ that is the number of additional colors needed to color all the uncolored nodes in the graph.

The next line contains n positive integers that are colors of the nodes.

Sample input/output:

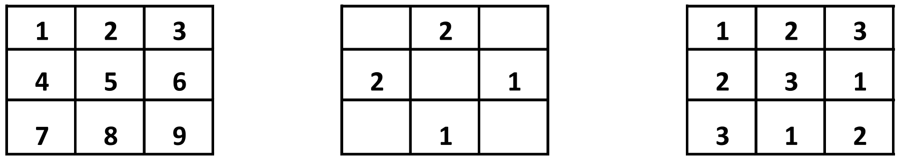

Figure 1. From left to right a) The nodes in the graph are equivalent to the 9 squares in the leftmost figure. There is an edge between two squares if and only if they are on a same row or column b) The initial coloring of the nodes in the graph c) The final coloring of 9 nodes using 3 colors

Input	Output
9 23	1
1 2	1 2 3 2 3 1 3 1 2
1 3	
1 4	
1 7	
2 3	
2 5	
2 8	
3 6	
3 9	
4 5	
4 6	
4 7	
5 6	
5 8	
6 9	
7 8	
7 9	
8 9	
0 2 0 2 0 1 0 1 0	

Explain of the input/output: The nodes in the graph are equivalent to 9 squares as shown in the above figure. There is an edge between two squares if and only if they are on a same row or column. The initial coloring of the nodes in the graph is shown in the middle figure. We can use one additional color (color 3) to color all the remaining nodes.

Submission: Your submission in Blackboard should include the following items

I. Your program in Java/C++ that solves the above problem following the above input/output format. A makefile and/or compiling instruction should be included if you have multiple source files. Your program should not take more than 2 minutes to terminate on any graph within the limits described in the Input section.

II. A report outline the numbers of colors used by your program on random graphs assuming *no nodes have any color at the beginning.* The report should have at least

		1						4
	2	3	1			9		
	8	6		5		3		
	6			4				
3			9		1			5
				7			3	
		4		3		6	1	
		2			5	8	4	
7						5		

Figure 2. A sudoku puzzle that is equivalent to an instance of our super graph coloring with 81 nodes

two parts. Run your program for graphs of sizes 50, 100,..., 500 and report both the number of used colors and the maximum degree.

III. A graph that corresponds to the following sudoku and the output of your program on that sudoku graph.

To generate random graphs, you can write your own program or use the source code here: `http://algs4.cs.princeton.edu/41graph/GraphGenerator.java.html`. Other options include Boost C++ library `http://www.boost.org/doc/libs/1_54_0/libs/graph/doc/erdos_renyi_generator.html` or Igraph `http://igraph.org/python/doc/tutorial/tutorial.html`

Grading: The grading will be based on your report and your (relative) performance of your program on unpublished test files.