CS 156
Spring 2011

Assignment 2
Parallel Constraint Solving
You can take up to 2 weeks
First Progress Report Due Thursday, 27 January 2011

Develop a Java program that uses concurrent threads to solve *SudoGraph* problems. The ultimate objective is speed-up.

SudoGraph is generalization of Sudoku and map coloring. A SudoGraph problem has the following components:

- A number N of nodes (which are implicitly numbered 1, 2, …, N).
- A number C of colors (which are implicitly numbered 1, 2, …, C).
- A set of 0 or more constraints. Each constraint is a subset of the set of nodes.

A *solution* to a SudoGraph problem consists of an assignment of one of the colors to each of the nodes, such that the colors in each constraint are unique.

Example:

N = 7, C = 4, Constraints = {{1, 2, 3, 4}, {2, 3, 4, 7}, {2, 4, 5, 6}}

This example has a solution, the sequence of colors

[3, 1, 4, 2, 3, 4, 3]

because if we assign that sequence of colors to nodes 1 through 7, then the colors assigned to the constraints are as shown in this table.

{{3, 1, 4, 2}, {1, 4, 2, 3}, {1, 2, 3, 4}}

as you can see by this table:

Node #	1	2	3	4	5	6	7
Assigned color	3	1	4	2	3	4	3
Constraint 1	3	1	4	2			
Constraint 2		1	4	2			3
Constraint 3		1		2	3	4	

Your program should read a single problem from System.in. The format of the problem will be a sequence of numbers, beginning with N followed by C followed by the constraints. Each constraint is ended by a 0 delimiter. (This is so you don't have to deal with read-line issues.)

For the above example, the input would be:

7 4
1 2 3 4 0
2 3 4 7 0
2 4 5 6 0

The output should be a table such as the one above (space-separated, no graphics), with 0's where there are blank entries.

If there is no solution, the output should simply be a single 0.