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

*SudoGraph* is a 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:

<table>
<thead>
<tr>
<th>Node #</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
</tr>
</thead>
<tbody>
<tr>
<td>Assigned color</td>
<td>3</td>
<td>1</td>
<td>4</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>3</td>
</tr>
<tr>
<td>Constraint 1</td>
<td>3</td>
<td>1</td>
<td>4</td>
<td>2</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Constraint 2</td>
<td>1</td>
<td>4</td>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td>3</td>
</tr>
<tr>
<td>Constraint 3</td>
<td>1</td>
<td></td>
<td>2</td>
<td>3</td>
<td>4</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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.