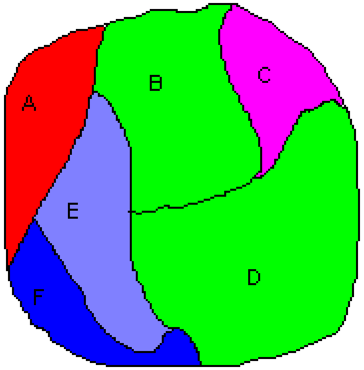
Parallel Graph Coloring

What is Graph Coloring?

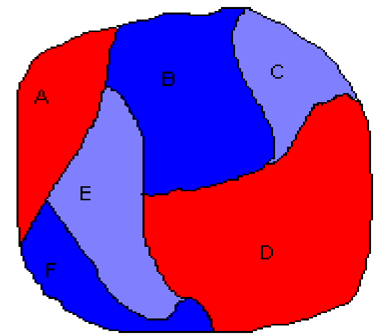
Graph coloring problem is to assign colors to certain elements of a graph subject to certain constraints.

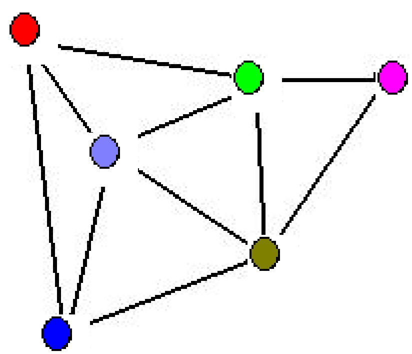
Vertex coloring is the most common graph coloring problem. The problem is, given m colors, find a way of coloring the vertices of a graph such that no two adjacent vertices are colored using same color.

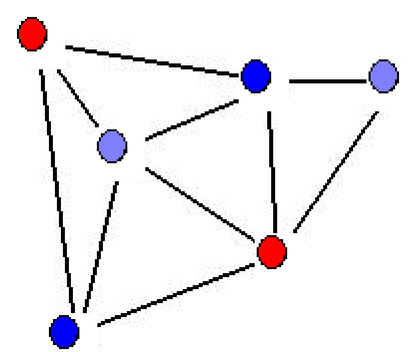
Origin of the problem



Origin of the problem

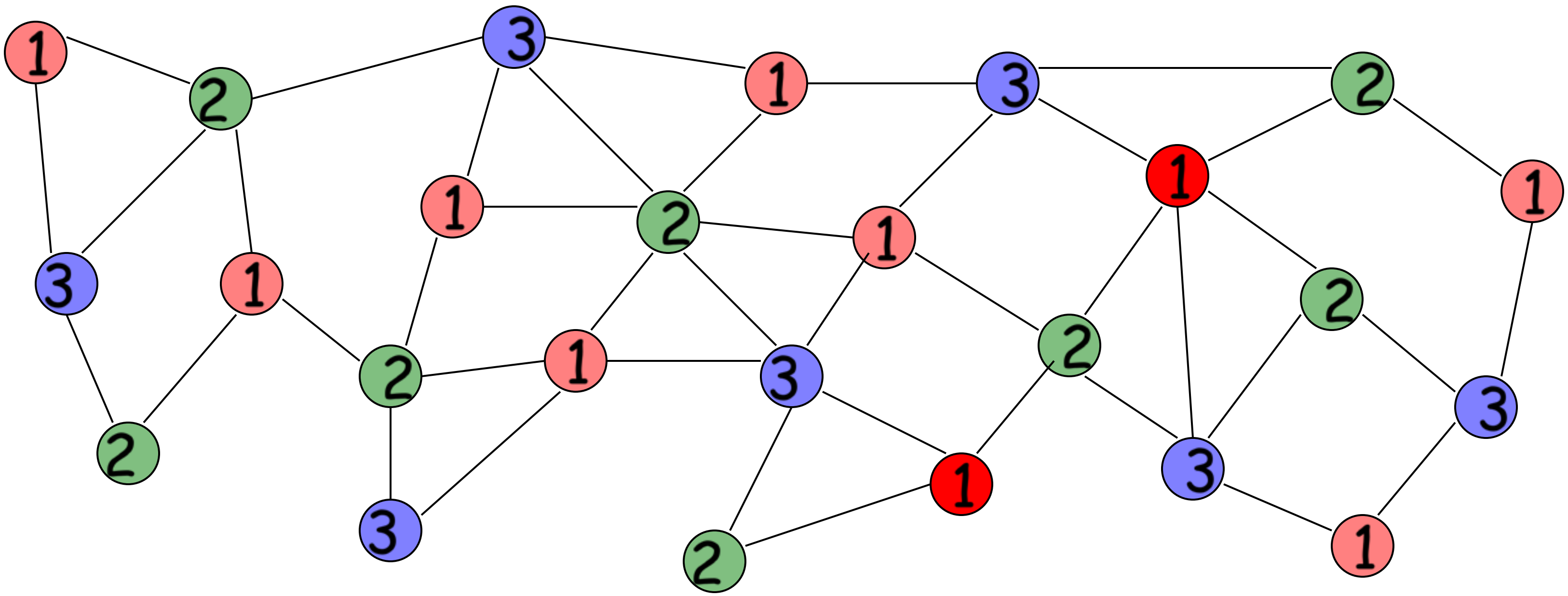






$k-$coloring: a valid coloring with k colors

Example: $3-coloring$



Chromatic number $\varphi(G)$:

The smallest number of colors that can be used to give a valid coloring in graph G

NP Complete!!!

Sequential $\Delta + 1$-coloring

For any graph G
there is a $\Delta + 1$-coloring

Therefore, $\varphi(G) \leq \Delta + 1$

Sequential Coloring Algorithm

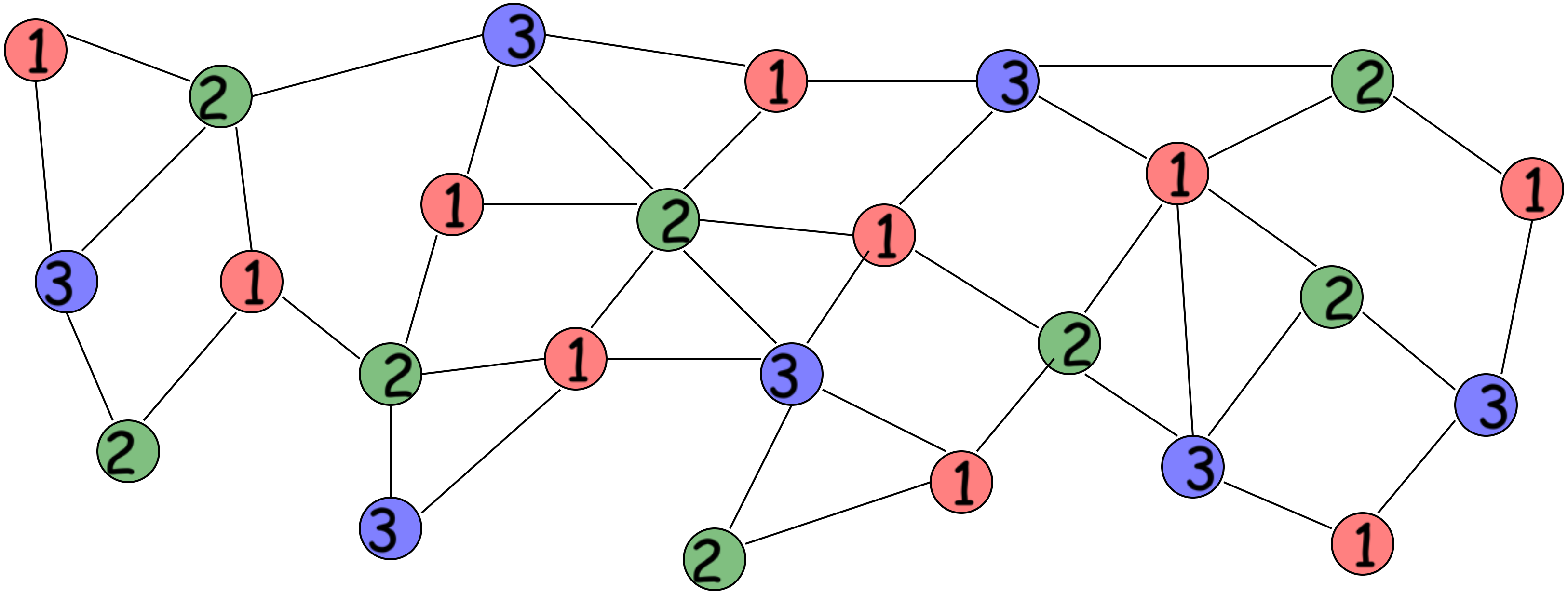
Mark all entries in the palettes of all the nodes as available

Repeat:

1. Pick an uncolored node v
2. Let c be an available color (from v's palette) (such a color always exists)
3. Color node v with color c
4. Mark c as unavailable in the color palette of every neighbor of v

Until all nodes are colored

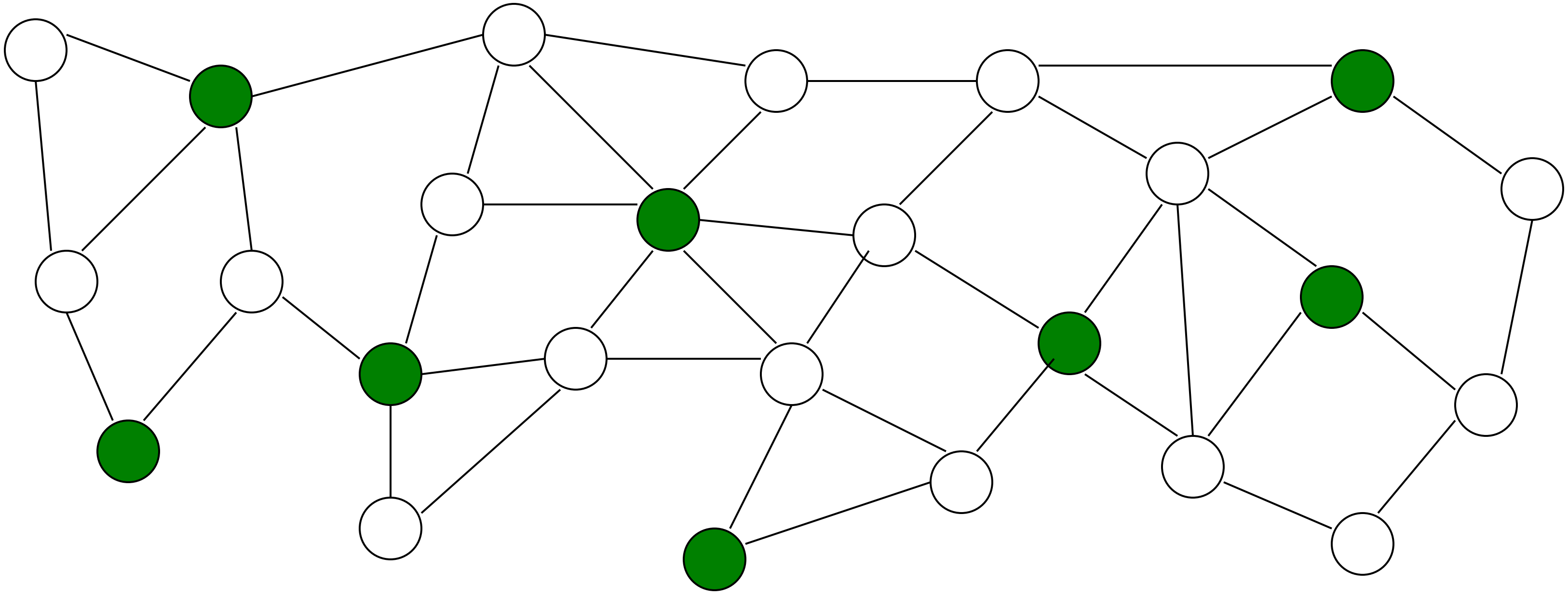
Example coloring



Coloring and MIS

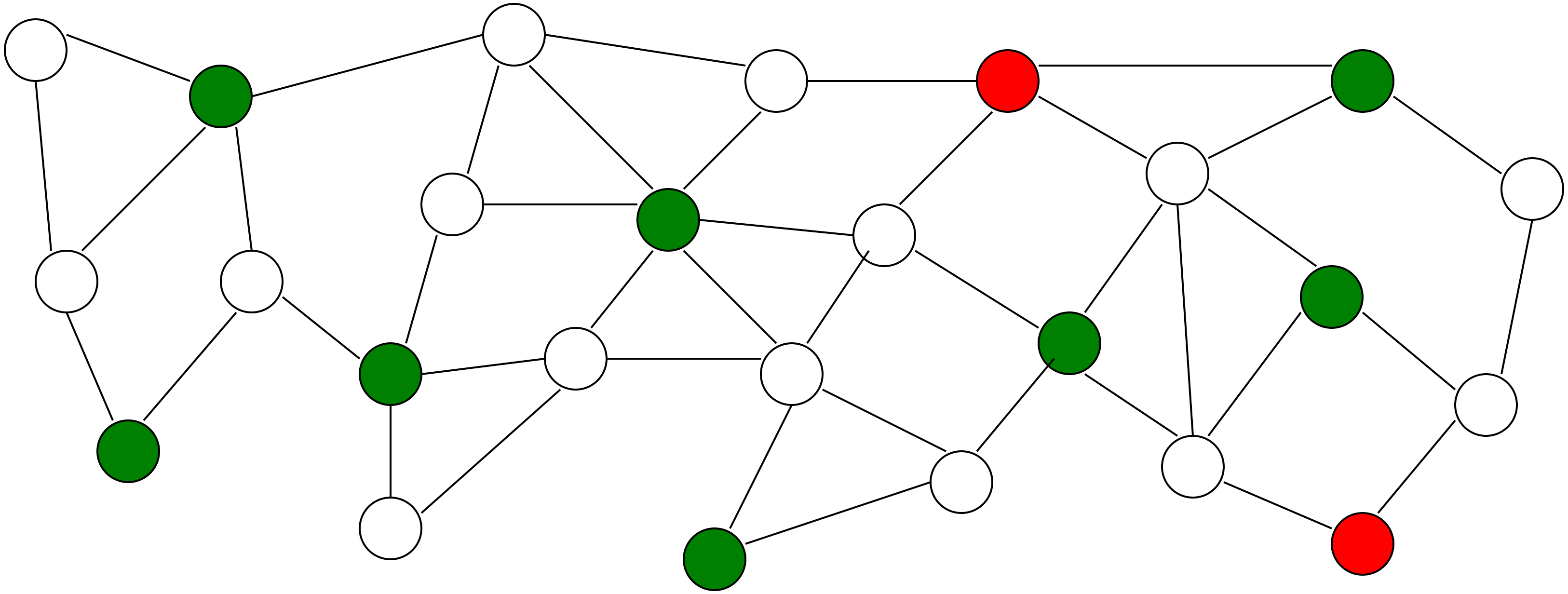
In a valid coloring, the nodes of same color form an independent set

Independent Set



However, the independent set may <u>not</u> be maximal:

New Independent set (Maximal)



Vertex Coloring is reduced to MIS

Consider an uncolored graph G

Coloring algorithm for G using MIS:

$c \leftarrow 1;$

Repeat:

Find a MIS in the uncolored nodes;

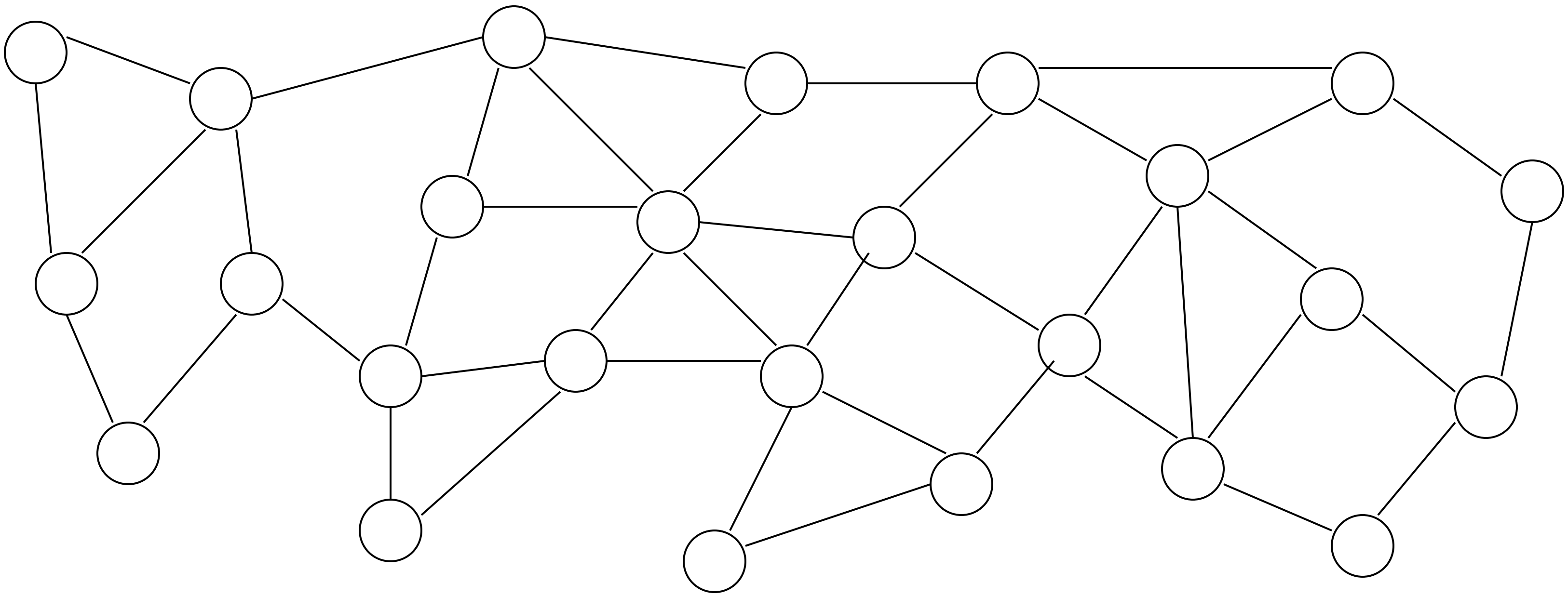
Assign color c to each node in MIS;

$c \leftarrow c + 1;$

Until every node is colored;

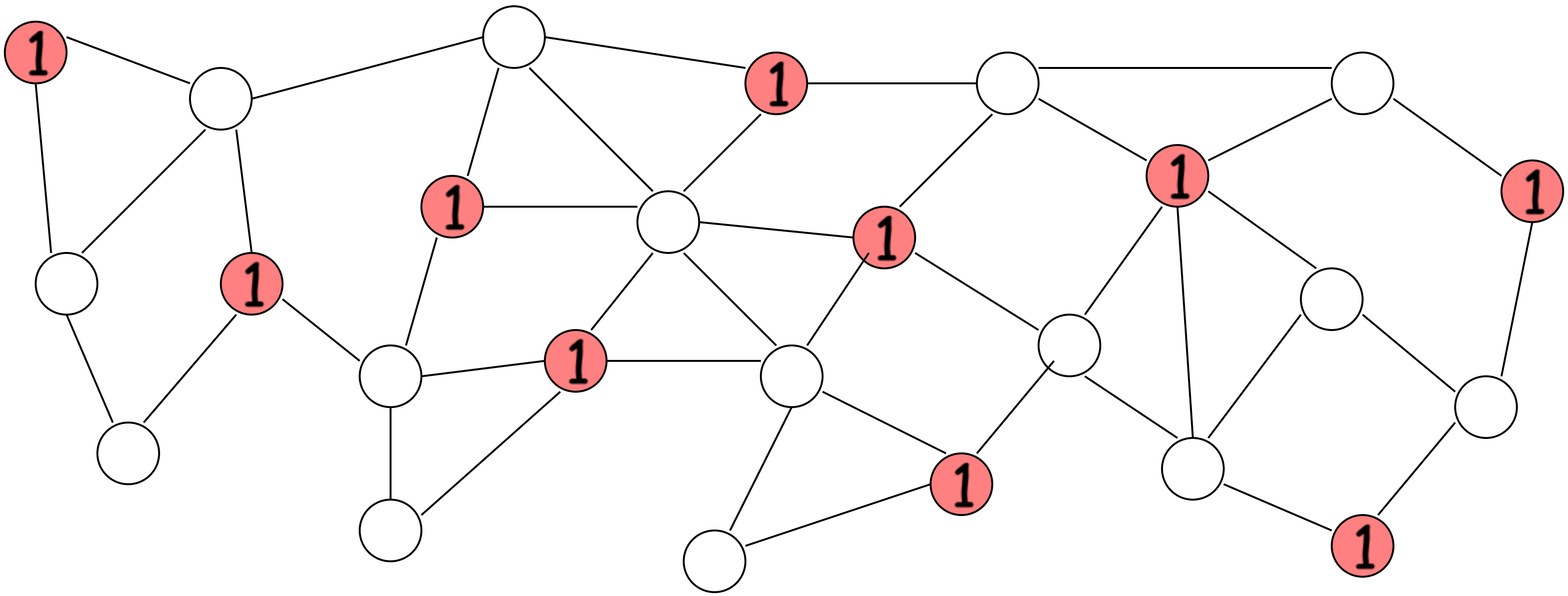
Example:

initially, all nodes are uncolored



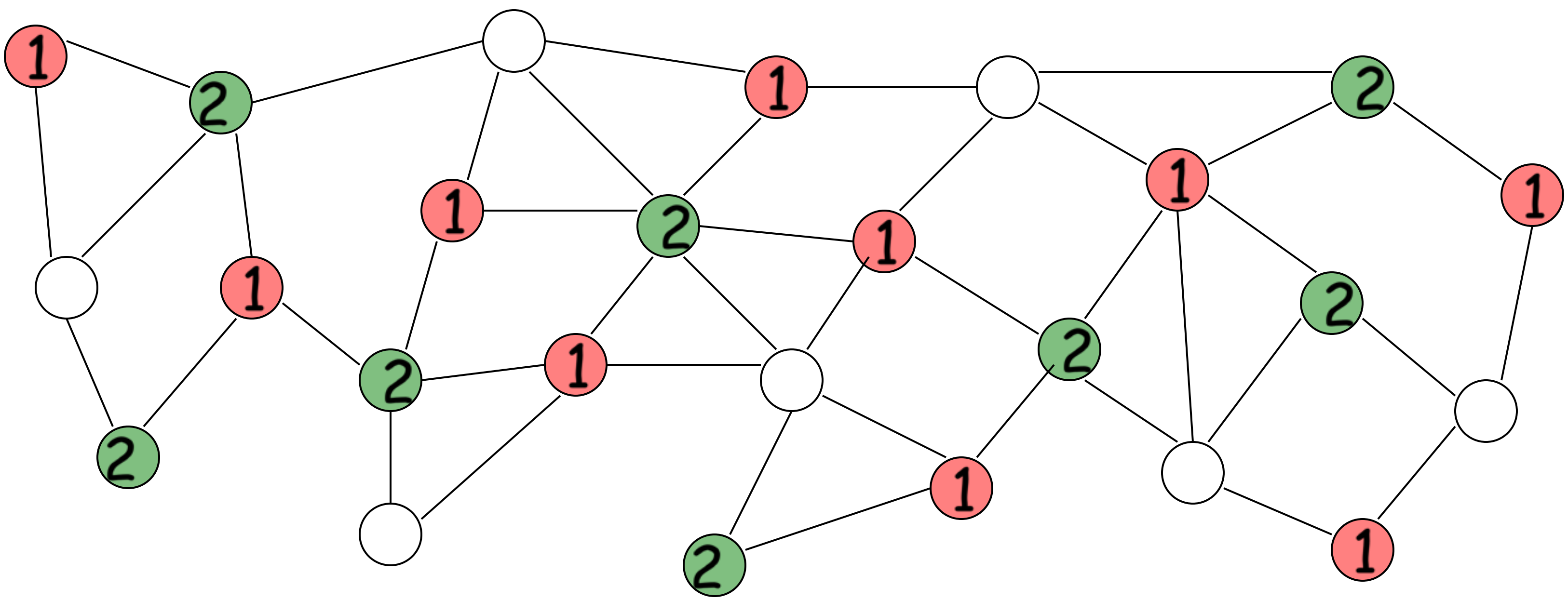
Iteration 1:

Find an MIS of the uncolored nodes and give to the nodes color 1



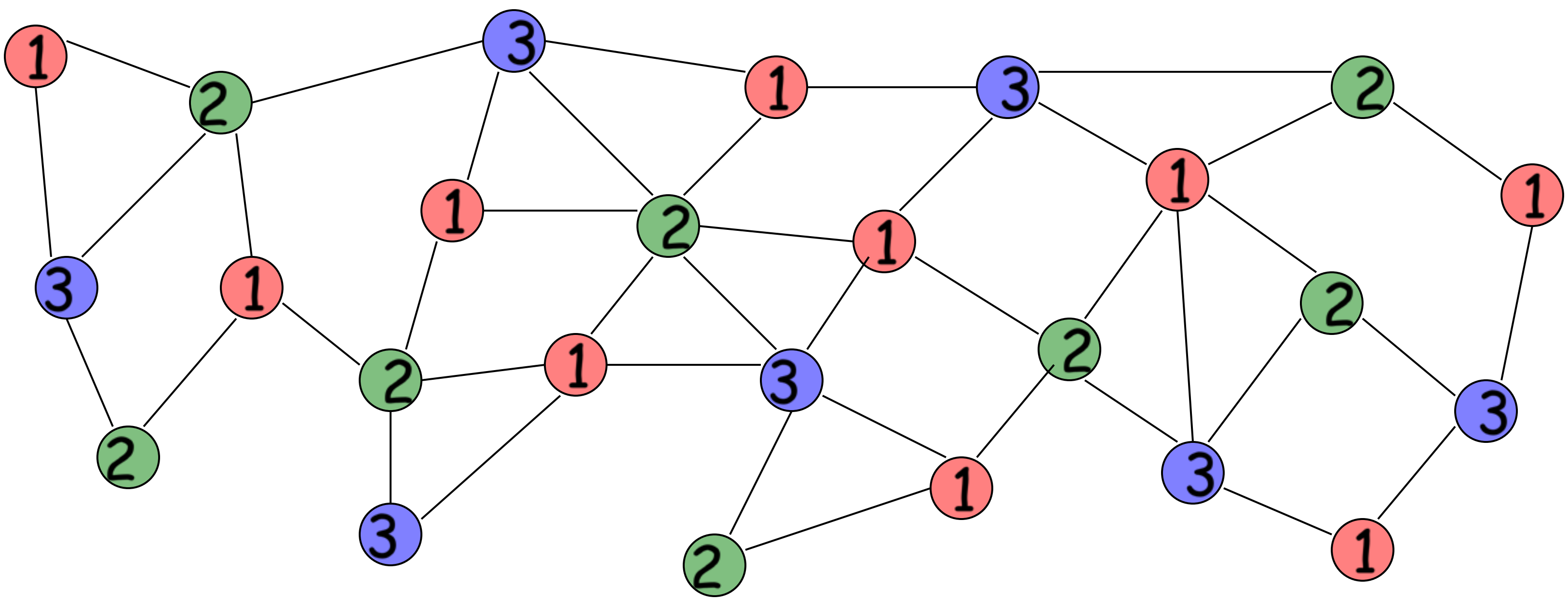
Iteration 2:

Find an MIS of the uncolored nodes and give to the nodes color 2



Iteration 3:

Find an MIS of the uncolored nodes and give to the nodes color 3



A Simple Randomized 2Δ-Coloring Algorithm

- Parallel Algorithm
- Randomized Algorithm

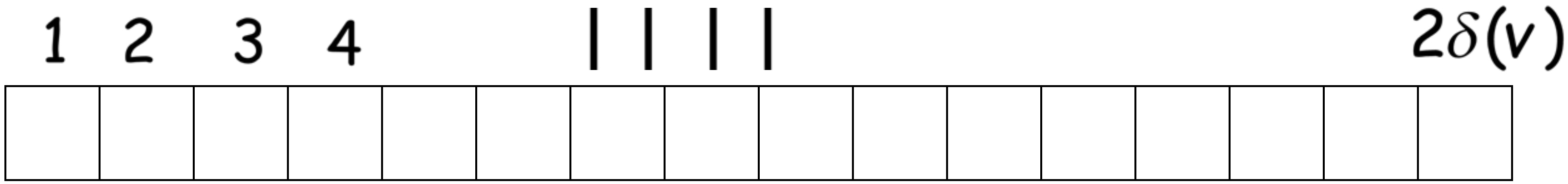
Running time: $O(\log n)$

with high probability

(n is the number of nodes)

Each node v has a palette with $2\delta(v)$ colors

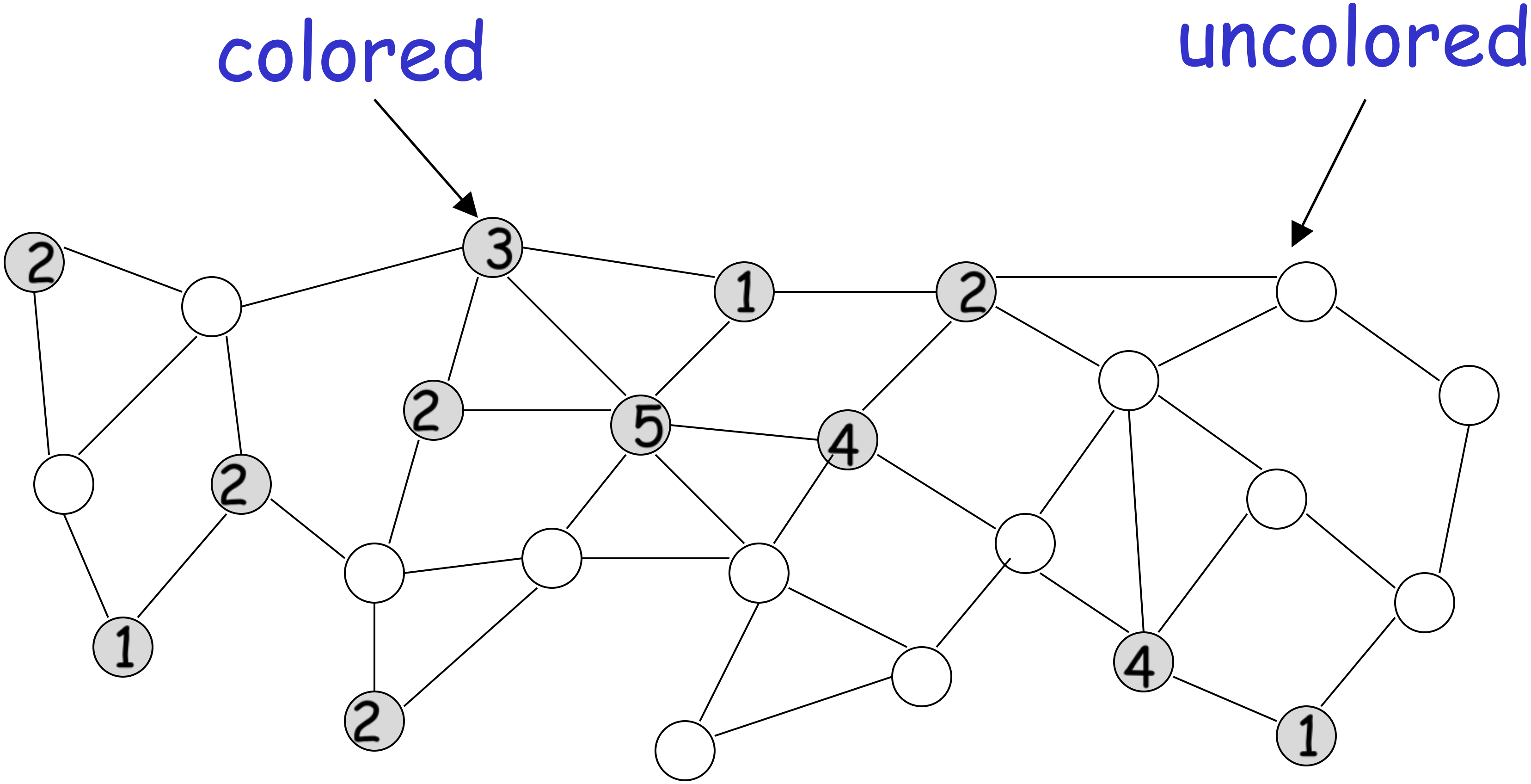
Palette of node v



Initially all colors in palette are available

The algorithm works in phases

At the beginning of a phase,
there are two kinds of nodes:



Algorithm for node v

Repeat (iteration = phase)

Pick a color c uniformly at random
from available palette colors;

Send color c to neighbors;

If (some neighbor chose same color c)

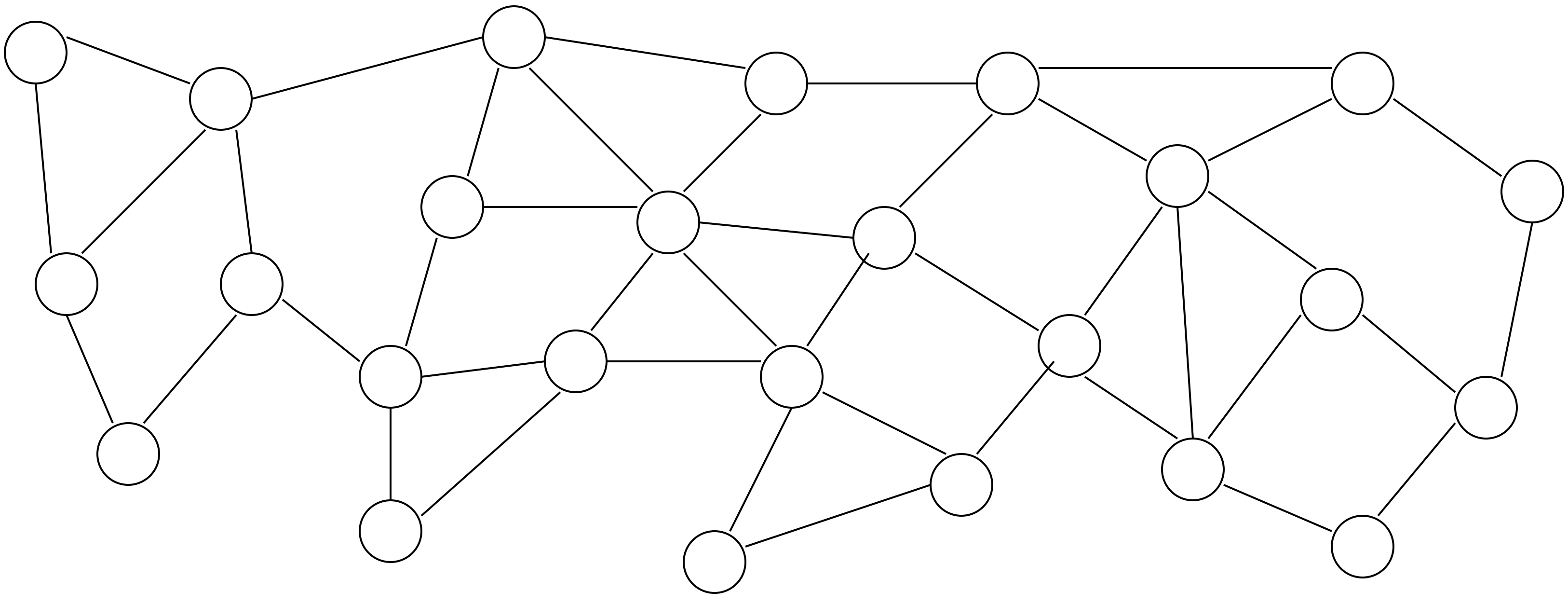
Then Reject color c ;

Else Accept color c ;

Inform neighbors about color c ;
(so that they mark color c as unavailable)

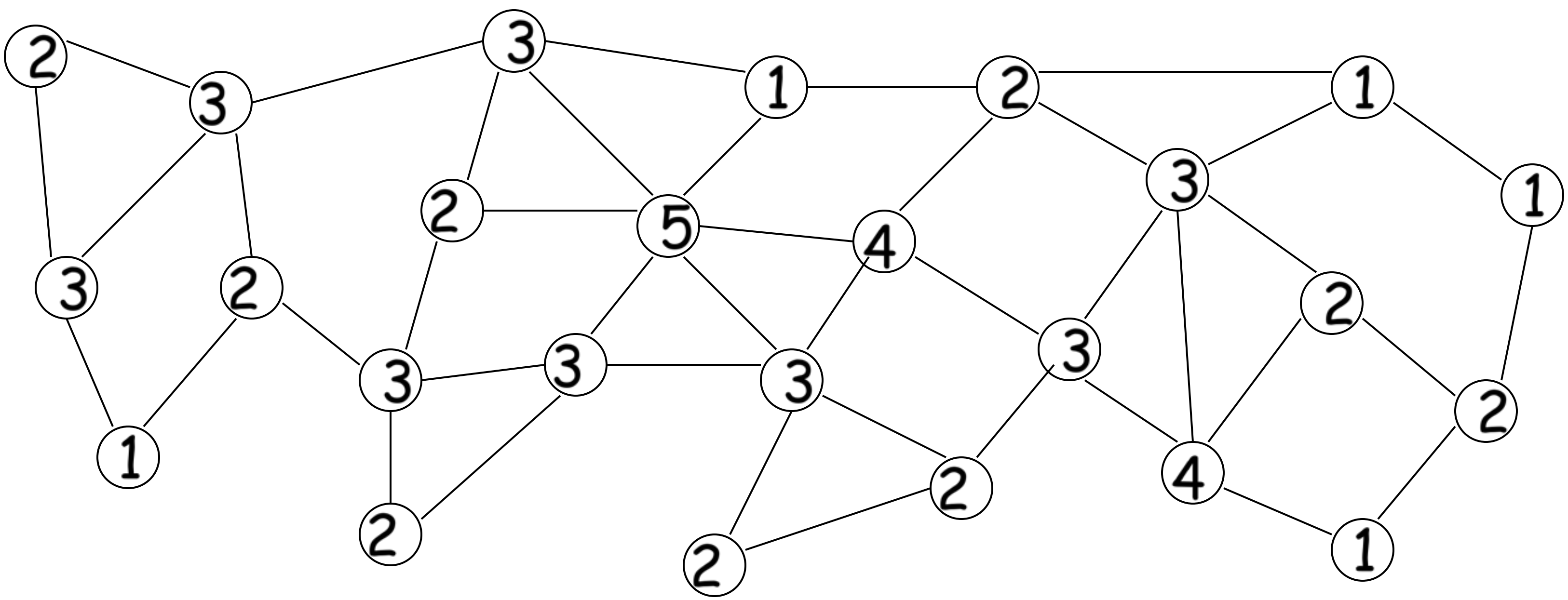
Until color is accepted;

Example execution

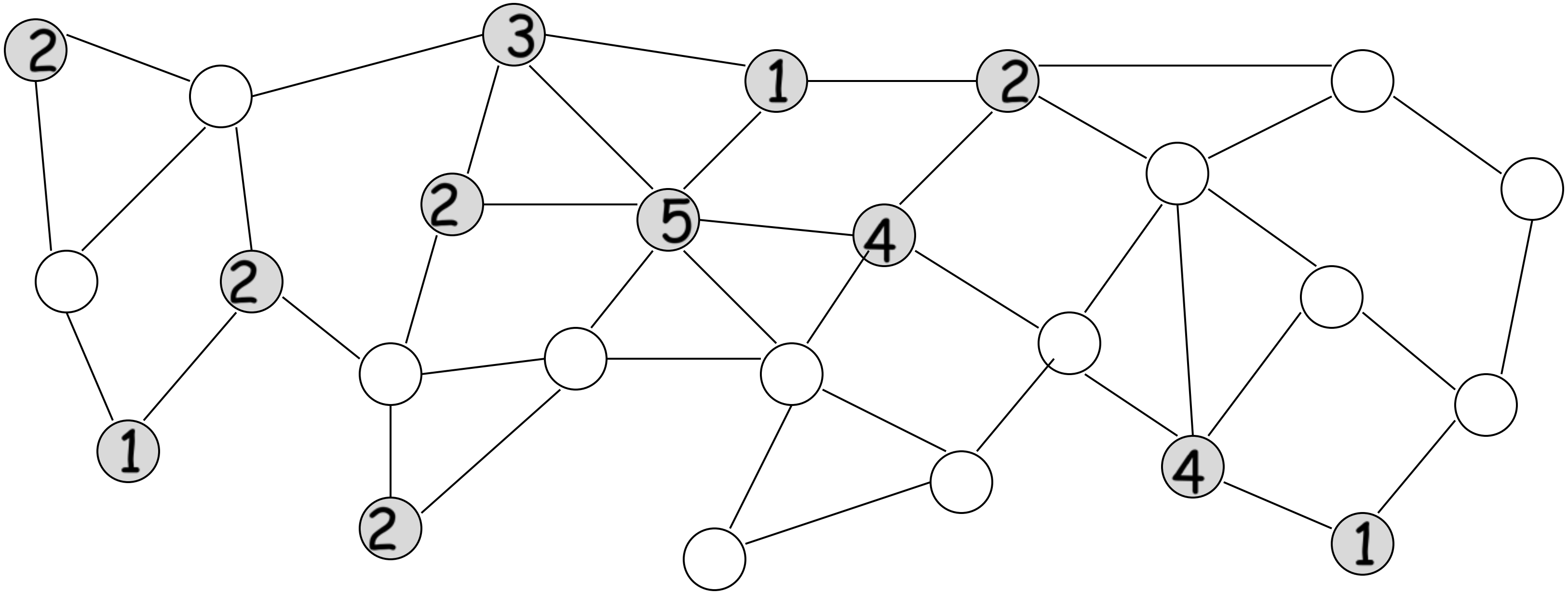


Phase 1: (iteration 1 of synchronous exectution)

Nodes pick random colors

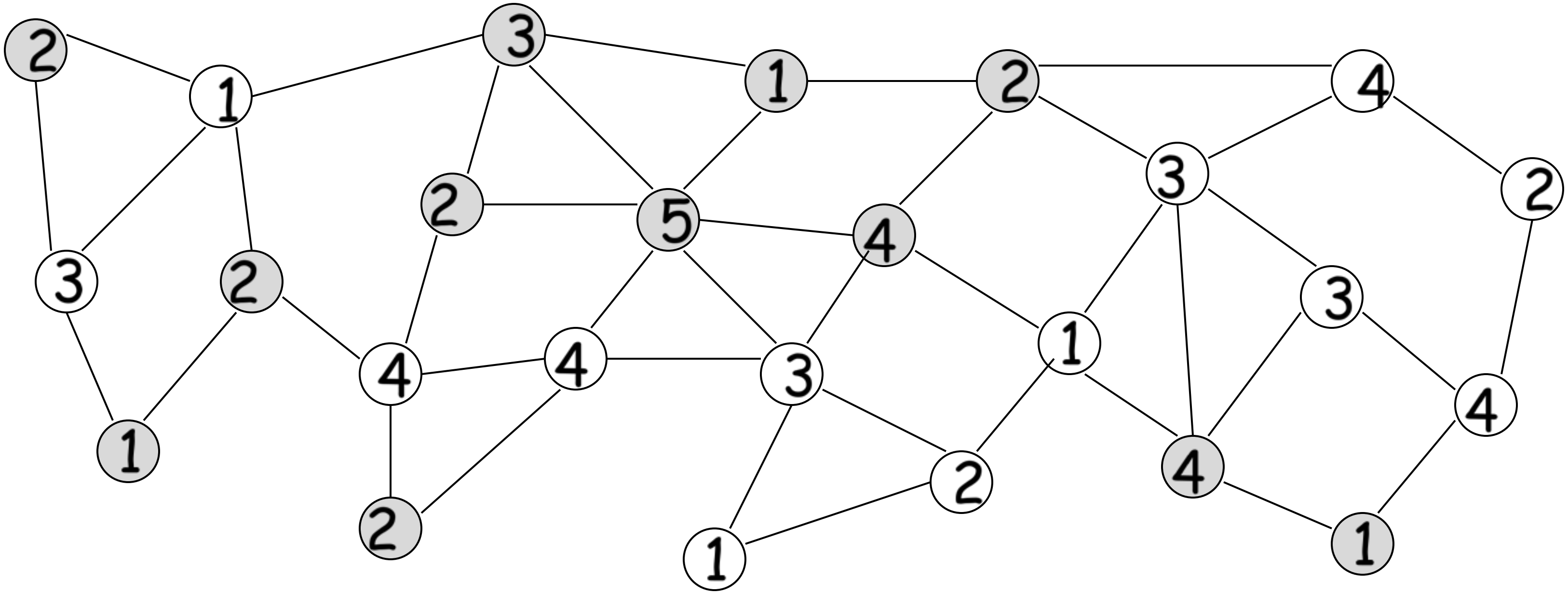


Successful Colors

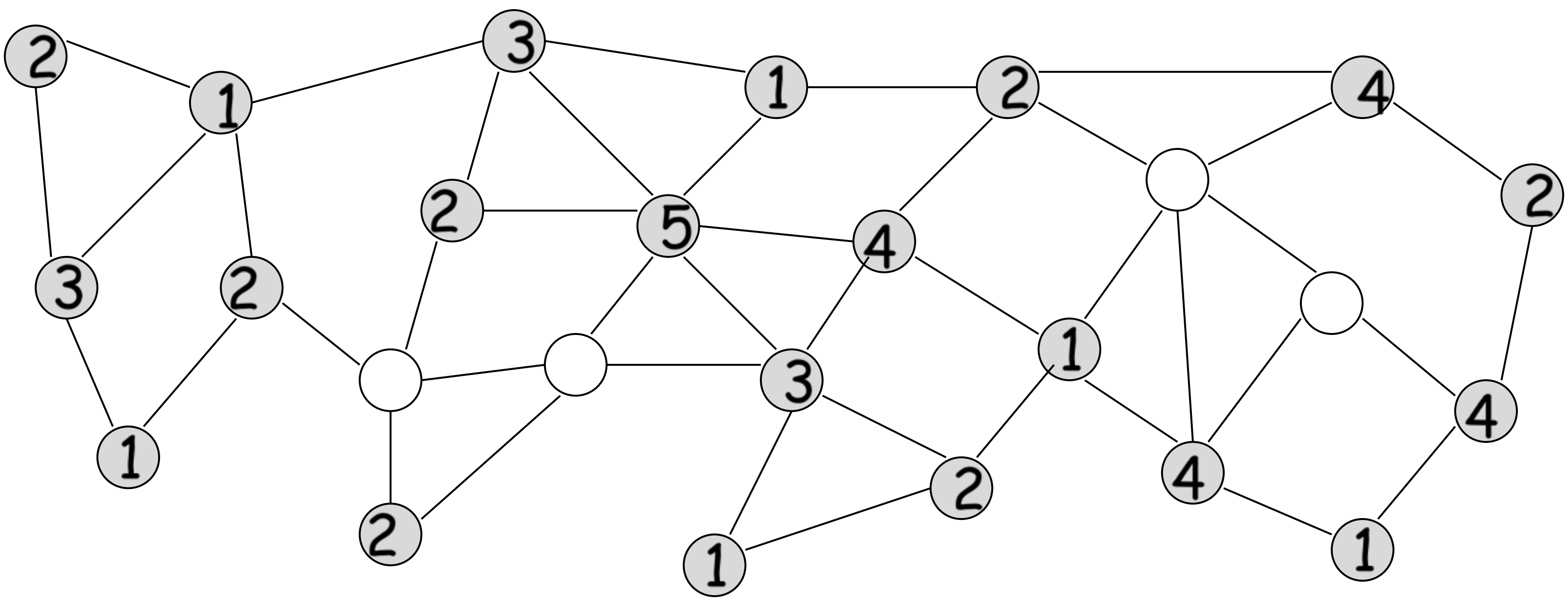


Phase 2: (iteration 2)

Nodes pick random colors

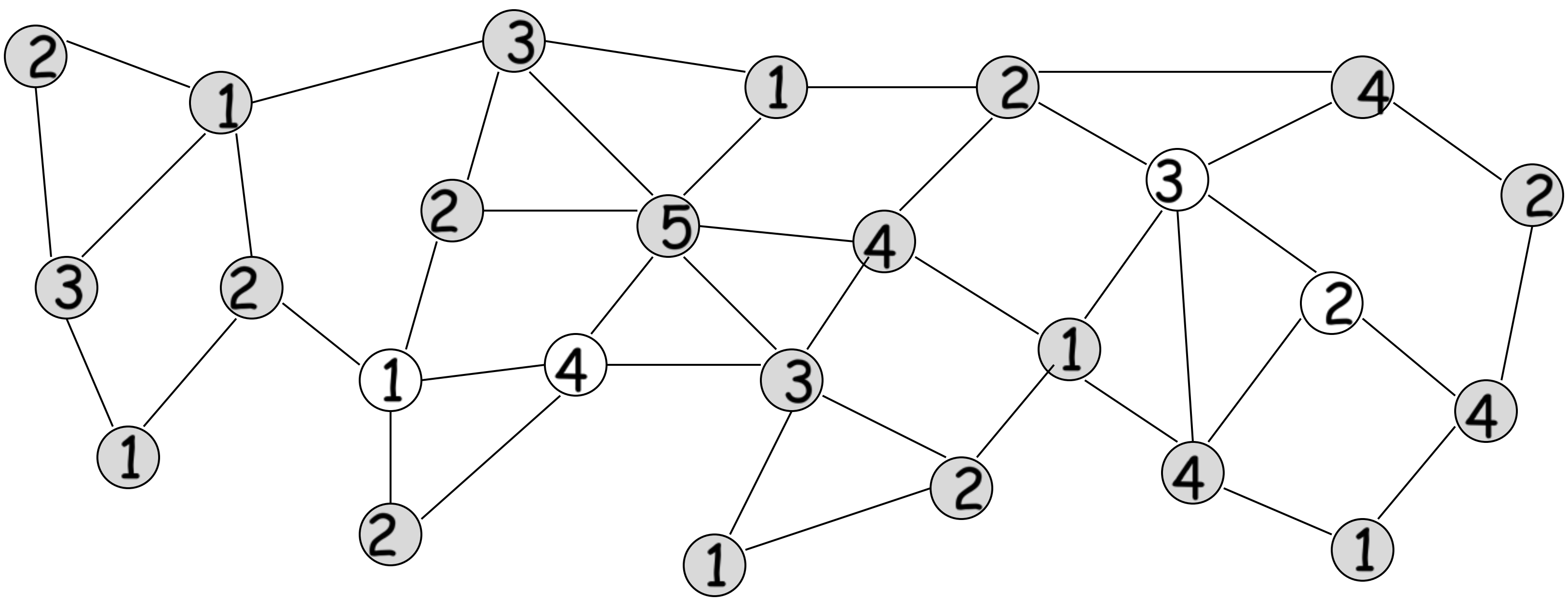


Successful Colors

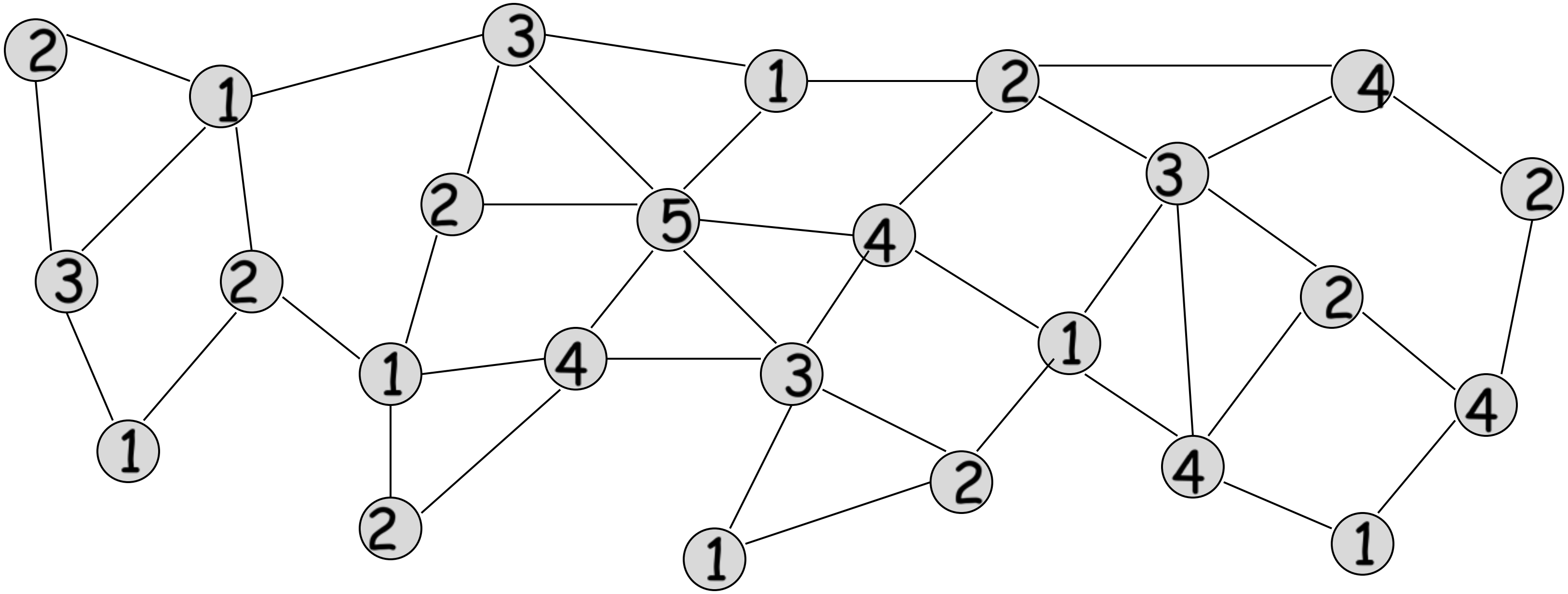


Phase 3: (iteration 3)

Nodes pick random colors



End of execution



A Randomized $\Delta + 1$-Coloring Algorithm

- Parallel Algorithm

- Randomized Algorithm

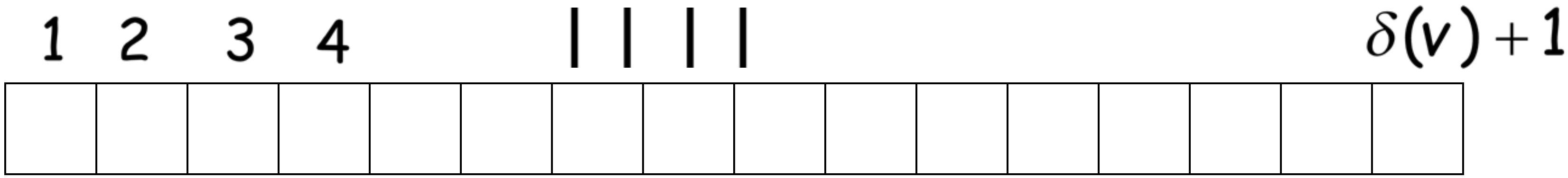
  Running time: $O(\log n)$

  with high probability

(similar with the 2Δ -coloring algorithm, but now the color palette size is $\delta(v)+1$)

Each node v has a palette with $\delta(v)+1$ colors

Palette of node v



Initially all colors in palette are available

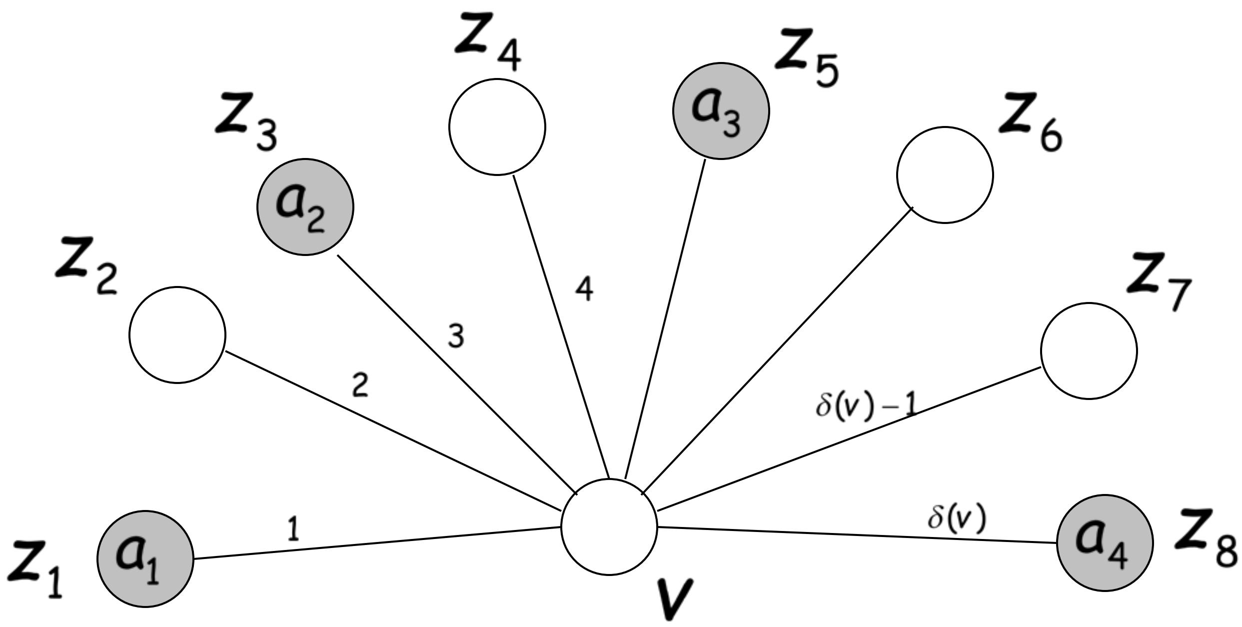
(Recall: $\delta(v)$ is the node's degree)

At the beginning of a phase:

$U(v)$: uncolored neighbors of v

$|U(v)|$: uncolored degree of v

Example: $U(v) = \{z_2, z_4, z_6, z_7\}$



Algorithm for node v

Repeat (iteration = phase)

Pick a color c uniformly at random from available palette colors;

Send color c to neighbors;

If (some neighbor z with $|U(z)| \geq |U(v)|$ chose same color c)

Then Reject color c ;

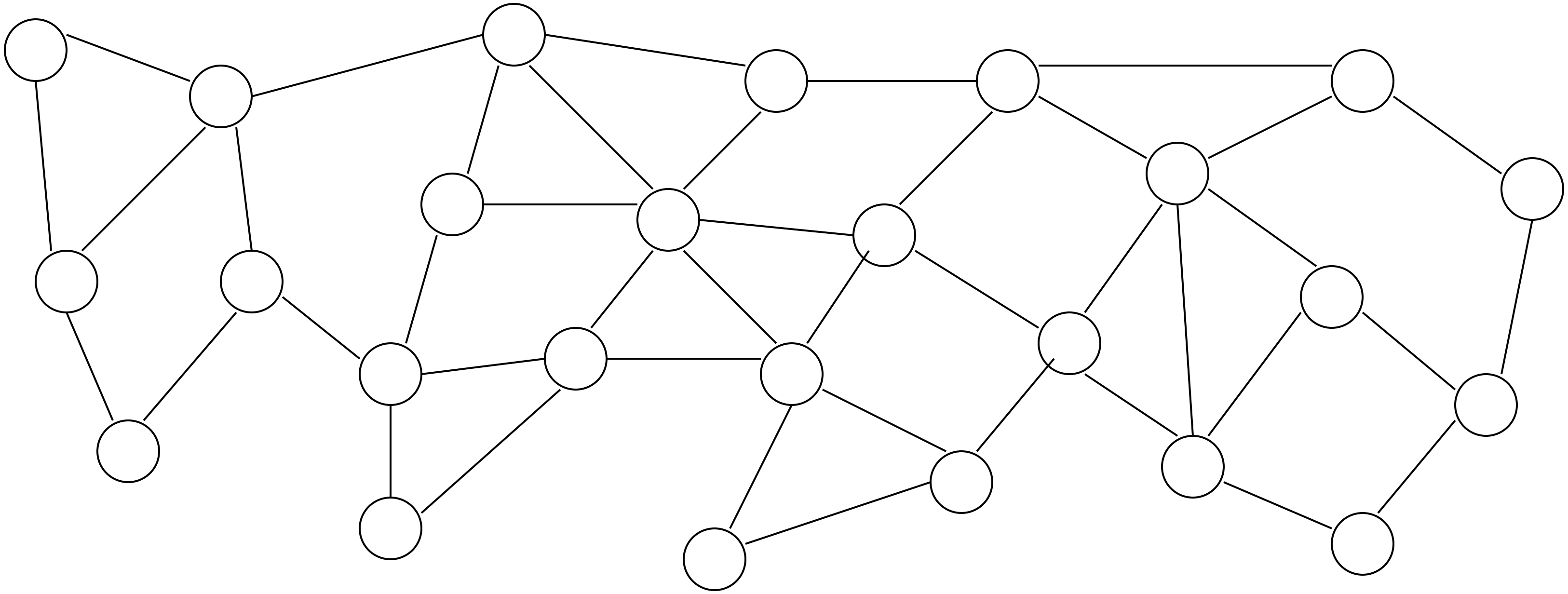
Else Accept color c ;

Inform neighbors about color c ;

(so that they mark color c as unavailable)

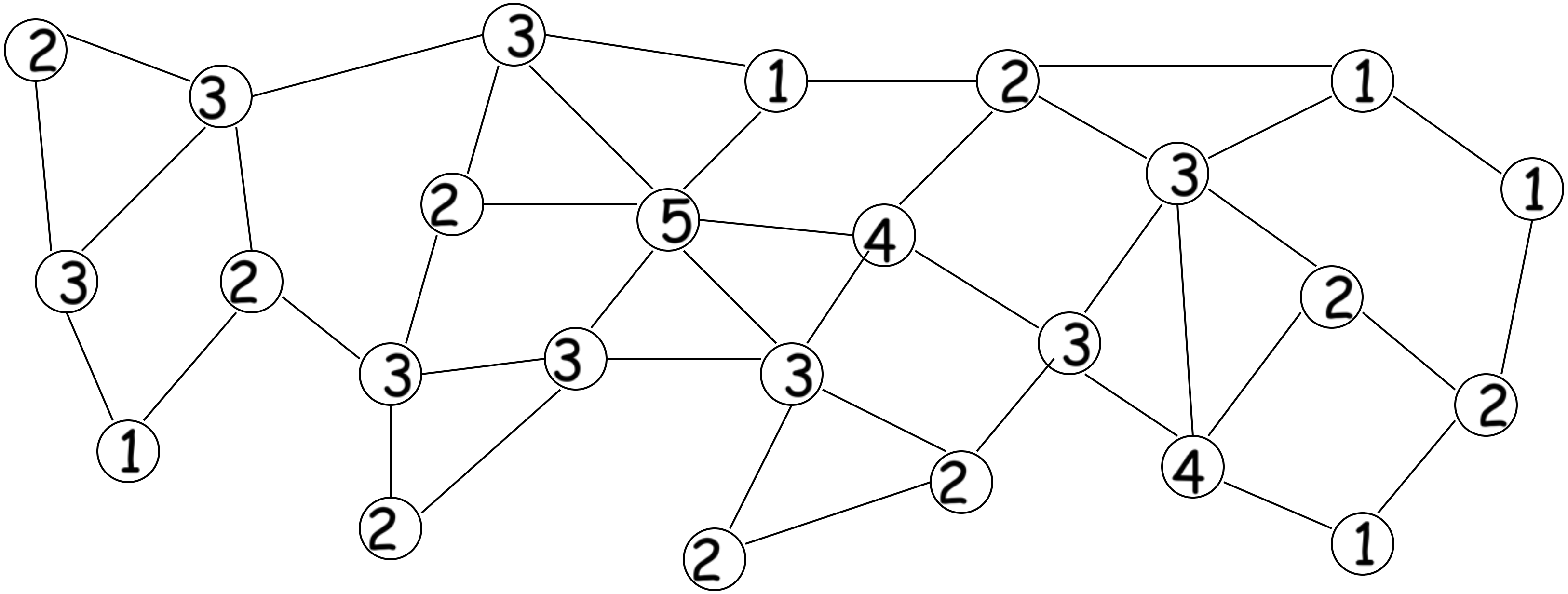
Until color is accepted;

Example execution



Phase 1: (iteration 1)

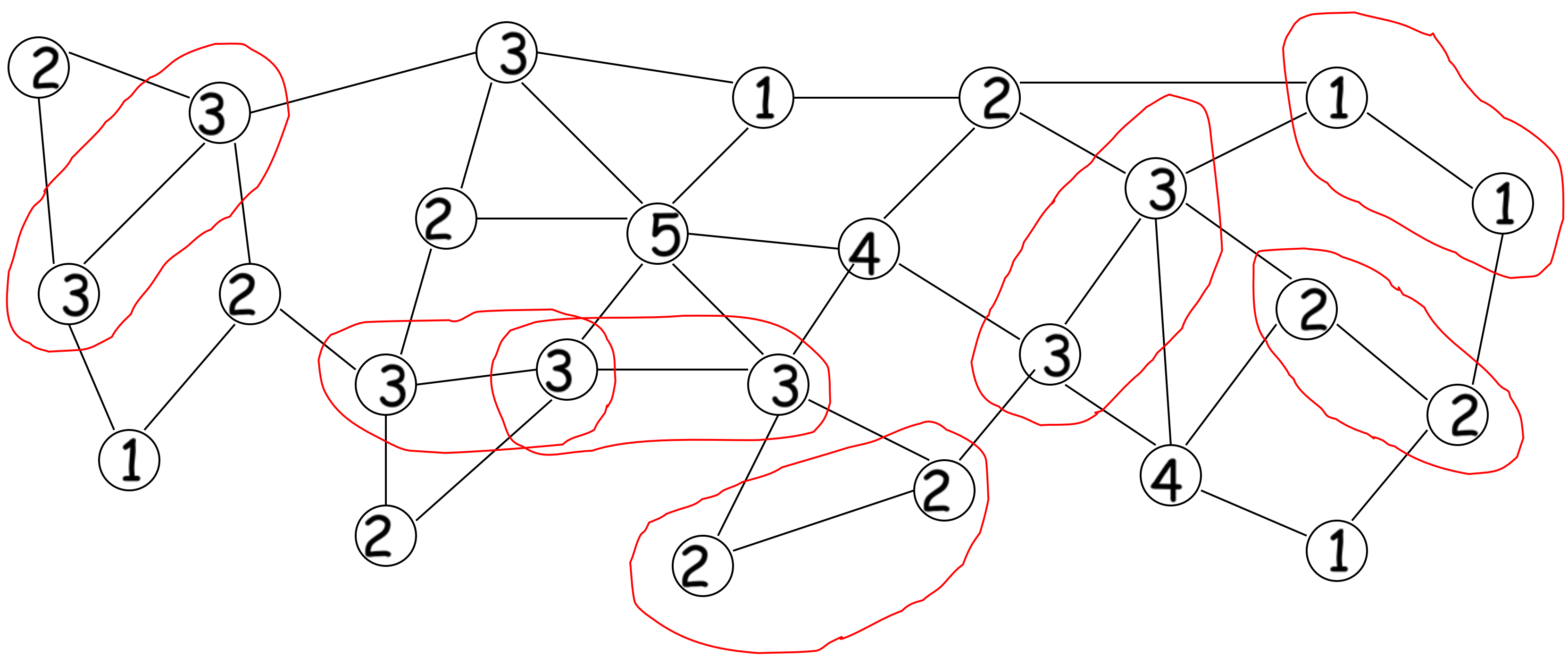
Nodes pick random colors



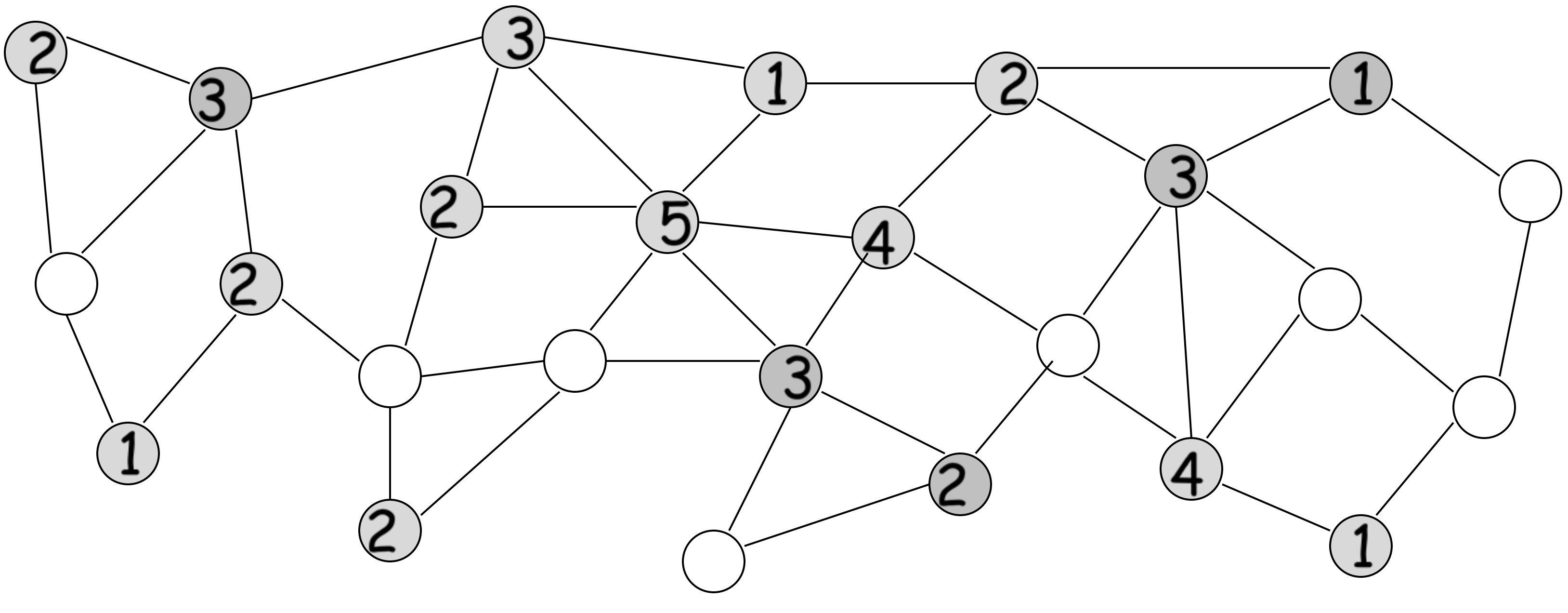
Conflicts

For this phase, uncolored degree = degree

The nodes of higher uncolored degree win

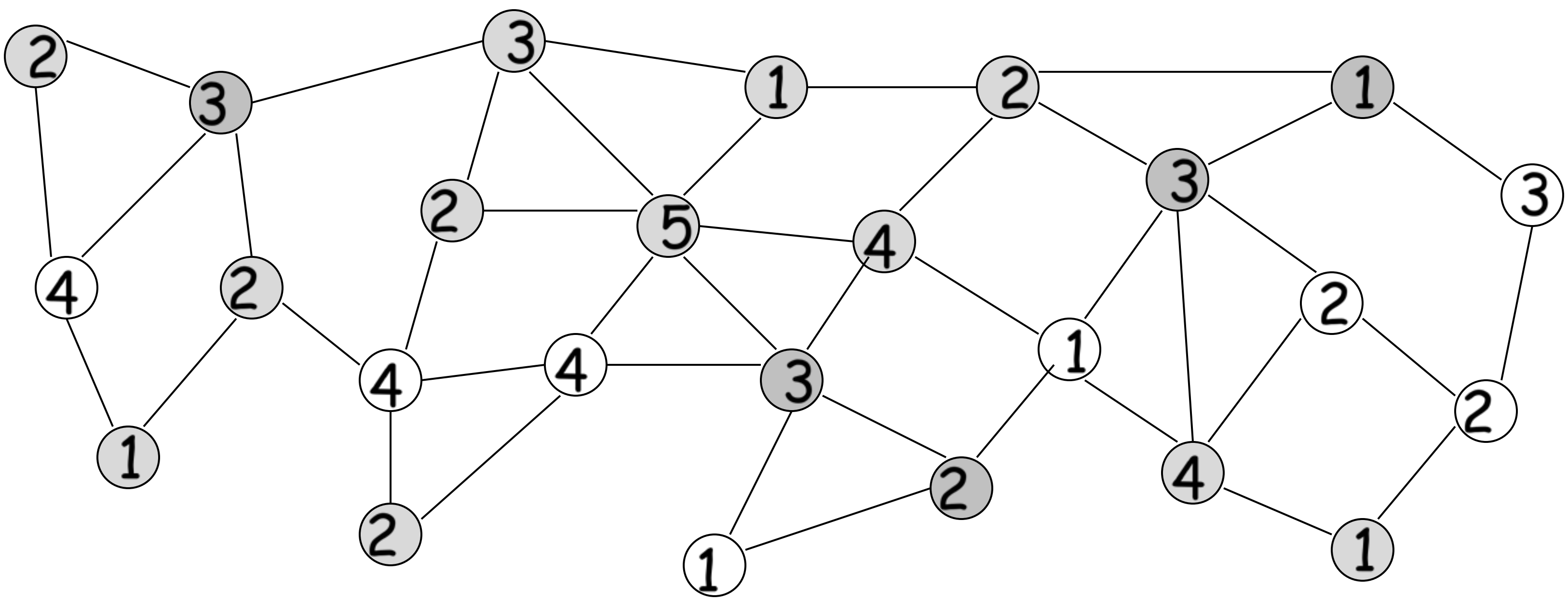


Successful colors



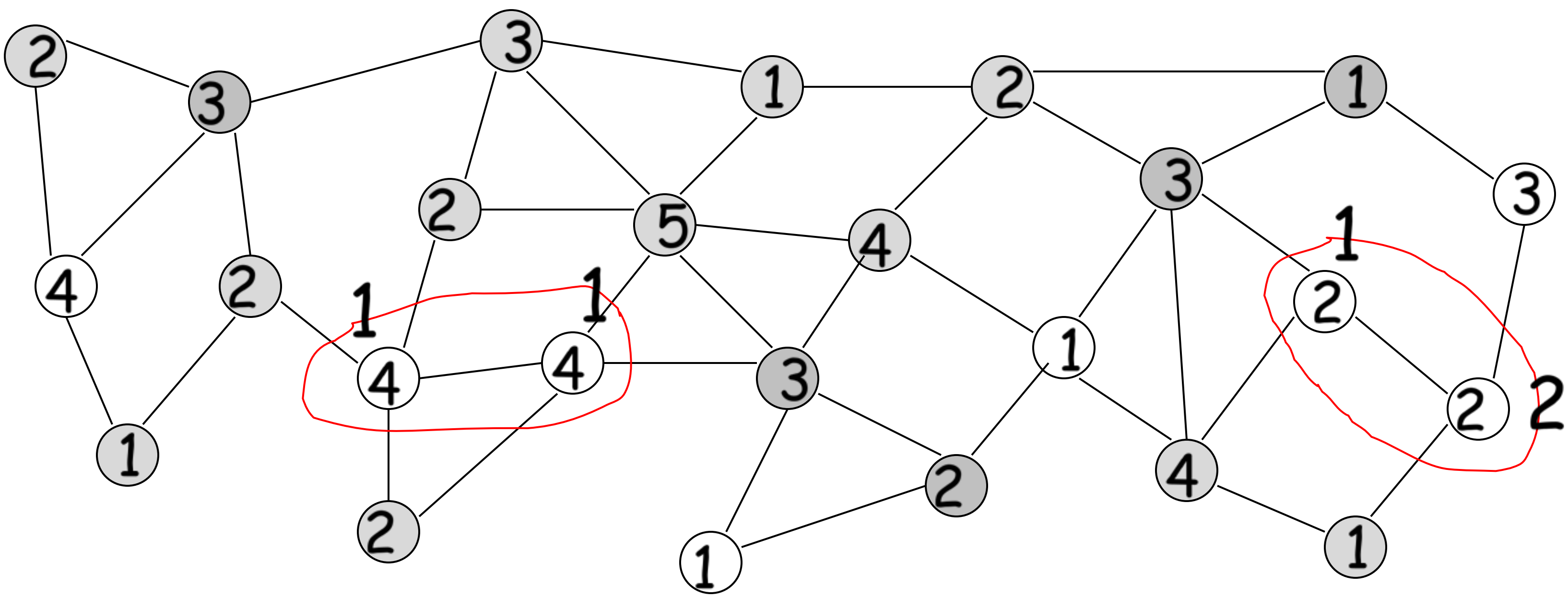
Phase 2: (iteration 2)

Nodes pick random colors

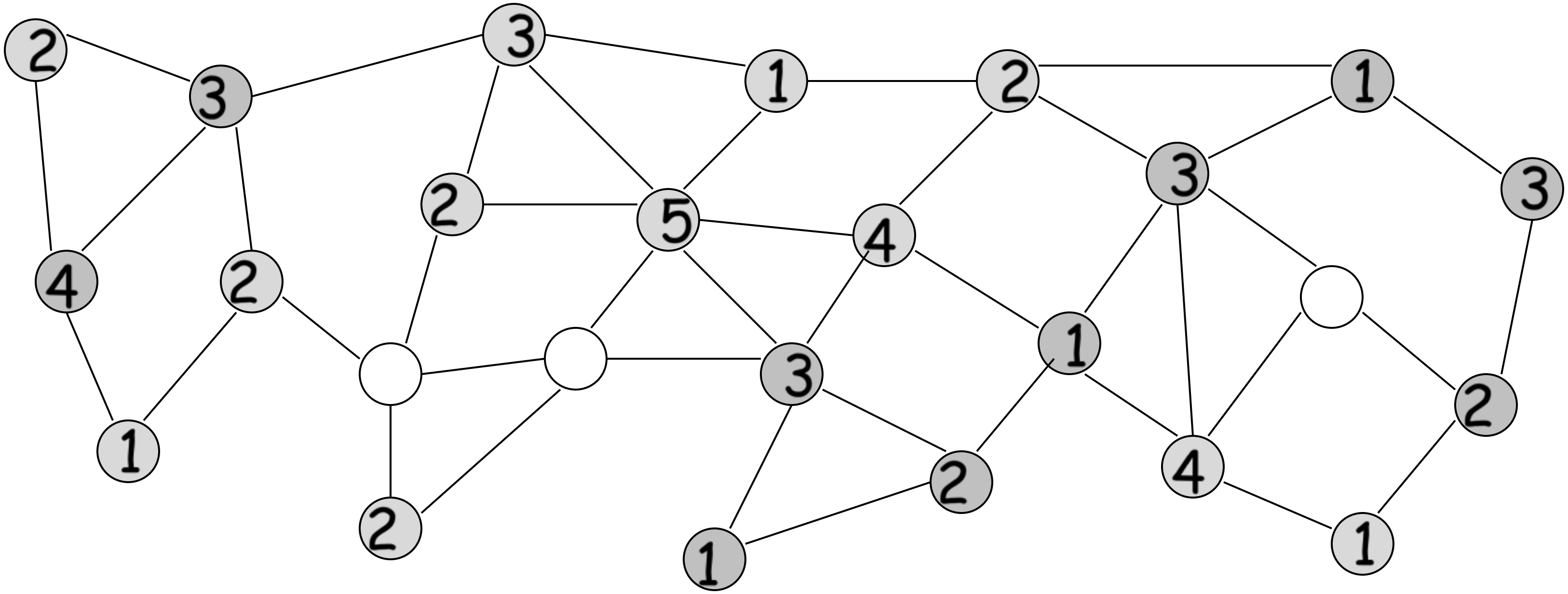


Conflicts

The nodes of higher uncolored degree win

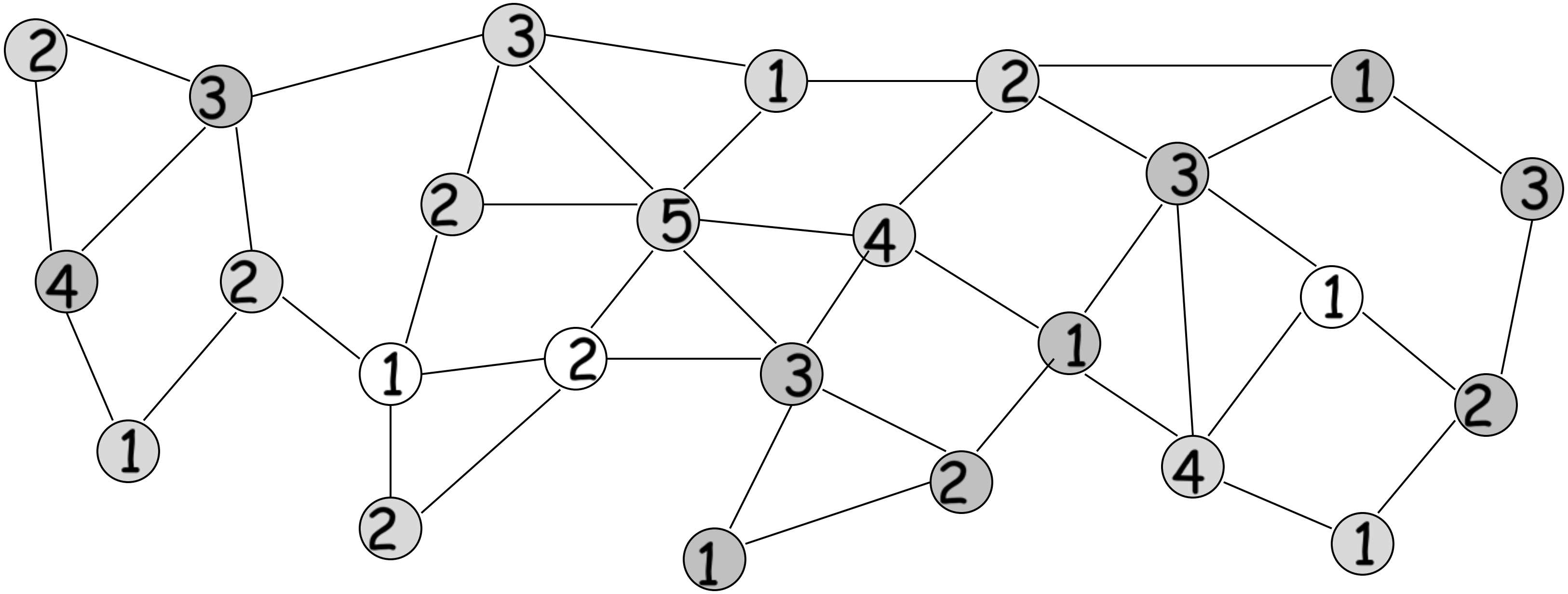


Successful colors



Phase 3: (iteration 3)

Nodes pick random colors



Successful colors

End of execution