Backtracking (Graph coloring)

Dr. Tarunpreet Bhatia

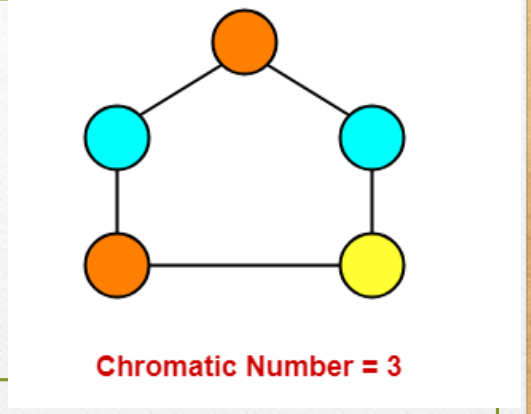
Associate Professor

CSED, TIET

Graph coloring

- Graph Coloring is a process of assigning colors to the vertices of a graph such that no two adjacent vertices of it are assigned the same color.
- Graph Coloring is also called as **Vertex Coloring**.
- Variants: m-coloring decision problem, m-coloring optimization problem, m-coloring all ways.
- Some important applications of graph coloring are Map Coloring, Scheduling the tasks, Preparing Time Table, Frequency Assignment, Bipartite graph etc.

Chromatic Number

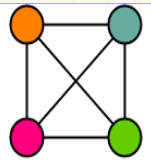


- The smallest number of colors needed to color a graph G is called its chromatic number.
- Chromatic Number of any Complete Graph = Number of vertices in that Complete Graph
- Chromatic Number of any tree = 2
- Chromatic Number of any Planar Graph = Less than or equal to 4
- If number of vertices in cycle graph is even, then its chromatic number = 2.
- If number of vertices in cycle graph is odd, then its chromatic number = 3.
- Chromatic Number of any Bipartite Graph= 2

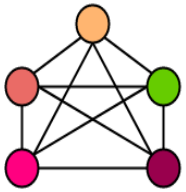
Planar and Cycle Graph

- A Planar Graph is a graph that can be drawn in a plane such that none of its edges cross each other.
- A simple graph of ‘n’ vertices (n>=3) and ‘n’ edges forming a cycle of length ‘n’ is called as a cycle graph.
- In a cycle graph, all the vertices are of degree 2.

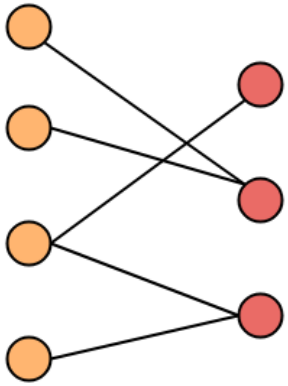
Chromatic Number of Complete Graph Examples



Chromatic Number = 4

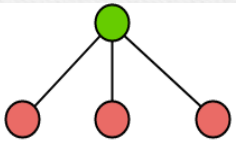


Chromatic Number = 5

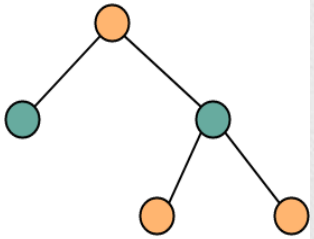


Chromatic Number = 2

Chromatic Number of Tree Examples

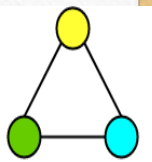


Chromatic Number = 2

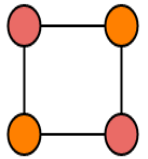


Chromatic Number = 2

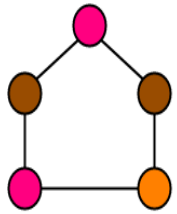
Chromatic Number of Planar Graph Examples



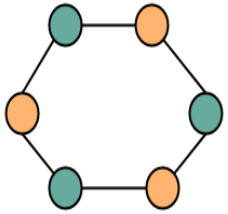
Chromatic Number = 3



Chromatic Number = 2



Chromatic Number = 3



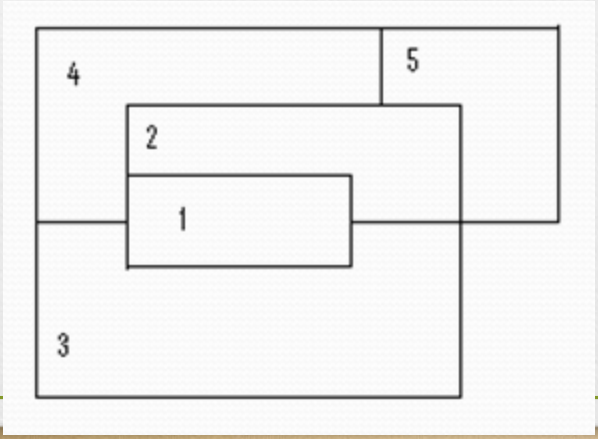
Chromatic Number = 2

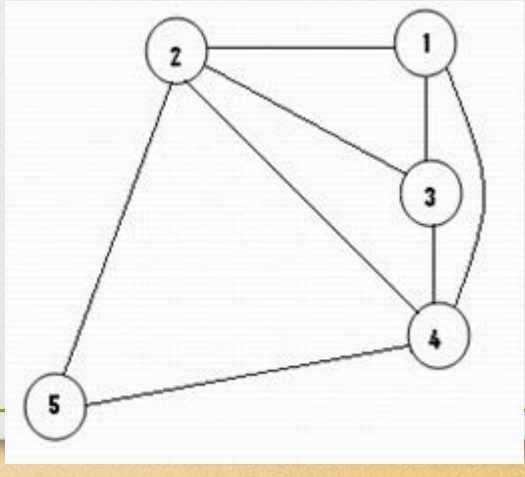
Coloring maps (coloring faces of planar graphs)

Problem:

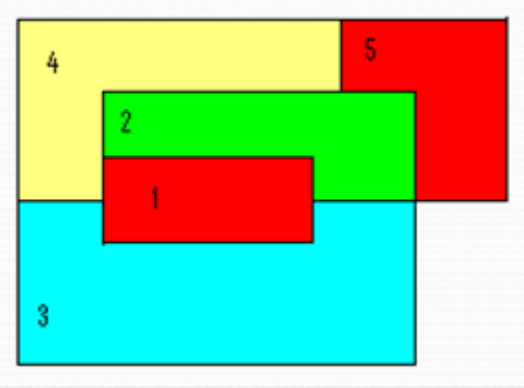
Let **G** be a graph and **m** be a given positive integer. We want to discover whether the nodes of **G** can be colored in such a way that no two adjacent node have the same color yet only **m** colors are used. This technique is broadly used in "map-coloring"; Four-color map is the main objective.

Consider the following map and it can be easily decomposed into the following planner graph beside it :

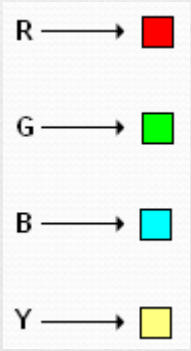




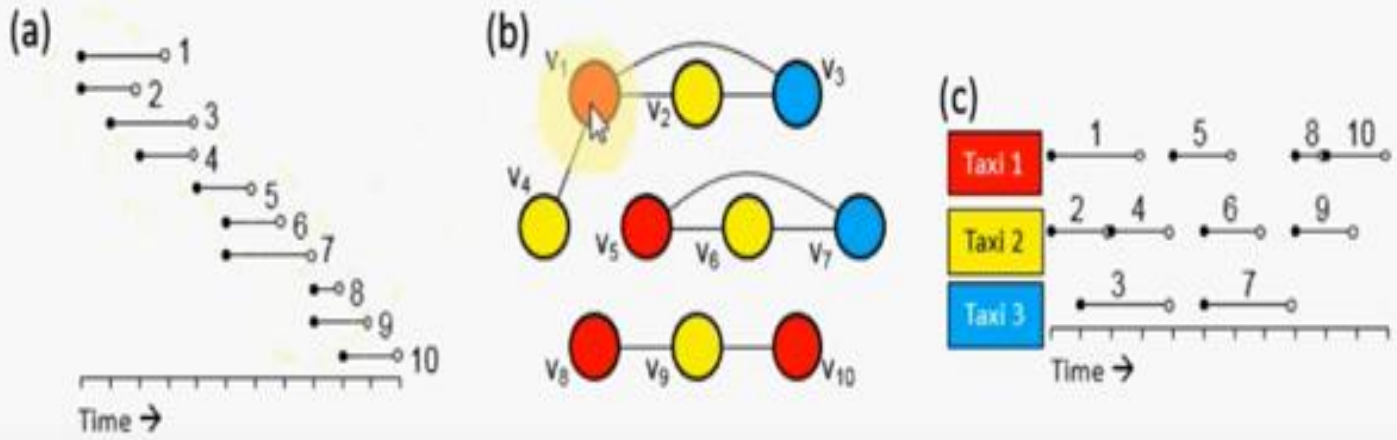
Now the map can be colored as shown here:-



Four colors are chosen as - Red, Green, Blue and Yellow



Scheduling Taxis



Question

You are a network administrator responsible for assigning frequencies to N wireless devices in a wireless network. Each device can only operate on a single frequency channel at a time, and adjacent devices must operate on different channels to avoid interference. Your goal is to represent this network scenario as an undirected graph G and then assign frequencies to the devices such that the minimum number of channels are used. Design an appropriate algorithm to solve this problem and determine its time complexity.

Question

TIET has a summer school session that offers eight courses (A-H). An "X" in **Table 2** shows which courses have students in common. There are time slots available (9:00 to 12:00 p.m.) each day (Monday, Tuesday, Wednesday, Thursday, Friday, Saturday and Sunday) to schedule final exams. TIET wishes to schedule the eight final exams over as few days as possible without creating conflicts for the students scheduled to take them. Having more than one exam scheduled on the same day is fine as long as the courses do not have any students in common. Find the complete schedule of the exams. Show the steps involved in finding the schedule.

	A	B	C	D	E	F	G	H
A		X	X	X				
B	X		X		X	X		
C	X	X		X	X	X	X	
D	X		X			X	X	
E		X	X			X		X
F		X	X	X	X		X	X
G			X	X		X		X
H					X	X	X	

Example (State space tree)

Finding all m-colorings of a graph

```
1   Algorithm mColoring(k)
2   // This algorithm was formed using the recursive backtracking
3   // schema. The graph is represented by its boolean adjacency
4   // matrix G[1 : n, 1 : n]. All assignments of 1, 2, ..., m to the
5   // vertices of the graph such that adjacent vertices are
6   // assigned distinct integers are printed. k is the index
7   // of the next vertex to color.
8   {
9       repeat
10      {// Generate all legal assignments for x[k].
11          NextValue(k); // Assign to x[k] a legal color.
12          if (x[k] = 0) then return; // No new color possible
13          if (k = n) then     // At most m colors have been
14                              // used to color the n vertices.
15              write (x[1 : n]);
16          else mColoring(k + 1);
17      } until (false);
18  }
```

Generating a next color

```
1   Algorithm NextValue(k)
2   // x[1], ..., x[k − 1] have been assigned integer values in
3   // the range [1, m] such that adjacent vertices have distinct
4   // integers. A value for x[k] is determined in the range
5   // [0, m]. x[k] is assigned the next highest numbered color
6   // while maintaining distinctness from the adjacent vertices
7   // of vertex k. If no such color exists, then x[k] is 0.
8   {
9       repeat
10      {
11          x[k] := (x[k] + 1) mod (m + 1); // Next highest color.
12          if (x[k] = 0) then return; // All colors have been used.
13          for j := 1 to n do
14          {   // Check if this color is
15              // distinct from adjacent colors.
16              if ((G[k, j] ≠ 0) and (x[k] = x[j]))
17              // If (k, j) is and edge and if adj.
18              // vertices have the same color.
19                  then break;
20          }
21          if (j = n + 1) then return; // New color found
22      } until (false); // Otherwise try to find another color.
23  }
```

Example (Running algorithm)

Time and Space Complexity

- In the backtracking approach of the graph coloring problem, the time complexity is $O(m^V)$. In the backtracking approach, the average time complexity is less than $O(m^V)$.
- In the backtracking approach to the graph coloring problem, we are not using any extra space but we are using the recursive stack for the recursive function call. So, the space complexity is O(V).
- The backtracking approach and the brute force approach have the same time complexity as in the backtracking algorithm we are just eliminating the bad decisions while coloring the vertices.