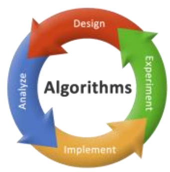
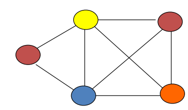


CAP538
Algorithm Design & Analysis

Graph Coloring & Hamiltonian Cycles

Graph Coloring

- *Graph Coloring* is an assignment of colors (or any distinct marks) to the vertices of a graph.
- A graph is said to be **colored** if a color has been assigned to each vertex in such a way that adjacent vertices have different colors.
- Its also known as ***optimization decision problem***

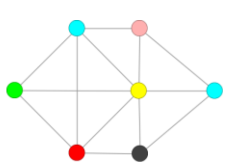


- The **chromatic number** *m* of a graph G is the smallest number of colors with which it can be colored.

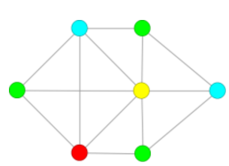
In the above example, the *chromatic number* is 4.

Examples

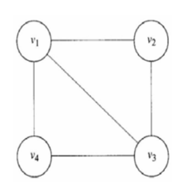
Proper 6-coloring



Optimal 4-coloring



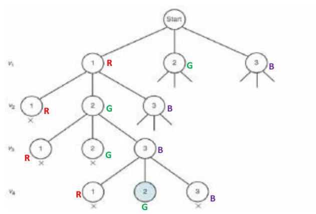
Examples

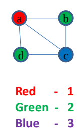


A Graph for which there is no solution to the **2-Coloring Problem**

State Space Tree

- A tree of All possibilities
- Each possible color is tried for vertex v_i at level i such that no two adjacent vertices are of the same color

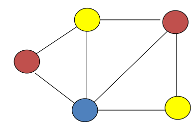




- A portion of the pruned state space tree produced using backtracking to do a **3-coloring** of the graph.
- First solution is found at the shaded node. Each non-promising node is marked with a X.

Coloring Planar Graphs

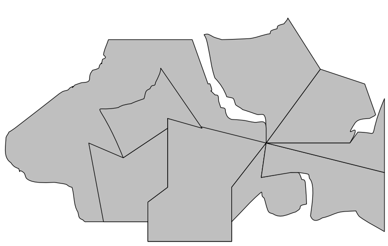
- **Definition**:
  A graph is said to be *planar* iff it can be drawn in a plane in such a way that no two edges cross each other.



- **Four Color Problem:**
  – For every planar graph, the chromatic number is ≤ 4.
  – The mathematicians, *Appel* and *Haken* proved that 4 colors are sufficient to color any graph with the help of computers.

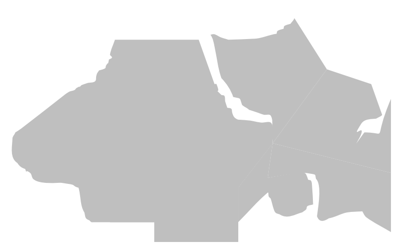
Map Coloring

Consider a Fictional Continent.



Map Coloring

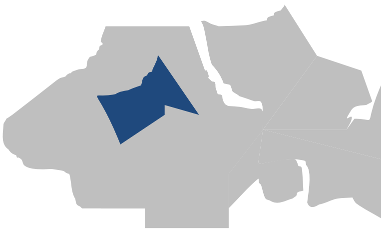
Suppose removed all borders but still wanted to see all the countries. 1 color insufficient ?



Map Coloring

So add another color.

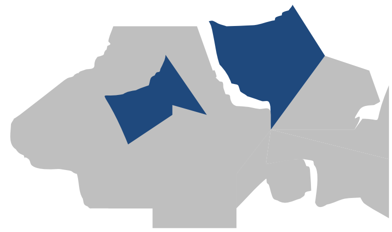
Try to fill in every country with one of the two colors.



Map Coloring

So add another color.

Try to fill in every country with one of the two colors.



Map Coloring

So add another color.

Try to fill in every country with one of the two colors.

Map Coloring

So add another color.

Try to fill in every country with one of the two colors.

Map Coloring

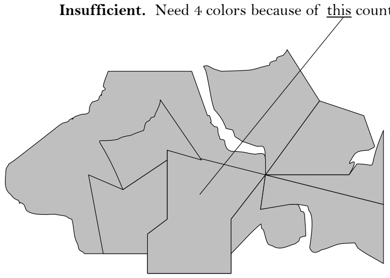
PROBLEM: Two adjacent countries forced to have same color. Border unseen.

Map Coloring

So add another color:

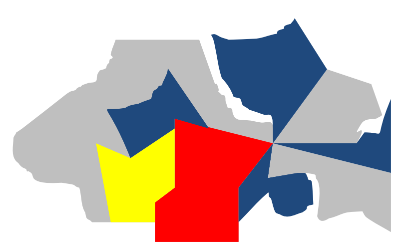
Map Coloring

Insufficient. Need 4 colors because of <u>this</u> country.



Map Coloring

With 4 colors, could do it.



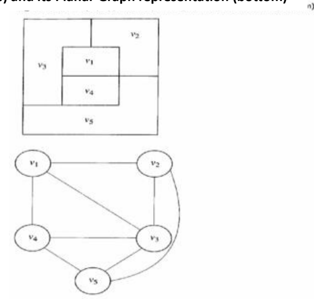
Map Coloring

- Color a map such that two regions with a common border are assigned different colors.

- Each **map** can be represented by a **graph**:
  - Each region of the map is represented by a vertex
  - Edges connect two vertices if the regions represented by these vertices are adjacent.

Map Coloring to Graph Coloring

Map (top) and its Planar Graph representation (bottom)

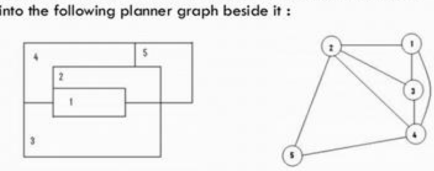


Coloring a map

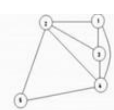
Problem:

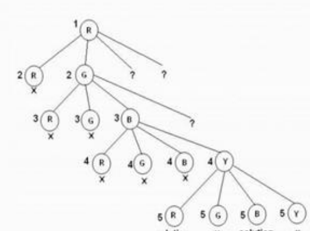
Let **G** be a graph and **m** be a given positive integer. We want to discover whether the nodes of **G** can be colored in such a way that no two adjacent node have the same color yet only **m** colors are used. This technique is broadly used in "map-coloring"; Four-color map is the main objective.

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



This map-coloring problem of the given map can be solved from the planner graph, using the mechanism of backtracking. The state-space tree for this above map is shown below:

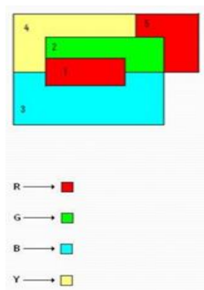




Map Coloring

Now the map can be colored as shown here:

Four Colors are chosen as:- Red, Green, Blue and Yellow



Graph Coloring Algorithm

```
MainAlgo
{       Create Adjacency Matrix G[1:n, 1:n] of a given graph G
        Set x[1:n] = 0
        mColoring(1)
        Return
}

mColoring(k)
{
        Repeat
        {       NextValue(k)
                if(x[k] = 0) then               // No New color possible
                        return
                if(k = n) then                  // All vertices colored
                        write 'x[1:n]'
                else
                        mColoring(k+1)
        } Until(false)
}
```

Graph Coloring Algorithm

```
NextValue(k)                        // Generating a next color
{
        Repeat
        {
                x[k] = (x[k] + 1) mod (m+1)          // next higher color
                if(x[k] = 0) then                    // All colors used
                        return
                for j = 1 to n do
                {
                        if(G[k, j] != 0 and x[k] = x[j])
                                Break
                }
                if(j = n+1) then                     // new color found
                        Return
        } Until(false)
}
```

Hamiltonian Cycles

- Let G = {V, E} be a connected graph with n vertices.
- **Hamiltonian Path:**
  - A path which visits every vertex in a graph *G* exactly once.
- **Hamiltonian Circuit/Cycle**:
  - Also known as a *Round-Trip Path.*
  - It was proposed by **William Hamilton**
  - It is a cycle which visits every vertex exactly once, *except for the first vertex*, which is also visited at the end of the cycle.

Hamiltonian Cycles

- In other words, more formally,
  - If a ***Hamiltonian Cycle*** begins at some vertex $v_1 \in G$ and the vertices of G are visited in the order $v_1, v_2, \ldots, v_{n+1}$, then the edges (v_i, v_{i+1}) are in E, $1 \le i \le n$ and v_i are distinct except for v_1 and v_{n+1} which are equal.

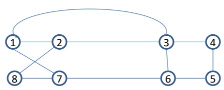


Fig-1: G1 contains Hamiltonian Cycle: 1, 2, 8, 7, 6, 5, 4, 3, 1

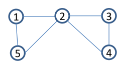


Fig-2: G2 contains no Hamiltonian Cycle

Hamiltonian Circuit Problem

Problem: This problem is concern about finding a Hamiltonian circuit in a given graph.

Hamiltonian circuit: Hamiltonian circuit is defined as a cycle that passes to all the vertices of the graph exactly once except the starting and ending vertices that is the same vertex.

For example consider the given graph and evaluate the mechanism:-

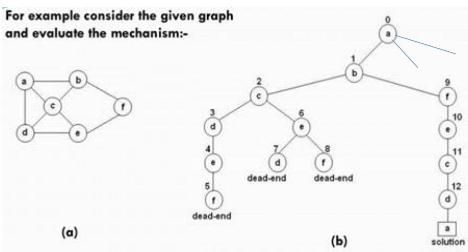


Figure: • (a) Graph.
• (b) State-space tree for finding a Hamiltonian circuit. The numbers above the nodes of the tree indicate the order in which nodes are generated.

Hamiltonian Algorithm

```
MainAlgo
{
        Create Adjacency Matrix G[1:n, 1:n] of a given graph G
        Set x[1] = 1
        Set x[2:n] = 0
        Hamiltonian(2)
}

Hamiltonian(k)          // Finds the Hamiltonian cycles of graph G
{
        Repeat
        {       NextValue(k)
                if(x[k] = 0) then
                        Return
                if(k = n) then
                        write 'x[1:n]'
                else
                        Hamiltonian(k+1)
        } Until(false)
}
```

Hamiltonian Algorithm

```
NextValue(K)            // Generating a next vertex
{
        Repeat
        {
                x[k] = (x[k] + 1) mod (n+1)             // next vertex
                if(x[k] = 0) then
                        Return
                if(G[x[k-1], x[[k]] != 0) then
                {
                        for j=1 to k-1 do
                        {       if(x[j] = x[k]) then    // if current vertex not same
                                        Break
                        }
                        if(j = k) then                  // vertex is distinct
                        {       if(k < n or (k = n and G[x[n], x[1]] != 0)) then
                                        Return
                        }
                }
        } Until(false)
}
```