Graph Coloring (Greedy Approach)

What is Graph Coloring?

- Graph Coloring is an assignment of colors to the vertices of a graph.
- A coloring is a **proper coloring** if no two adjacent vertices have the same color.

Why Graph Coloring?

- Many problems can be formulated as a graph coloring problem including Time Tabling, etc.

Terminology

- K-Coloring
  - A k-coloring of a graph G is a mapping of V(G) onto the integers 1..k such that adjacent vertices map into different integers.
  - A k-coloring partitions V(G) into k disjoint subsets such that vertices from different subsets have different colors.

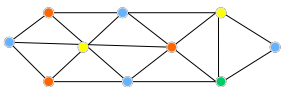
Terminology

- K-colorable
  - A graph G is k-colorable if it has a k-coloring.
- Chromatic Number
  - The smallest integer k for which G is k-colorable is called the chromatic number of G.
- A coloring of a graph G assigns colors to the vertices of G so that adjacent vertices are given different colors

Example

- Problem: A state legislature has a number of committees that meet each week for one hour. How can we schedule the committee meetings times such that the least amount of time is used but two committees with overlapping membership do not meet at the same time.

Example (cont)



An vertex represents a meeting

An edge represents a conflict between to meetings

The chromatic number of this graph is four. Thus four hours suffice to schedule committee meetings without conflict.

Graph Colouring Algorithm

- There is no efficient algorithm available for coloring a graph with minimum number of colors.
- Graph coloring problem is a known NP Complete problem.

NP Complete Problem

- NP complete problems are problems whose status is unknown.
- No polynomial time algorithm has yet been discovered for any NP complete problem
- It is not established that no polynomial-time algorithm exist for any of them.

Basic Greedy Algorithm

1. Color a vertex with color 1.
2. Pick an uncolored vertex v. Color it with the lowest-numbered color that has not been used on any previously-colored vertices adjacent to v. (If all previously-used colors appear on vertices adjacent to v, this means that we must introduce a new color and number it.)
3. Repeat the previous step until all vertices are colored.

Greedy Algorithm

The above algorithm doesn't always use minimum number of colors.

Also, the number of colors used sometime **depend on the order in which vertices are processed**

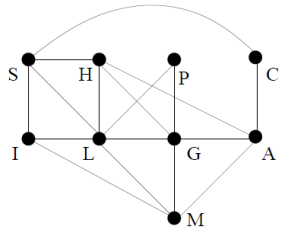
Example

- Suppose that you are responsible for scheduling times for lectures in a university. You want to make sure that any two lectures with a common student occur at different times to avoid a conflict. We could put the various lectures on a chart and mark with an "X" any pair that has students in common

Lecture	A	C	G	H	I	L	M	P	S
Astronomy		X	X	X			X		
Chemistry	X								X
Greek	X			X		X	X	X	
History	X		X			X			X
Italian						X	X		X
Latin			X	X	X		X	X	X
Music	X		X		X	X			
Philosophy			X			X			
Spanish		X		X	X	X			

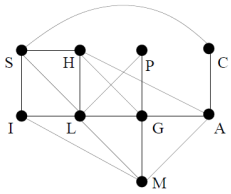
Graph representation

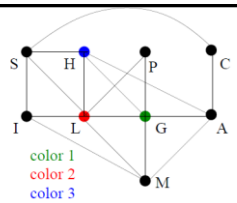
- One vertex for each lecture and in which two vertices are joined if there is a conflict between them

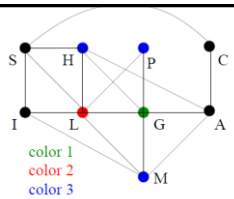


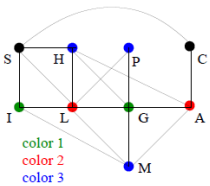
Example

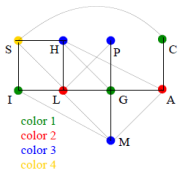
- Suppose we decide to color the course conflict graph using the Greedy Coloring Algorithm, and we decide to color the vertices in order G, L, H, P, M, A, I, S, C.
- Then we would color G with color 1 (green), L with color 2 (red) since adjacency with G prevents it from receiving color 1 (green), and we color H with color 3 (blue) since adjacency with G and L prevents it from receiving colors 1 and 2 (green and red).



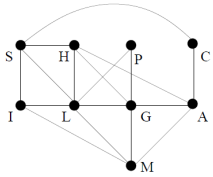


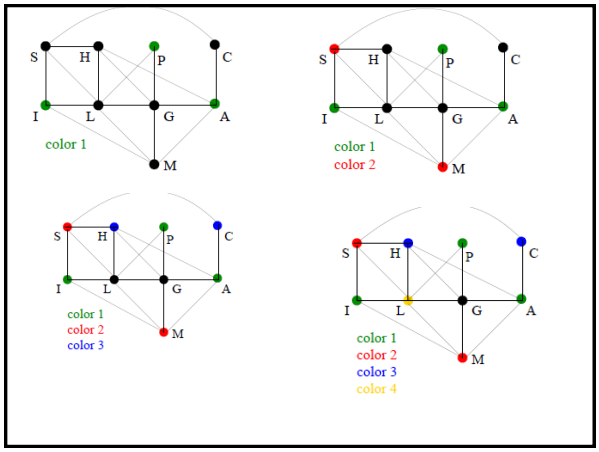


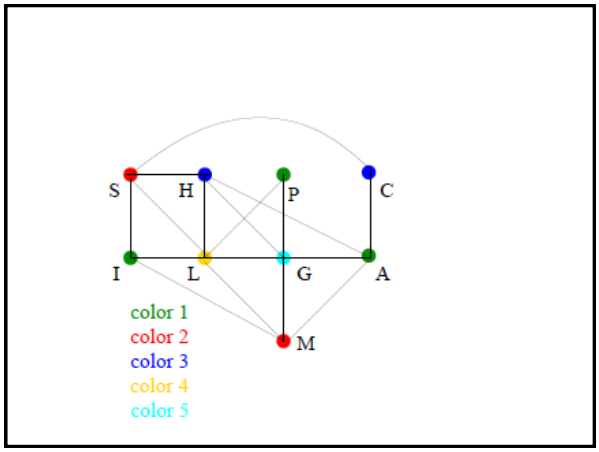




- Suppose we chose to color the vertices in the order A, I, P, M, S, C, H, L, G.
- First we color A with color 1 (green) and also I and P color 1




S
H
P
C
I
L
G
A
M
color 1
color 1
color 2
color 1
color 2
color 3
color 1
color 2
color 3
color 4


S
H
P
C
I
L
G
A
M
color 1
color 2
color 3
color 4
color 5