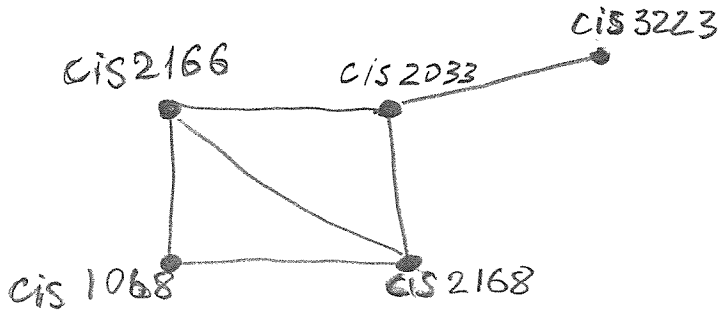
Graph Coloring

Exam Scheduling @ Temple



- Nodes: Classes offered
- Edge between classes means there's a student in both classes. Therefore Exam cannot be scheduled for the two classes at the same time.

Time slots Available for Exams:

- 5 - 7 pm
- 7 - 9 pm
- 9 - 11 pm
- 11pm - 1 am
- 1 - 3 am

Your job is to figure out how not to use later slots.

→ I.e. Assign classes (nodes in graph) to slots so that no adjacent classes fall into the same time slot, using as few time slots as you can.

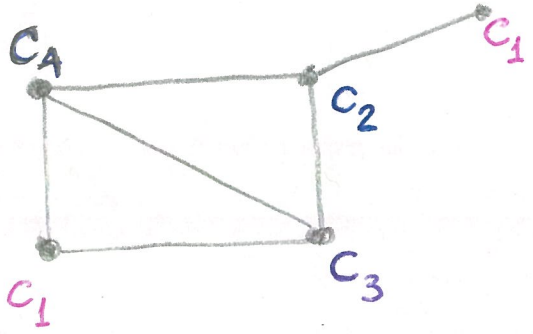
→ This is called a graph coloring problem: Given graph G and K colors, assign color to each node so that adjacent nodes get different colors.

The minimum number of colors needed for this is called $\chi(G)$, the chromatic number of graph G.

Thus "color" in this problem corresponds the time slot. We could assign 5 colors to the 5 nodes, but then somebody is taking exam from 1–3 am. Can we avoid this by using fewer colors?

Let colors be c_1 c_2 c_3 c_4 c_5

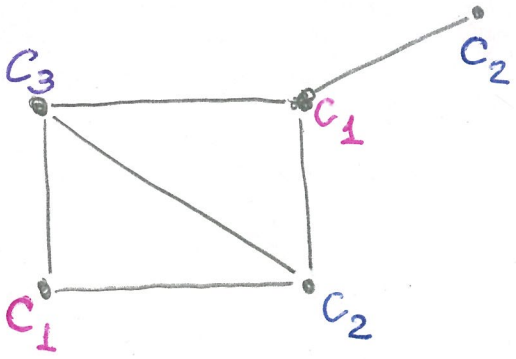
We can get away with only 4 colors like so:



Time Slots:

- 5–7 c_1
- 7–9 c_2
- 9–11 c_3
- 11–1 c_4
- 1–3

But we can do even better: we can use 3 colors only



- 5–7 c_1
- 7–9 c_2
- 9–11 c_3
- 11–1
- 1–3

Can we do even better? No! Chromatic number of this graph is 3.

$$\chi(G) = 3$$

(Note: the triangle (K_3 subgraph) needs 3 colors!)

O.K. This was a small problem that we could easily solve by trial and error. to determine the chromatic number of this graph

- In general, this is a very difficult problem: Nobody today knows how to solve this problem in time that is better than exponential in the number of vertices of the graph.

- However, it is easy to check if a given assignment of colors is correct: just look at every edge and check if the nodes it connects are of different colors.

- Even determining, for an arbitrary graph G, if 3 colors are enough (3-coloring problem) is difficult.

- This is an NP-complete problem

(NP complete problems: a class of problems whose solutions are easy to check, but it takes exponential time to solve them.

If you figure out how to solve just one of these problems in faster time, then all of these problems can be solved in this faster time.

- You get $1,000,000 from Clay Institute if you do this!

If you prove that no better-than-exponential-time solution exists for just any one of these problems, You still get $1,000,000 and eternal(?) fame.)

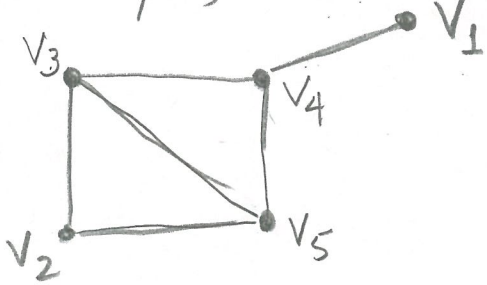
But there are simple algorithms that use "small" number of colors. Here's One:

1. Order the nodes $v_1, v_2, v_3, \ldots v_n$
2. Order the colors $c_1, c_2, c_3 \ldots \ldots$
3. For $i = 1, 2, \ldots, n$

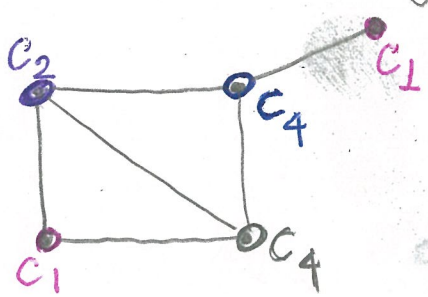
   Assign the lowest "legal" color to v_i

(By "legal" we mean that you do not color a node by a color of any node that it is adjacent to)

- Try this on the exam graph shown earlier. Chances are you unwittingly used this algorithm to find the coloring earlier.
- We note that different orderings of vertices produce different colorings
  - For example, this ordering of vertices:

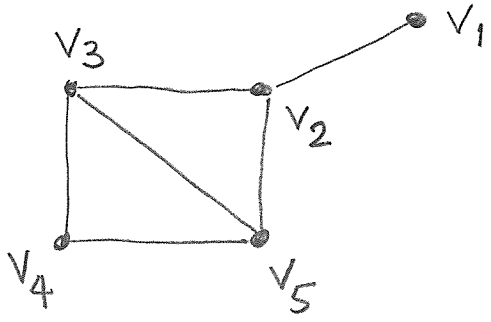


produces the coloring:

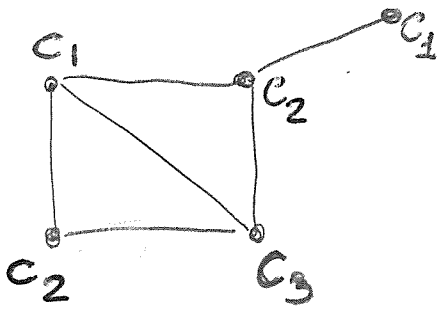


Therefore, the above ordering of vertices produced 4 colors.
But we know 3 colors are possible

- Let's try a different ordering of vertices:



This ordering produces the following coloring:



This is much better: with this particular vertex ordering, we just need 3 colors, not 4.

- Thus we can see that a particular vertex ordering determines the number of colors needed.
- So the trick is to use a clever ordering of vertices

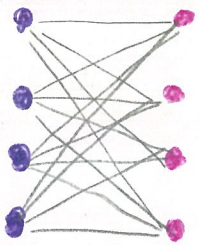
There are various clever ways of ordering vertices that depend on particular applications. E.g., largest degree vertices being colored first, or some other exotic orderings that may perform well for a given problem.

Theorem If every node in G has degree $\leq d$, then the above algorithm uses at most $d+1$ colors for G.

Proof This is not difficult to prove if you try induction of n - - the number of vertices in G. (Note: you may try induction on d, but it's very difficult to prove it this way)

Example 1

This can be a very loose upper bound on number of colors. For example, A bipartite graph has a chromatic number 2:



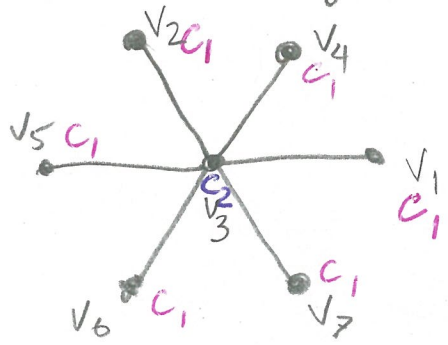
$\frac{n}{2}$ vertices + $\frac{n}{2}$ vertices = total of n vertices. Degree of each vertex is:

$\deg(v) = \frac{n}{2}$; However, for large n, $2 = \chi(G) < \frac{n}{2} + 1$

(< : much less.)

Example 2

On the following example, the algorithm does much better:

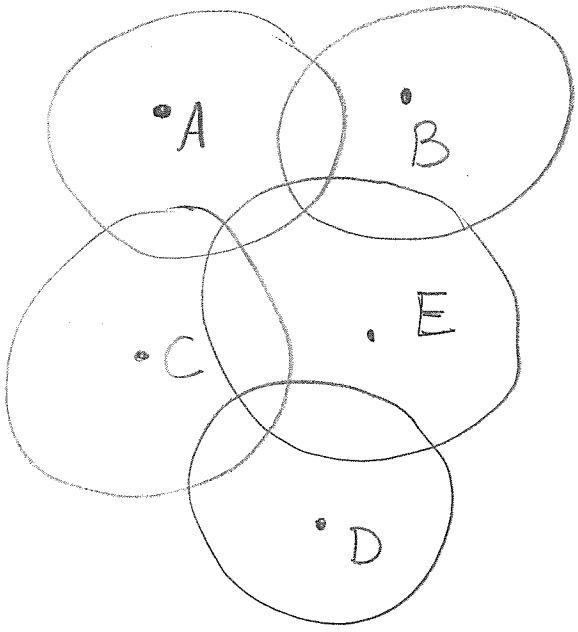


- Here, no matter how you order the vertices, you still get 2 colors.
- Thus the algorithm here does much better than the bound we proved in the theorem.

Another Application of Graph-Coloring

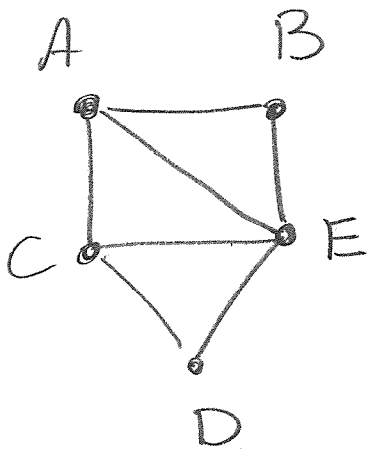
Need to assign frequencies to radio-communication towers.

If two radio stations serve overlapping geographical areas, they can not be assigned the same frequency. E.g., towers A, B, C, D, E serve areas represented by circles:



Question: What's the smallest number of radio frequencies you need to enable all towers to operate?

Answer: Let's build the corresponding graph G. Towers are nodes. There is an edge between towers if their service areas overlap. The chromatic number of this graph, $\chi(G)$, the minimum number of frequencies you need to operate all towers.



Here $\chi(G) = 3$.

Can you 3-color this graph?