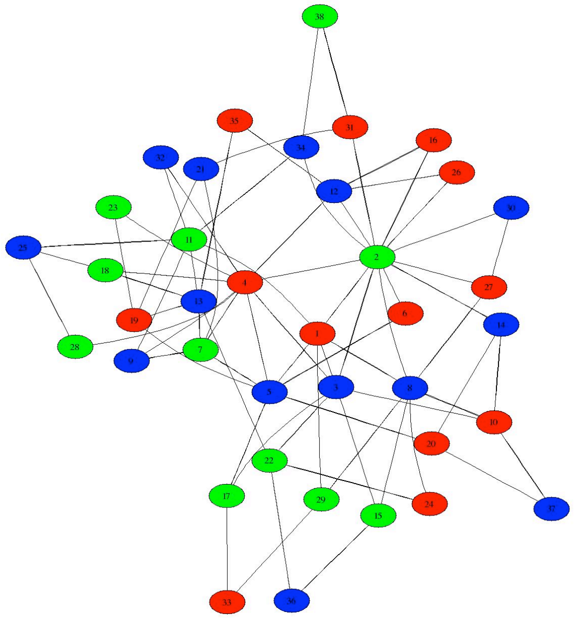
Graph Coloring: Background and Assignment

Networked Life
NETS 112
Fall 2014
Prof. Michael Kearns

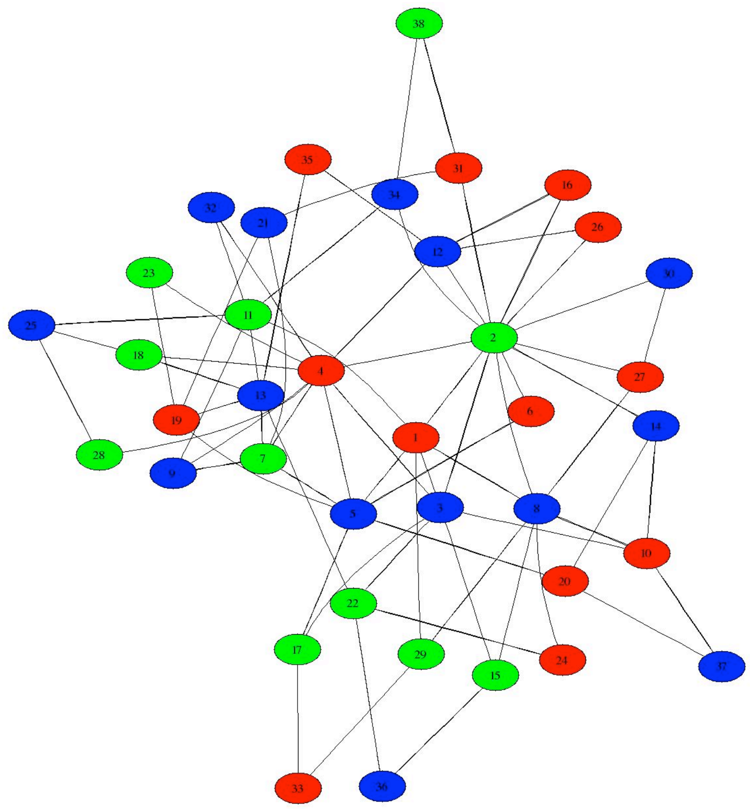
A Little Experiment

- Consider the network in which you are connected to the people sitting to your left, right, in front of and behind you
  - if you have no neighbors, change your seat
  - if no one is sitting to your (say) left, you're just "missing" that neighbor
- Hold up 1, 2, or 3 fingers
- Your "goal" is to pick a *different* number than all your neighbors
  - can change your number whenever you want, as many times as you want
- While your number matches at least one neighbor, raise your other hand



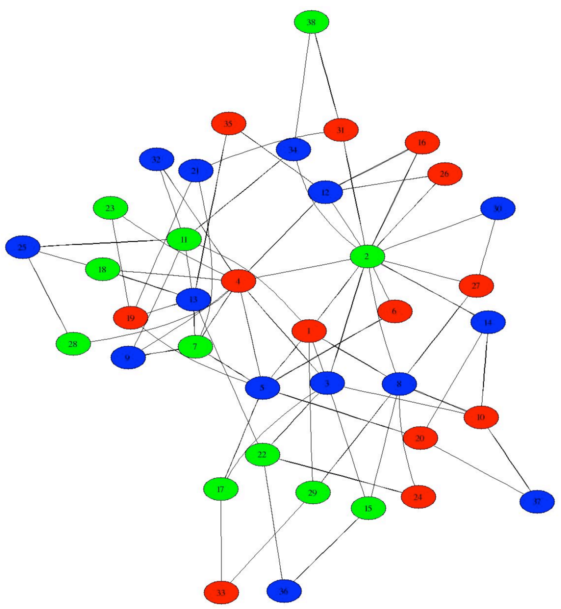
Graph Coloring

- We are given a graph or network G with N vertices (e.g. N ~ 100)
- We are given K values or “colors” (e.g. K = 3)
- We would like to find a labeling of vertices by colors such that for every edge (u,v) in G, u and v have *different colors*
- In general, for any given G, this problem is harder the smaller K is
- *Chromatic number* K(G) = smallest K for which there is a solution



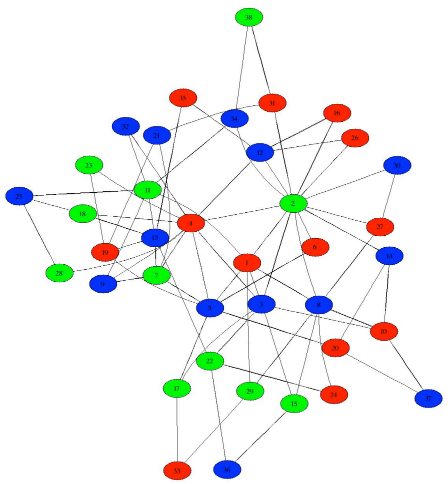
How Does Structure of G Influence K(G) (and Good Algorithms?)

- How many colors are always enough?
- If some vertex has degree D, must K(G) >= D?
- What's a sufficient condition to force K(G) >= 3? More generally?
- Is it a necessary condition?
- How did I know 2 colors was enough for the grid?
- What's a good algorithm if you know K(G) = 2?
- Is there any "local" property/algorithm determining K(G)/solutions?



A Famous, Important and Notorious Problem

- Optimization (minimizing scarce resource): e.g. exam scheduling
  - one vertex for each Penn class: NWLife, CIS 120, Intro to Chocolate,…
  - draw an edge between two classes if there is a student taking both
  - colors = final exam time slots
  - solution ensures no student has simultaneous exams!
  - would like to minimize the length of exam period…
- Cartography: 4-color theorem
- Graph Coloring as a model of social differentiation
- Graph Coloring is a computationally hard problem:
  - algorithm given graph (list of vertices and edges) as input
  - output a proper coloring (solution) with smallest number of colors (K(G))
  - best known algorithms not much better than exhaustive search
  - running time scale exponentially in N, e.g ~ 10^N
  - For N=100: 10^100 >> number of protons in the universe
  - centralized or "birds-eye" computation, vs distributed & local
- Even significant relaxations remain intractable
  - e.g. allow (much) more than K(G) colors



Your Assignment

- Later in the course, we will study experiments on human subjects solving graph coloring from local, distributed information
- For now, you are asked to try some coloring problems on your own
- Will employ a web app we have designed and developed
- Your score will give points for simply finding solutions, with bonus points for finding them quickly
- No collaboration/collusion of any kind --- not even discussion
- Assignment is due (app will close) in one week (midnight Tue 9/16)