Section 8.8 Graph Coloring

Coloring of a graph G: an assignment of colors to the vertices so that no two adjacent vertices are assigned the same color.

So a coloring partitions the vertex set V into sets $V_1, V_2, \ldots, V_k$ so that no two vertices in the same set are joined by an edge. Think of V_1 as the blue vertices. V_2 as the red vertices, etc.

The **chromatic number** $\chi(G)$ of G is the smallest number of colors possible (smallest k above).

Examples:

1. It is easy to check that $\chi(C_n)$ is 2 if n is even and 3 if n is odd..

2. $\chi(K_n) = n$ since no two vertices can get the same color.

3. A bipartite graph G with at least one edge satisfies $\chi(G) = 2$. In fact, this was basically our definition of bipartite; think of Left as Red and Right as Blue.

4. $\chi(Q_n) = 2$ since Q_n is bipartite (even number of 1's corresponds to Red and odd number of 1's corresponds to Blue).

Application/Motivation: **Scheduling** classes, exams, events, ...

Form a graph G whose vertices are the events and put an edge between two events if they conflict (cannot be scheduled in the same time slot). Then $\chi(G)$ is the smallest number of time slots needed to schedule all events.

Remark: Computation of $\chi(G)$ is a NP-Hard problem; best known algorithms have exponential complexity.

Clearly, if a graph has n vertices, then $\chi(G) \leq n$. We can do better.

Theorem: If Δ is the max degree of a vertex of G, then $\chi(G) \leq \Delta + 1$.

Proof. Consider 'colors' $1, 2, \ldots, \Delta, \Delta + 1$. List the vertices $v_1, v_2, \ldots, v_n$. The following recursive/iterative algorithm colors G using at most $\Delta + 1$ colors:

1. Color v_1 with color 1.

2. For $i = 2, \ldots, n$, color v_i with the smallest color **not** assigned already to the vertices among $\{v_1, \ldots, v_{i-1}\}$ joined to v_i (i.e. colors among the vertices adjacent to v_i that have already been colored.). Since the max degree is Δ and we have $\Delta + 1$ colors, there will always be an available color.

Remark: If G is connected, the only graphs for which $\chi(G) = \Delta + 1$ are complete graphs and odd length cycles.

9.1 Trees

Trees are important dats structures in CS.

A **tree** is a connected graph that is barely connected in the sense that each edge is a cut-edge (bridge).: removing any edge of a tree leaves a non-connected graph. So a tree cannot have any cycles, since removing an edge of a cycle in a connected graph cannot disconnect the graph.

Up to isomorphism, there are 1, 1, 1, 2, 3, trees on 1, 2, 3, 4, 5, vertices, respectively. Drwa them!

Characterising properties of tree with $n > 1$ vertices.

1. connected and every edge a bridge (our definition).
2. connected with no cycles.
3. connected with a unique simple path joining each pair of vertices.
4. connected graph with $n - 1$ edges.

(I planned to but didn't get to the following but will resume this on Friday.)

Remarks:

1. If the degrees of the vertices of a tree are $d_1, d_2, \ldots, d_n$, then

$$d_1 + d_2 + \cdots + d_n = 2(n-1).$$

This implies that a tree with $n > 1$ vertices has at least two vertices of degree equal to 1. Such vertices are called **pendent** and the unique edge incident with a pendent vertex is called a **pendent edge**.

2. Recursively, removing a pendent vertex, pendent edge pair gets one down to a single vertex. (implying again that a tree with n vertices has $n-1$ edges).