Graph Coloring and Algorithms

CSCI 170 Spring 2021

Sandra Batista

1.1-1.2

Graph Coloring and Algorithms

- **Vertex Coloring**
- Bipartite Graphs and Matchings
- Graph Algorithms: BFS and DFS

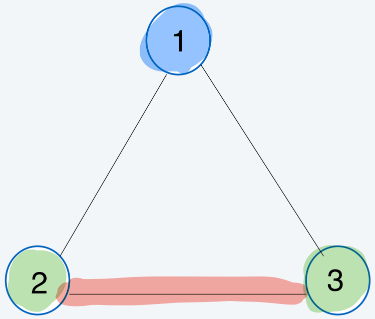
1.1–1.2

Vertex Coloring

A **valid vertex coloring** of a graph is an assignment of a color to each vertex of a graph such that no edge is **monochromatic**, i.e. no edge has endpoints of the same color.

A **k-coloring** is a valid vertex coloring using k colors

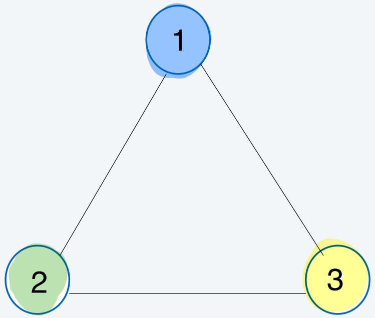
Invalid Vertex Coloring



NOT valid

because edge in red is monochromatic

Valid Vertex Coloring



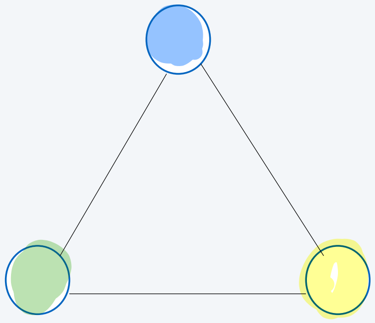
Chromatic Number of a Graph

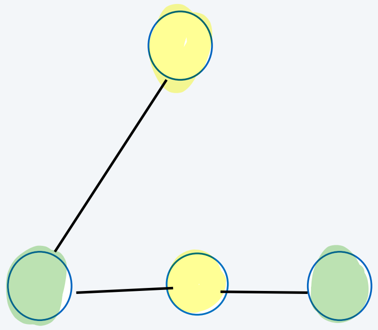
The minimum number of colors necessary for a valid vertex coloring of a graph is colored its **chromatic number.**

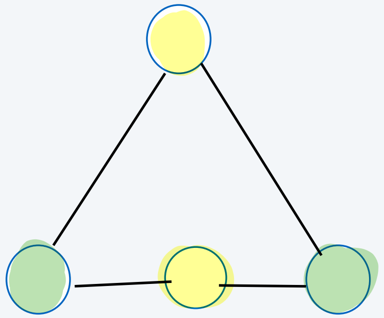
To show that a specific value, v, is a chromatic number of a graph, G:

1)Show the vertices of G can be colored using only v colors

2)Show the vertices of G cannot be colored in fewer colors than v.





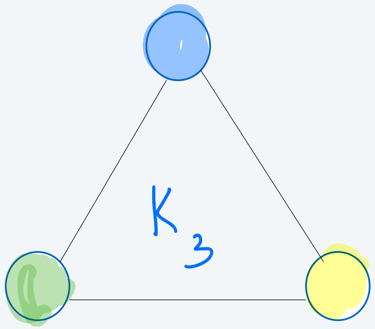


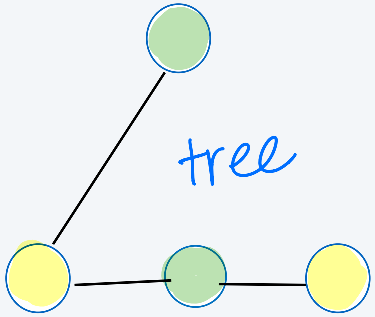
Chromatic Number of a Graph

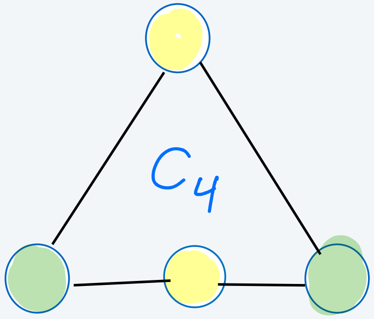
The chromatic number of a complete graph on n vertices is n.

The chromatic number of an odd cycle is 3 and the chromatic number of an even cycle is 2.

The chromatic number of a non-trivial tree is 2.







Application: Chromatic Number of a Graph

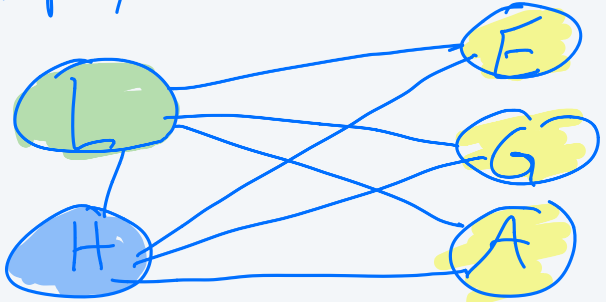
A zoo would like to use the fewest enclosures possible. The main concern is that the zoo cannot house predators in the same enclosures as their prey. The zoo will have lions, elephants, giraffes, hyenas, and antelope.

Elephants, giraffes and antelopes are vegetarians, but they are prey for lions and hyenas. Lions and hyenas will eat any of the other species. What are the fewest number of enclosures needed?

let $G = (V, E)$ V = species = $\{L, E, G, H, A\}$

$E = \{\langle u, v\rangle$: predator prey relationship between u and $v\}$

$\chi(G) = 3$



zoo needs 3 enclosures

Bipartite Graphs

A Graph, G=(V,E), is called **bipartite** if its vertices may be partitioned into two sets, L and R, such that

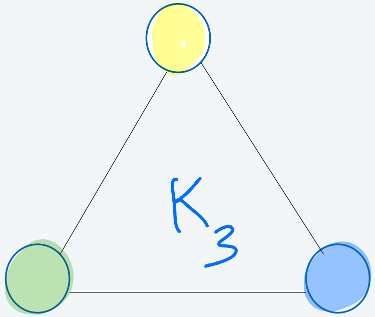
i) $L \bigcup R = V$

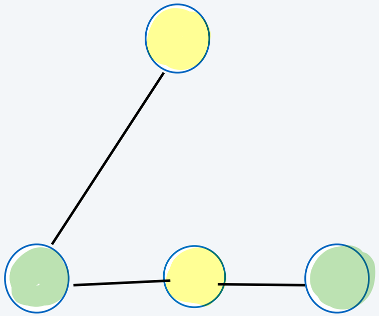
ii) $L \cap R = \varnothing$

iii) Every edge has one endpoint in L and the other in R.

A graph is bipartite if and only if it is two-colorable.

Every tree is bipartite.





Graph Coloring and Algorithms

- Vertex Coloring
- **Bipartite Graphs and Matchings**
- Graph Algorithms: BFS and DFS

1.1–1.2

Bipartite Graphs

A Graph, G=(V,E), is called **bipartite** if its vertices may be partitioned into two sets, L and R, such that

i) $L \bigcup R = V$

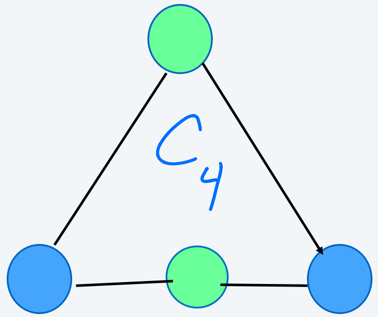
ii) $L \cap R = \varnothing$

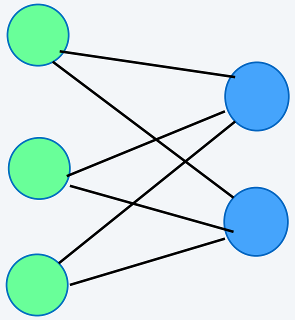
iii) Every edge has one endpoint in L and the other in R.

A graph is bipartite if and only if it is two-colorable.

A graph is bipartite if and if it contains no odd length cycles.

A bipartite graph is complete if it has every possible edge between the partitions.





Complete Bipartite Graph $K_{3,2}$

Exercise: Bipartite Graph Exercise

A graph is bipartite if and only if it is two-colorable.

Proof:

If a graph is bipartite, then it is two-colorable.

$G = (V, E)$ is bipartite $L \cup R = V$, $L \cap R = \emptyset$

let's color all vertices in L blue and all vertices in R green

$\Rightarrow \forall \; e = (u, v) \in E$ $u \in L$ and $v \in R$ (WLOG)

$\Rightarrow$ each edge is not monochromatic since u is blue and v is green

$\Rightarrow$ G is 2-colorable.

Exercise: Bipartite Graph Exercise

A graph is bipartite if and only if it is two-colorable.

Proof (continued):

If a graph is two-colorable, then it is bipartite.

$G = (V, E)$ has valid vertex coloring using only blue & green.

$\Rightarrow$ Let L be blue vertices and R the green

$\Rightarrow L \cup R = V$ and $L \cap R = \emptyset$ (every vertex must have exactly 1 color)

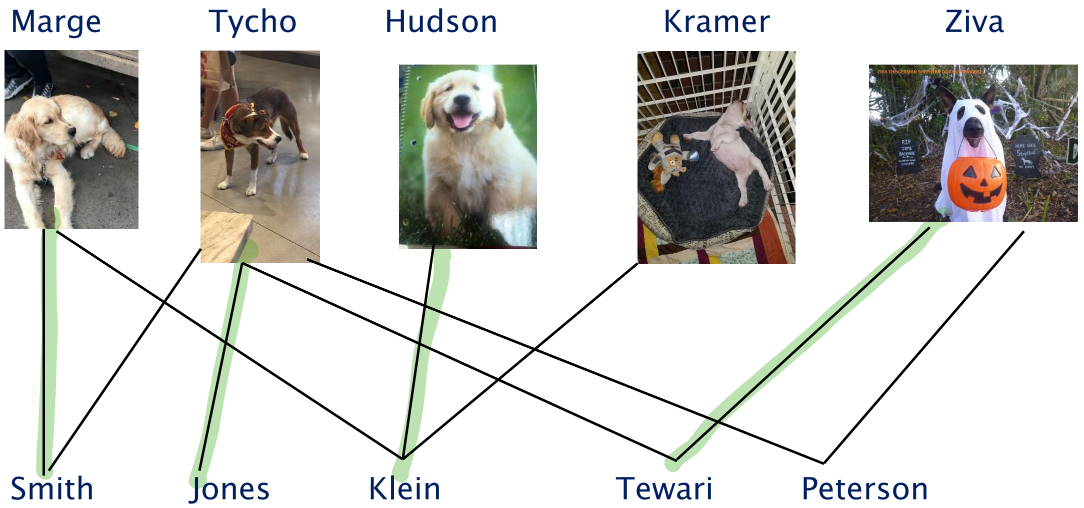
$\Rightarrow$ Since it is a valid vertex coloring (WLOG)

$\forall e = (u, v) \in E$ u is blue and v is green

$\Rightarrow u \in L$ and $v \in R$ $\Rightarrow$ G is bipartite

Puppy Adoption Exercise: Bipartite Graph Application

Families arrive at a shelter to adopt dogs. We modeled their preferences as a bipartite graph. There exists an edge between puppies and family if and only if the family wants to adopt the puppy. Can all the puppies be adopted?

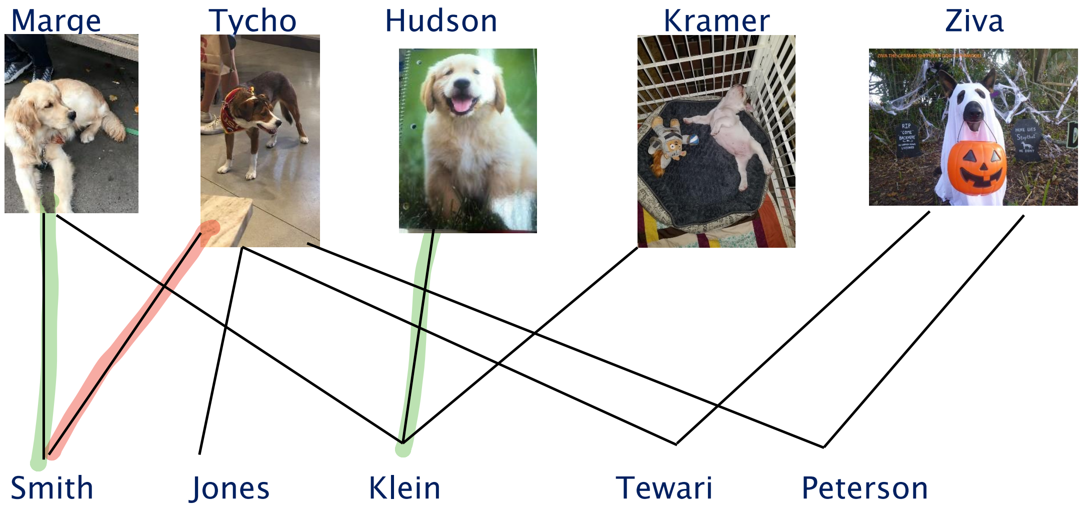


Each family may adopt at most one dog.

Matchings

A matching, M, in an undirected bipartite graph is a subset of the edges of the graph such that each vertex appears at most once in M.

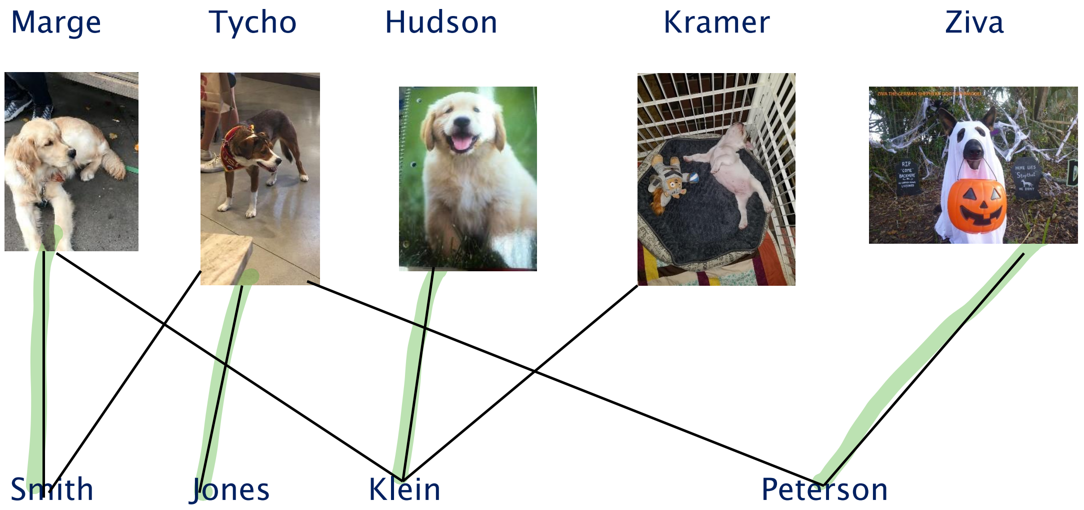
Example: {(Klein, Hudson), (Smith, Marge)}



Complete Matchings

A matching is complete if every vertex in one partition is included in the matching.

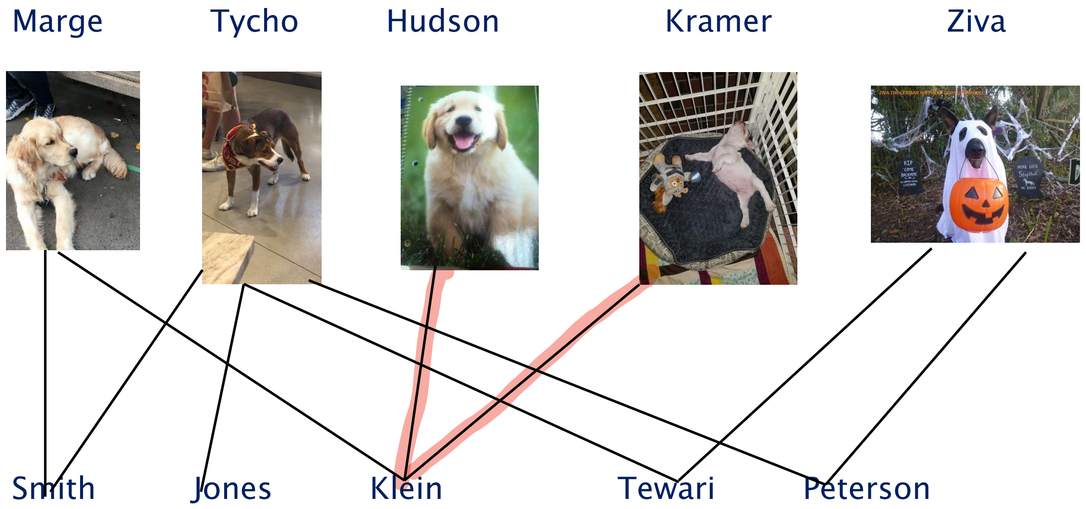
Example: {(Klein, Hudson), (Smith, Marge), (Jones, Tycho), (Peterson, Ziva)}



Perfect Matchings

A matching is perfect if every vertex in the graph is in it.

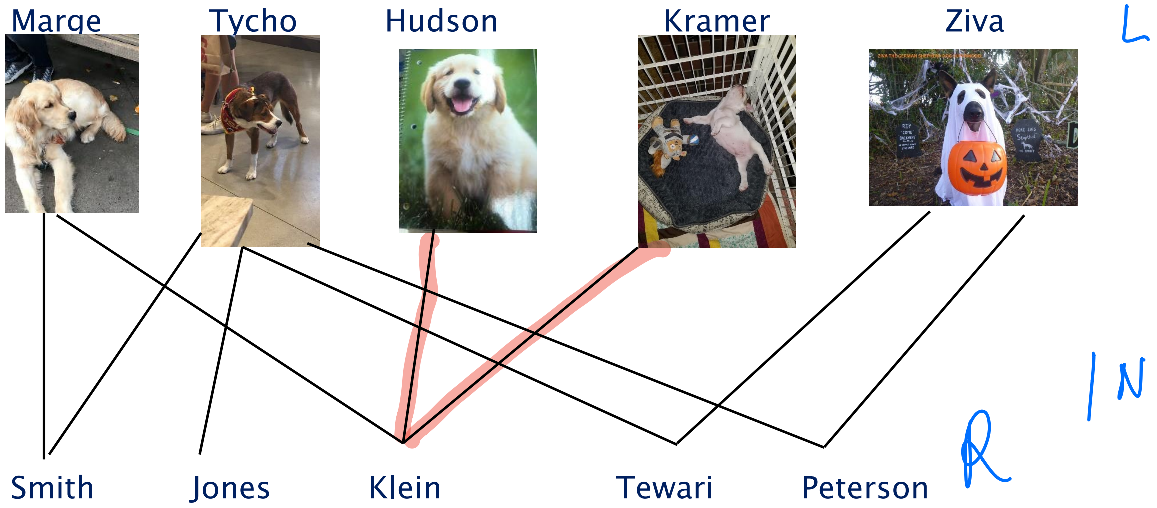
There is no perfect matching in this graph and that is why no adoption for all puppies.



Hall’s Theorem

A bipartite graph with partitions L and R has a complete matching between L and R if and only if all subsets of L have at least as many neighbors in R.

Let L be puppies and R the families. |{Hudson, Kramer}|= 2, but |N({Hudson, Kramer})= {Klein}|=1



$\{\text{Hudson}, \text{Kramer}\} \subseteq L$

$|\{H, K\}| = 2$

$|N(\{H, K\}) = \{\text{Klein}\}|$
$= 1$

Graph Coloring and Algorithms

- Vertex Coloring
- Bipartite Graphs and Matchings
- **Graph Algorithms: BFS and DFS**

1.1–1.2

Breadth First Search (BFS)

Given a connected graph G and a starting vertex s, BFS will find a path between s and every other vertex in G.

```
BFS(s){
  Mark s as discovered and all other vertices as not discovered.  Set List[0] = {s} and i = 0.
  While (List[i] is not empty) {
    Make List[i+1] be the empty list.  for (all vertices u on List[i]){
      for (all neighbors v of u){
        if (v has not been discovered){ Mark v as discovered.
          Add v to List[i+1]. }
}}  Increment i.
  }
}
```

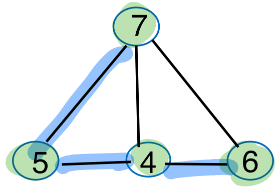
BFS(5)

List[0] = {5}   i = 0

List[1] = {4, 7}   i = 1

List[2] = {6}   i = 2

List[3] = {}   i = 3



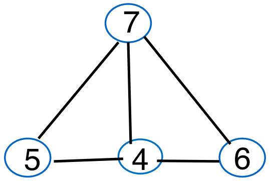
Breadth First Search (BFS)

Given a connected graph G and a starting vertex s, BFS will find a path between s and every other vertex in G.

The runtime of BFS is O(|V|+|E|)

BFS finds the shortest path between the source, s, and every other vertex in the graph.

How to modify BFS to detect if a graph is bipartite?

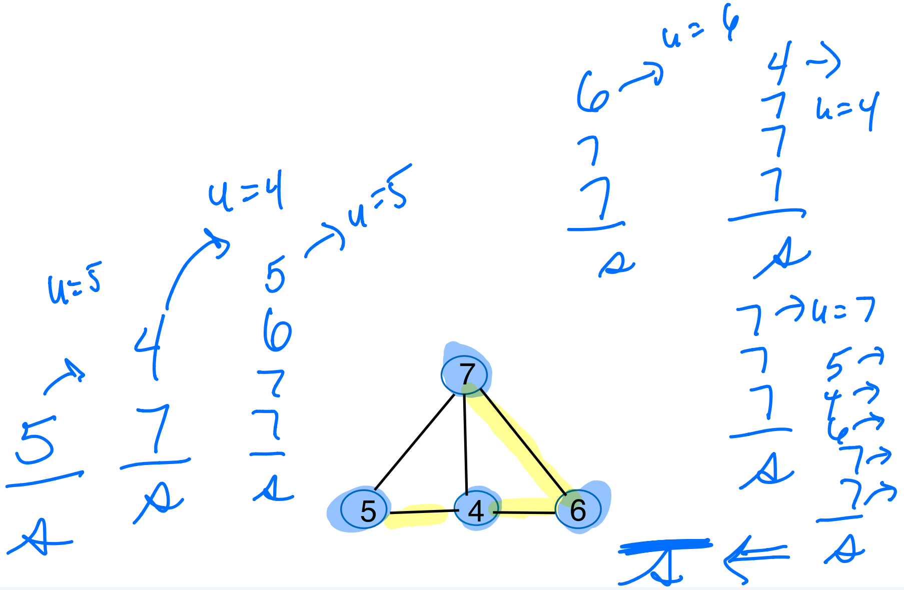


Depth First Search (DFS)

Given a connected graph G and a starting vertex s, DFS will find a path between s and every other vertex in G.

```
DFS(s){
  Mark all vertices as not discovered.  Initialize stack to only contain s.
  While (stack is not empty) {
    Let u be vertex popped off the stack.
    if (u has not been discovered) {
      Mark u as discovered.
      for (all of u's neighbor's v) {
        Push v onto the stack.
      }
    }
  }
}
```

DFS(5)



Depth First Search (DFS)

Given a connected graph G and a starting vertex s, DFS will find a path between s and every other vertex in G.

The runtime of DFS is O(|V|+|E|)

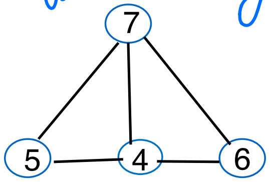
DFS does not finds the shortest path between the source, s, and every other vertex in the graph.

How to modify DFS to detect if a graph has a cycle?

1) how to implement DFS recursively

2) how to use stack more efficiently

3) how to use DFS for cycle detection in (directed or) undirected graphs



Graph Coloring and Algorithms

- Vertex Coloring
- Bipartite Graphs and Matchings
- Graph Algorithms: BFS and DFS

1.1–1.2