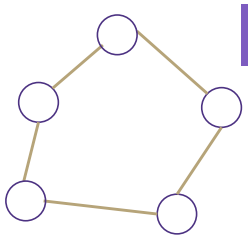
A detailed application (example)

Bipartite (also called "2-colorable")

A graph is bipartite (also called 2-colorable) if the vertex set can be divided into two sets V_1, V_2 such that the only edges go between V_1 and V_2.

Called "2-colorable" because you can "color" V_1 red and V_2 blue, and no edge connects vertices of the same color.



If a graph contains an odd cycle, then it is not bipartite.

Try the example on the right, then proving the general theorem in the light purple box.

Help Robbie figure out how long to make the explanation
Pollev.com/robbie

16

BFS With Layers

```
search(graph)
   toVisit.enqueue(first vertex)
   mark first vertex as seen
   toVisit.enqueue(end-of-layer-marker)
   l=1
   while(toVisit is not empty)
      current = toVisit.dequeue()
      if(current == end-of-layer-marker)
          l++
          toVisit.enqueue(end-of-layer-marker)
      current.layer = l
      for (v : current.neighbors())
         if (v is not seen)
            mark v as seen
            toVisit.enqueue(v)
```

It's just BFS!

With some extra bells and whistles.

19

Lemma 2

If BFS has an intra-layer edge, then the graph has an odd-length cycle.

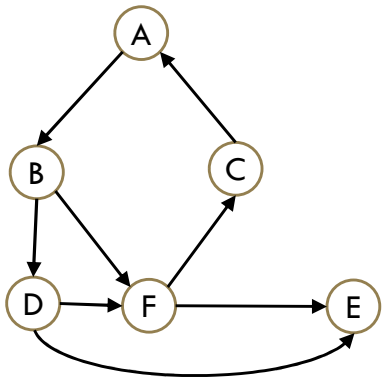
An "intra-layer" edge is an edge "within" a layer.

22

Try it Yourselves!

```
DFSWrapper(G)
     counter = 0
     For each vertex u of G
          If u is not "seen"
               DFS(u)
          End If
     End For
DFS(u)
     Mark u as "seen"
     u.start = counter++
     For each edge (u,v) //leaving u
          If v is not "seen"
               DFS(v)
          End If
     End For
     u.end = counter++
```

Type	Definition	When is (u, v) that edge type?
Tree	Edges forming the DFS tree (or forest).	v was not seen before we processed (u, v).
Forward	From ancestor to descendant in tree.	u and v have been seen, and `u.start < v.start < v.end < u.end`
Back	From descendant to ancestor in tree.	u and v have been seen, and `v.start < u.start < u.end < v.end`
Cross	Edges going between vertices without an ancestor relationship.	u and v have not been seen, and `v.start < v.end < u.start < u.end`



47