Lemma 3

If a graph has no odd-length cycles, then it is bipartite.

Prove it by **contrapositive**

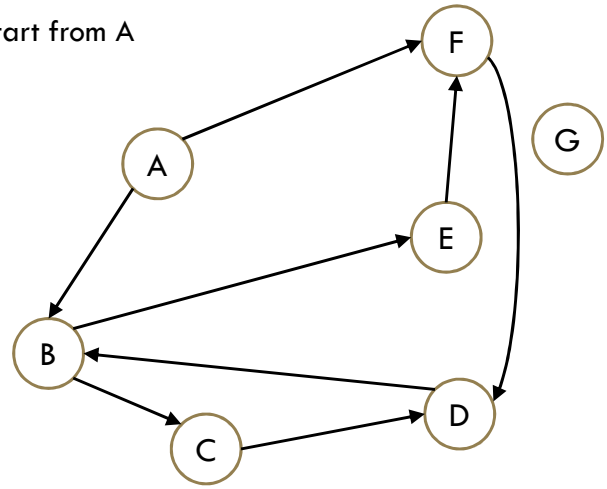
The contrapositive implication is "the same one" prove that instead!

Wrapping it up

```
BipartiteCheck(graph) //assumes graph is connected!
   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)
         else //v is seen
            if(v.layer == current.layer)
                 return "not bipartite" //intra-level edge
  return "bipartite" //no intra-level edges
```

Running DFS

Start from A




```
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++
```

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`