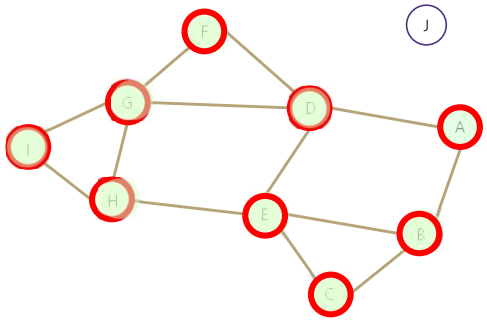
Breadth First Search

```
search(graph)
   toVisit.enqueue(first vertex)
     mark first vertex as visited
   while(toVisit is not empty)
      current = toVisit.dequeue()
      for (V : current.neighbors())
         if (v is not visited)
            toVisit.enqueue(v)
                 mark v as visited
      finished.add(current)
```



Current node: I

Queue: B D E C F G H I

Finished: A B D E C F G H I

4

Edge Classification (DFS,directed graphs)

Edge 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`

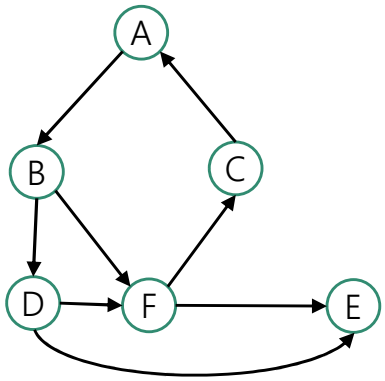
14

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`



16

How Do We Find a Topological Ordering?

```
TopologicalSort(Graph G, Vertex source)
   count how many incoming edges each vertex has
       Collection toProcess = new Collection()
       foreach(Vertex v in G){
              if(v.edgesRemaining == 0)
                     toProcess.insert(v)
   }
       topOrder = new List()
       while(toProcess is not empty){
              u = toProcess.remove()
              topOrder.insert(u)
              foreach(edge (u,v) leaving u){
                     v.edgesRemaining--
                     if(v.edgesRemaining == 0)
                            toProcess.insert(v)
              }
       }
```

39