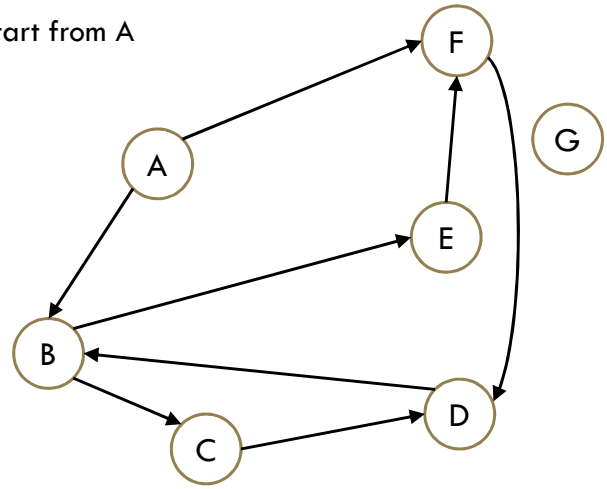
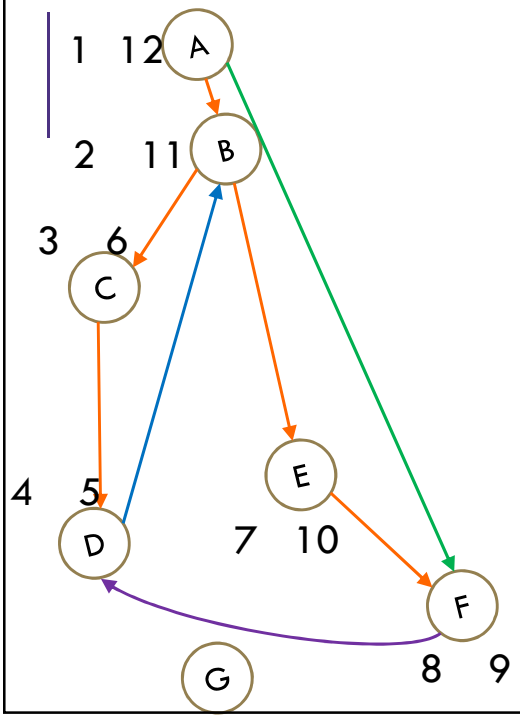
Running DFS

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++
```



The orange edges (the ones where we discovered a new vertex) form a tree!*
We call them **tree edges.**

That blue edge went from a descendent to an ancestor
B was still on the stack when we found (B,D).
We call them **back edges.**

The green edge went from an ancestor to a descendant
F was put on and come off the stack between putting A on the stack and finding (A,F)
We call them **forward edges.**

The purple edge went…some other way.
D had been on and come off the stack before we found F or (F,D)
We call those **cross edges.**

*Conditions apply. Sometimes the graph is a forest. But we call them tree edges no matter what.

Edge Classification (for DFS on 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`

The third column doesn't look like it encompasses all possibilities.
It does – the fact that we're using a stack limits the possibilities:
e.g. `u.start < v.start < u.end < v.end` is impossible.
And the rules of the algorithm eliminate some other possibilities.

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`