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.

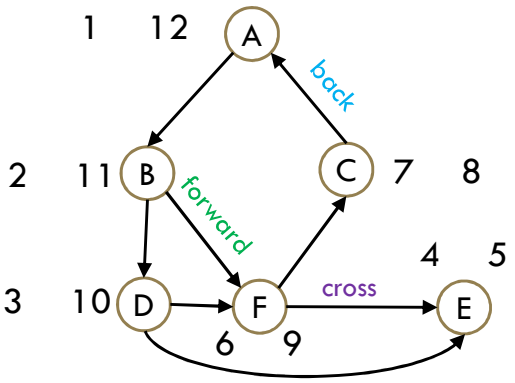
Edge Classification (for DFS on directed graphs)

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++
```



Backward direction

This direction is trickier.
Here's a "proof" – it has the right intuition, but (at least) one bug.

Suppose G has a cycle $v_0, v_1, \ldots, v_k$.

Without loss of generality, let v_0 be the first node on the cycle DFS marks as seen.

For each i there is an edge from v_i to v_{i+1}.

We discovered v_0 first, so those will be tree edges.

When we get to v_k, it has an edge to v_0 but v_0 is seen, so it must be a back edge.

Talk to your neighbors to find a bug –then try to fix it.