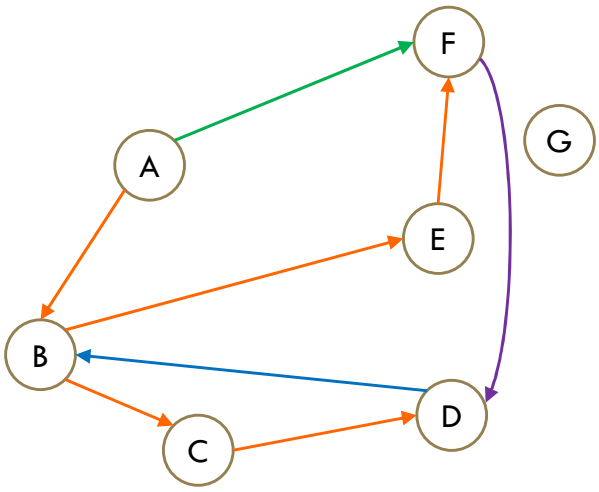


```
DFS(u)
      Mark u as "seen"
      u.start = counter++
      For each edge (u,v) //leaving u
            If v is not "seen"
                  DFS(v)
            End If
      End For
      u.end = counter++
```

18

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.

We will give you this table on exams!

20

Problem Solving Suggestions

Read the problem carefully.

Are there any technical terms in the question? Any formulas?

What kind of object will you get as input? What type is your output?

Do you understand it? Write sample inputs and outputs

We'll often give you samples, but it helps to add your own.

Now start thinking about solutions

On those examples, how would you get the solution?

Does this remind you of any algorithms from class?

Can you think of a new idea?

It's ok to start with slow solutions and try to speed them up!

Try the graph modeling process.

42

Scenario #1

You've made a new social networking app, Convrs. Users on Convrs can have "asymmetric" following (I can follow you, without you following me). You decide to allow people to form multi-user direct messages, but only if people are probably in similar social circles (to avoid spamming).

You'll allow a messaging channel to form only if for every pair of users a,b in the channel: a must follow b or follow someone who follows b or follow someone who follows someone who follows b, or …
And the same for b to a.

You'd like to be able to quickly check for any new proposed channel whether it meets this condition.

What are the vertices?

What are the edges?

What are we looking for?

What do we run?

45