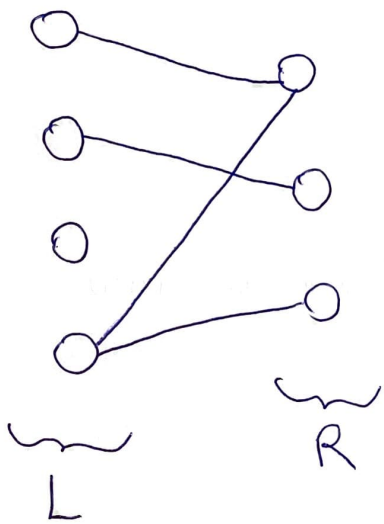
Undirected graph $G=(V,E)$ is <u>bipartite</u> if V can be split into two sets L and R such that each edge has one vertex in L and the other in R.



How to decide if a graph is bipartite?

Consider the DFS-forest. For each tree in this forest, label its vertices:

* root: label L
* child of L-node: label R
* child of R-node: label L

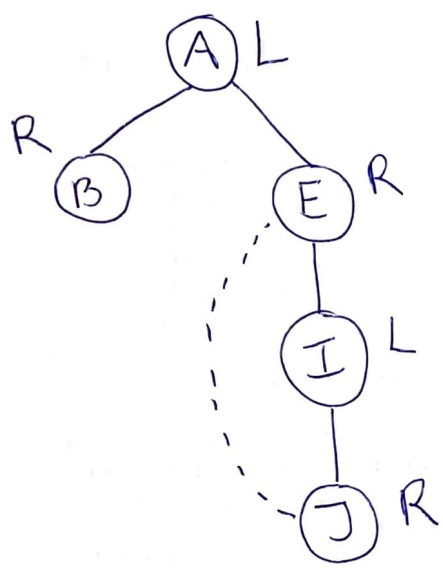
Why: <u>if</u> G is bipartite, this is the only way to split V into L and R.

Test if the two vertices of each back edge have different labels.

If yes : bipartite.

If no : not bipartite.

For the leftmost tree on page 74:

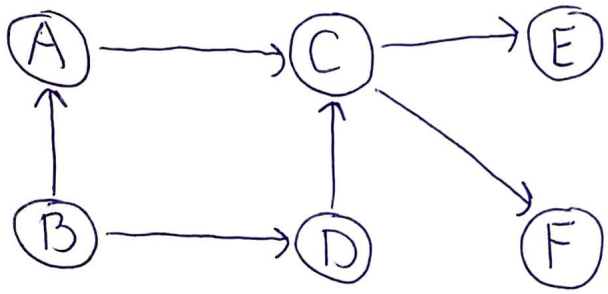


back edge {E, J} : both E and J have label R ∴ not bipartite

<u>Conclusion:</u> Testing if G is bipartite can be done in $O(|V| + |E|)$ time.

Directed graph $G = (V, E)$.

Assume G is acyclic: no directed cycles in G.



Topological sorting: number the vertices $1, 2, \ldots, n$ such that for each edge (u, v):

$$\text{number of } u < \text{number of } v$$

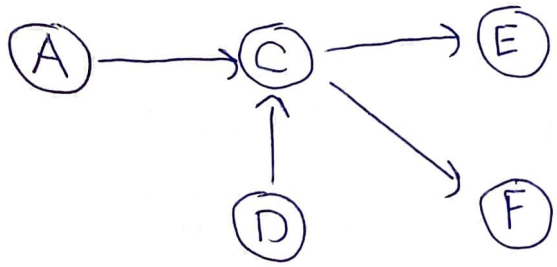
(If G is cyclic, this is not possible.)

How to compute such a numbering:

$k = 1$.
choose a vertex u with in-degree 0. // why is this possible?
give u the number k.
$k = k + 1$.
remove u.
repeat.

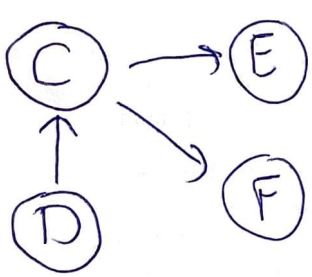
For the example graph:

B gets number 1.

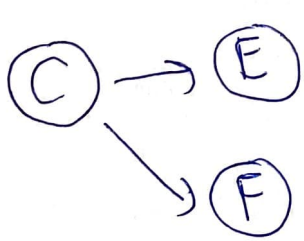


now we can choose A or D; let's say, we choose A.

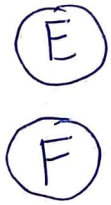
A gets number 2.



D gets number 3.

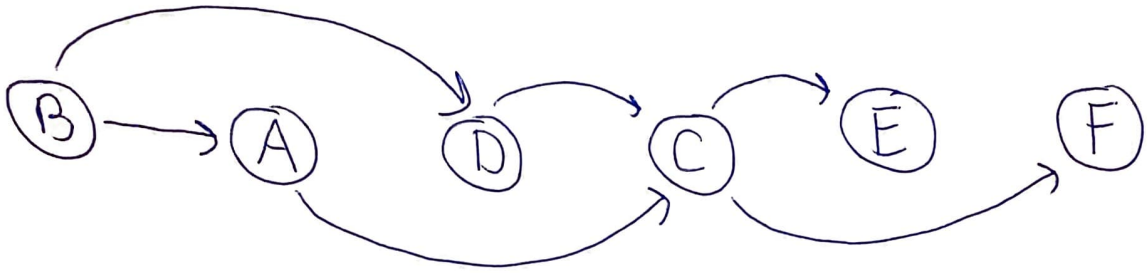


C gets number 4



E gets number 5, F gets number 6

(or vice versa)



all edges go from left to right.

We now show: DFS can be modified to directed graphs such that we can decide if a graph is acyclic.

```
Algorithm explore(v):
    visited(v) = true;
    previsit(v);
    for each edge (v,u) ∈ E        // edges leaving v
        if visited(u) = false : explore(u);
    postvisit(v)
```

We also compute for each vertex v :

pre(v) : the first time we visit v, i.e., the time at which explore(v) is called.

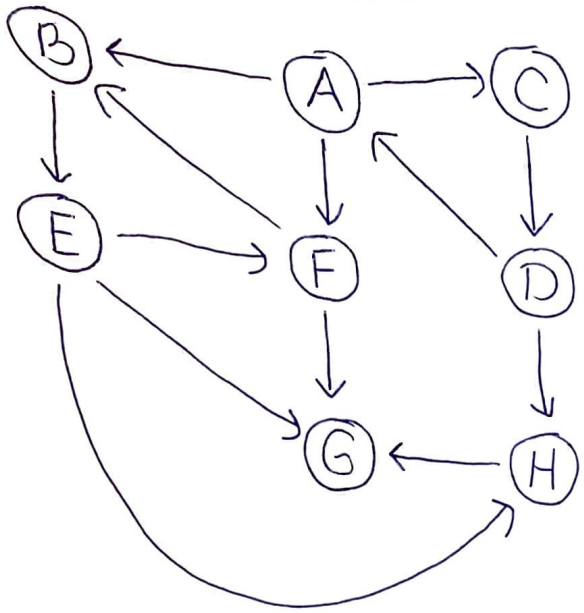
post(v): the time at which explore(v) is finished.

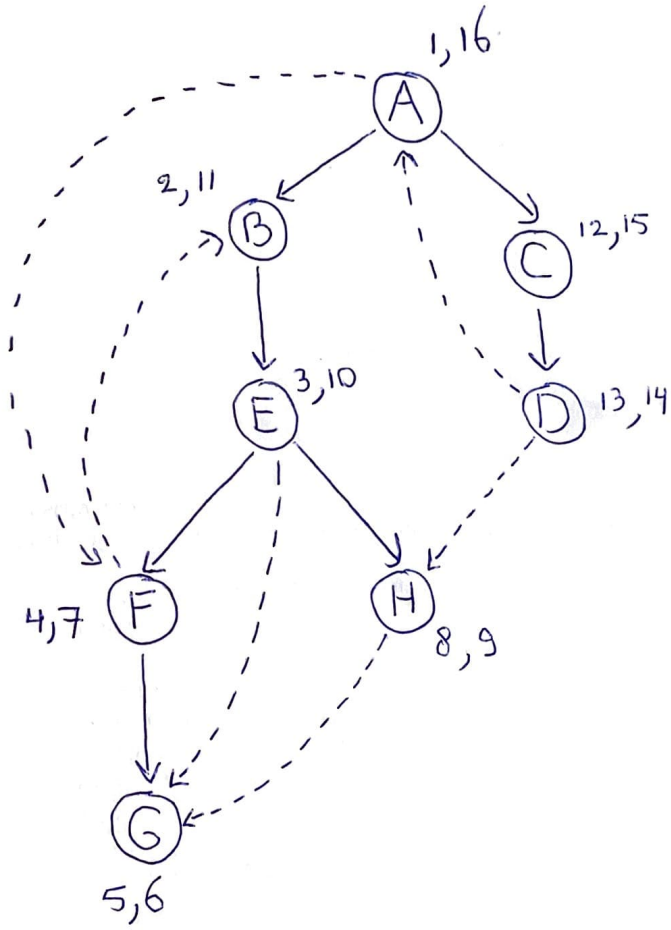
Use variable clock.

At the start : clock = 1.

previsit(v) $\equiv$ pre(v) = clock;
clock = clock + 1

postvisit(v) $\equiv$ post(v) = clock;
clock = clock + 1



Run explore(A).

numbers are pre- and post-numbers.