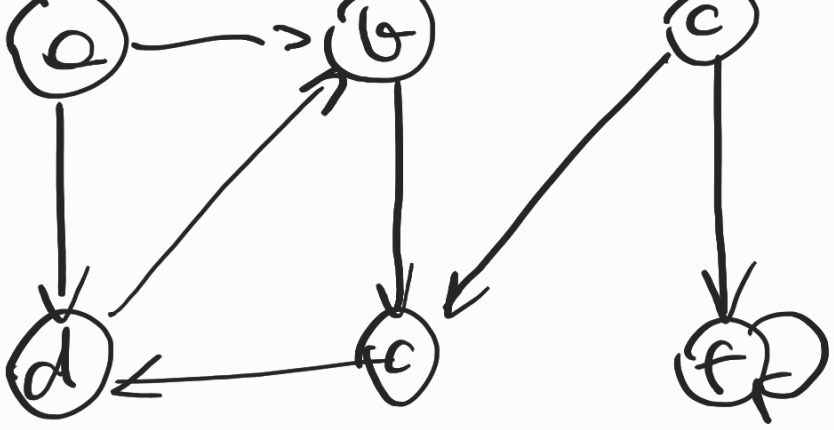
# DEPTH-FIRST SEARCH, TOPOLOGICAL SORT

- recursively explore graph, backtracking as necessary
- careful not to repeat

```
parent = {s: None}
DFS-visit(V, Adj, s):
    for v in Adj[s]:
        if v not in parent:
            parent[v] = s
            DFS-visit(V, Adj, v)
```

```
DFS(V, Adj):
    parent = {}
    for s in V:
        if s not in parent:
            parent[s] = None
            DFS-visit(V, Adj, s)
```

$\Theta(V+E)$




Analysis: $\Theta(V+E)$
(linear time)

- visit each vertex once in DFS
  alone $O(V)$
- DFS-visit(...) called once per vertex $v$
  ↳ pay $|Adj[v]|$

$$\Rightarrow O\left(\sum_{v \in V} |Adj[v]|\right) = O(E)$$

## Edge classification:

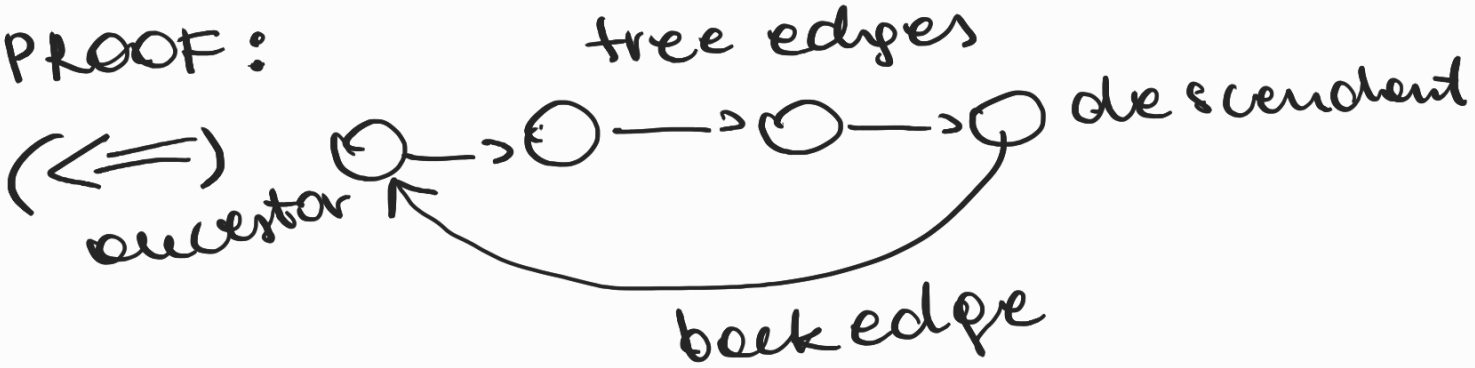
- tree edge (parent pointer)
  visit new vertex via edge
- forward edge:
  node -> descendant in tree
- backward edge:
  node -> ancestor in tree
- cross edges: between two
  neither ancestor related

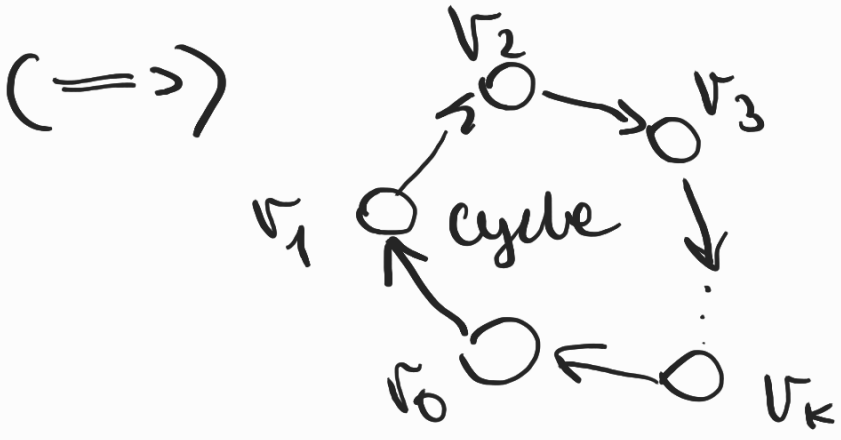
subtree.

## Cycle detection:

G has a cycle $\Leftrightarrow$ DFS has a back edge

PROOF:




assume $v_0$ is first vertex in a cycle visited by DFS.

Claim $(v_k, v_0)$ is back edge.

$v_1$ visited before finish $v_0$

$v_i$ visited before finish $v_{i-1}$

$v_k$ visited before finish $v_0$.

consider $(v_k, v_0)$ →

start $v_0$

⋮

start $v_k$

$(_0 \ldots (_k \ldots )_k \ldots )_0$

back edge

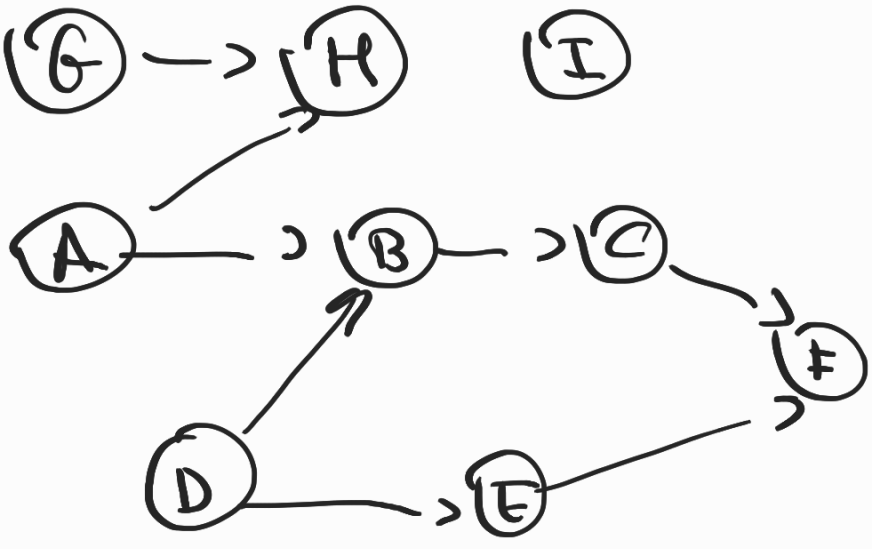
finish $v_k$

:

finish $v_0$

## Job scheduling:

given directed acyclic graph, order vertices so that all edges point from lower order to higher order.

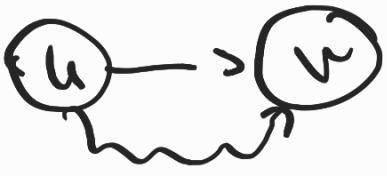



### Correctness:

for any edge $e = (u, v)$
$v$ finishes before $u$ finishes

### Case 1:

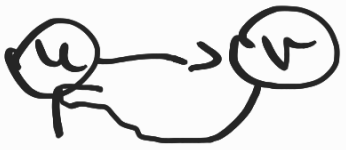
$u$ starts before $v$




=> visit $v$ before $u$ finishes

### Case 2:

$v$ starts before $u$




✗ v finish before visit u