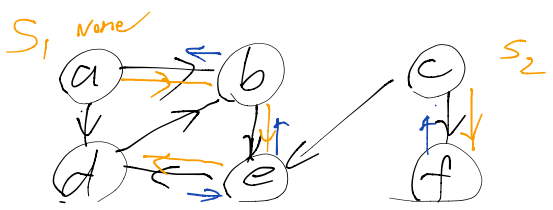
Depth-First Search, Topological Sort

1 Depth-first search (DFS)

- recursively explore the graph, backtracking as necessary
- Be 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):          find all possible starting point
    parent = { }      works for non-connected graph
    for s in V:
        if s not in parent:
            parent[s] = None
            DFS-visit(V, Adj, s)
```

e.g.



Time: $\Theta(|V|+|E|)$ (linear time)

- Visit each vertex once in DFS alone $O(V)$
- DFS_Visit(·, ···, v) called once per vertex v, pay length of adj[v],
  $\Rightarrow O(\sum_{v \in V} |Adj[v]|) = O(|E|)$

DFS doesn't find the shorst path !

DFS going as deep as it can before back tracking !

Edge Classification

- tree edge: (parent pointers) visit new vertex via the edge $\longrightarrow$ keys in parent{}

- forward edge: node $\rightarrow$ descendant in tree
- backward edge: node $\rightarrow$ Ancestor in tree
- cross edge: between two non-ancestor-related subtrees

only tree edges and backward edge exist in undirected graph

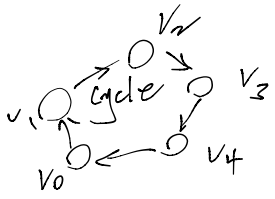
Cycle detection

G has a cycle $\rightleftharpoons$ DFS has a backward edge

o Proof

⟵ :

⟶ :



assume V_0 is ~~the~~ first vertex visited by DFS, claim (V_k, V_0) is backward edge.

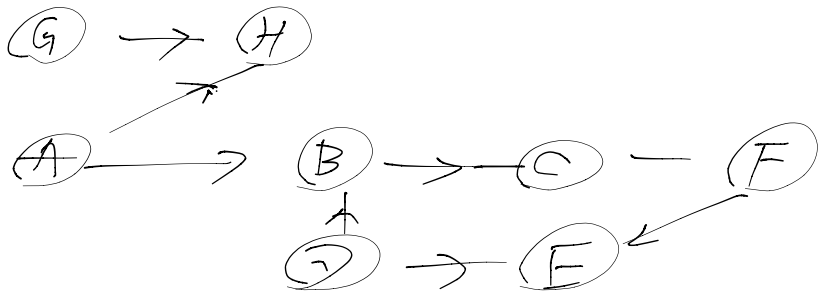
V_1 visited before finish visiting V_0
V_i visited before finish visiting V_{i-1}
V_k visited before finish visiting V_0

start V_0
$\equiv$
start V_k
finish V_k ← Consider (V_k, V_0) ↓ backward edge
$\equiv$
finish V_0

I Topological Sort

o Job Scheduling

Given directed cyclic graph, order vertices so that all edges point from lower order to higher order



o Topological Sort

Run DFS, run reverse of finishing times of vertices.

Proof:

- Correctness:

  for any edge (u, v), v finishes before u

  case 1: u starts before v

  ↳ visit v before u finishes

  case 2: v starts before u

  only happens if there is cycle

  → no cycle in DFS

  ↳ v finish before u □