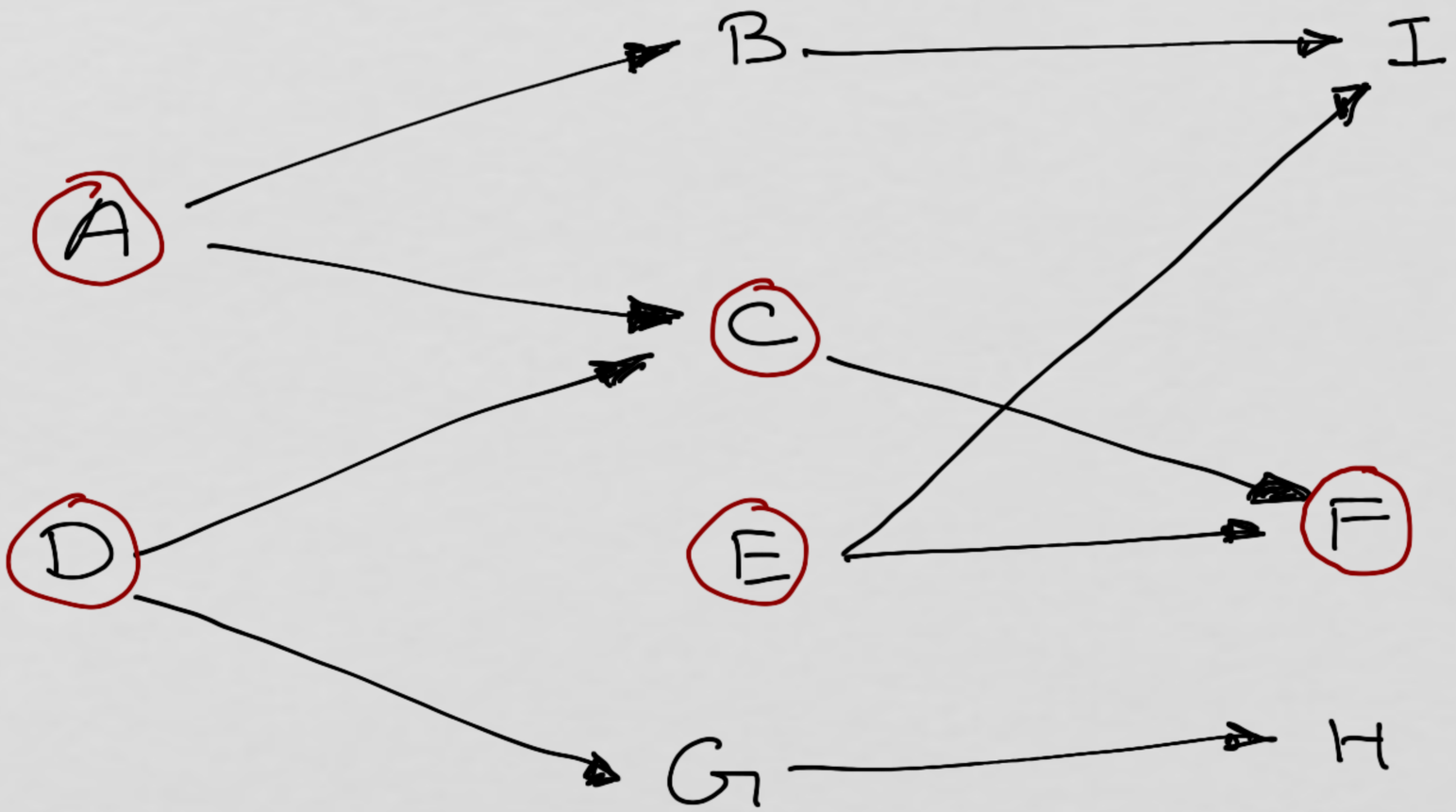
Topological Sort

- TS orders vertices of a DAG s.t. if $(v, w) \in E$ then v appears before w in the ordering

- Example Applications
  - Course pre-requisite structure
  - Event scheduling, program compilation
  - ⋮



In order to take F, I must have finished (A,D); (C,E)

with that order

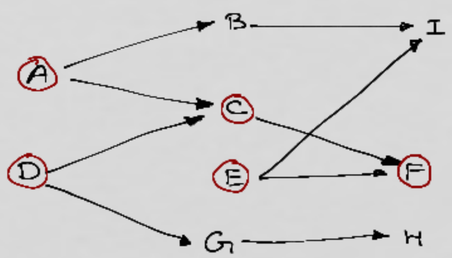
A D B C E G I F H

A B E D C . . .

Topological orderings need not be unique.

Algorithm 1

- Find any vertex with inDegree = 0
- Print the vertex & remove it along with its edges from the graph
- Pseudocode




```
for ( i = 0 to i < |V|)
    V = FindVertexOfInDegree0 ()
    if ( V == NULL)
        Throw ERROR // G is NOT DAG
    TSId [V] = i
    for each w adjacent to V
        inDegree [w] --
```

Time Complexity

Requires a scan of vertex array → $O(|V|)$

Since there $O(|V|)$ many calls to FindVertex...()

$T \in O(|V|^2)$

Algorithm 2

- Idea: reduce the inefficiency of Alg 1 by maintaining a stack of vertices with inDegree 0

- Pseudo Code:

```
Counter = 0
for each v ∈ V:
    if (inDegree[v] == 0)
        Stack.push(v)
while (!Stack.Empty()):
    v = Stack.pop()
    TSId[v] = ++counter
    for each w adjacent to v:
        inDegree[w]--
        if (inDegree[w] == 0)
            Stack.push(w)
```

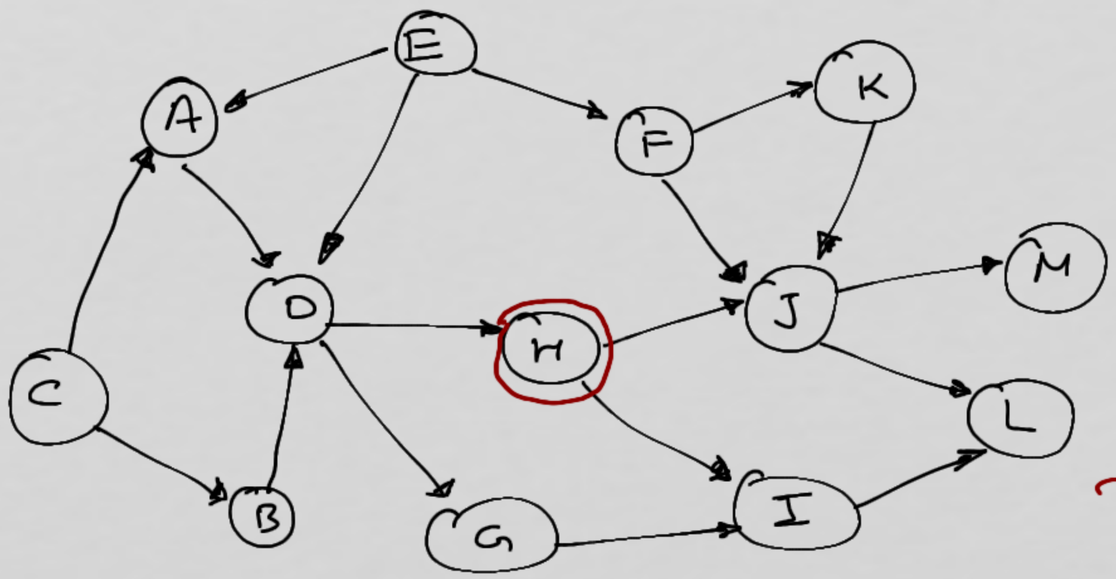
Time Complexity

$O(|V|+|E|)$

- Body of for loop executed once / edge
- Stack ops are done atmost once per vertex

Algorithm 3

- Idea : use DFS
- Pick an unvisited node & apply DFS
- On callback add the current node to the topological ordering in reverse order!



H
~~J~~
~~M~~
~~K~~
I

CBEFKADGHIJLM

Runtime : $O(|V|+|E|)$

Quiz 3

- Given a directed graph $G = (V, E)$, a *safe node* is defined as a node from which all paths lead to a *terminal node*. Terminal nodes have an outdegree 0. Give an algorithm that finds all *safe nodes* in G. Also, provide a succint argument of correctness.