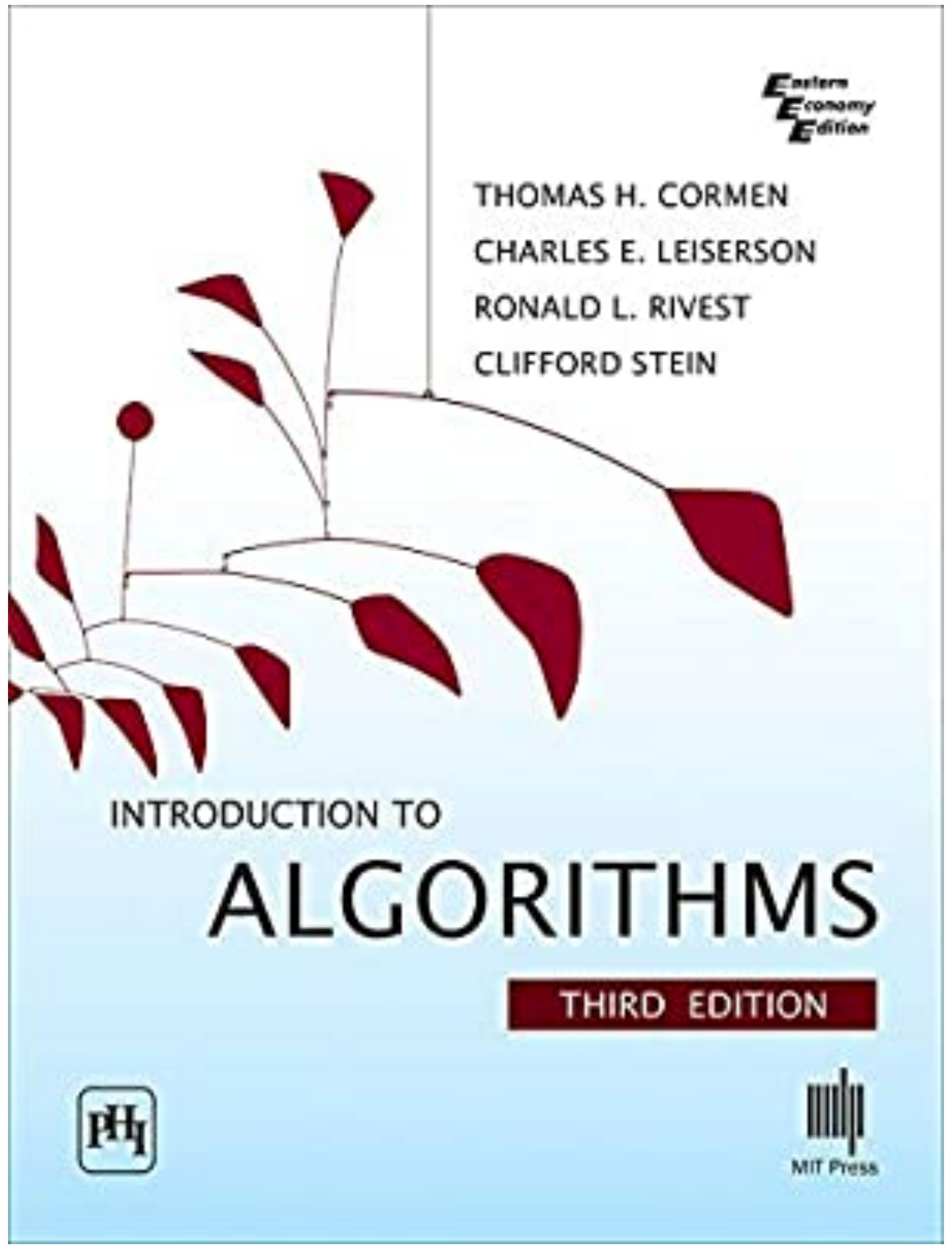
Lecture 2
Graphs II: Deep First Search

Overview

- Depth-First Search
- Edge Classification
- Cycle Testing
- Topological Sort

Readings

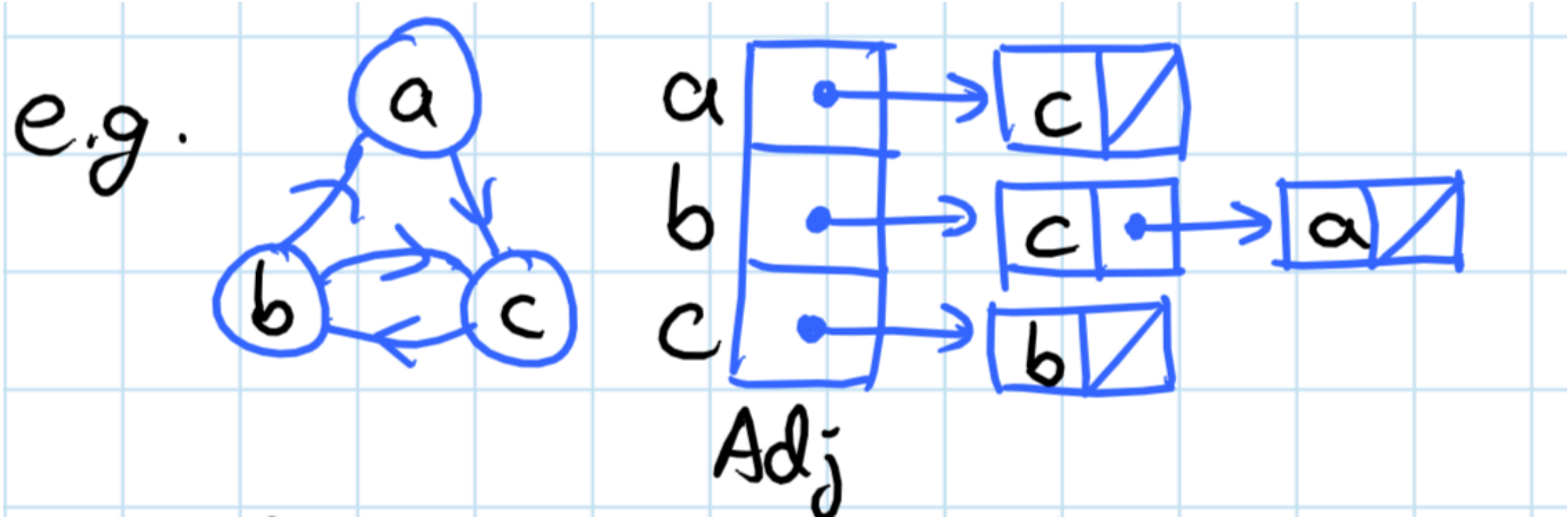


CLRS Chapter 22.3

Graph Representations: recall

Adjacency lists:

for each vertex u ∈ V, Adj[u] stores u's neighbors, i.e., {v ∈ V | (u, v) ∈ E}. (u, v) are just outgoing edges if directed.



Graph Search:

Find a path from start vertex s to a desired vertex

BFS:

Explore Level by level from S and find the shortest path

Deep-First Search

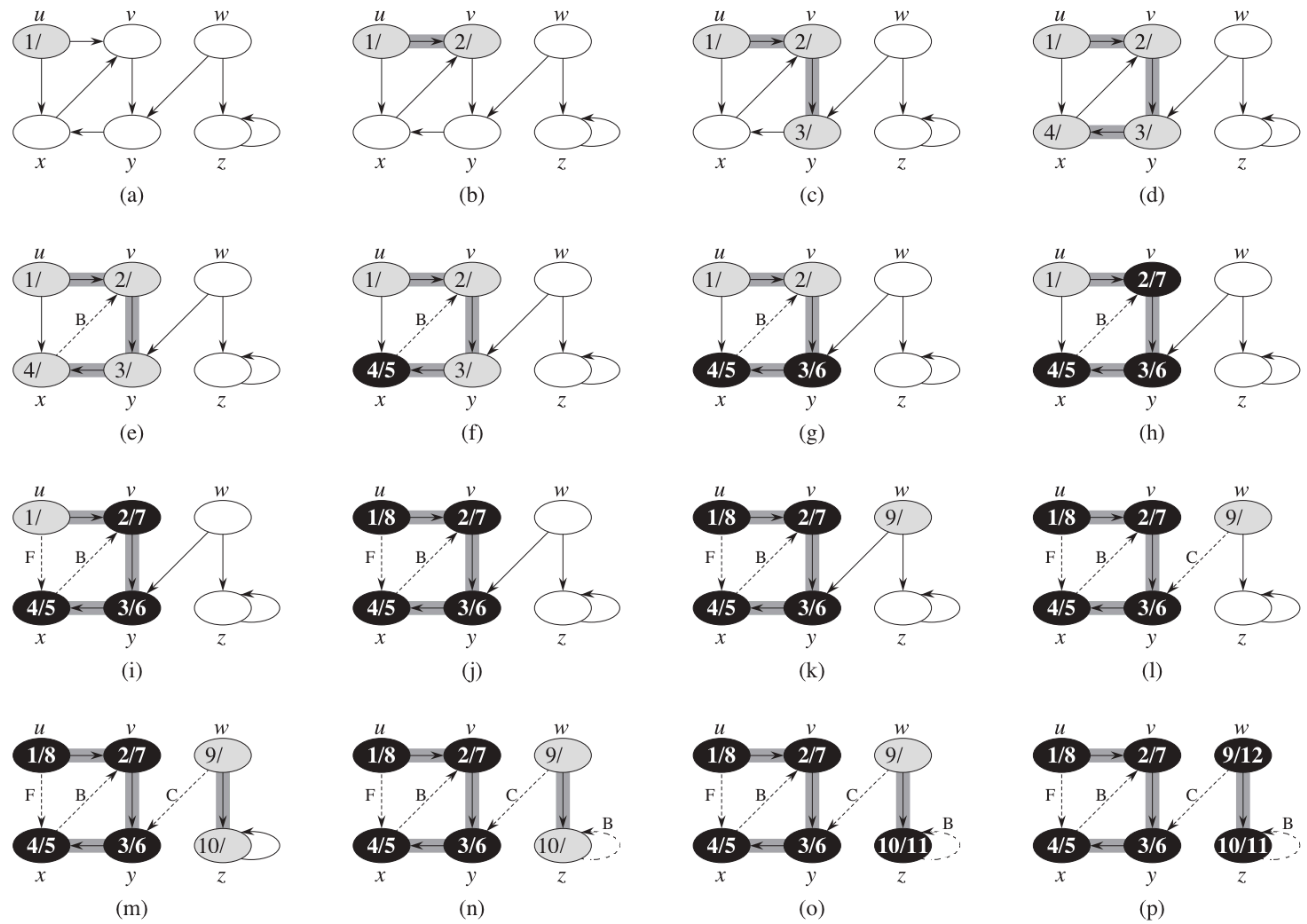
- follow path until you get stuck
- backtrack along breadcrumbs until reach unexplored neighbour
- recursively explore
- careful not to repeat a vertex

Deep first search

```
DFS(G)
for each vertex u ∈ G.V
    u.color = WHITE
    u.π = NIL
time = 0
for each vertex u ∈ G.V
    if u.color == WHITE
        DFS-VISIT(G, u)
```

```
DFS-VISIT(G, u)
time = time + 1                 // white vertex u has just been discovered
u.d = time
u.color = GRAY
for each v ∈ G.Adj[u]           // explore edge (u, v)
    if v.color == WHITE
        v.π = u
        DFS-VISIT(G, v)
u.color = BLACK                 // blacken u; it is finished
time = time + 1
u.f = time
```

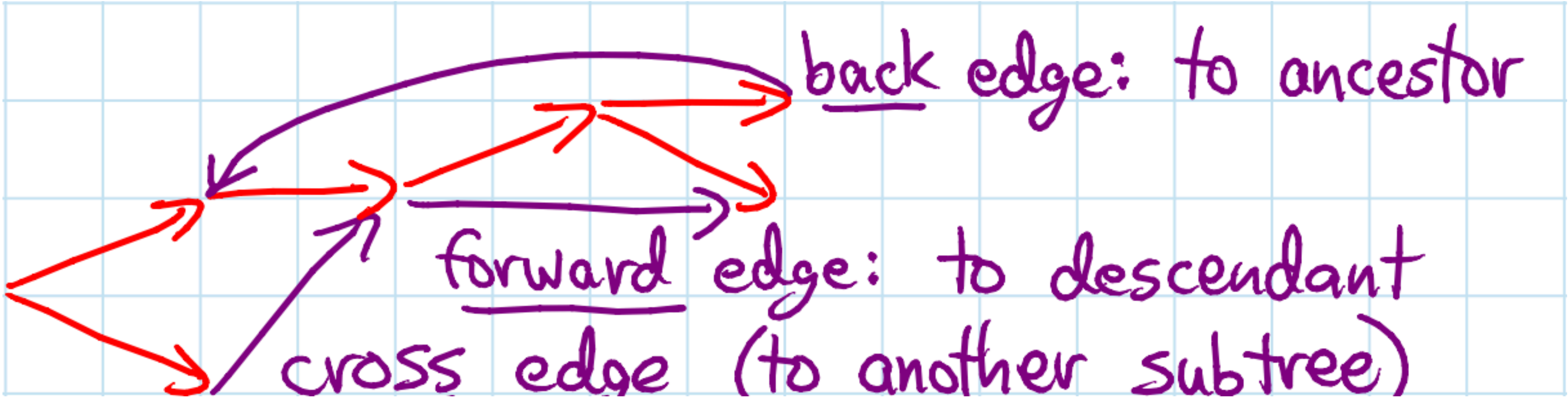
Example



Classification of Edges

1. ***Tree edges*** are edges in the depth-first forest. Edge (u, v) is a tree edge if ***v*** was first discovered by exploring edge (u, v).
2. ***Back edges*** are those edges (u, v) connecting a vertex ***u*** to an ancestor ***v*** in a depth-first tree. We consider self-loops, which may occur in directed graphs, to be back edges.
3. ***Forward edges*** are those non-tree edges (u, v) connecting a vertex ***u*** to a descendant ***v*** in a depth-first tree.
4. ***Cross edges*** are all other edges. They can go between vertices in the same depth-first tree, as long as one vertex is not an ancestor of the other, or they can go between vertices in different depth-first trees.

Example



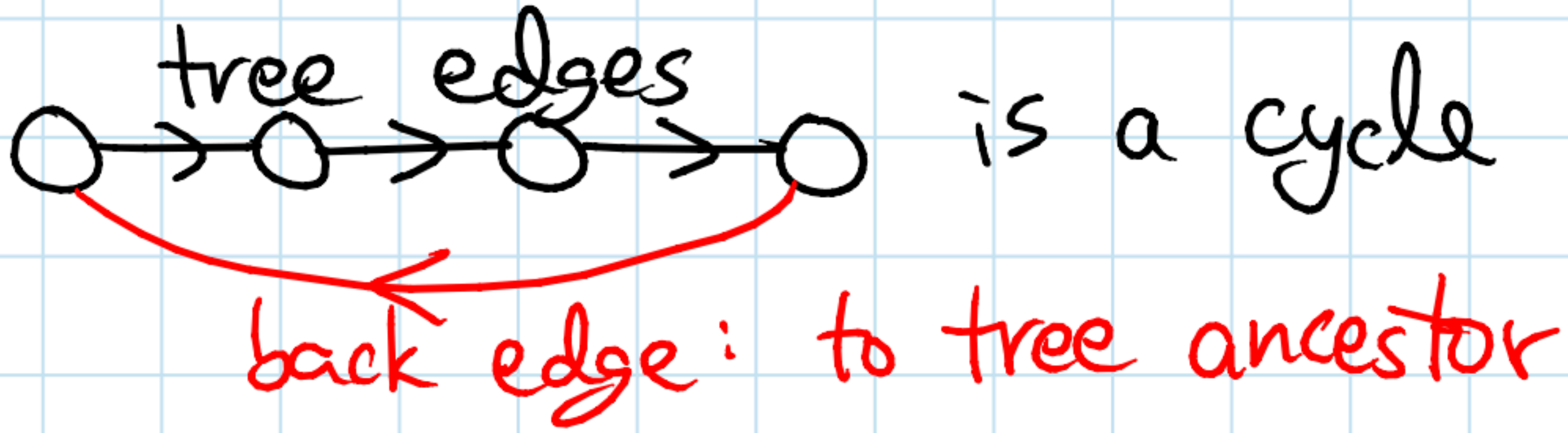
The key idea is that when we first explore an edge (u, v), the color of vertex tells us something about the edge:

1. WHITE indicates a tree edge
2. GRAY indicates a back edge
3. BLACK indicates a forward or cross edge.

Cycle Detection

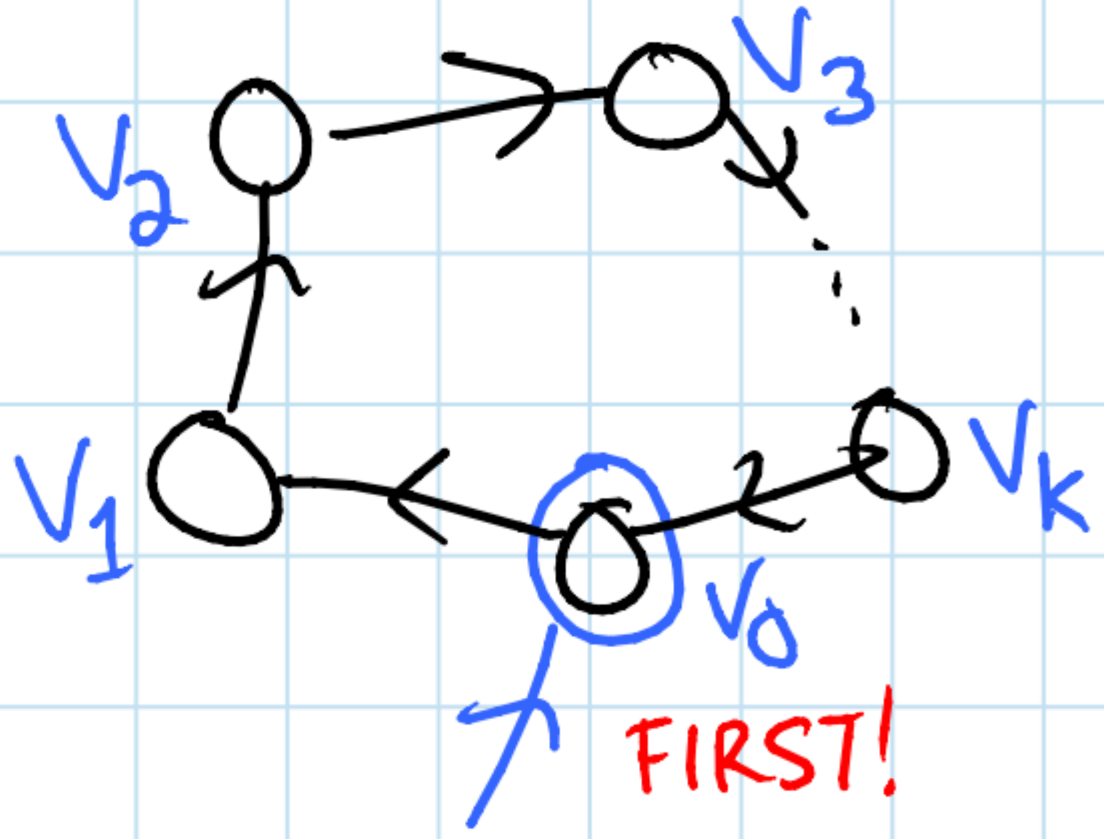
Graph G has a cycle ⇔ DFS has a back edge

Proof: (⇐)



is a cycle

(⇒) consider first visit to cycle:



Job scheduling

Given Directed Acylic Graph (DAG), where vertices represent tasks & edges represent dependencies, order tasks without violating dependencies

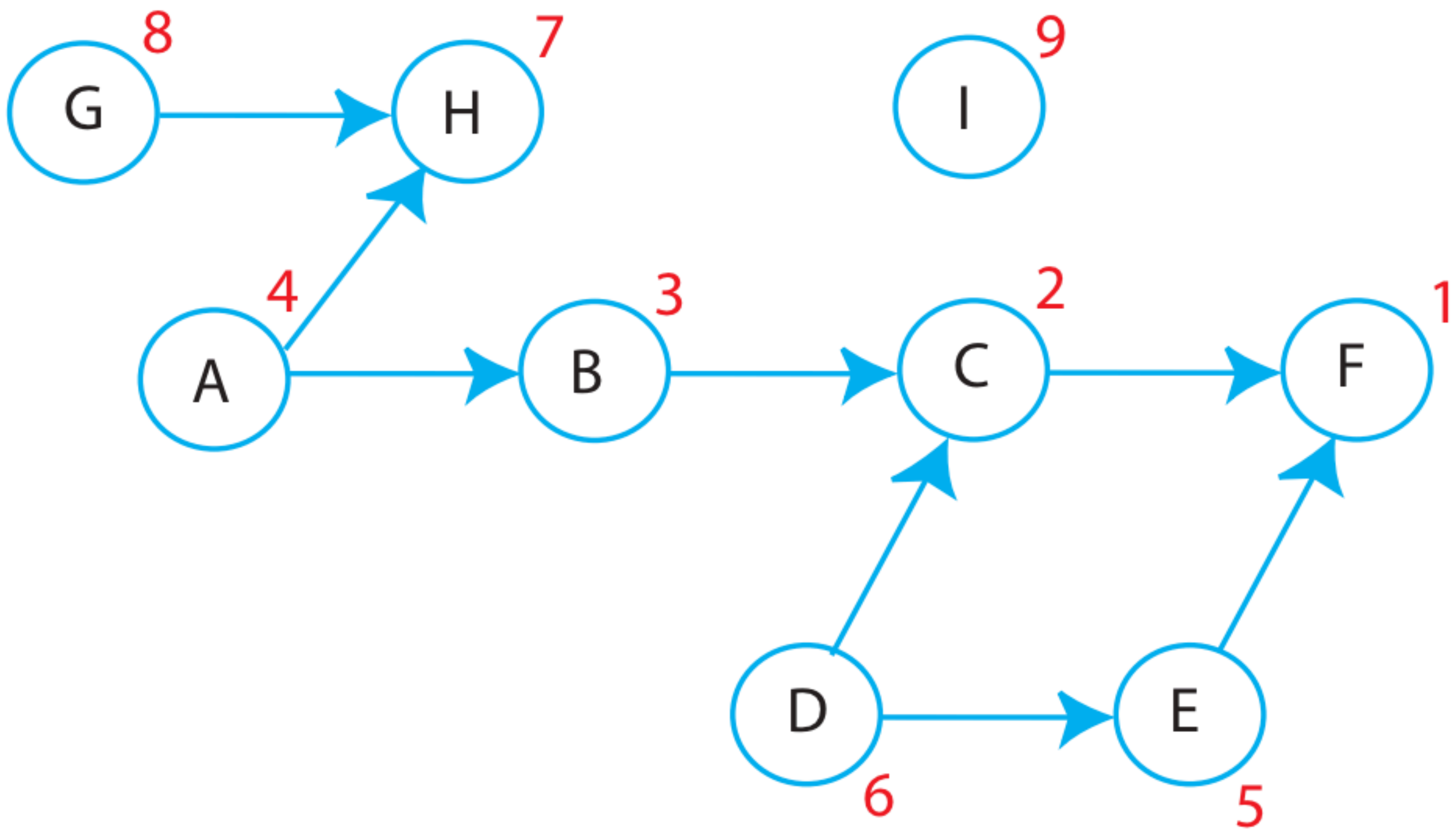


Figure 6: Dependence Graph: DFS Finishing Times

Topological sort

Source:
Source = vertex with no incoming edges = schedulable at beginning (A,G,I)

Reverse of DFS finishing times (time at which DFS-Visit(v) finishes)

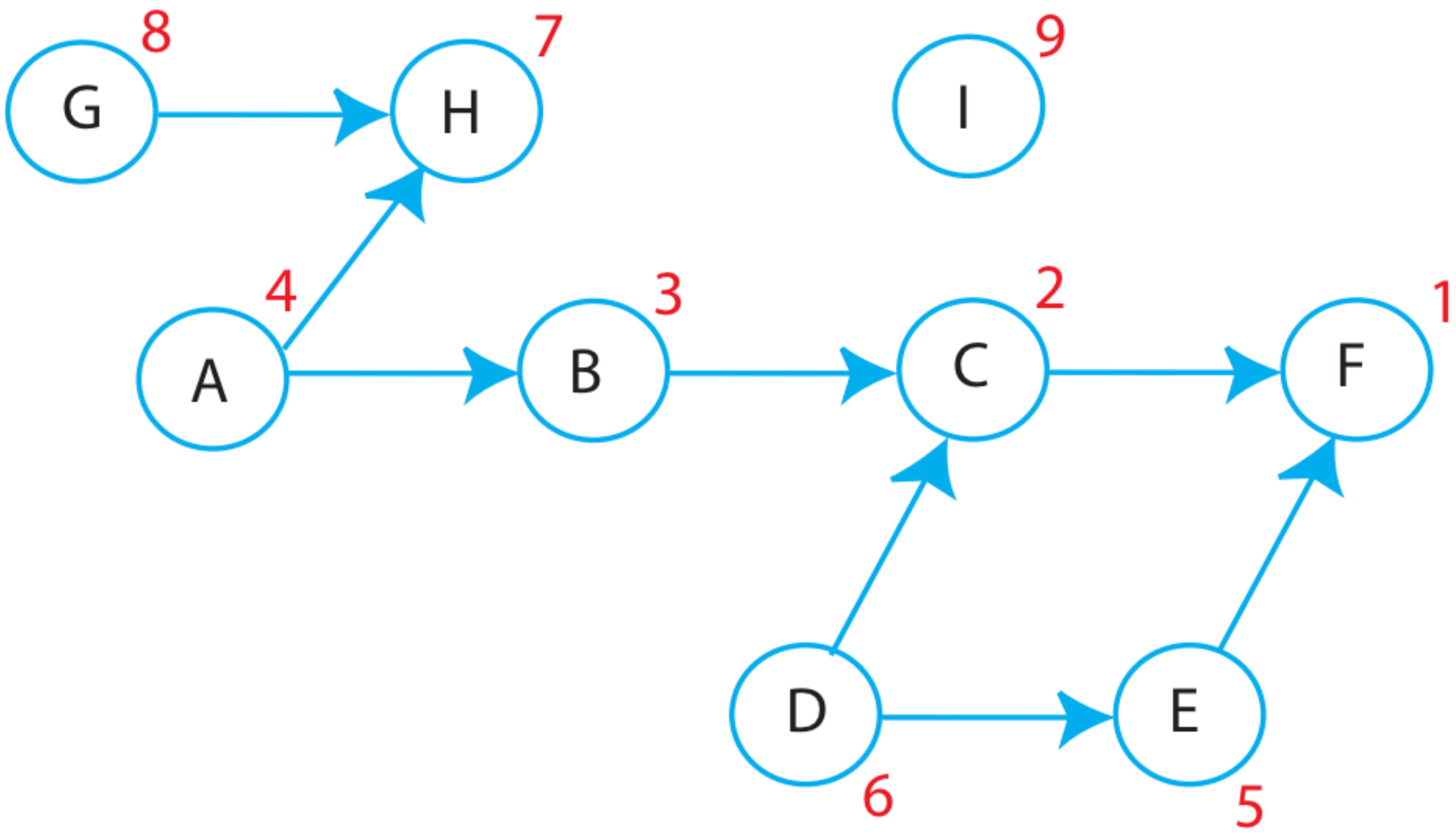


Figure 6: Dependence Graph: DFS Finishing Times