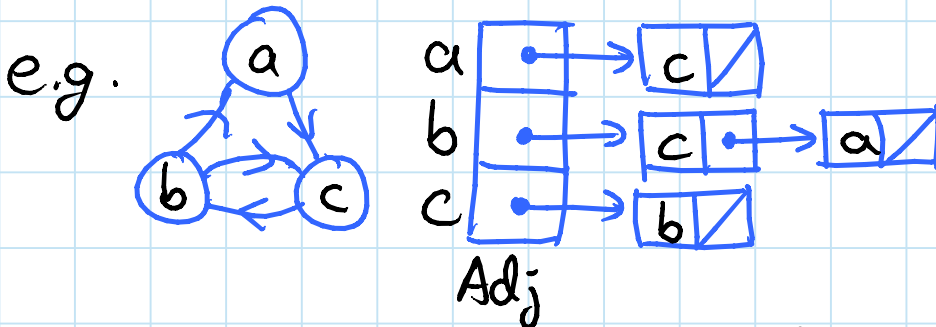


Outline: Search II: DFS (II of 2)

- depth-first search
- edge classification
- cycle testing
- topological sort

Recall:

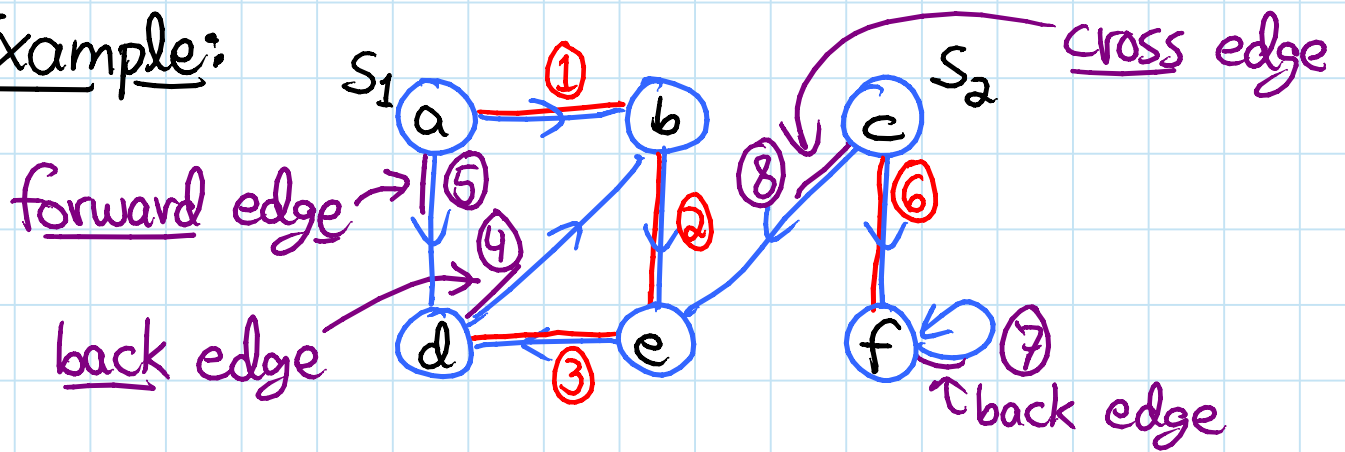
- graph search: explore a graph  
e.g. find a path from start vertex  $s$  to a desired vertex
- adjacency lists: array  $Adj$  of  $|V|$  linked lists
  - for each vertex  $u \in V$ ,  $Adj[u]$  stores  $u$ 's neighbors, i.e.  $\{v \in V \mid (u,v) \in E\}$   
just outgoing edges if directed



- BFS: explore level-by-level from  $s$ 
  - find shortest paths

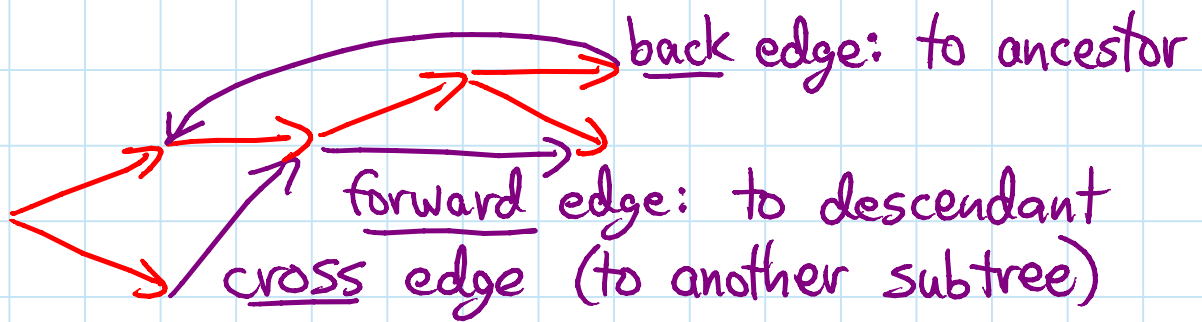


Example:



Edge classification:

tree edges (formed by parent)  
nontree edges




- to compute this classification, mark nodes for duration they are "on the stack" → back or not
- only tree & back edges in undir. graph

Analysis:

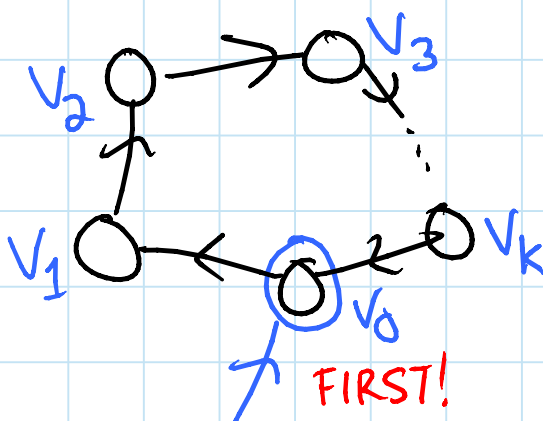
- DFS-visit gets called with a vertex  $s$  only once (because then  $\text{parent}[s]$  set)
- $\Rightarrow \text{time in DFS-visit} = \sum_{s \in V} |\text{Adj}[s]| = O(E)$

- DFS outer loop adds just  $O(V)$
- $\Rightarrow O(V+E)$  time (linear time)

Cycle detection: graph  $G$  has a cycle  $\Leftrightarrow$  DFS has a back edge

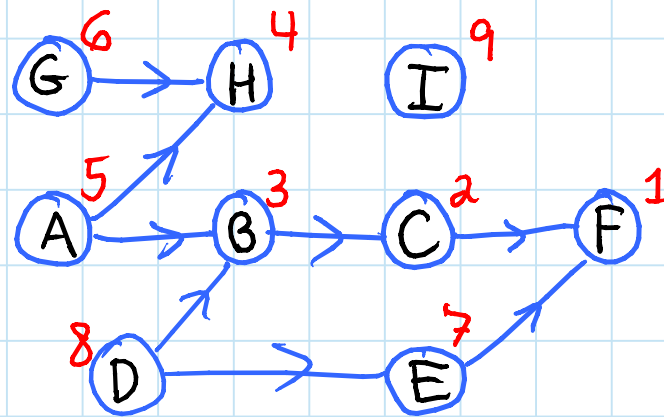
Proof: ( $\Leftarrow$ )  is a cycle

( $\Rightarrow$ ) consider first visit to cycle:



- before visit to  $v_i$  finishes, will visit  $v_{i+1}$  (& finish):  
will consider edge  $(v_i, v_{i+1})$   
 $\Rightarrow$  visit  $v_{i+1}$  now or ~~already~~ did
  - $\Rightarrow$  before visit to  $v_0$  finishes, will visit  $v_k$  (& didn't before)
  - $\Rightarrow$  before visit to  $v_k$  (or  $v_0$ ) finishes, will see  $(v_k, v_0)$  as back edge.
-

Job scheduling: given directed acyclic graph (DAG),  
where vertices represent tasks  
& edges represent dependencies,  
order tasks without violating dependencies



DFS  
finishing  
times

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

Attempt: BFS from each source:

- from A finds A, B, H, C, F
- from D finds D, B, E, C, F ←
- from G finds G, H
- from I finds I

slow...  
and  
wrong!

Topological sort: reverse of DFS finishing times  
(time at which DFS-Visit(v) finishes)

```
[ DFS-Visit(v)
  ...
  order.append(v)
  order.reverse()
```

Correctness: for any edge  $(u, v)$ ,  
u ordered before v  
i.e. v finished before u



- if u visited before v:
  - before visit to u finishes,  
will visit v (via  $(u, v)$  or otherwise)  
 $\Rightarrow$  v finishes before u
- if v visited before u:
  - graph is acyclic  
 $\Rightarrow$  u can't be reached from v  
 $\Rightarrow$  visit to v finishes before  
visiting u

□