CSCE 411: Design and Analysis of Algorithms

Lecture 11: Graph Algorithms: DFS

Date: February 25 *Nate Veldt, updated by Samson Zhou*

Course Logistics

- Graph algorithms: Chapter 22
- Homework 4 due this Friday

1 Depth First Search: Background and Motivating problems

Recall that a *breadth-first* search explores nodes that are k steps away from node s before exploring any nodes that are $k+1$ steps away.

A *depth-first search* instead explores the *most recently discovered vertex* before backtracking and exploring other previously discovered nodes.

Roughly speaking, this is accomplished by ________________________________.

Depth first search is used in several applications for analyzing directed graphs. We will take a closer look at these applications before exploring how to solve them using DFS.

Directed graph reminders

1.1 Reachability and Connected Components

Reachability. Given a graph $G = (V, E)$ and node set $S \subseteq V$, node $v \in S$ is *reachable* from node $u \in S$ if ____.

Connected components. For an undirected graph $G = (V, E)$ a connected component is a maximal subgraph in which every node in is ____________________________________

Weakly Connected components If $G = (V, E)$ is directed, a *weakly connected component* is ____

Strongly Connected components If $G = (V, E)$ is directed, a *strongly connected component* is subgraph $S \subseteq V$ in which there is ____

Question 1. *How many weakly connected components and strongly connected components are there in the following graph, respectively?*

A *1 and 3*

B *1 and 2*

C *0 and 1*

D *2 and 3*

1.2 Directed Acyclic Graphs

A *cycle* in a directed graph is a directed path ____

A *Directed acyclic* graph is a directed graph that ________________________.

Examples

1.3 Topological Sorting

A topologically ordering of a directed acyclic graph $G = (V, E)$ is an ordering of nodes so that:

2 Depth First Search Algorithm

Unlike in a BFS, a depth-first search (DFS):

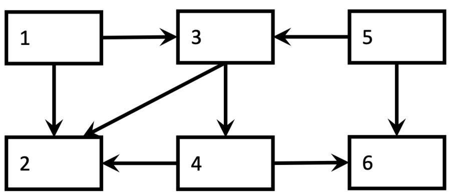
- Explores the *most recently discovered vertex* before backtracking and exploring other previously discovered vertices
- All nodes in the graph are explored (rather than just a DFS for a single node s)
- We keep track of a global *time*, and each node is associated with two timestamps for when it is *discovered* and *explored*.

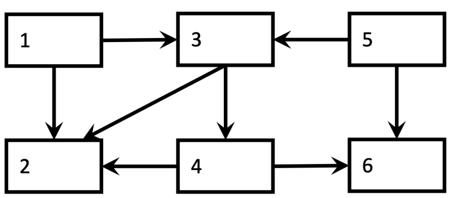
Each node $u \in V$ is associated with the following attributes

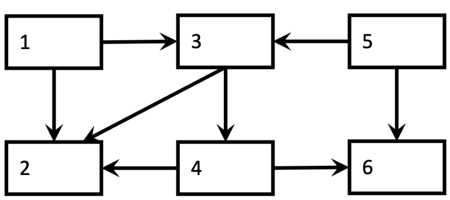
Attribute	Explanation	Initialization
u.status	tells us whether a node has been *undiscovered*, *discovered*, and *explored*	
u.D	timestamp when u is first discovered	
u.F	timestamp when u is finished being explored	
u.parent	predecessor/"discoverer" of u	

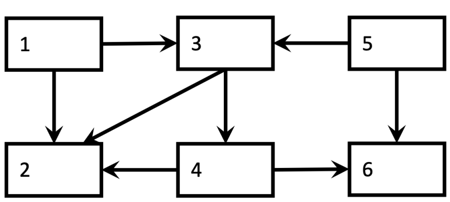
```
DFS(G)
  for v ∈ V do
      v.parent = NIL
      v.status = U
  end for
  time = 0
  for u ∈ V do
      if u.status == U then
          DFS-Visit(G, u)
      end if
  end for
```

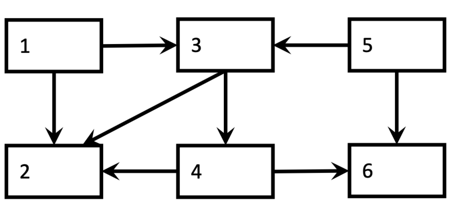
```
DFS-Visit(G, u)
  time = time + 1
  u.D = time
  u.status = D
  for v ∈ Adj[u] do
      if v.status == U then
          v.parent = u
          DFS-Visit(G, v)
      end if
  end for
  u.status = E
  time = time + 1
  u.F = time
```











2.1 Runtime Analysis

Question 2. *What is the runtime of a depth first search, assuming that we store the graph in an adjacency list, and assuming that* $|E| = \Omega(|V|)$*?*

A $O(|V|)$

B $O(|E|)$

C $O(|V| \times |E|)$

D $O(|V|^2)$

E $O(|E|^2)$

2.2 Properties of DFS

Theorem 2.1. *In any depth-first search of a graph $G = (V, E)$, for any pair of vertices u and v, exactly one of the following conditions holds:*

- $[u.D, u.F]$ *and* $[v.D, v.F]$ *are disjoint;* ____________________________________

- $[v.D, v.F]$ *contains* $[u.D, u.F]$ *and* ______________________________________

- $[u.D, u.F]$ *contains* $[v.D, v.F]$ *and* ______________________________________

2.3 Classification of Edges

Given a graph $G = (V, E)$ performing a DFS on G produces a graph $\hat{G} = (V, \hat{E})$ where

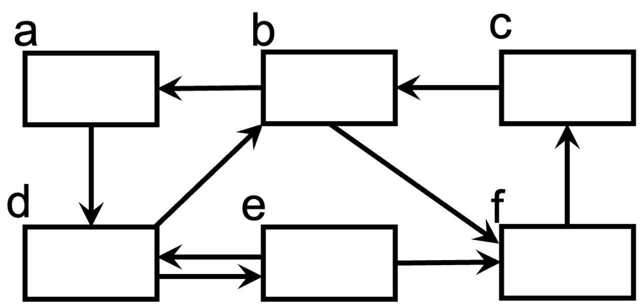
$$\hat{E} = \{(u.\text{parent}, u) \colon v \in V \text{ and } v.\text{parent} \neq NIL\}$$

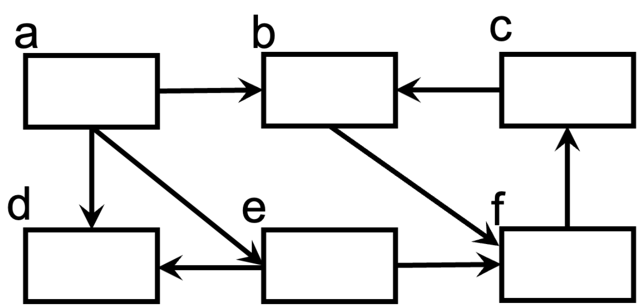
This is called a *depth-first* forest of G.

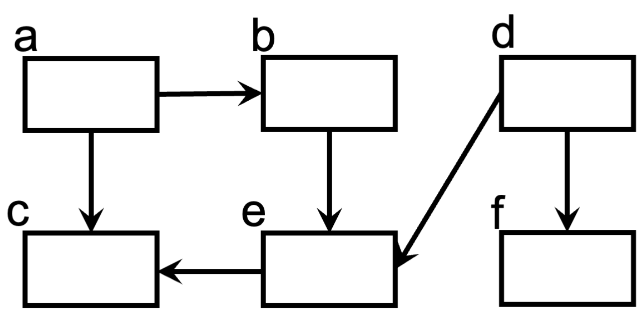
Given any edge $(u, v) \in E$, we can classify it based on the status of node v when we are performing the DFS:

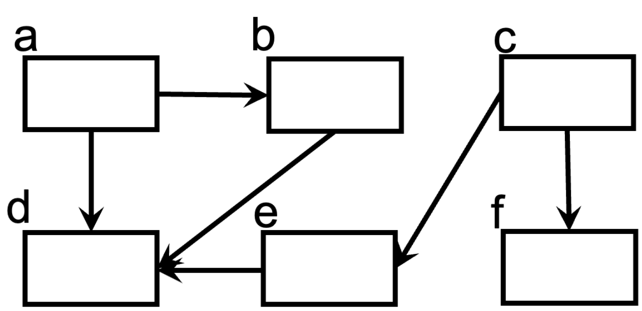
Edge	Explanation	How to tell when exploring (u, v)?
Tree edge	edge in $\hat{E}$	
Back edge	connects u to ancestor v	
Forward edge	connects vertex u to descendant v	*and* $u.D < v.D$
Cross edge	either (a) connects two different trees or (b) crosses between siblings/cousins in same tree	*and* $u.D > v.D$

3 Practice









Question 3. *How many of the above graphs were directed acyclic graphs?*

A *1*

B *2*

C *3*

D *4*

E *none of them*