CSCE 411: Design and Analysis of Algorithms

Week 6: Graph Algorithms: BFS and DFS

Date: February 17, 2026 *Nate Veldt, updated by Samson Zhou*

Course Logistics

- Graph algorithms: Chapter 22
- Homework 4 out this weekend, due next Friday

1 Graph Notation and Terminology

An (undirected) graph $G = (V, E)$ is defined by

An edge between nodes i and j is denoted by ____________________

We can also denote an edge by ____________________

If $(i, j) \in E$, we say i and j are _______________. The neighborhood of node i is the set of nodes adjacent to it:

The *degree* of i is the number of neighbors it has: __________

1.1 Generalized graph classes

- Weighted:

- Directed:

1.2 Basic graphs and edge structures

- A **complete graph** is a graph in which

- A **bipartite graph** is a graph in which

- **Triangle**: set of three nodes that all share edges:

$$\{i, j, k\} \subseteq V \text{ such that } \{(i, j), (i, k), (j, k)\} \in E$$

- **Path**: is a sequence of edges joining a sequence of vertices:

$$\{i_1, i_2, \dots i_k\} \subseteq V \text{ where } (i_1, i_2) \in E, (i_2, i_3) \in E, \dots, (i_{k-1}, i_k) \in E.$$

- **Matching**: is a set of edges without common vertices

$$\mathcal{M} \subseteq E \text{ such that for all } e_i, e_j \in \mathcal{M} \text{ with } e_i \neq e_j, e_i \cap e_j = \emptyset.$$

- **Connected component**: a maximal subgraph in which there is a path between every pair of nodes in the subgraph.

1.3 Optimization Problems on Graphs

Many graph analysis problems amount to optimizing an objective function over a graph.

Example 1. Shortest path. Given a source node $s \in V$ and target node $t \in V$, find the shortest path of edges between s and t.

Example 2. Maximum bipartite matching. Let $G = (V, E)$ be a bipartite graph. Find a matching $\mathcal{M}$ with maximum sum of edge weights.

Example 3. Find connected components. Return the connected components of a graph:

2 Graph Representation

Consider a graph $G = (V, E)$ with a fixed node ordering $V = \{1, 2, \ldots, n\}$.

Adjacency Matrix The **adjacency matrix** $\mathbf{A}$ of G is defined so that

Adjacency List The **adjacency list** Adj of G is

Graph Activity

Consider the following graph:

- Write down the adjacency matrix
- Write down the adjacency list
- Write down the degree of each node
- Write down the neighborhood of node 3
- Find the number of connected components

Question 1. *Assume that $G = (V, E)$ is a graph in which each node has at least one edge touching it. Let $n = |V|$ and $m = |E|$. How much space is needed to store the graph?*

A $O(n)$

B $O(m)$

C $O(n^2)$

D $O(mn)$

3 Breadth First Search

Shortest Path Problem: Given a graph $G = (V, E)$ and source node $s \in V$, find the shortest path from s to every other $v \in V$.

We will do this using the *breadth first search* algorithm.

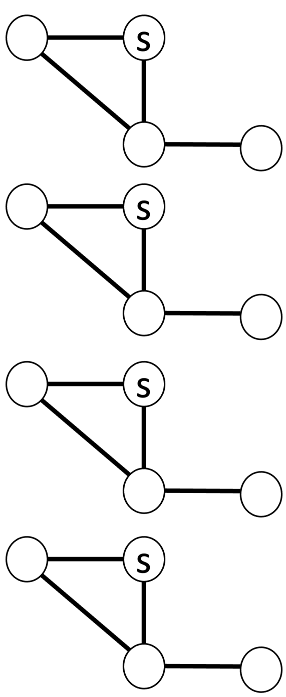
Attribute	Explanation	Initialization
u.status	tells us whether a node is	
u.dist		
u.parent		

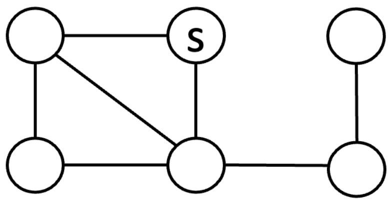
We will also make use of

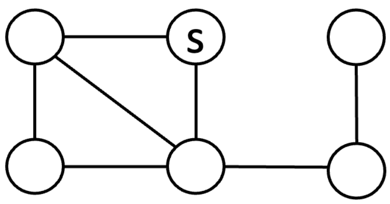
Basic Idea

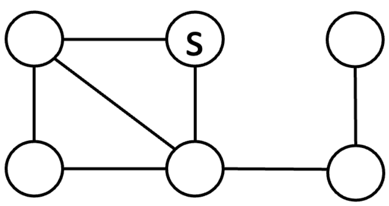
- Mark s as
- Iteratively *explore* discovered nodes
- Continuously update

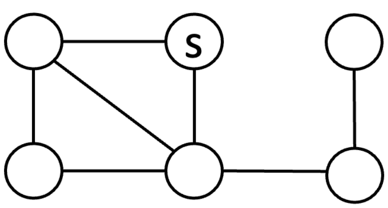
Example

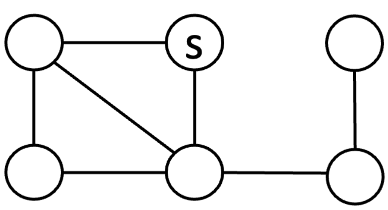


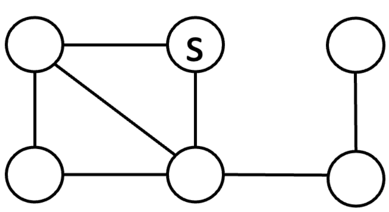

S

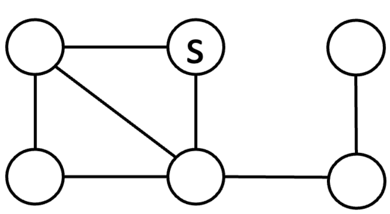

S


S


S


S


S


S

3.1 Shortest Paths and Breadth First Trees

Definition 1. Given a graph $G = (V, E)$, source node s, and a *parent* attribute for each node, a ____________________________ is a subgraph $\hat{G} = (\hat{V}, \hat{E})$ where

It is furthermore a ______________________________ if it contains a unique simple path

from s to v that is ____

Benefits of the BFS algorithm

- If G is undirected, it finds the ____

- It tells us the ____

- It provides a ______________________________

```
BFS(G,s)
  for v ∈ V do
    v.parent = NIL
    v.dist = ∞
    v.status = U
  end for
  s.dist = 0
  s.status = D
  Initialize Q
  Enqueue(s)
  while |Q| > 0 do
    u = Dequeue(Q)
    N(u) = Adj[u]
    for v in N(u) do
      if v.status == U then
        v.status = D
        v.parent = u
        v.dist = u.dist + 1
        Enqueue(v)
      end if
    end for
    u.status = E
  end while
```

3.2 Code and Runtime Analysis

- We assume G is undirected and stored as an adjacency list.
- Initializing attributes takes __________________
- Each node u only enters Q once, and entering/leaving Q takes _________
- When we *explore* u, we discover up to ______________________

Using aggregate analysis, what is the overall runtime of this method?

4 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

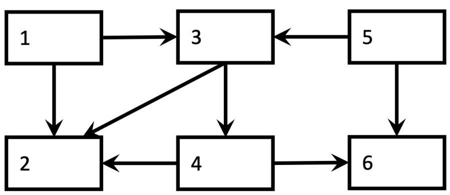
Attribute	Explanation	Initialization
u.status	tells us whether a node has been *undiscovered*, *discovered*, and *explored*	
u.D	timestamp when u is first discovered	
u.F	timestamp when u is finished being explored	
u.parent	predecessor/"discoverer" of u	

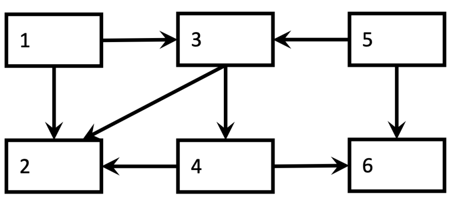
DFS(G)

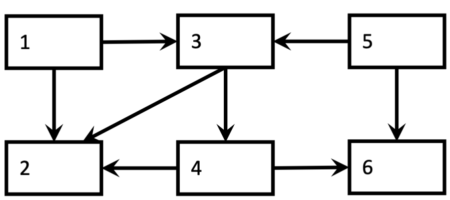
  for $v \in V$ **do**
    v.parent $= NIL$
    v.status $=$ U
  end for
  time $= 0$
  for $u \in V$ **do**
    if u.status $== U$ **then**
      DFS-Visit(G, u)
    end if
  end for

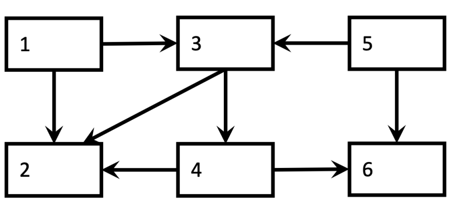
DFS-Visit(G, u)

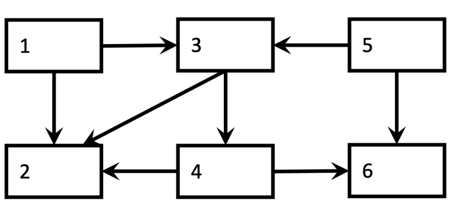
  time $=$ time $+ 1$
  $u.D =$ time
  u.status $= D$
  for $v \in$ Adj$[u]$ **do**
    if v.status $== U$ **then**
      v.parent $= u$
      DFS-Visit(G, v)
    end if
  end for
  u.status $=$ E
  time $=$ time $+ 1$
  $u.F =$ time











4.1 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 $O(|E|^2)$

4.2 Properties of DFS

Theorem 4.1. *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* ______________________________________

4.3 Classification of Edges

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

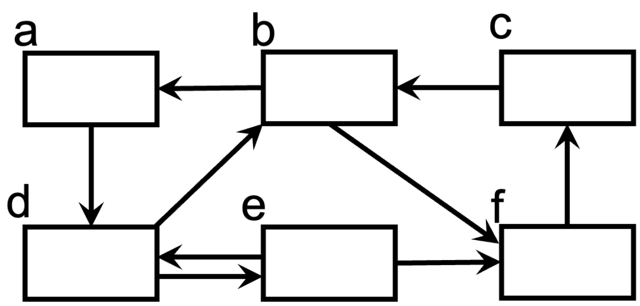
$$\hat{E} = \{(u.\text{parent}, u) \colon v \in V \text{ and } v.\text{parent} \neq NIL\}$$

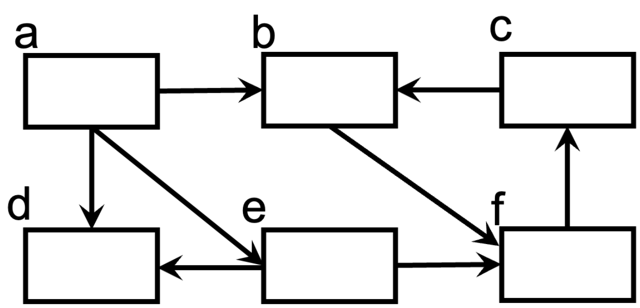
This is called a *depth-first* forest of G.

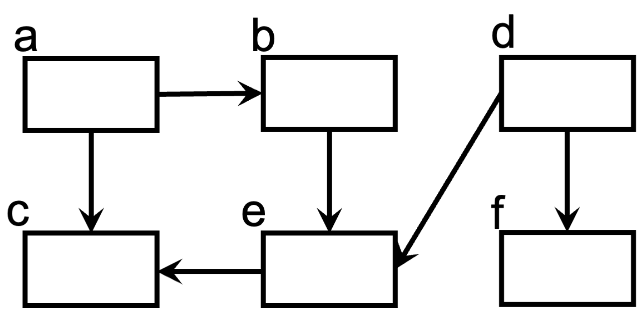
Given any edge $(u, v) \in E$, we can classify it based on the status of node v when we are performing the DFS:

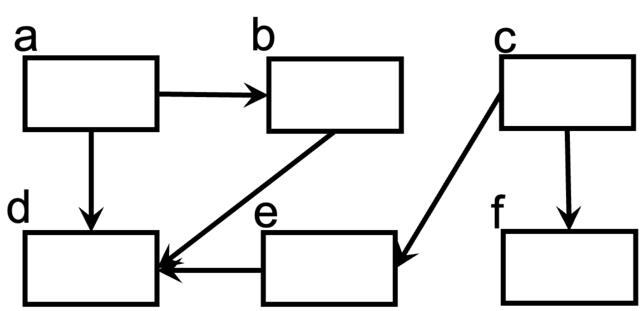
Edge	Explanation	How to tell when exploring (u, v)?
Tree edge	edge in $\hat{E}$	
Back edge	connects u to ancestor v	
Forward edge	connects vertex u to descendant v	*and* $u.D < v.D$
Cross edge	either (a) connects two different trees or (b) crosses between siblings/cousins in same tree	*and* $u.D > v.D$

5 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*