

# Things you should know!

- Today's attendance is a sign-in sheet!
- Midterms are next week!



# Midterms:

## ● Format

- 30 minute technical phone interview
  - 1 Behavioral Question
  - 1 Technical Question
  - Google Docs/Google Hangouts

## ● Sign ups

- [ter.ps/fen](https://ter.ps/fen)
- Once 9 people have signed up for an instructor, all of that instructor's time slots are closed
  - Check before you sign up!

## ● Rubric

- [ter.ps/feo](https://ter.ps/feo)

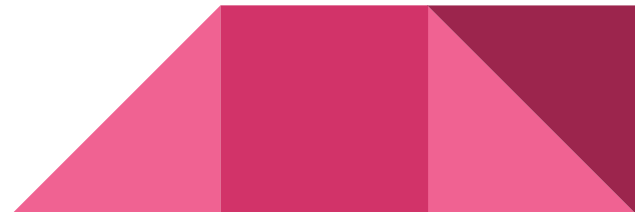


# Week 7!

CMSC3890: The Coding Interview

# Today

- Trees
- Graphs



# Tree Traversals

- Pre-order

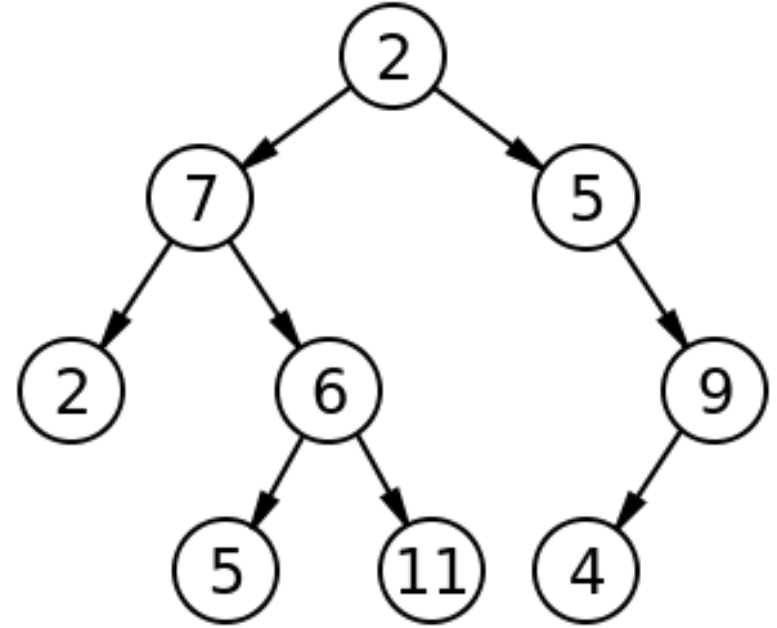
- 2 7 2 6 5 11 5 9 4

- In-order

- 2 7 5 6 11 2 5 4 9

- Post-order

- 2 5 11 6 7 4 9 5 2



# Breadth-First Search

Given a Graph of Nodes, graph, and a starting node, startNode:

```
Queue<Node> queue = new LinkedList<Node>();
```

```
Set<Node> visited = new HashSet<Node>();
```

```
queue.add(startNode);
```

```
while(!queue.isEmpty()) {
```

```
    Node node = queue.remove();
```

```
    process(node);
```

```
    for (Node n : graph.getNeighbors(node)) {
```

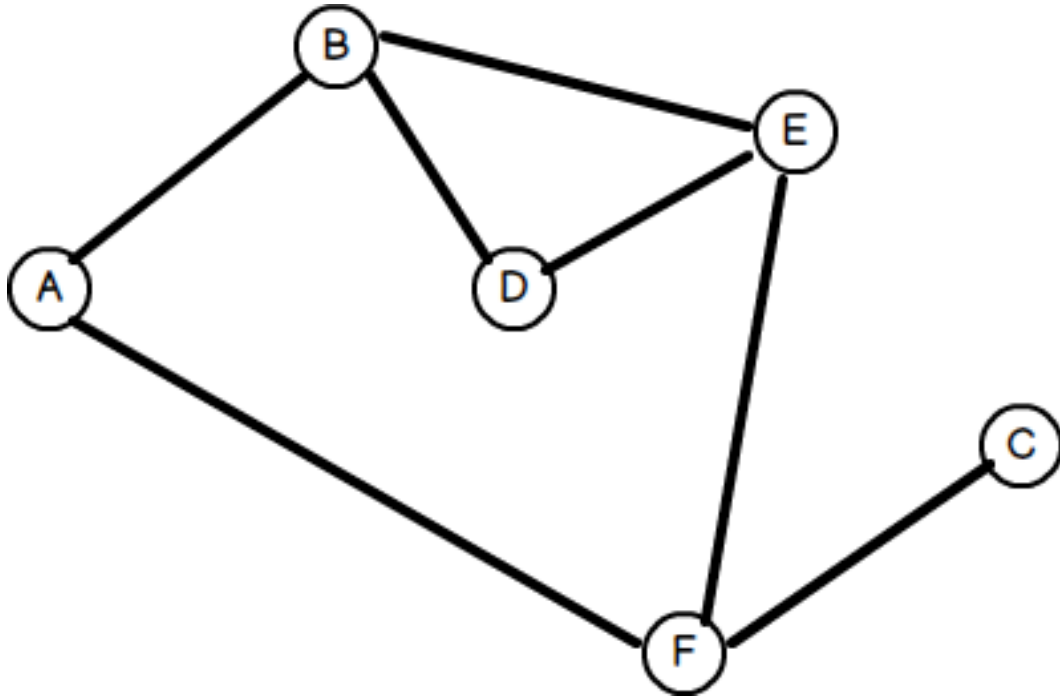
```
        if ( !visited.contains(n) ) queue.add(n);
```

```
    }
```

```
}
```




# Let's Run Through An Example!



# Depth-First Search

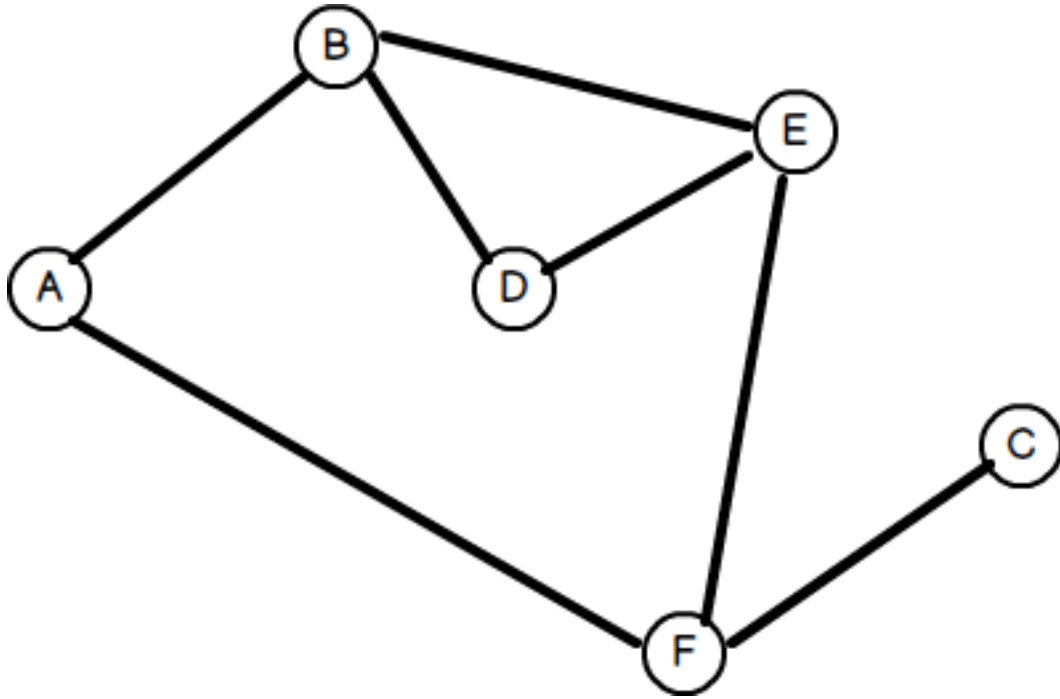
Given a Graph of Nodes, graph, and a starting node, startNode:

```
Stack<Node> stack = new Stack<Node>();  
Set<Node> visited = new HashSet<Node>();  
stack.push(startNode);  
while(!stack.isEmpty()) {  
    Node node = stack.pop();  
    process(node);  
    for (Node n : graph.getNeighbors(node)) {  
        if ( !visited.contains(n) ) stack.push(n);  
    }  
}
```



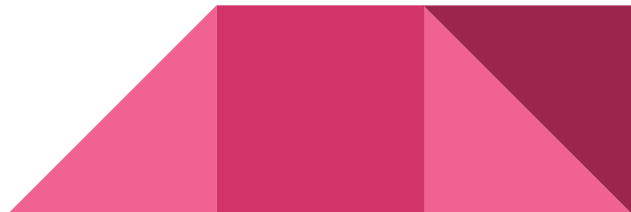


# Let's Run Through An Example!



# When is DFS and BFS useful?

- Can be useful when you need to search within a tree or graph
- TIP: exploit the process(node) part of DFS/BFS



Given a 2d grid map of '1' s (land) and '0' s (water), count the number of islands. An island is surrounded by water and is formed by connecting adjacent lands horizontally or vertically. You may assume all four edges of the grid are all surrounded by water.

**Example 1:**

```
11110
11010
11000
00000
```

Answer: 1

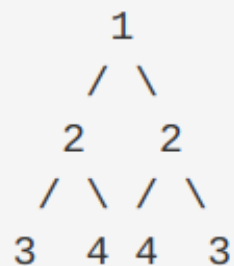
**Example 2:**

```
11000
11000
00100
00011
```

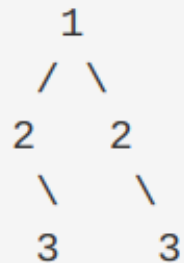
Answer: 3

Given a binary tree, check whether it is a mirror of itself (ie, symmetric around its center).

For example, this binary tree `[1, 2, 2, 3, 4, 4, 3]` is symmetric:



But the following `[1, 2, 2, null, 3, null, 3]` is not:



# Dijkstra's Algorithm

```
dijkstra(G,w,s) {  
    for each (u in V) {                                // initialization  
        d[u] = +infinity  
        mark[u] = undiscovered  
        pred[u] = null  
    }  
    d[s] = 0                                            // distance to source is 0  
    Q = a priority queue of all vertices u sorted by d[u]  
    while (Q is nonEmpty) {                            // until all vertices processed  
        u = extract vertex with minimum d[u] from Q  
        for each (v in Adj[u]) {                      // relax all outgoing edges from u  
            if (d[u] + w(u,v) < d[v]) {  
                d[v] = d[u] + w(u,v)  
                decrease v's key in Q to d[v]  
                pred[v] = u  
            }  
        }  
        mark[u] = finished  
    }  
    [The pred pointers define an "inverted" shortest path tree]  
}
```

# Dijkstra's Algorithm

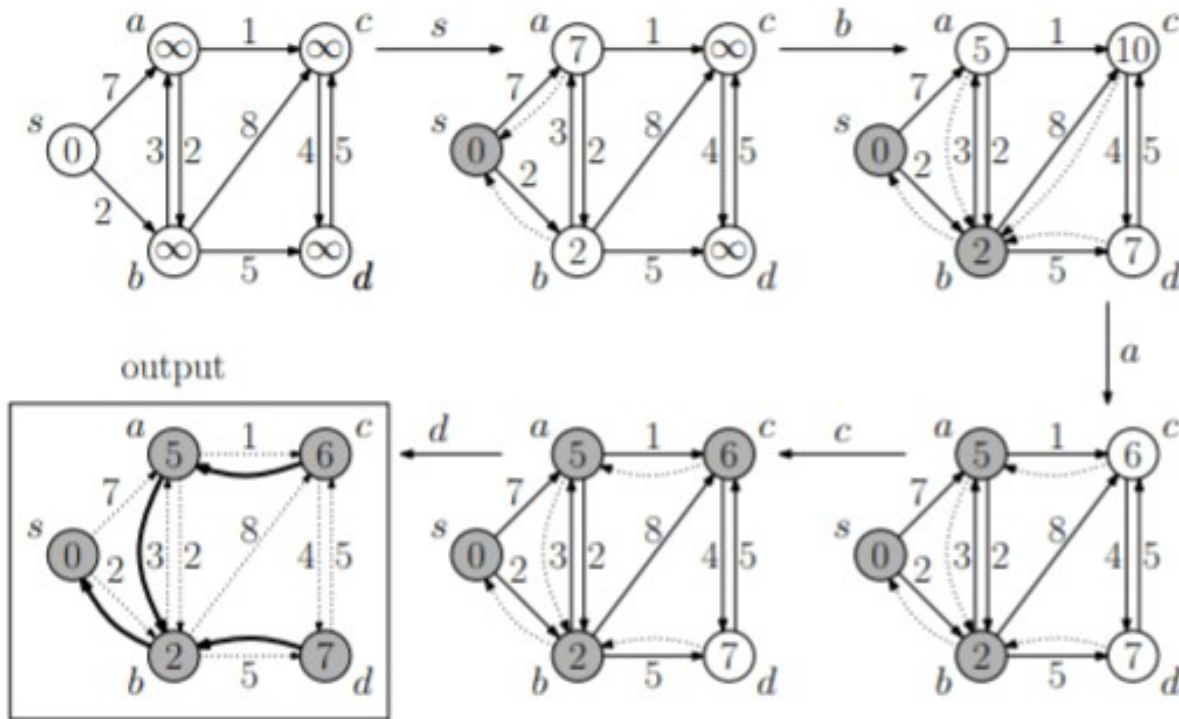



Fig. 2: Dijkstra's Algorithm example.

# Heaps

```
PriorityQueue<Integer> PQ = new PriorityQueue<Integer>(1000, new  
Comparator<Integer>() {  
    public int compare(Integer num1, Integer num2) {  
        if (num1 > num2)  
            return 1;  
        else if (num1 < num2)  
            return -1;  
        return 0;  
    }  
});
```

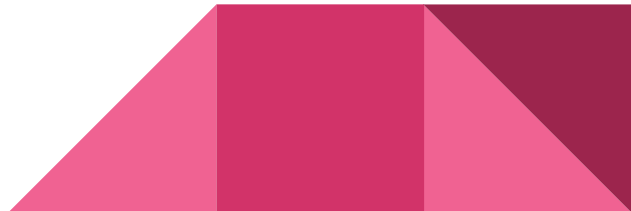
If you choose to use Python... good for you. (import minheap)  
C on the other hand...



# Dictionaries:

Problem: You have a set of words and need to determine if a word is within your set.

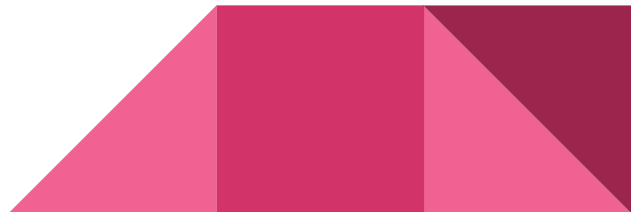
- What data structure should you use to store your dictionary?
- What is the time/space complexity for
  - Adding a word
  - Deleting a word
  - Searching for a word
  - Space complexity for your dictionary as a whole



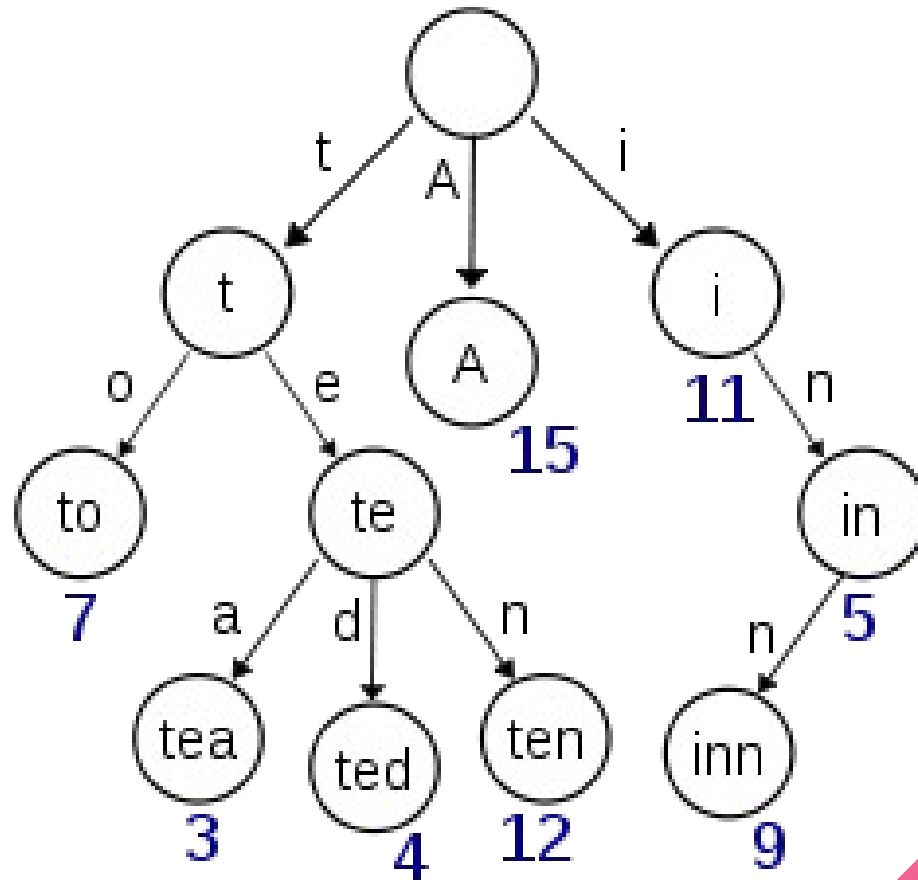


# Tries!

- Ordered tree data structure that is used to store a dynamic sector associative array where the keys are usually strings
- Complexities
  - $O(L)$  to insert
  - $O(L)$  to search

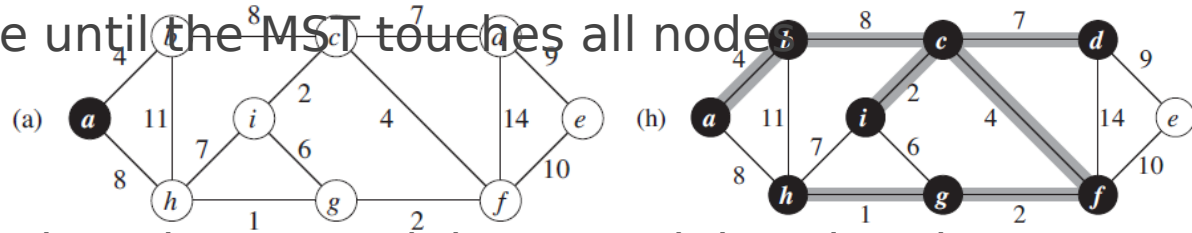


# Tries

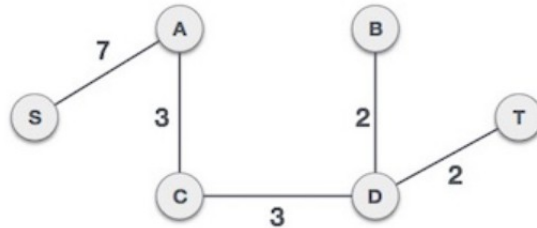
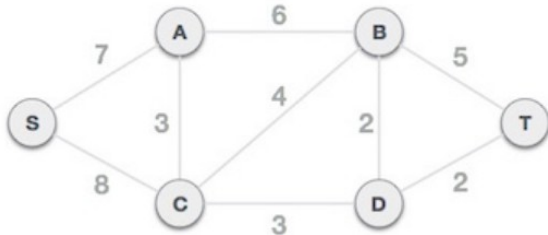


# Minimum Spanning Trees

- Prim's Algorithm: start with a randomly selected start node and select the minimum-weight edge touching it. select the next minimum-weight edge until the MST touches all nodes



- Kruskal's Algorithm: select the next minimum-weight edge that does not cause a circuit until the MST touches all nodes



# Serializing and Deserializing

Let's figure this out together!

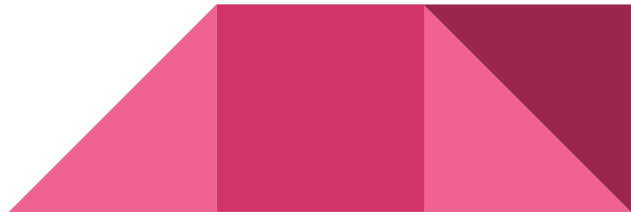
What do we mean by Serializing and Deserializing?

Storing and restoring a tree in a file or an array.

How can we store this information?

Think of traversals. Think of the type of tree it is. Is it a BST?

Complete Tree? Full Tree?



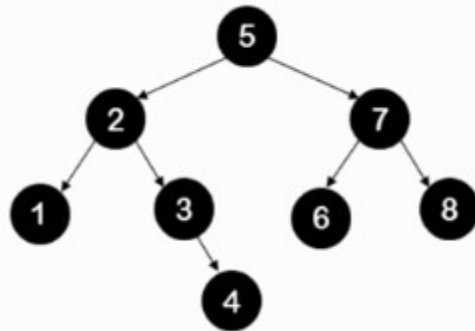
# Serializing a BST

Think of Pre-Order Traversal:

5, 2, 1, -1, -1, 3, -1, 4, 7, 6, -1, -1, 8, -1, -1

The '-1's mean that it is a NULL value! (If we are using negatives and positives, pick another character.)

Traverse the list in-order. If it does not have any children, put in a -1 as the base case. This creates the list as above.



# Deserialize a BST

Keep recursively building the tree to the left until you get to a '-1' value. Then, it undoes itself in its recursion, and goes to the right, doing the same building step as before.

This keeps going no matter what, since the tree building is independent of the other branches.



# Reminders

- Fill out feedback form at [ter.ps/ffe](https://ter.ps/ffe) !
- Send us your photos from the career fair for extra credit!

