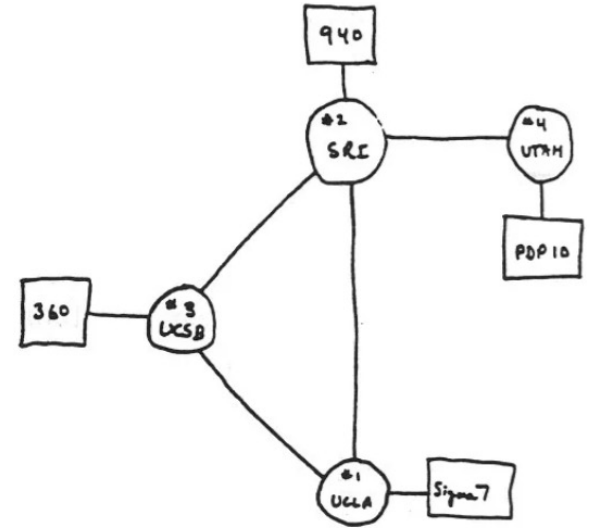
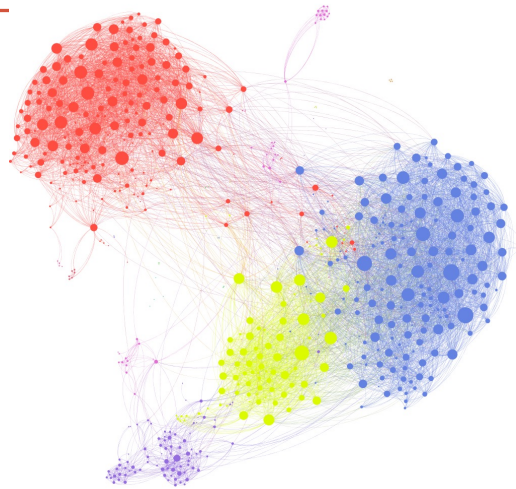
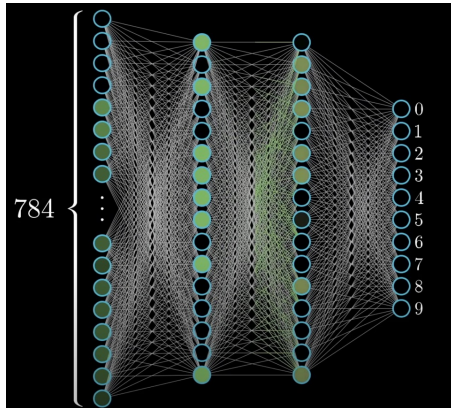


<https://bit.ly/CS24-W24-Graph-Search-Complexity-Handout>

COMPLEXITY ANALYSIS GRAPH SEARCH ALGO



THE ARPA NETWORK

DEC 1969

4 NODES

BFS Traverse: Time Complexity

$$T(n) = O(n) + O(1) + \underset{\text{pops}}{O(n)} + \underset{\text{w. visited check}}{O(m)} + \underset{\text{w. visited = true}}{O(n)} + \underset{\text{push to queue}}{O(n)} = O(n+m)$$

Algo exploreBFS(v):

Mark all the vertices as "not visited" $O(n)$

v.visited ← true $O(1)$

Push v into a queue $O(1)$

While queue is not empty:

→ Pop the vertex from the front of the queue (v) $O(n)$

For each edge (v,w)

If not w.visited → $O(m)$

w.visited ← true → $O(n)$

Push w into the queue → $O(n)$

Over all iterations of while loop

Total

n: number of vertices
m: number of edges

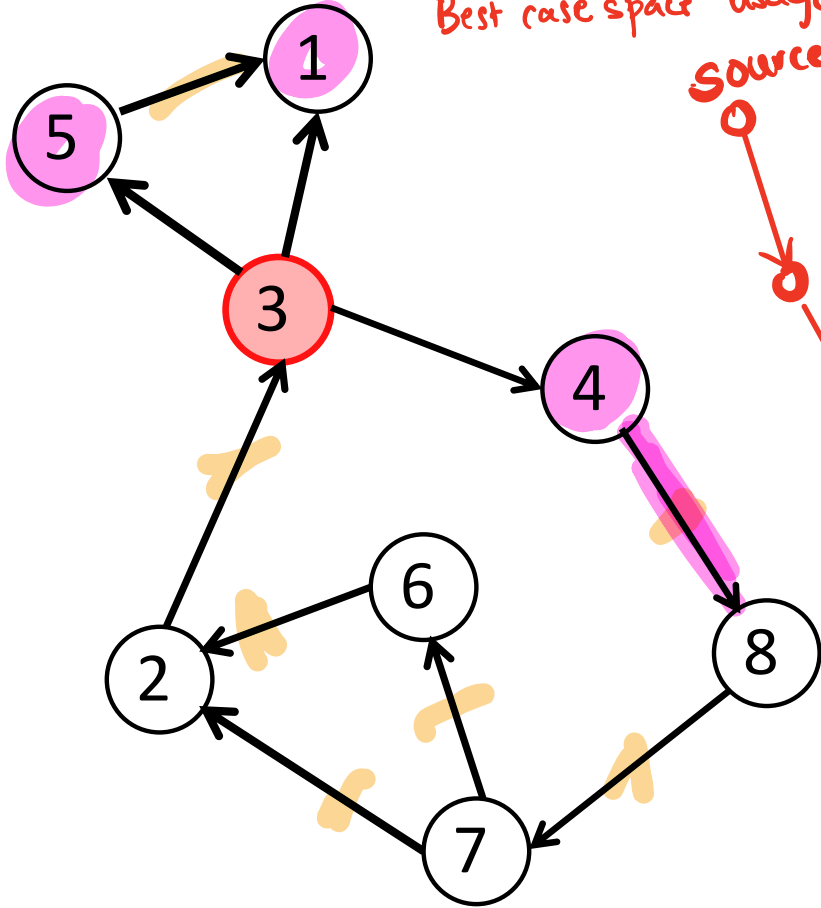
What is the time complexity of exploreBFS?

- A. $O(n)$
- B. $O(m)$
- C. $O(n + m)$**
- D. $O(n^2)$
- E. None of the above

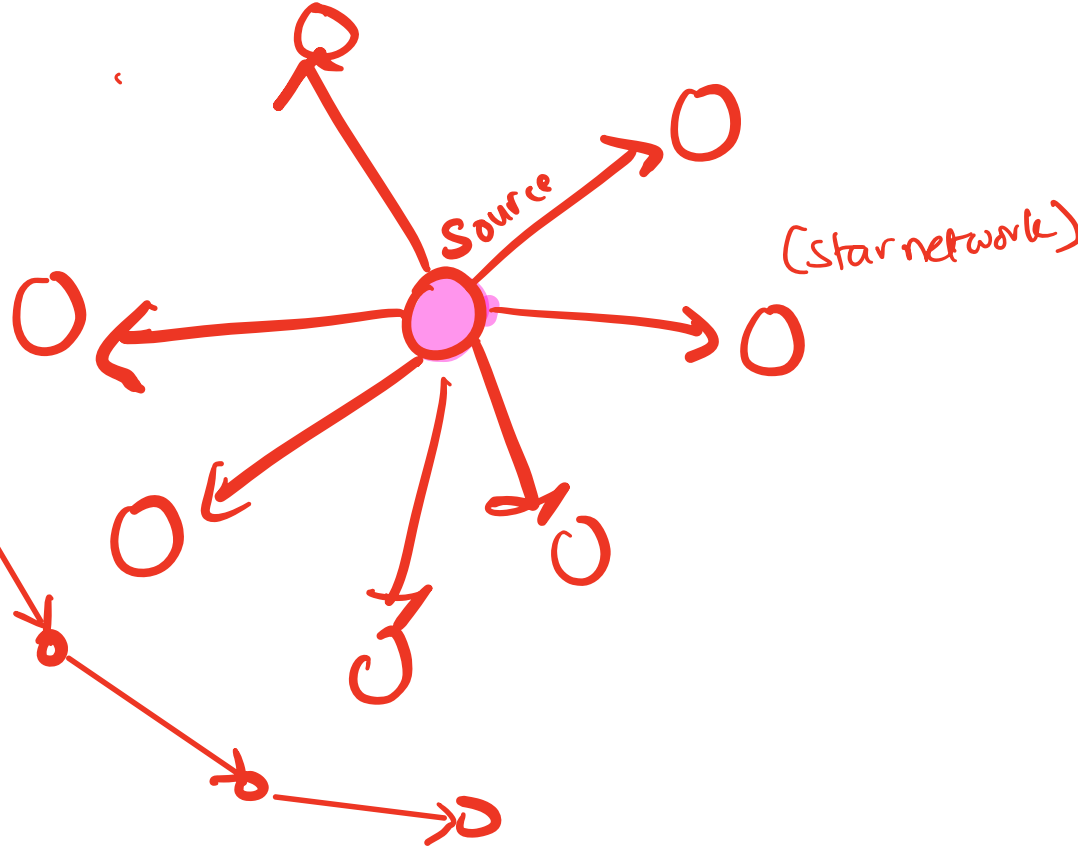
Breadth First –The Game

Best case space usage of BFS: $O(1)$

worst case space usage of BFS: $O(n)$



source



BFS Traverse: Space Complexity

Algo exploreBFS(v):

Mark all the vertices as “not visited”

$v.visited \leftarrow true$

Push v into a queue

While queue is not empty:

Pop the vertex from the front of the queue (v)

For each edge (v,w)

If not $w.visited$

$w.visited \leftarrow true$

Push w into the queue

If space used to store 'visited' flag is part of input, best case auxiliary space complexity is $O(1)$
(star network)



n : number of vertices

m : number of edges

What is the Big -O auxiliary space complexity of exploreBFS?

A. $O(n)$ - worst case

B. $O(m)$

C. $O(n + m)$

D. $O(n^2)$

E. None of the above

- Auxiliary Space complexity: Additional space usage (not including input and output)

exploreDFS: Time Complexity

```
exploreDFS (v)
```

```
v.visited ← true
```

```
For each edge (v, w)
```

```
If not w.visited
```

```
    exploreDFS (w)
```

Total

$O(n)$

$O(m)$

$O(n)$

n: number of vertices

m: number of edges

$$T(n) = O(n) + O(m) + O(n) \\ = O(n+m)$$

What is the time complexity of exploreDFS?

A. $O(n)$

B. $O(m)$

C. $O(n + m)$

D. $O(n^2)$

E. None of the above

Depth First Search: Time Complexity

```
DepthFirstSearch(G)
```

```
  Mark all  $v \in G$  as unvisited
```

```
  For  $v \in G$ 
```

```
    If not  $v.visited$ , (exploreDFS(v))
```

\downarrow
 $O(n+m)$

n : number of vertices
 m : number of edges

What is the time complexity of Depth First Search?

A. $O(n)$

B. $O(m)$


C. $O(n + m)$

D. $O(n^2)$

E. None of the above

Best case space complexity for DFS - $O(1)$ (assuming visited field is part of input) Worst case $O(n)$

exploreDFS: Space Complexity



```
exploreDFS (v)
```

```
  v.visited ← true
```

```
  For each edge (v, w)
```

```
    If not w.visited
```

```
      exploreDFS (w)
```

n: number of vertices
m: number of edges

What is the worst-case space complexity of exploreDFS?

- A. $O(n)$
- B. $O(m)$
- C. $O(n + m)$
- D. $O(n^2 + n.m)$
- E. None of the above



Leetcode: Lowest Common Ancestor (LCA) of a binary tree

What is the LCA of each of the following?

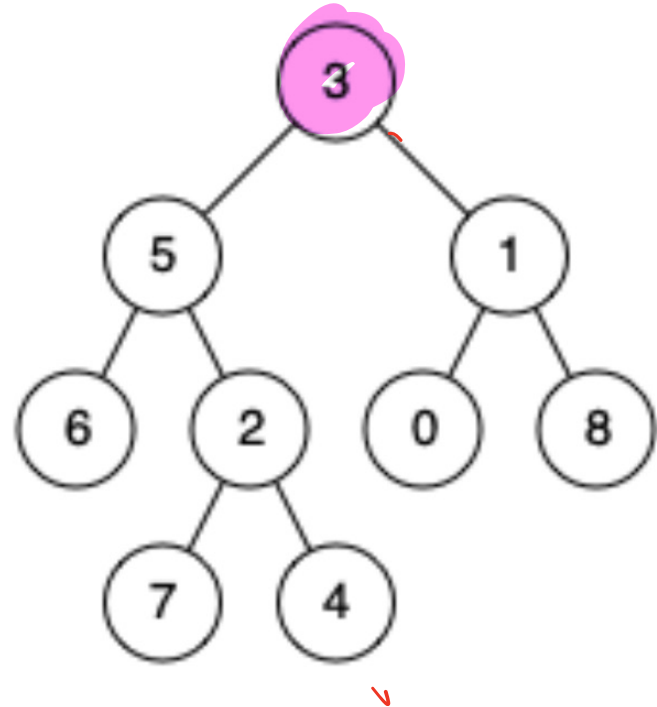
5 and 1: 3

5 and 4: 5

3 and 1: 3

6 and 8: 3

Discuss how you would solve the problem with your neighbor, implement in leetcode to check your answer



Implement these functions from last week's handout

```
class graph{
public:
    graph(int n = 0) { // n is the number of vertices
        _____
    }
    A void addEdge(int from, int to); bool hasEdge(int i, int j) const;
    B vector<bool> bfs(int source) const;
    C bool isValidPath(const vector<int> & path) const; // returns true if the input path exists
    D bool isReachable(int source, int dest) const; // returns true if a path exists from source to dest
private:
    vector<_____> adjList;
};
```

