

# ECE2810J

## Data Structures and Algorithms

### **Graph Search**

#### **Learning Objectives:**

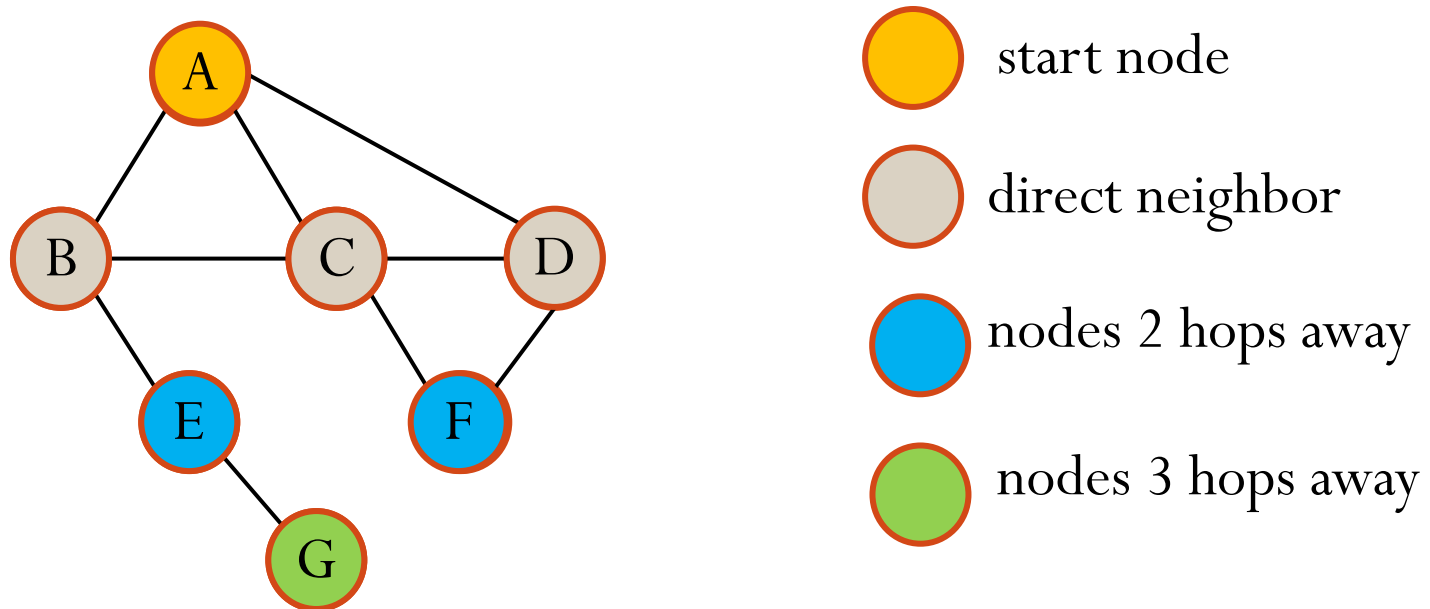
- Know two widely-used graph search algorithms, breadth-first search and depth-first search
- Know their runtime complexity

# Graph Search

- A node  $u$  is **reachable** from a node  $v$  if and only if there is a path from  $v$  to  $u$ .
- A graph search method starts at a given node  $v$  and visits **every** node that is **reachable** from  $v$  **exactly once**.
- Many graph problems are solved using a search method.
  - Find a path from one node to another.
  - Find if the graph is connected.
- Commonly used search methods:
  - Breadth-first search.
  - Depth-first search.

# Breadth-First Search (BFS)

- Given a start node, visit all directly connected neighbors first, then nodes 2 hops away, 3 hops away, and so on.



$A \rightarrow B \rightarrow C \rightarrow D \rightarrow E \rightarrow F \rightarrow G$

# Breadth-First Search (BFS)

## Implementation

- BFS can be implemented using a queue.

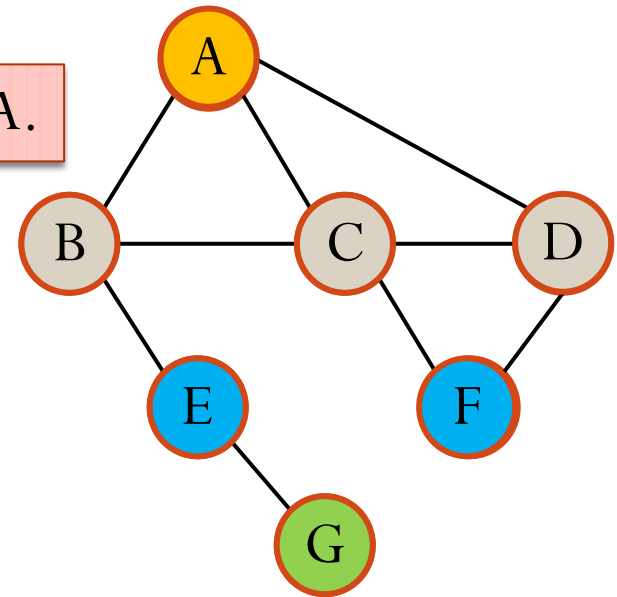
```
BFS(s) {  
    queue q; // An empty queue  
    visit s and mark s as visited;  
    q.enqueue(s);  
    while(!q.isEmpty()) {  
        v = q.dequeue();  
        for(each node u adjacent to v) {  
            if(u is not visited) {  
                visit u and mark u as visited;  
                q.enqueue(u);  
            }  
        }  
    }  
}
```

# Breadth-First Search (BFS)

## Example

Start node is node A.

```
BFS(s) {  
  queue q; // An empty queue  
  visit s and mark s as visited;  
  q.enqueue(s);  
  while(!q.isEmpty()) {  
    v = q.dequeue();  
    for(each node u adjacent to v) {  
      if(u is not visited) {  
        visit u and mark u as visited;  
        q.enqueue(u);  
      }  
    }  
  }  
}
```



Queue: 

A	B	C	D	E	F	G
---	---	---	---	---	---	---

Visit Order: A B C D E F G

# Breadth-First Search (BFS)

## Time Complexity

- If graph is implemented as **adjacency matrix**:
  - Visit each node exactly once:  $O(V)$ .
  - The row of each node in the adjacency matrix is scanned once:  $O(|V|)$  for each node.
  - Total running time:  $O(|V|^2)$ .
- If graph is implemented as **adjacency list**:
  - Visit each node exactly once:  $O(|V|)$ .
  - Adjacency list of each node is scanned once.
  - Size of entire adjacency list is  $2|E|$  for undirected graph and  $|E|$  for directed graph.
  - Total running time:  $O(|V| + |E|)$ .

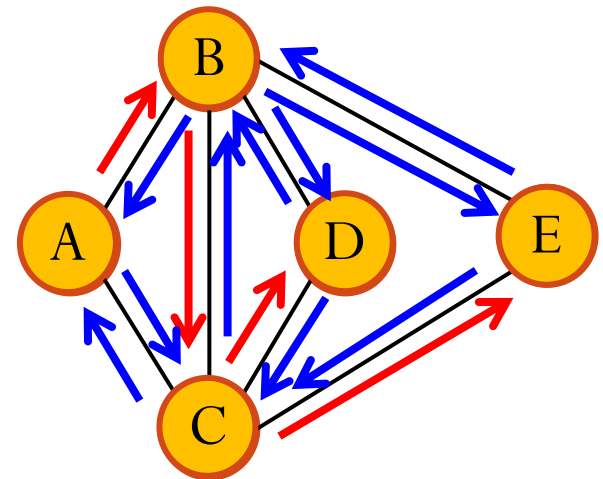
# Depth-First Search (DFS)

```
DFS(v) {  
    visit v;  
    mark v as visited;  
    for(each node u adjacent to v)  
        if(u is not visited) DFS(u);  
}
```

- How to mark a node “visited”?
  - Keep a “visited” field in the node

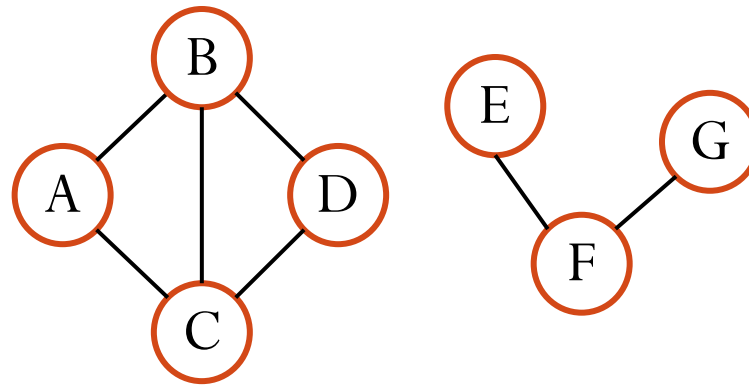
Time complexity?

Same as BFS



# Traverse All the Nodes in a Graph

- The graph may not be connected. How can we traverse all the nodes in the graph?



```
for(each node v in the graph)
  if(v is not visited)
    DFS(v) ;
```