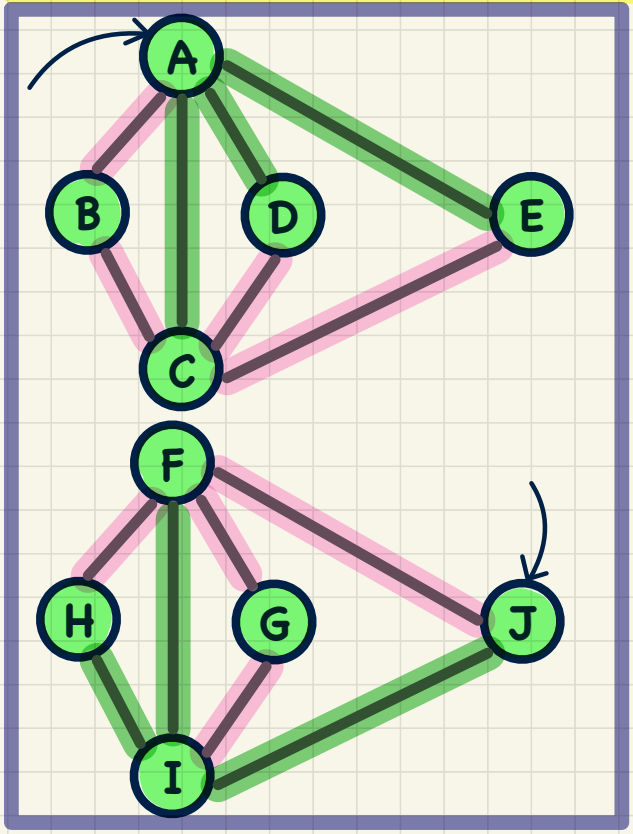
## Test 1 (WSC, 4:30 PM to 5:20 PM)

- **Coverage**
  - + Lecture materials (slides, notes, example code)
    up to and including Monday, October 20
  - + Tutorials 1 to 4
  - + Assignment 1
- **Format**
  - + **Programming** Part (Eclipse):
    - * Import a Java starter project (like A1)
    - * Implement Java classes/methods to pass test cases
  - + **Written** Part (eClass):
    - * Primarily MCQs
    - * Written questions (e.g., short answers, justifications, proofs)

# Graph Traversals: Adapting DFS

**Efficient Traversal of Graph G:**



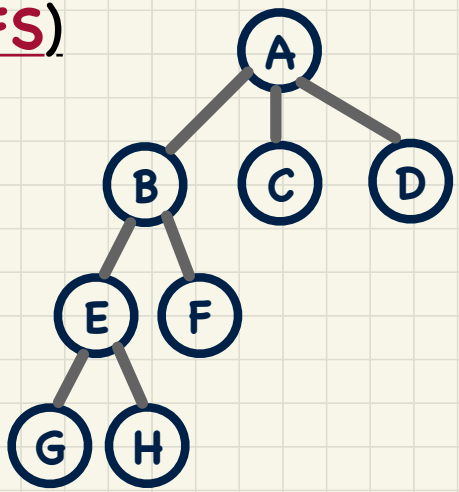

**Graph Questions:**

- Fina a **path** between {u, v} ⊆ V
- Is v **reachable** from v
- Find **all** **connected components** of G.
- Compute a **spanning tree** of a connected G.
- Is G **connected**?
- If G is cyclic, return a **cycle**.

# Graph Traversal: Breadth-First Search (BFS)

A ***breadth-first search (BFS)*** of graph G = (V, E), starting from some vertex $v \in V$:

- Visits every vertex ***adjacent*** to $v$ before visiting any other (**more distant**) vertices
  - ***BFS*** attempts to stay as **close** as possible, whereas ***DFS*** attempts to move as **far** as possible
  - ***BFS*** proceeds in rounds and divides the vertices into ***levels***
- **No backtracking** in ***BFS***: it is completed **as soon as** the **most distant level** of vertices from the start vertex $v$ are visited.




Q. What data structure should be used to keep track of the visited nodes?

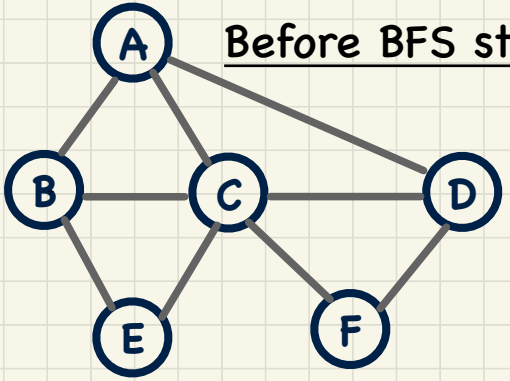
# Breadth-First Search (BFS): Marking Vertices & Edges

Before the ***BFS*** starts:

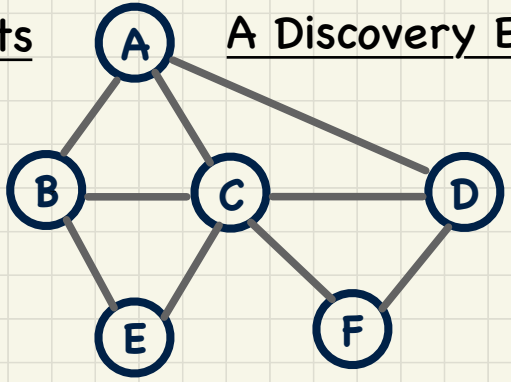
- All vertices are ***unvisited***.
- All edges are **unexplored/unmarked**.

Over the course of a ***BFS***, we **mark** vertices and edges:

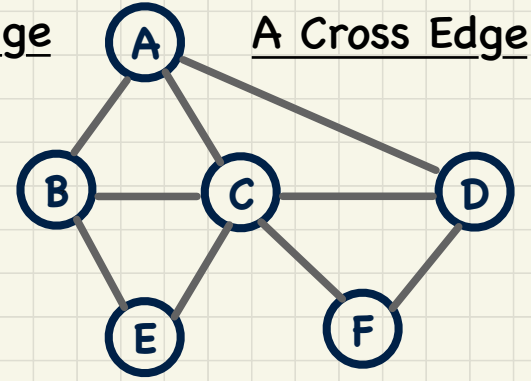
- A vertex is marked ***visited*** when it is **first** encountered.
- Then, we iterate through each of v's **incident edges**, say *e*:
  - If edge *e* is already **marked**, then skip it.
  - Otherwise, for an *undirected* graph, an edge is marked as:
    - A ***discovery*** edge if it leads to an ***unvisited*** vertex
    - A ***cross*** edge if it leads to a ***visited*** vertex
      (i.e., from a different ***branch*** at the same ***level***).




Before BFS starts

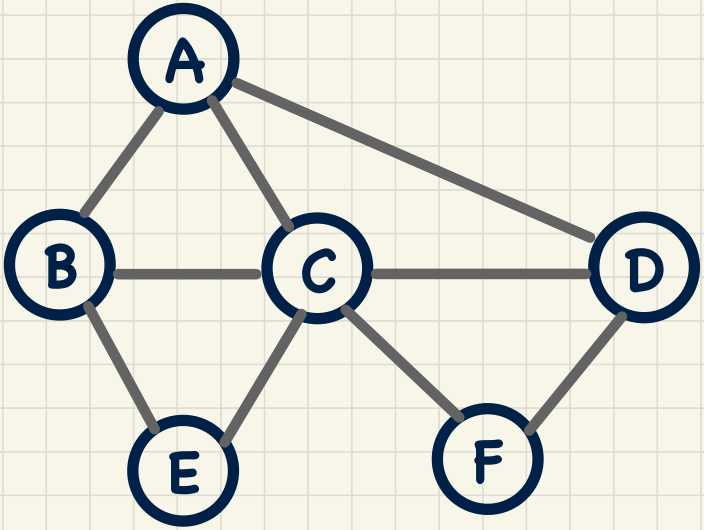



A Discovery Edge




A Cross Edge

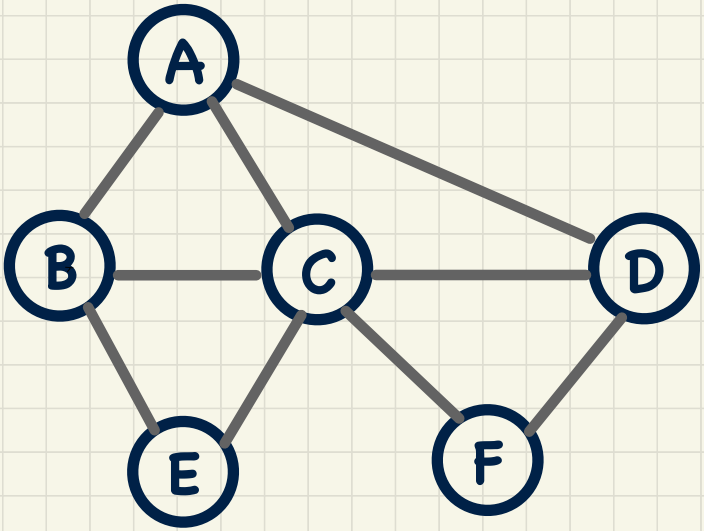
# Breadth-First Search (BFS): Example 1 (a)




Assumptions:

- Adjacent vertices visited in alphabetic order

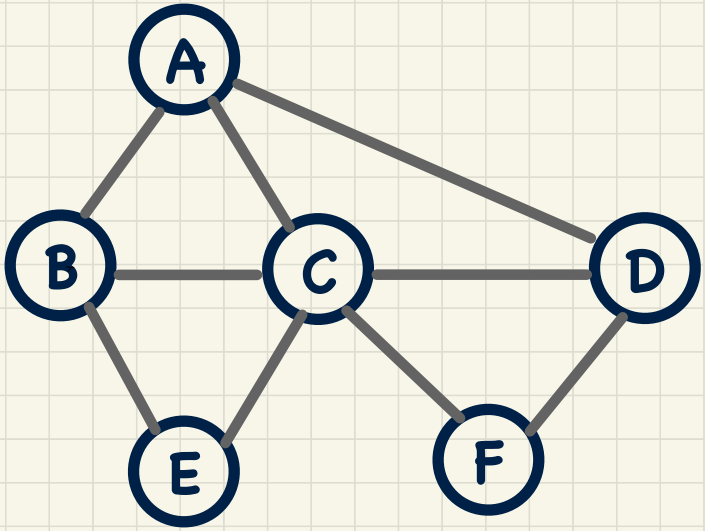
# Breadth-First Search (BFS): Example 1 (b)




Assumptions:

- Adjacent vertices visited in alphabetic order
- Exception: Edge AC visited first

# Breadth-First Search (BFS): Example 1 (c)

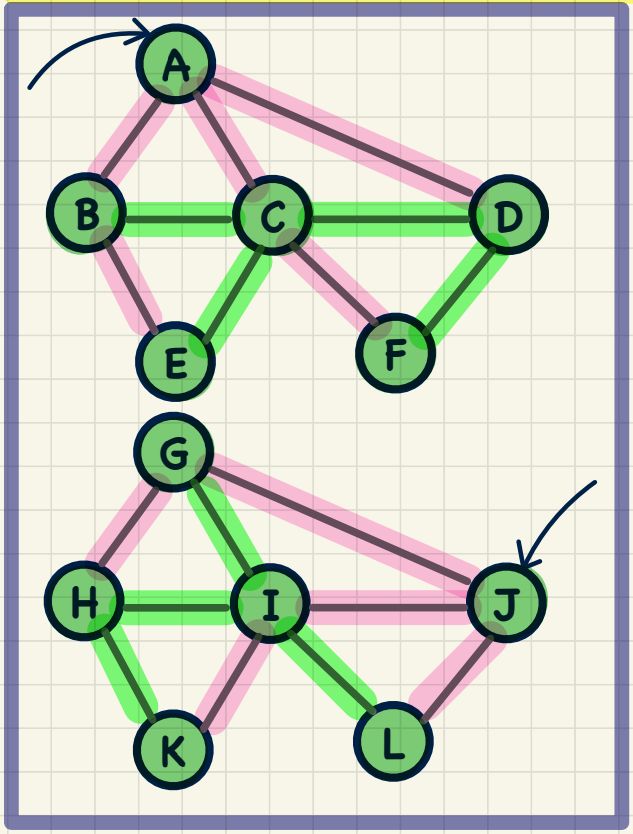



Assumptions:

- Adjacent vertices visited in alphabetic order
- Exception: Edge AD visited first

# Graph Traversals: Adapting BFS

**Efficient Traversal of Graph G:**




Graph Questions:

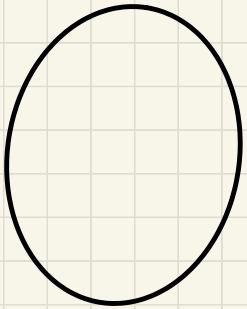
- Fina a **path** between {u, v} ⊆ V
- Is v **reachable** from v
- Find **all** **connected components** of G.
- Compute a **spanning tree** of a connected G.
- Is G **connected**?
- If G is cyclic, return a **cycle**.

# Big-O Properties (1): Members in a Family

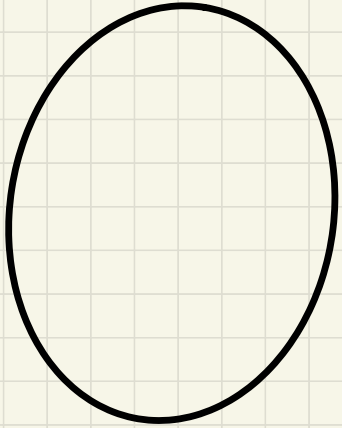
Each member f(n) in O( g(n) ) is such that:

Higest Power of f(n) <= Highest Power of g(n)

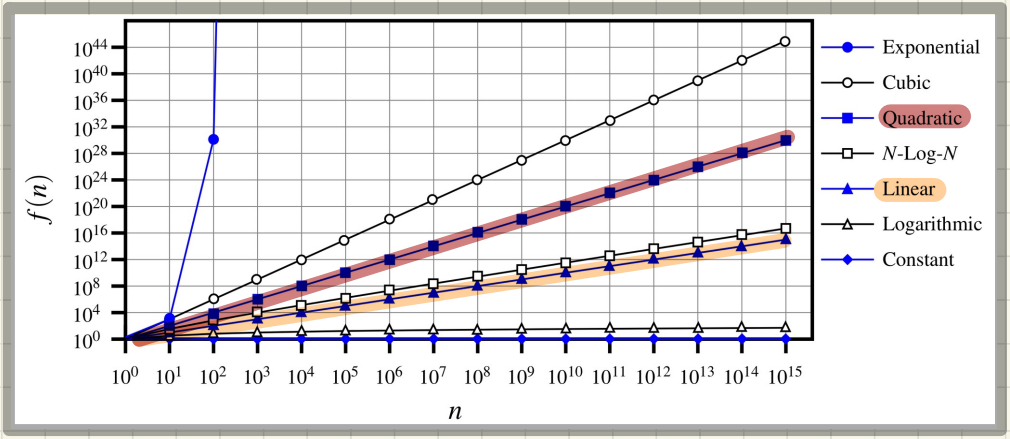
O(n)



O(n²)



Functions: Rates of Growth



# Big-O Properties (2): Relating Families

# Big-O Properties (3): Deciding Correct & Accurate Bound