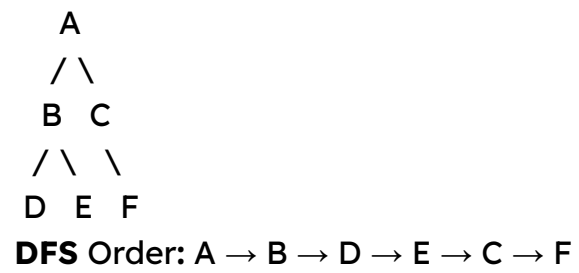


Recitation Recap

Concept	DFS (Depth-First Search)	BFS (Breadth-First Search)
Goal	Explore as <i>deeply</i> as possible before backtracking	Explore <i>level by level</i> (nearest neighbors first)
Data Structure	Stack (explicit or recursive call stack)	Queue
Traversal Order	"Dive down a path" — go deep, then backtrack	"Wave outward" — visit all neighbors before moving deeper
When to Use	Searching paths, checking connectivity, and topological sorting	Finding shortest paths (in unweighted graphs), minimum moves
Key Feature	Goes deep before wide	Goes wide before deep
Time Complexity (not relevant right now)	$O(V + E)$	$O(V + E)$
Space Complexity (not relevant right now)	$O(V)$ (recursive stack)	$O(V)$ (queue)

Walkthrough of DFS and BFS



BFS Order: $A \rightarrow B \rightarrow C \rightarrow D \rightarrow E \rightarrow F$

Dijkstra = tells us the shortest distance between one node to every other node

1. First, set all distances to infinity except the source node
2. Then update the chart with corresponding costs that are immediately reachable from the current node.
3. From here, we go to the vertex with the smallest path weight - make sure we don't go to a node we already visited!
4. Repeat until all nodes are visited

Questions

1. What problem does Dijkstra's algorithm solve?
2. **True or False:**
 - (a) Dijkstra's algorithm can handle negative edge weights.
 - (b) Dijkstra's algorithm always finds the shortest path from one node to all others.
 - (c) Dijkstra's algorithm stops as soon as all nodes are visited.
3. Arrange these steps in the correct order:
 - Update the distances of all neighboring nodes.
 - Pick the unvisited node with the smallest distance.
 - Mark the current node as visited.
 - Initialize all distances to infinity (except the start node).

Graph = always defined by vertices, edges, and weights:

Edge Weight

A-B	2
A-C	5
B-C	1
B-D	3
C-D	2

Task:

Run Dijkstra's algorithm from node **A**.

Fill in a table like this:

Node	Minimum Distance / Path Weight from A	Visited?
A		
B		
C		
D		

- What is the shortest distance from $A \rightarrow D$?
- Which path gives that distance

Multiple choice:

1. Dijkstra's algorithm can't be used if:
 - a) There are negative edge weights
 - b) The graph is directed
 - c) The graph has cycles
 - d) All edges have equal weight
2. In Dijkstra's algorithm, if the shortest path to node X is found to be 7, and there's another path that gives cost 9 later, what happens?
 - a) We update it to 9
 - b) We ignore it
 - c) We add it to a queue to re-check
 - d) We restart the algorithm
3. The data structure most commonly used for selecting the next node with the smallest distance is:
 - a) Stack
 - b) Queue
 - c) Priority Queue
 - d) Hash Map

Extra Questions (might not get to, but good to think about)

1. Suppose we're halfway through Dijkstra's algorithm and a node X is marked visited. Why is it guaranteed that its shortest distance is finalized?
2. How does Dijkstra's algorithm differ from BFS when all edge weights are 1?