Graph Algorithms

- انواع الگوریتم های گراف

Part
1. Graph Traversal Algorithms
   1.1 Breadth-First Search (BFS)
   1.2 Depth-First Search (DFS)

Part
2. Shortest Path Algorithms
   2.1 Dijkstra's Algorithm
   2.2 Bell-man Ford Algorithm
   2.3 Floyd-Warshall Algorithm
   2.4 A* Search Algorithm

Part
3. Cycle Detection
   3.1 UnDirected Graph
   3.2 Directed Graph

بخش اول

What is Graph Traversal?
Graph Traversal means visiting all the vertices (and possibly edges) of a graph in a systematic order.

تاريخ: / /

موضوع:

We need traversal when:

* We want to check if two nodes are connected.

* We want to find the shortest path in an unweighted graph.

* We need to process all components.

* We want to explore a structure (e.g., a maze, social network, file system).

There are two main strategies

* Breadth-First Search (BFS)
  Explore layer by layer

* Depth-First Search (DFS)
  Dive deep before backtracking

موضوع: تاريخ: / /

1. Breadth-First Search (BFS)

* Intuition:

Imagine throwing a stone into a pond. Waves ripple outward in all directions. That is how BFS explores a graph — level by level.

* How it works? (Step-by-Step)

1. Start from a given node.
2. Use a queue to keep track of nodes to visit next.
3. Visit all neighbors of the current node
4. Add unvisited neighbors to the queue.
5. Repeat until the queue is empty.

Pseudo Code ↓

Next page.

BFS (graph, start):
Create an empty queue Q
Mark start as visited.
Enqueue start into Q

while Q is not empty:
current = Q.dequeue()
Process current

for neighbor in graph[current]:
if neighbor is not visited:
Mark neighbor as visited
Q.enqueue(neighbor)

* Time and Space Complexity

Case	Time Complexity	Space Complexity
All graphs	$O(V+E)$	

V → number of vertices

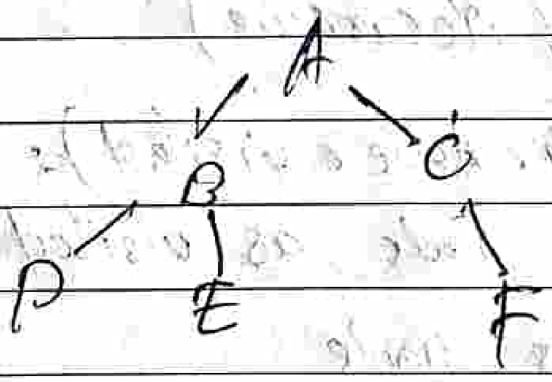
E → Number of Edges

Applications

- Shortest path in an unweighted graph (in maps).
- Level-order Traversal
- Finding connected components
- Web crawlers
- Peer-to-peer networks (e.g. BitTorrent).

Visual Example

Consider the below graph:



BFS from A visited A → B → C → D → E → F

2. Depth-First Search (DFS)

Intuition: Imagine exploring a maze. You go as far as you can in one direction before backtracking. That's DFS — it goes deep first.

التاريخ: موضوع:

* How it works?

1. Start at the root node.
2. Go to the first unvisited neighbor.
3. Continue deeper until there's nowhere to go
4. Backtrack and try another path.

It can be implemented in two ways:

* Recursively
* Iteratively (using a Stack)

Pseudo code (Recursive)

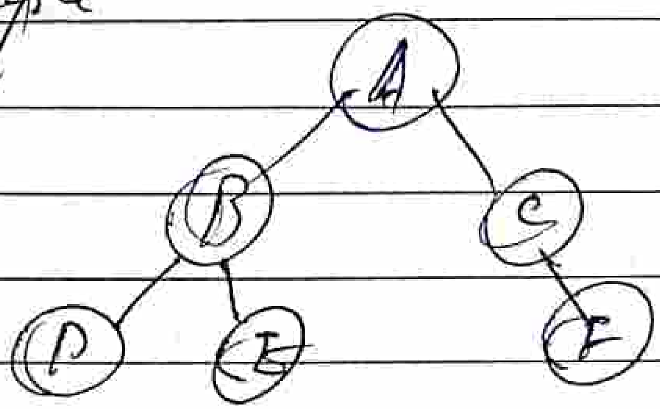
```
DFS(graph, node, visited):
    Mark node as visited
    Process node

    for neighbor in graph[node]:
        if neighbor is not visited:
            DFS(graph, neighbor, visited)
```

* Applications
- Detecting Cycles in graphs.

- Topological Sorting in DAGs
- Maze Solving
- Generating mazes
- Connected component detection

A Visual Example



DFS from A might visit: A → B → D → E → F → C

(depends on graph structure)

BFS vs DFS Comparison

Feature	BFS	DFS
Structure used	Queue	Stack (or recursion)
Path finding	Yes (shortest in unweighted)	Not guaranteed
Memory usage	More in wide graphs	More in deep graphs
Complete search	yes	yes
Cycle detection	yes	

Shortest Path Algorithms

I. Dijkstra's Algorithm:

Use case: Graphs with non-negative edge width only.