T.Y. CSF V Sem.

ARTIFICIAL INTELLIGENCE & MACHINE LEARNING TECHNIQUES

ASSIGNMENT - 1

Breadth First Search & Depth First search

Aim: Write a program to implement BFS & DFS

OBJECTIVE: To study BFS and DFS. To understand their working.

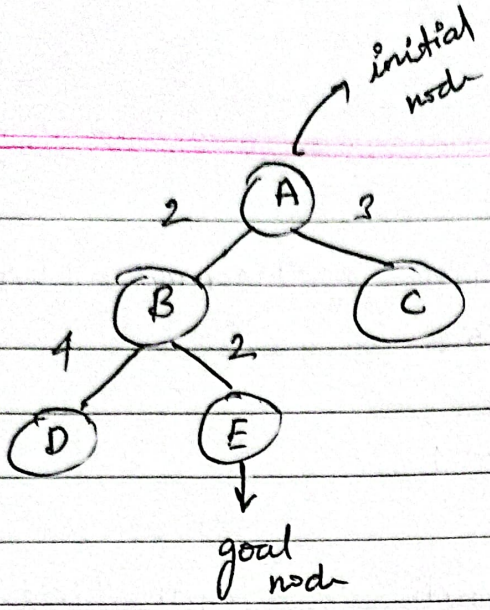
Theory: (1) Breadth First Search

Def: It is a graph traversal algorithm that explores all the vertices of a graph level by level, starting from a chosen start vertex.

→ It starts from the source vertex, explores its neighbours, then moves to their unvisited neighbours and so on, until all reachable vertices are visited.

→ It uses a queue to keep track of vertices to visit. The source is enqueued, it visited and its neighbours are enqueued, to be processed later.

eg. -1



BFS :

queue: | A | B | C | D | E |

SB output : A, B, C, D, E

distance = 2 + 3 + 4 + 2

= 11

→ Time complexity : O(V+E)

Space complexity : O(V)

→ Applications :

1. shortest path : BFS can find the shortest path between 2 nodes in an unweighted graph

2. Network traversal : Used in network routing algorithms to explore neighbouring routers.

3. Web crawlers : Used by search engines to index web pages by following links.

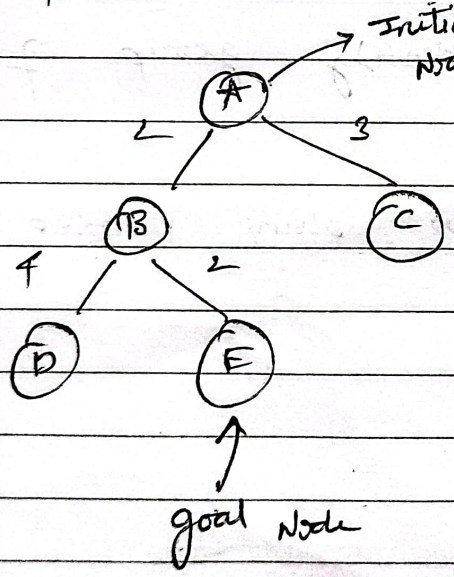
4. Puzzle solving : Can be applied to solve puzzles like sliding tile puzzles.

(2) Depth First Search

Def: DFS is a graph traversal algorithm that explores as far as possible along each branch before backtracking. It traverses a path from its starting vertex to the deepest level before backtracking to explore other paths.

→ It starts at the source vertex, explores as deep as possible until it reaches a dead-end, and backtracks after that.

Eg:



DFS –

Stack. 1. A

2. B, C, A → A, pop A

3. D, E, C → pop B

4. E, C → pop D

5. C → pop E

6. → C

So A, B, D, E, C.

distance. 2 + 4 + 2 = 8

(*) Space & Time complexity:

1. Time : O(V+E)
2. Space : O(V)

DFS uses stack. An element is inserted, then when it's popped, its children are inserted. All elements are popped, until none remain.

Applications

1. Path Finding : Used to find paths between nodes, like in maze solving.

2. Topological Sorting : Arranges items based on dependencies, like scheduling tasks.

3. Connected Components : To identify group of connected nodes in a graph

4. Solving Puzzles : Used in solving puzzles like the N-queens problem

Conclusion :

→ DFS and DFS on undirected is detail, and executed successfully.

Algorithms :

Next page.

AIMLT - Assignment 2

BFS - pseudo code.

→ procedure BFS (graph, start)
    init queue
    enqueue start
    start → visited = true

    while queue not empty:
        current = dequeue from queue
        process current node.

        for each neighbor of current:
            if neighbor not visited,
                mark neighbor as visited
                enqueue neighbor to queue

→ procedure DFS : (graph, node)

    mark node as visited
    process node.

    for each neighbours of node;:
        if neighbours not in visited :
            DFS (graph, neighbours)

FAQs

(*) Is BFS & DFS complete? are they optimal?

→ Both BFS and DFS are complete algorithms, meaning they will always find a solution if one exists in the graph.

→ BFS is optimal for finding shortest path in unweighted graph. DFS may not guarantee an optimal solution, due to its deep exploration before backtracking.

(*) Why are queues preferred over other structures for BFS?

→ queues are preferred for implementing BFS as BFS requires visiting all neighbours of a vertex before moving on their neighbours. Queues follow FIFO. This ensures level by level visits.

(*) Why not queue for DFS? Why stack?

→ DFS uses a stack in recursive manner. Stack is used to explore deeply before backtracking. It aligns with stack's LIFO behaviour.

(*) Why can't we use DFS for shortest path possible?

→ DFS doesn't guarantee finding shortest path as it explores the path deeply. It may encounter long paths, before finding shorter one, leading to non optimal solution.