MILITARY INSTITUTE OF SCIENCE AND TECHNOLOGY

Department of Computer Science and Engineering

==========

CSE-404

ARTIFICIAL INTELLIGENCE SESSIONAL

COMPARATIVE ANALYSIS BETWEEN BFS & DFS IN FINDING SHORTEST PATH IN A GRID

Members :

Name	Student ID
Nazia Shehnaz Joynab	201914032
Fairooz Nawar Nawme	201914033
Sazia Tabasum Mim	201914040
A.S.M. Rakibul Hasan	201914056

Introduction :

In Artificial Intelligence the search algorithm plays a vital role to figure out the problems of the shortest path finding. Pathfinding algorithm is required to determine the fastest path possible. There are several algorithms to find the fastest path. Breadth First Search(BFS) and Depth First Search(DFS) are two pathfinding algorithms and are used mostly.

BFS : BFS is an algorithm that is used to graph data or searching tree or traversing structures. The algorithm efficiently visits and marks all the key nodes in a graph in an accurate breadthwise fashion.This algorithm selects a single node (initial or source point) in a graph and then visits all the nodes adjacent to the selected node. Once the algorithm visits and marks the starting node, then it moves towards the nearest unvisited nodes and analyzes them. Once visited, all nodes are marked. These iterations continue until all the nodes of the graph have been successfully visited and marked.

DFS : DFS is an algorithm for finding or traversing graphs or trees in depth-ward direction. The execution of the algorithm begins at the root node and explores each branch before backtracking. It uses a stack data structure to remember, to get the subsequent vertex, and to start a search, whenever a dead-end appears in any iteration. The full form of DFS is Depth-first search.

Literature Review :

From paper with title “Comparative Analysis of Various Uninformed Searching Algorithms in AI” we can get the definitions of BFS and DFS.

In Breadth First Search (BFS) all the vertices are explored or traversed level by level. That is, it first expands all the nodes at first level in the search tree, then expands all the nodes of the second level and this way it reaches the goal. [1]

In Depth First Search (DFS) expansion starts from the initial node in the graph and it explores or traverses deepest unexplored node of that vertex, in this way it reaches the goal node. It is also called edge based method and it works in the recursive fashion where the vertices are explored along a path. [1]

A good number of journals explain the comparative relation between BFS and DFS in details.

Comparison between BFS and DFS :

- BFS algorithm can be implemented by using queue data structure, which works based on the First In First Out (FIFO) principle. On the other hand, DFS algorithm can be implemented by using stack data structure. That is, it works based on the Last In First Out (LIFO) principle. [2]

- The Time complexity of BFS is O(bd+1) and space complexity is O(bd+1), where 'b' is branching factor and 'd' is the solution depth. The Time complexity of DFS is O(bm) and space complexity is O(bm), where 'b' is branching factor and 'm' is maximum depth. [2]

- The result of the depth-first search algorithm is a certain route, following which you can go around all the graph vertices accessible from the initial vertex. In this way, it is fundamentally different from the breadth-first search, where many vertices are simultaneously processed, and only one vertex is processed in the depth-first search at each moment of the algorithm execution. [1]

- Most BFS algorithms perform an abbreviated DFS before traversing each adjacency list, placing this result in a queue, which is then bypassed. For this reason, when used in trees, it is twice as slow as DFS. However, the advantage is that if you are looking for close neighbors, BFS is faster than DFS. [1]

- The implementation of the breadth-first search algorithm allows to simply parallelize the program, with a greater effect than a parallel algorithm for depth-first search. [1]

- The DFS does not find the shortest paths, but it is applicable in situations where the graph is not completely known, but is being investigated by some automated device. So DFS can't be used to find shortest path in an unweighted graph, whereas BFS is the most efficient algorithm in this case. [1]

- From the standpoint of the algorithm, all child nodes obtained by expanding a node are added to a First in First out (FIFO) queue. In typical implementations, nodes that have not yet been examined for their neighbors are placed in some

container (such as a queue or linked list) called “open” and then once examined are placed in the container “closed”. [3]

- DFS is an uninformed search that progresses by expanding the first child node of the search tree that appears and thus going deeper and deeper until a goal node is found, or until it hits a node that has no children. Then the search backtracks, returning to the most recent node it has not finished exploring. [3]

Results and discussion with the Table :

Comparison points	BFS	DFS
Execution time	12.314804	7.668805
Number of nodes of the found path	30	42
Path length	29	41
Number of turns	13	21
Number of iteration	76	43
Number of explored nodes	76	43
Max use of the Queue and Stack	80	70
Number of nodes in the Queue or Stack at the end	3	26

Conclusion :

In this assignment, we have modified the given grid by changing obstacles and perform BFS and DFS to find shortest path. We have also analysed some journals to find a brief comparison between the efficiency of BFS and DFS. It can be concluded that , BFS and DFS, both of the graph searching techniques have similar running time but different space consumption, DFS takes linear space because we have to remember single path with unexplored nodes, while BFS keeps every node in memory.

References :

[1] Kapitan, Dmitrii Yu, et al. "The Comparison of DFS and BFS Methods on 2D Ising Model." CEUR Workshop Proceedings. Vol. 2426. 2019.

[2] Gayathri, R. "Comparative Analysis of Various Uninformed Searching Algorithms in AI." (2019).

[3] Akanmu, T. A., et al. "Comparative study of complexities of breadth-first search and depth-first search algorithms using software complexity measures." Proceedings of the World Congress on Engineering. Vol. 1. 2010.