**Question 1**: Compare Breadth First Search (BFS) and Depth First Search(DFS) with respect to the number of steps required to reach the solution if they are reachable.

Breadth First Search (BFS) and Depth First Search (DFS) are “Uninformed Search” algorithms. BFS can be implemented using FIFO i.e. Queue & DFS using LIFO i.e. Stack.

The goal of both the algorithms is to traverse and search a pre-defined goal in tree/graph structures. Although the moto of both algorithms is same, but they have different method to reach the goal due to which the characteristics and performance of both are different in different scenarios.

The BFS explores all the nodes present at the depth. After that it explores all the nodes present at next depth. So, we can say that BFS completes the “Breath” at a depth and then moves to next depth. Due to this behavior the BFS will always find the shortest path to the goal.

On other hand, the DFS explores as deeply as possible along one branch before backtracking It continues this process until it finds the solution or reaches at the end. If the solution is not found, then it backtrace and starts exploring the next branch and moves towards the depth. Due to this behavior the DFS may not find the shortest path to the goal and it can also get trapped into the fact if the depth is infinite.

Let is now validate this using some examples:

**Example 1**:

|  |  |
| --- | --- |
| **Start State** | **Target State** |
| 1 2 3  4 B 5  6 7 8 | 1 2 3  4 5 6  7 8 B |

When we run our algorithm implementation for above mentioned start & goal state then we have observed the below outcome

|  |  |  |
| --- | --- | --- |
|  | **BFS** | **DFS** |
| **States Explored** | 4693 | 37067 |
| **Solution found in steps** | 14 | 34838 |

We can see that

* The BFS has found the **shorter** path as compared to the DFS.
* The BFS has explored **less** number of states to find the solution as compared to DFS

**Example 2:**

|  |  |
| --- | --- |
| **Start State** | **Target State** |
| 3 2 1  4 5 6  8 7 B | 1 2 3  4 5 6  7 8 B |

When we run our algorithm implementation for above mentioned start & goal state then we have observed the below outcome

|  |  |  |
| --- | --- | --- |
|  | **BFS** | **DFS** |
| **States Explored** | 130052 | 158318 |
| **Solution found in steps** | 24 | 27816 |

We can see that

* The BFS has found the **shorter** path as compared to the DFS.
* The BFS has explored **less** number of states to find the solution as compared to DFS

**Example 3**:

|  |  |
| --- | --- |
| **Start State** | **Target State** |
| B 1 2  3 4 5  6 7 8 | 1 2 3  4 5 6  7 8 B |

When we run our algorithm implementation for above mentioned start & goal state then we have observed the below outcome

|  |  |  |
| --- | --- | --- |
|  | **BFS** | **DFS** |
| States Explored | 79712 | 15288 |
| Solution found in steps | 22 | 14832 |

We can see that

* The BFS has found the **shorter** path as compared to the DFS.
* The BFS has explored **more** number of states to find the solution as compared to DFS

**Conclusion**:

The BFS find the shortest path to the solution. The number of sates explored to reach the solution may depend on the start state. As the solution moves towards the depth, there is high possibility that BFS will explore more number of states as compared to DFS. So, it will be safe to say that DFS is good in finding the problem when solution lies towards the bottom of the tree. Because the in that case BFS will keep on exploring the breath at each depth. This will put burden on BFS in term of memory storage as well as number of states explored.

**Moral**:

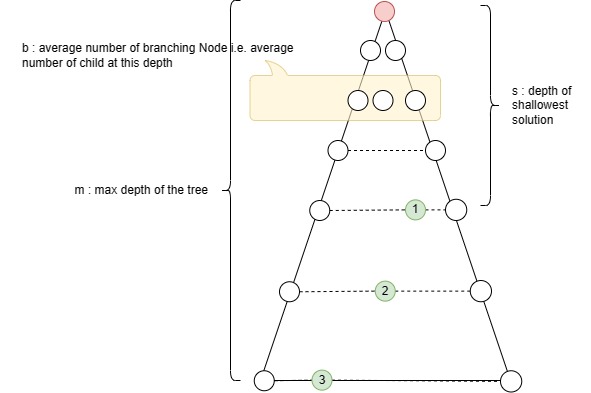
* If we want to find the shortest path then BFS should be used.
* If the solution is deeper in the tree and the memory used is limitation, then we should consider DFS

**Question 2**: Comment on which algorithm will be faster and when, by mentioning proper intuition and examples.

There are two aspects of the term “faster algorithm”

* Time taken by the algorithm to reach the solution i.e. to reach the goal in less number of steps.
* Time taken by the algorithm to find the solution i.e. explore less number of states before finding the solution.

Now lest us discuss both BFS and DFS one by one let us take the below picture for reference



Let us say we have the above shown tree. The red circle is the start state. The green circles (marked with 1, 2, 3) are the possible solution to this problem.

The BFS will be faster if

* The branching factor “b” is low
* If the depth of the tree is infinite then BFS will be able to find the solution faster (simple DFS will not be able to find solution if depth is infinite)

The DFS will be faster if

* If the solution lies towards the bottom of the tree. e.g. the solution 3 (marked in green circle) will be reached quickly by DFS as compared to the BFS because here the DFS has to explore less number of states as compared to BFS. (BFS will explore bs + 1 states while DFS will explore bm+1 ).
* If the breath factor b is more than the DFS will get advantage over BFS.
* If we have limited memory then there is high possibility that the BFS will get exhausted in memory uses & will fail. DFS will take advantage in such cases.

Let us take some outcomes of examples:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Start State** | **BFS** | | **DFS** | |
| Steps to reach solution | Number of states explored | Steps to reach solution | Number of states explored |
| 1 2 3  4 B 5  6 7 8 | **14** | 4693 | 34838 | 37067 |
| B 1 2  3 4 5  6 7 8 | **22** | 79712 | 14832 | 15288 |

* If reaching the shortest path is the criteria then **BFS will be fast**
* If we need to find the solution fast i.e. in less number of explored states then DFS will be fast in some cases. But final verdict will depend on the initial state.

So, BFS will always find shortest path. DFS may find solution in less number of time, if there is only one solution and it lies at the bottom of the tree.