Bi-directional search

Introduction of the algorithm:

Bidirectional search is a graph search algorithm that finds the shortest path between the source and the goal vertex. It performs two searches at the same time -

- Searching forward from the source/initial vertex to the goal vertex
- Searching backwards from the goal/target vertex to the source vertex

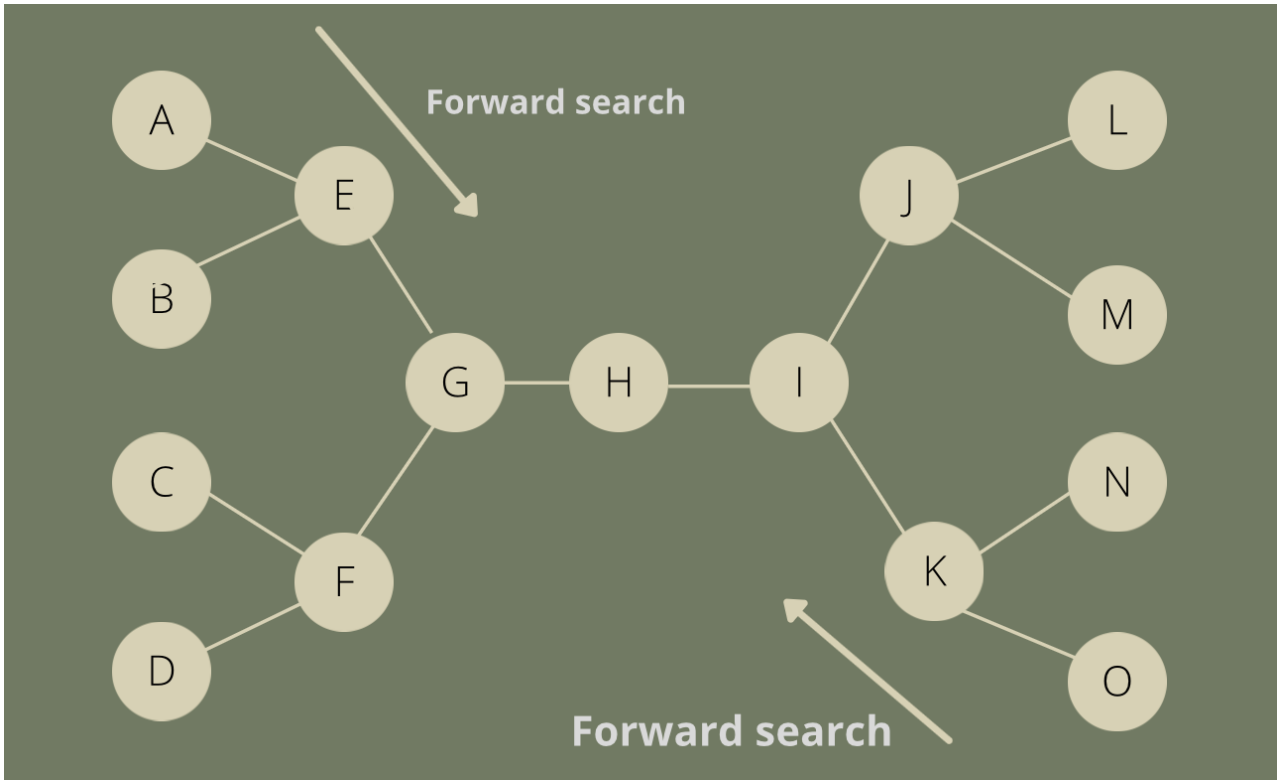
Bidirectional search divides a single search graph (which is expected to develop exponentially) into two smaller subgraphs, one beginning at the initial vertex and the other beginning at the destination vertex. When two graphs intersect, the search comes to an end.

Details and methodology:

The shortest path from the present node in the graph to the desired node is referred to as 'heuristic'. The shortest path to the goal node is always chosen. A bidirectional heuristic search employs this approach. The sole distinction is that both the beginning point and the goal vertex are searched simultaneously. The fundamental goal of bidirectional searches is to dramatically minimize the amount of time it takes to do a search.

This occurs when both searches are conducted concurrently, starting with the node depth or breadth-first search and working backwards from target nodes meeting somewhere in the graph. Because of this search, the path traversing via the beginning node, intersecting point, and goal vertex is now the shortest path discovered. This is the shortest path and found in a fraction of time taken by other search algorithms.

This can be simplified by the following example:



- Step 1: Assume A is the starting point, O is the destination, and H is the junction point.
- Step 2: We'll begin searching from the beginning to the end, as well as backwards from the end to the beginning.
- Step 3: When the forward and backward searches converge at a single node, the search comes to a halt.

It is therefore conceivable if both the Start node and the Goal node are known and distinct from one another. In addition, the branching factor for both traversals in the graph is the same. Bidirectional searches are complete if a breadth-first search is performed for both traversals, i.e. for both pathways from the start node to the intersection and from the destination node to the intersection.

The bidirectional search's time and space complexity is given as O(b^(d/2).The following are the two primary types of bidirectional searches:

- Front to back or BFEA
- Front to Front or BFFA

1. BFEA (back to front)

Two heuristic functions are required for bidirectional Front to Front Search. The first is the estimated distance between a node and the target state using forwards search, while the second is the estimated distance between a node and the start state using reverse action. The heuristic value of the distance between the node n and the root of the opposite search tree s or t is determined in the method, and it is h. Of the three bidirectional search algorithms, this is the most extensively employed.

2. BFFA (Back to Front)

The minimum of all heuristic distances from the current node to nodes on opposing fronts is determined here, and h is calculated as the minimum of all heuristic distances from the current node to nodes on opposing fronts.

Performance metrics:

- *Completeness:* If BFS is used in both searches, the bidirectional search is complete.
- *Optimality:* When BFS is used for search and pathways have uniform costs, the result is optimum.
- *Time and Space Complexity:* O(b^(d/2)) is the time and space complexity.

Advantages

The benefits are as follows:

- Bidirectional searches have a number of advantages, one of which is the speed with which we may obtain the necessary results.
- By doing many searches at the same time, it substantially lowers the amount of time it takes to complete a search.
- It also saves consumers resources by using less memory to keep all of the searches.

Disadvantages

The following are the drawbacks:

- The main difficulty with bidirectional search is that the user must be aware of the objective state in order to utilize it, greatly reducing its use cases.
- Another problem is the implementation, which needs more code and instructions, as well as special attention to each node and step in order to conduct such searches.
- Otherwise, there's a risk of an infinite loop if the algorithm isn't strong enough to recognise the intersection when the search should terminate.
- It's also impossible to go backwards through all of the states.

Conclusion:

A bidirectional search, despite its disadvantages, is the most efficient and fastest approach to arrive at desired search results when the goal state is known before the search begins, and hence one of the most commonly used and researched search algorithms accessible. Anyone interested in a career in the Database management system's 'Search' should be familiar with all search methods, with bidirectional being the most distinctive and sought-after.

References:

- https://www.sciencedirect.com/topics/computer-science/bidirectional-search
- https://www.javatpoint.com/ai-uninformed-search-algorithms
- https://www.geeksforgeeks.org/bidirectional-search/
- https://en.wikipedia.org/wiki/Bidirectional_search