Lesson 2: Searching as a Problem-Solving Technique

2.1 Introduction

Many interesting problems in science and engineering are solved using search.
A search problem is defined by:

a) A search space: The set of objects among which we search for the solution.
Examples: routes between cities, or n-queens configuration
This problem space can be represented as a graph, tree, or any structure where you need to navigate to find a specific goal or solution.

b) A goal condition: Characteristics of the object we want to find in the search space?
Examples: Path between cities A and B, Non-attacking n-queen configuration

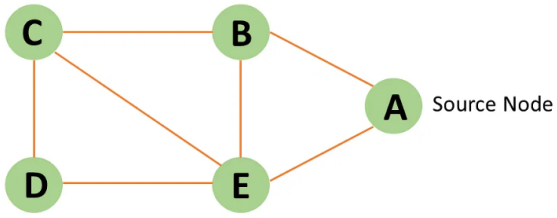
2.2 Breadth-first search (BFS)

BFS is a search algorithm that explores a graph or tree by visiting all neighbors of a node before moving on to their neighbors. It is often used to find the shortest path in unweighted graphs. BFS explores all the neighbors of a node before moving on to their neighbors.

In general, all the nodes are expanded at a given depth in the search tree before any nodes at the next level are expanded.

This is achieved by using a FIFO queue for the frontier. Accordingly, new nodes go to the back of the queue, and old nodes, which are shallower than the new nodes are expanded first.

Example: Given below graph with start point as node A, the BFS will be: ABECD

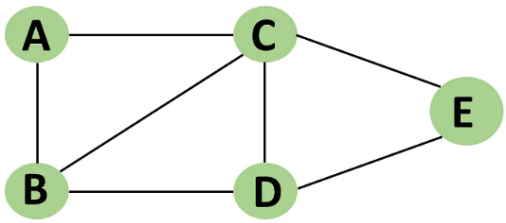


2.3 Depth-first search (DFS)

DFS is another graph traversal algorithm that explores as far as possible along a branch before backtracking. It is often used in scenarios such as maze solving and topological sorting.

The algorithm starts at the root node (selecting some arbitrary node as the root node in the case of a graph) and explores as far as possible along each branch before backtracking. It uses LIFO strategy and hence it is implemented using a stack.

Example: Given below graph with start point as node A, the DFS will be: ABCDE

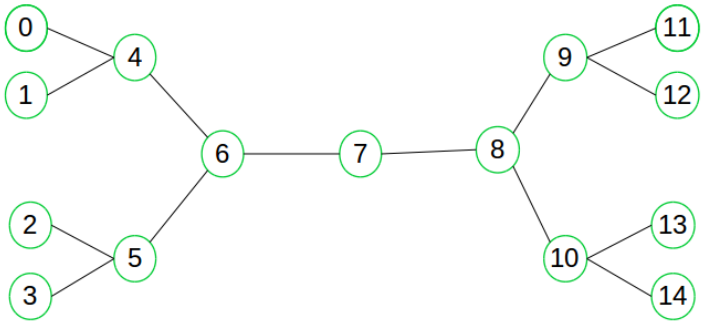


2.4 Bi-directional

Bi-directional search is a strategy that involves running two simultaneous searches, one from the start state and another from the goal state. The search terminates when the two searches meet in the middle.

In this scenario, the two searches will meet in the middle, saving time and resources.

Consider following example:



Suppose we want to find if there exists a path from vertex 0 to vertex 14. Here we can execute two searches, one from vertex 0 and other from vertex 14. When both forward and backward search meet at vertex 7, we know that we have found a path from node 0 to 14 and search can be terminated now. We can clearly see that we have successfully avoided unnecessary exploration.

2.5 Best-first search

Best-First Search is a family of search algorithms that select the most promising nodes to explore based on a heuristic function. A heuristic estimates the cost to reach the goal from a given state. A* search is a popular example of a best-first search algorithm.

2.6 Heuristic functions

A heuristic function is a function that estimates the cost or distance from a given state to the goal state. It helps guide search algorithms in making informed decisions about which nodes to explore first. The quality of the heuristic function significantly impacts the efficiency of search algorithms.

Example: In the A* search, a common heuristic is the Manhattan distance in grid-based problems. For each cell (i, j), the heuristic value h (i, j) could be:

h (i, j) = |i - i_Finish| + |j - j_Finish|, where (i_Finish, j_Finish) is the goal location.

The heuristic helps the algorithm prioritize nodes that seem closer to the goal.

2.7 Impact of heuristic functions on the performance of search algorithms

The choice of heuristic function can greatly affect the performance of search algorithms. A well-designed heuristic can lead to faster convergence to the solution, while a poor heuristic may lead to inefficient or even incorrect results. It's essential to strike a balance between accuracy and computational efficiency when designing heuristic functions.

Example: In the 8-puzzle problem, a perfect heuristic would exactly predict the number of moves to solve it. A less accurate heuristic may lead to more nodes being expanded before reaching the solution, making the search slower.

Lesson 2: Review Questions

1. Compare and Contrast BFS and DFS as search strategies.
2. Differentiate between informed search and uninformed search.
3. Discuss the A* algorithm.