**Malnad College Of Engineering**

Under the auspices of M.T.E.S

(An Autonomous Institution Affiliated to VTU, Belgaum)

P.B No. 21, Hassan-573 202, Karnataka



**Artificial Intelligence (20IS603)**

“ITERATIVE DEEPENING DEPTH FIRST SEARCH”

**Report Submitted By**

Jhenkar 4MC20IS019

Noor Moinuddin H S 4MC20IS033

Rahul H 4MC20IS038

Sumukha K Y 4MC20IS054

Under The Guidance Of

DR. B.V Balaji Prabhu

Associate professor

**Department of Information Science and Engineering**

**Malnad College of Engineering**

**Hassan-573 201**

**2023-2024**

Title: Iterative deepening depth first search

1. Introduction

Iterative Deepening Depth-First Search (IDDFS) is a graph traversal algorithm that combines the benefits of depth-first search (DFS) and breadth-first search (BFS). It is often used in scenarios where the search space is large or infinite, and the depth of the target node is unknown.

IDDFS incrementally increases the maximum depth limit for DFS in each iteration until the target node is found. It performs a series of DFS searches, starting from a depth of 0 and incrementing the depth limit by 1 in each iteration. This allows IDDFS to explore the search space in a depth-first manner while also avoiding the drawbacks of BFS, such as high memory consumption.

The basic idea behind IDDFS is to re-visit nodes multiple times but with increasing depth limits. By doing so, it guarantees that all nodes at depths within the limits are explored before moving to deeper levels. This ensures that the shortest path from the source to the target node is found.

2. Algorithm Description

1. Set the initial depth limit to 0.
2. Perform a depth-first search (DFS) starting from the source node, limiting the search to the current depth limit.
3. If the target node is found at the current depth limit, return the path from the source to the target.
4. If the target node is not found at the current depth limit, increment the depth limit by 1.
5. Repeat steps 2-4 until the target node is found or the search space is exhausted.

3. Working Principle

The working principle of Iterative Deepening Depth-First Search (IDDFS) revolves around gradually increasing the depth limit in each iteration while performing a depth-first search. Let's explore the working principle in more detail:

1. Start with a depth limit of 0: Initially, set the depth limit to 0, indicating that only the source node can be explored.
2. Perform a depth-first search (DFS): Begin the depth-first search from the source node. Traverse the graph by exploring one branch as deeply as possible before backtracking.
3. Check the depth limit and target node: At each visited node, compare the current depth with the depth limit. If the depth exceeds the limit, stop exploring that branch and backtrack. If the target node is found at the current depth limit, return the path from the source to the target.
4. Increment the depth limit: If the target node is not found at the current depth limit, increment the depth limit by 1.
5. Repeat the process: Repeat steps 2-4, performing another depth-first search but with the new depth limit. This time, the search will explore nodes at a greater depth than in the previous iteration.
6. Termination condition: The iterations continue until either the target node is found or the search space is exhausted.

4. Advantages of Iterative deepening depth first search

1. Completeness: IDDFS guarantees completeness, meaning that if a solution exists within the search space, IDDFS will eventually find it.
2. Memory efficiency: IDDFS combines the memory efficiency of depth-first search (DFS) with the completeness of breadth-first search (BFS).
3. Optimal solutions: IDDFS is optimal for problems with uniform path costs.
4. Time efficiency for finite depths: In cases where the depth of the target node is known or bounded, IDDFS can be more time-efficient than other algorithms like BFS.
5. Incremental results: IDDFS provides incremental results during its iterations.
6. Simplicity of implementation: IDDFS is relatively simple to implement compared to other complex graph traversal algorithms.

5. Limitations of Iterative deepening depth first search

1. Inefficiency in time and space: IDDFS can be inefficient in terms of time and space complexity. In each iteration, the algorithm explores the entire search space up to the current depth limit. As the depth limit increases, the number of nodes explored grows exponentially.
2. Redundant exploration: IDDFS may redundantly explore nodes multiple times.
3. Lack of pruning: IDDFS does not employ pruning techniques to avoid unnecessary exploration of suboptimal branches.
4. Inefficient for graph structures with large branching factors: IDDFS can struggle with graphs that have high branching factors, where the number of child nodes at each level is large. Limited to problems with known or bounded depths: IDDFS is not suitable for problems where the depth of the target node is unknown or not bounded.
5. Non-optimal for problems with non-uniform path costs: IDDFS is optimal only for problems with uniform path costs.

6. Applications of Iterative deepening depth first search

1. Game tree searching: IDDFS is commonly used in game tree searching algorithms, particularly in games with large or infinite state spaces.
2. Artificial Intelligence planning: IDDFS can be employed in AI planning systems to find optimal or near-optimal plans for complex tasks.
3. Web crawling and search algorithms: IDDFS can be used in web crawling and search algorithms to explore web pages or documents efficiently.
4. Solving puzzles and mazes: IDDFS is well-suited for solving puzzles or navigating mazes where the optimal solution is desired.
5. Graph algorithms and traversal: IDDFS can be used as a graph traversal algorithm when the graph structure is unknown or the depth of the target node is not specified.
6. AI problem solving and constraint satisfaction: IDDFS is applicable to various AI problem-solving scenarios, including constraint satisfaction problems (CSPs).

Conclusion

The applications of IDDFS span various problem domains, including game tree searching, AI planning, web crawling, puzzle solving, graph algorithms, and AI problem solving.

Overall, IDDFS strikes a balance between memory efficiency and completeness, making it a valuable algorithm in scenarios where the search space is large, unknown, or infinite, and when memory limitations are a concern. It serves as a useful tool for searching and exploring graphs while gradually increasing the depth to find solutions or optimal paths efficiently.