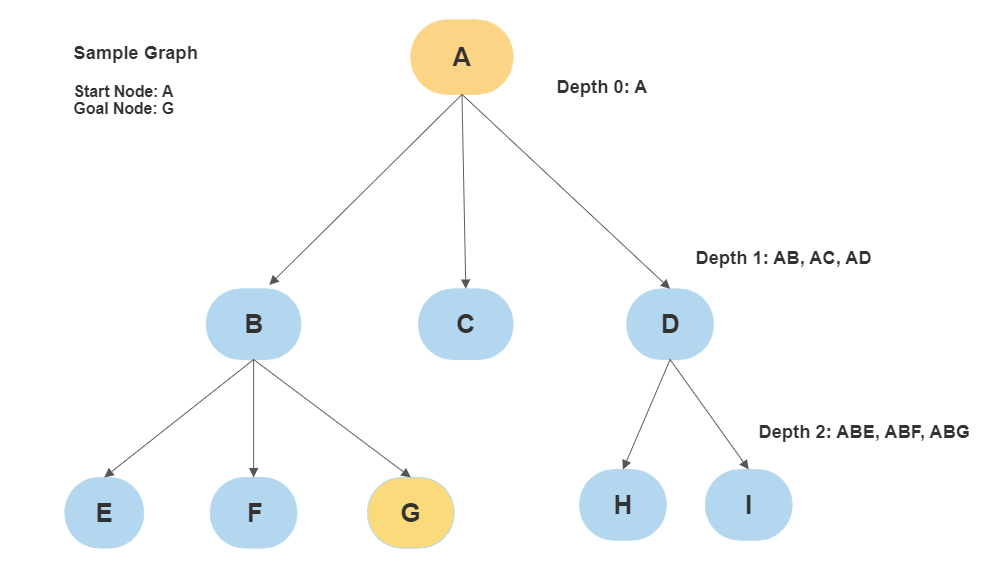
**IDS:**

Iterative Deepening Search (IDS) is a graph traversal and search algorithm that is based on Depth-First Search (DFS). It is used to find the shortest path between a starting node and a goal node in a graph or a tree.

In IDS, the search starts at the root node and proceeds down through the tree or graph until a specified depth is reached or the goal node is found. If the goal node is not found, the search returns to the root node and starts again with a greater depth limit. This process is repeated until the goal node is found.

IDS is called iterative deepening because it performs a series of DFS searches with increasing depth limits until the goal node is found. This approach allows IDS to find the shortest path between the starting node and the goal node while avoiding the exponential time complexity of traditional DFS algorithms.

**Graphical Representation**



**8 Puzzle Game Algorithm:**

The 8-puzzle game is a classic sliding puzzle where a player must slide numbered tiles on a 3x3 grid to arrange them in order from 1 to 8. The empty tile is used to move the other tiles around the board. The goal of the game is to arrange the tiles in the fewest number of moves possible.

**For Example:**

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Start State**   |  |  |  | | --- | --- | --- | | 1 | 5 | 3 | | 2 | 7 | 4 | | 6 | 0 | 8 | | **Goal State**   |  |  |  | | --- | --- | --- | | 1 | 2 | 3 | | 4 | 5 | 6 | | 7 | 8 | 0 | |

**Here's how the IDS algorithm can be used to solve the 8-puzzle game:**

1. Define the initial state of the puzzle and the goal state.
2. Set the initial depth limit to 0.
3. Perform a depth-limited search (DLS) with the current depth limit, starting from the initial state of the puzzle.
4. If the goal state is found, return the solution.
5. If the search reaches the depth limit without finding the goal state, increase the depth limit and repeat from step 3.
6. If the depth limit exceeds the maximum depth of the search space, terminate the search and return failure.

Here's the Depth-Limited Search (DLS) algorithm that is used in step 3:

1. Check if the current state is the goal state. If yes, return the solution.
2. If the current depth is equal to the depth limit, return failure.
3. Generate all possible successor states by moving one tile in each of the four directions (up, down, left, right).
4. For each successor state, check if it has already been visited. If yes, skip to the next successor state. If not, mark the state as visited.
5. Recursively perform DLS on each unvisited successor state with an increased depth.
6. If any of the recursive calls to DLS returns a solution, return the solution.

Note that in step 3, we check whether the current depth is equal to the depth limit. This is what limits the search to a specific depth, making it a depth-limited search.

In step 4, we check if the goal state has been found. If yes, we return the solution. If not, we increase the depth limit and repeat the search from the initial state with an increased depth limit.

By combining the IDS algorithm with the DLS algorithm, we can efficiently search through the entire state space of the 8-puzzle problem, systematically increasing the depth of the search until a solution is found.

Top of Form

**LAB TASK:**

Implement **Depth first with iterative deepening** to solve 8 puzzle problems using python. Make sure that the problem instance is solvable.