# Iterative deepening Search

# Introduction

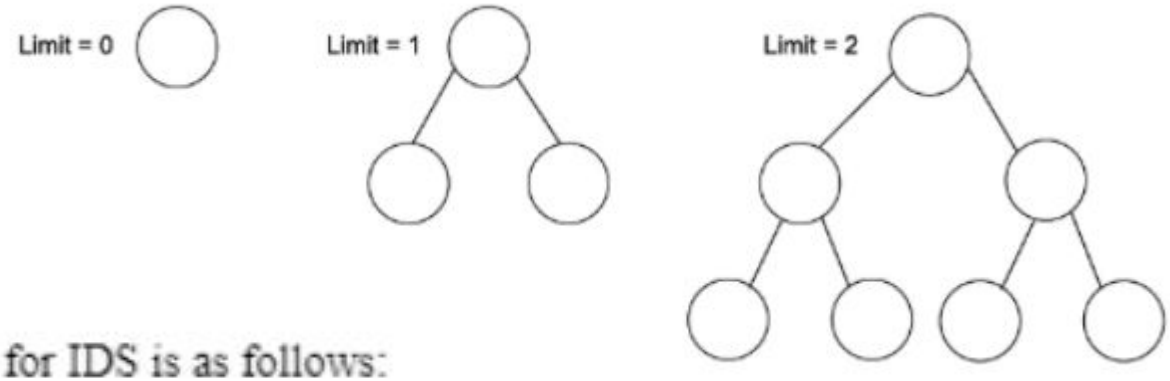
- The iterative deepening algorithm is a combination of DFS and BFS algorithms. This search algorithm finds out the best depth limit and does it by gradually increasing the limit until a goal is found.
- This algorithm performs depth-first search up to a certain "depth limit", and it keeps increasing the depth limit after each iteration until the goal node is found.

# IDS

- This Search algorithm combines the benefits of Breadth-first search's fast search and depth-first search's memory efficiency.
- The iterative search algorithm is useful uninformed search when search space is large, and depth of goal node is unknown.

- **Advantages:**
- It combines the benefits of BFS and DFS search algorithm in terms of fast search and memory efficiency.
- **Disadvantages:**
- The main drawback of IDDFS is that it repeats all the work of the previous phase.

# IDS

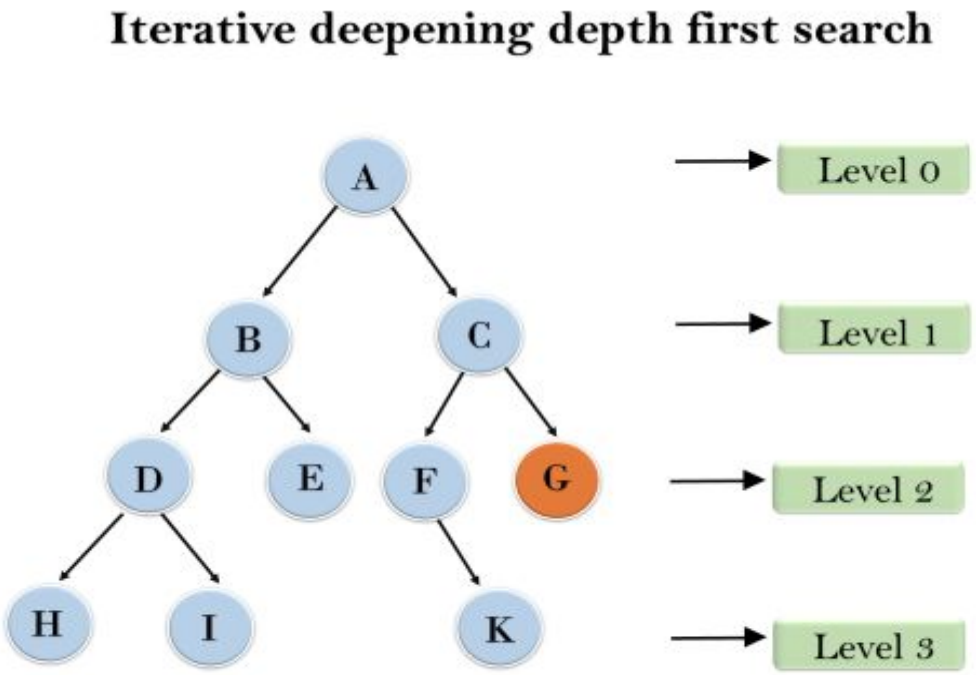



The algorithm for IDS is as follows:

1. Set depth-limit ← 0
2. Do
   Solution = DLS(depth-limit, initial node)

   If(solution= goal state) then return
   else
   Depth-limit = depth-limit + 1
   continue

# Example:




1'st Iteration-----> A
2'nd Iteration----> A, B, C
3'rd Iteration------>A, B, D, E, C, F, G
4'th Iteration------>A, B, D, H, I, E, C, F, K, G
In the fourth iteration, the algorithm will find the goal node.

- **Completeness:**
- This algorithm is complete is ifthe branching factor is finite.
- **Time Complexity:**
- Let's suppose b is the branching factor and depth is d then the worst-case time complexity is **$O(b^d)$**.
- **Space Complexity:**
- The space complexity of IDDFS will be **O(bd)**.
- **Optimal:**
- IDDFS algorithm is optimal if path cost is a non-decreasing function of the depth of the node.

# Bidirectional Search Algorithm

- Bidirectional search algorithm runs two simultaneous searches, one form initial state called as forward-search and other from goal node called as backward-search, to find the goal node.
- Bidirectional search replaces one single search graph with two small subgraphs in which one starts the search from an initial vertex and other starts from goal vertex.

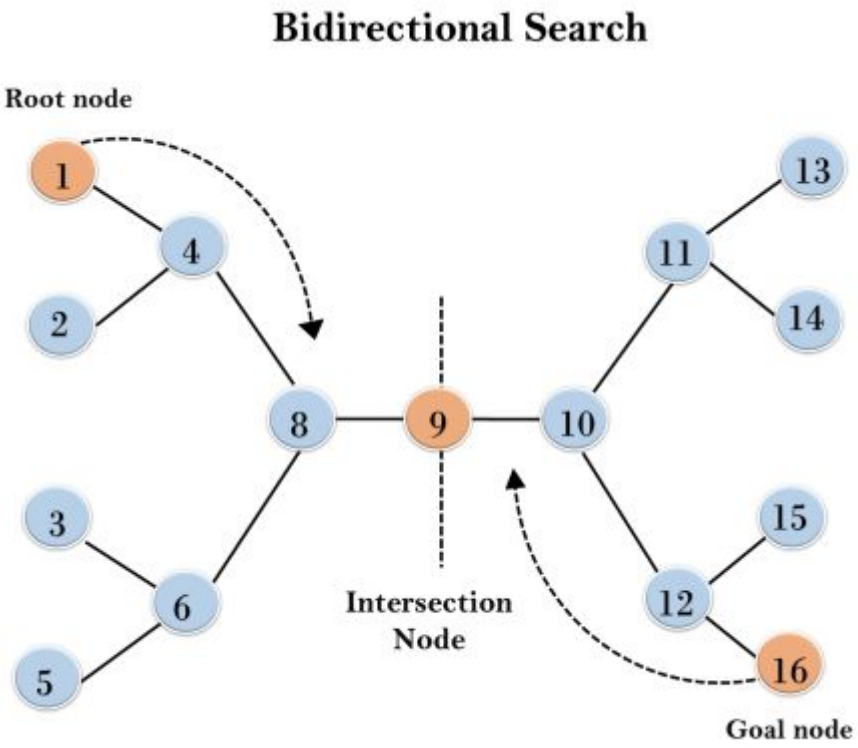
# Bidirectional Search Algorithm

- The search stops when these two graphs intersect each other.
- Bidirectional search can use search techniques such as BFS, DFS, DLS, etc.

- **Advantages:**
- Bidirectional search is fast.
- Bidirectional search requires less memory
- **Disadvantages:**
- Implementation of the bidirectional search tree is difficult.
- **In bidirectional search, one should know the goal state in advance.**

# Example:

- In the below search tree, bidirectional search algorithm is applied. This algorithm divides one graph/tree into two sub-graphs. It starts traversing from node 1 in the forward direction and starts from goal node 16 in the backward direction.
- The algorithm terminates at node 9 where two searches meet.




**Completeness:** Bidirectional Search is complete if we use BFS in both searches.
**Time Complexity: $O(b^d)$**.
**Space Complexity: $O(b^d)$**.
**Optimal:** Bidirectional search is Optimal.