Uninformed/Blind Search Algorithm AI(Artificial Intelligence)

Md. Alamgir Hossain

Lecturer

Dept. of CSE, Prime University

Mail: *alamgir.cse14.just@gmail.com*

Uninformed Search

- ***Uninformed search*** is a class of general-purpose search algorithms which operates in brute force-way.
- Uninformed search algorithms do not have additional information about state or search space other than how to traverse the tree, so it is also called blind search.
- ***Following are the various types of uninformed search algorithms:***
  - ✓ ***Breadth-first Search***
  - ✓ ***Depth-first Search***
  - ✓ ***Depth-limited Search***
  - ✓ ***Iterative deepening depth-first search***
  - ✓ ***Uniform cost search***
  - ✓ ***Bidirectional Search***

Breadth-first Search

- Breadth-first search is the most common search strategy for traversing a tree or graph. This algorithm searches breadthwise in a tree or graph, so it is called breadth-first search.
- BFS algorithm starts searching from the root node of the tree and expands all successor node at the current level before moving to nodes of next level.
- The breadth-first search algorithm is an example of a general-graph search algorithm.
- Breadth-first search implemented using FIFO queue data structure.

Advantages & Disadvantages of Breadth-first Search

➢ **Advantages:**

✓ BFS will provide a solution if any solution exists.

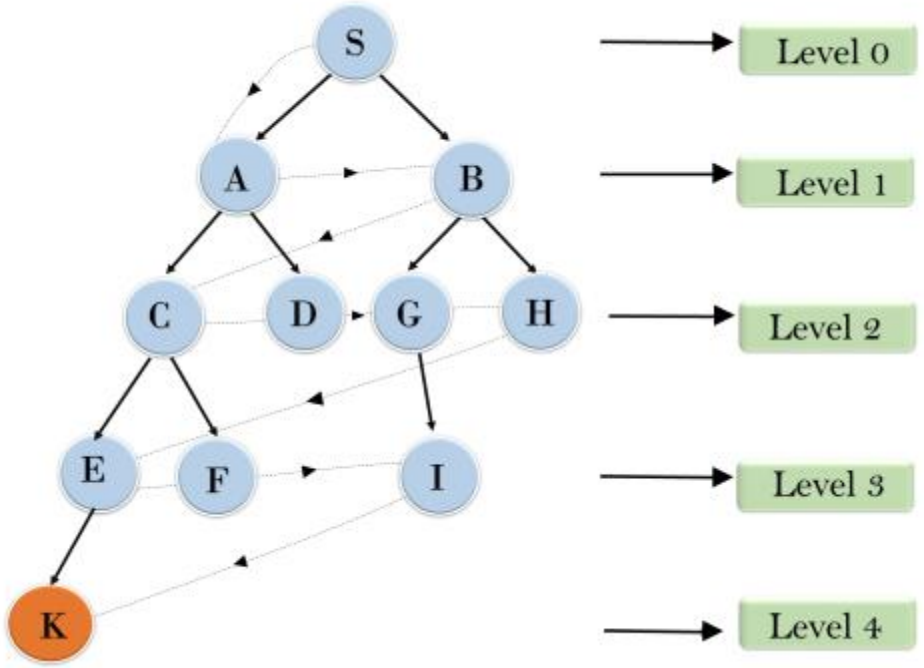
✓ If there are more than one solutions for a given problem, then BFS will provide the minimal solution which requires the least number of steps.

➢ **Disadvantages:**

✓ It requires lots of memory since each level of the tree must be saved into memory to expand the next level.

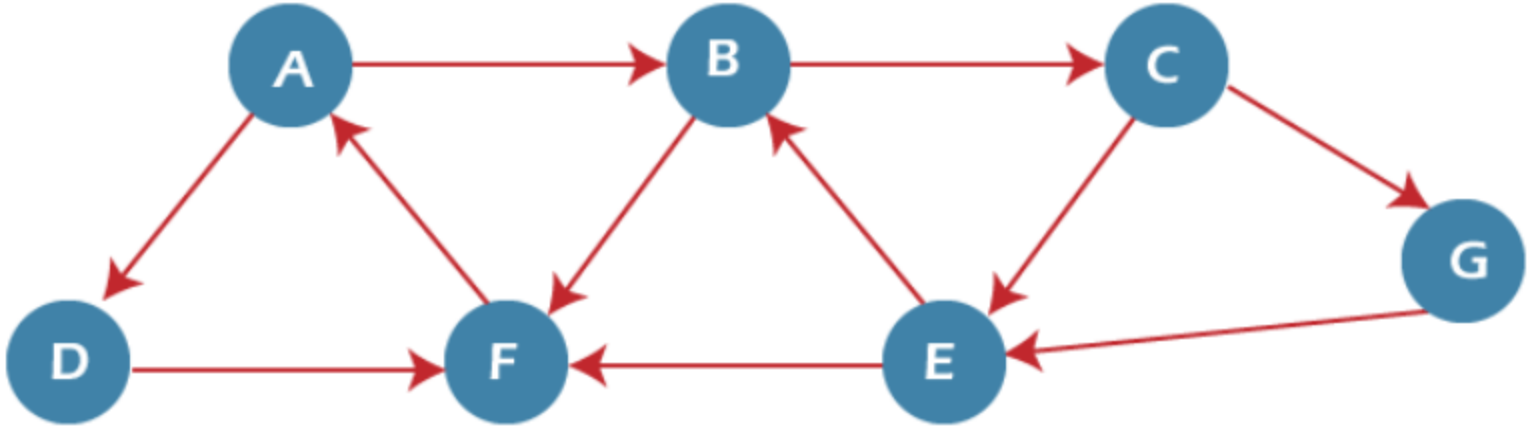
✓ BFS needs lots of time if the solution is far away from the root node.

Example of BFS



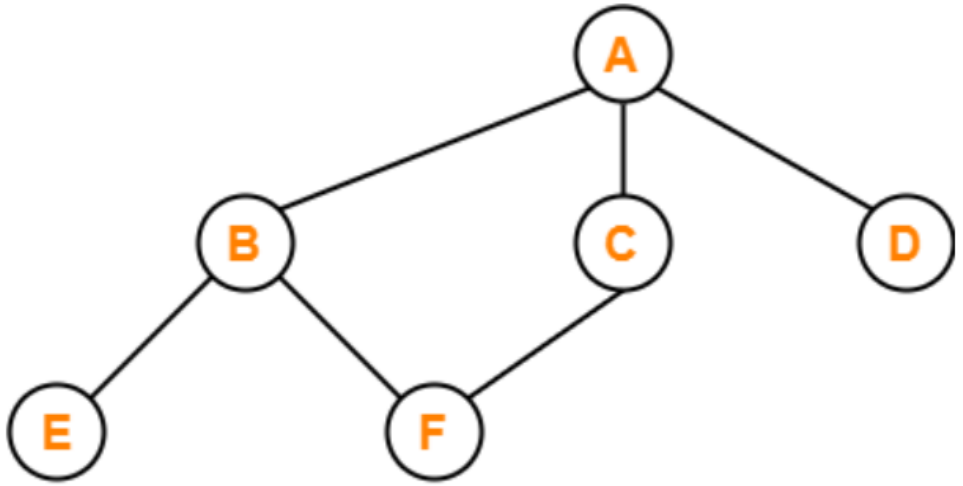
S---> A--->B---->C--->D---->G--->H--->E---->F---->I---->K

Practice Example of BFS



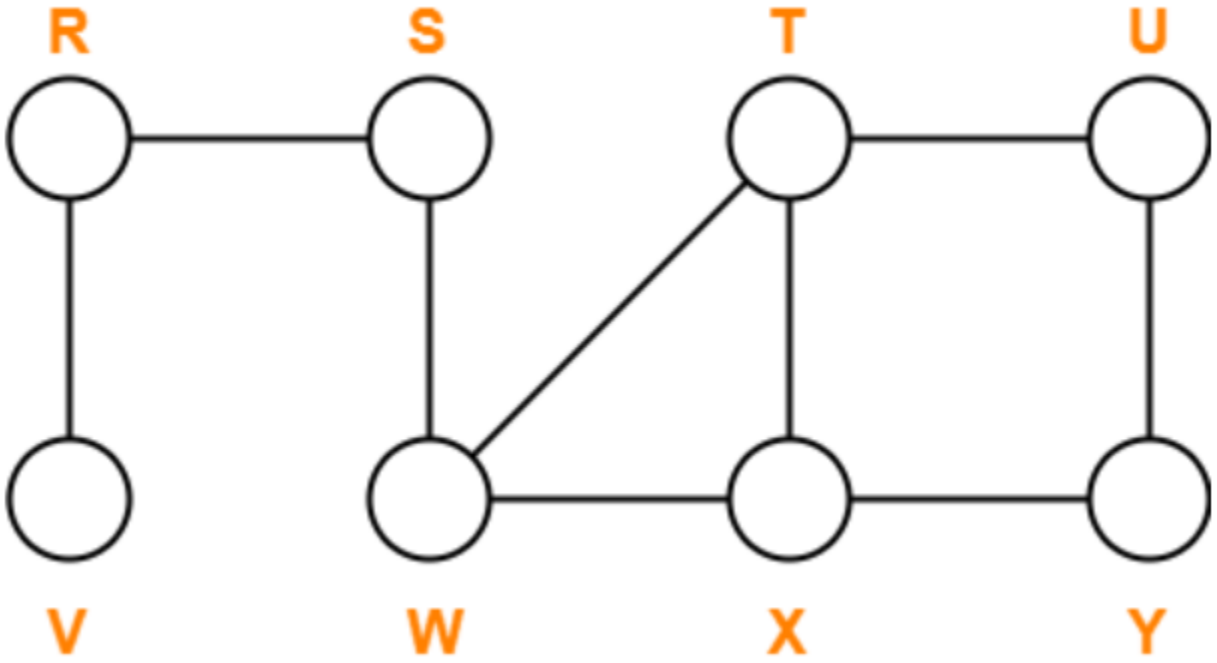
Visited Node: A->B->D->C->F->E->G

Practice Example of BFS



Visited Node: A->B->C->D->F->E

Practice Example of BFS



??

Breadth First Search Algorithm

- ***Time Complexity:*** Time Complexity of BFS algorithm can be obtained by the number of nodes traversed in BFS until the shallowest Node. Where the d= depth of shallowest solution and b is a node at every state.
- $T(b) = 1+b^2+b^3+.....+ b^d= O(b^d)$
- ***Space Complexity:*** Space complexity of BFS algorithm is given by the Memory size of frontier which is $O(b^d)$.
- ***Completeness:*** BFS is complete, which means if the shallowest goal node is at some finite depth, then BFS will find a solution.
- ***Optimality:*** BFS is optimal if path cost is a non-decreasing function of the depth of the node.

Depth-First Search

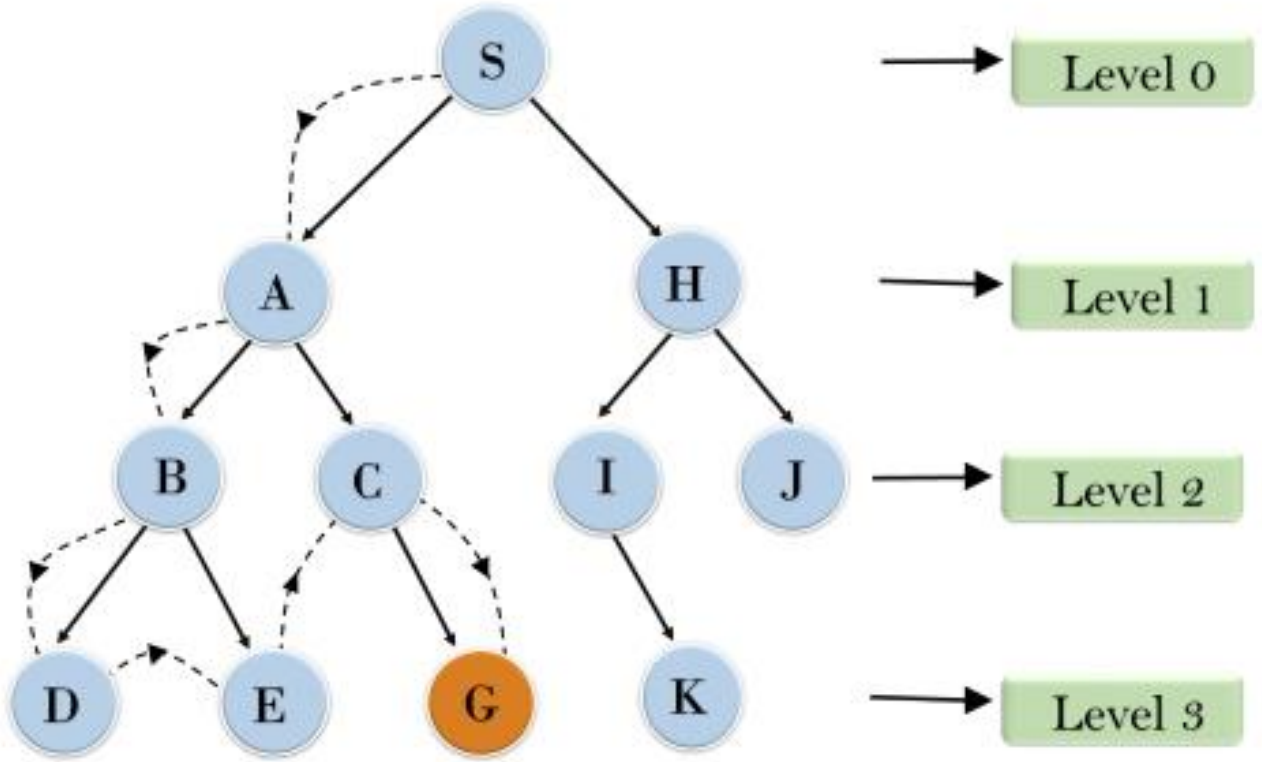
- Depth-first search is a recursive algorithm for traversing a tree or graph data structure.
- It is called the depth-first search because it starts from the root node and follows each path to its greatest depth node before moving to the next path.
- DFS uses a stack data structure for its implementation.
- The process of the DFS algorithm is almost similar to the BFS algorithm.

Advantages & Disadvantages of Depth-First Search

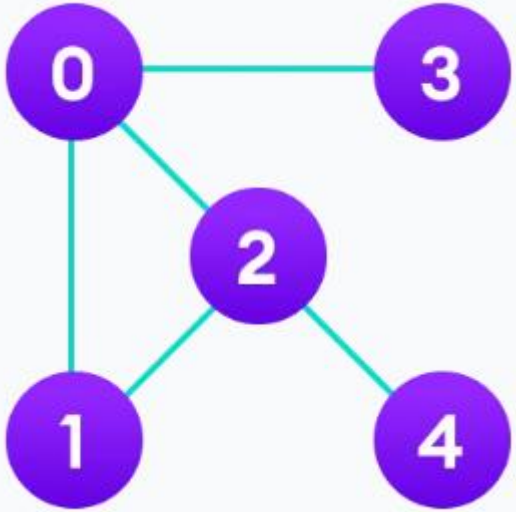
- **Advantage:**
- DFS requires very less memory as it only needs to store a stack of the nodes on the path from root node to the current node.
- It takes less time to reach to the goal node than BFS algorithm (if it traverses in the right path).

- **Disadvantage:**
- There is the possibility that many states keep re-occurring, and there is no guarantee of finding the solution.
- DFS algorithm goes for deep down searching and sometime it may go to the infinite loop.

Depth-First Search



Depth-First Search



0					Visited
1	2	3			Stack

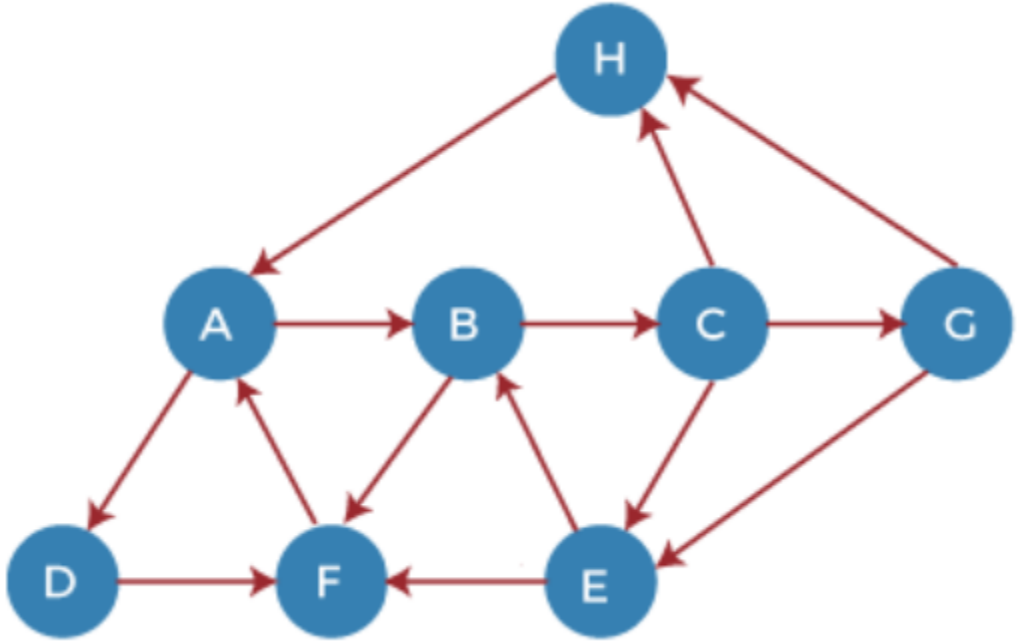
0	1				Visited
2	3				Stack

0	1	2			Visited
4	3				Stack

0	1	2	4		Visited
3					Stack

0	1	2	4	3	Visited
					Stack

Depth-First Search



Adjacency Lists

A : B, D
B : C, F
C : E, G, H
G : E, H
E : B, F
F : A
D : F
H : A

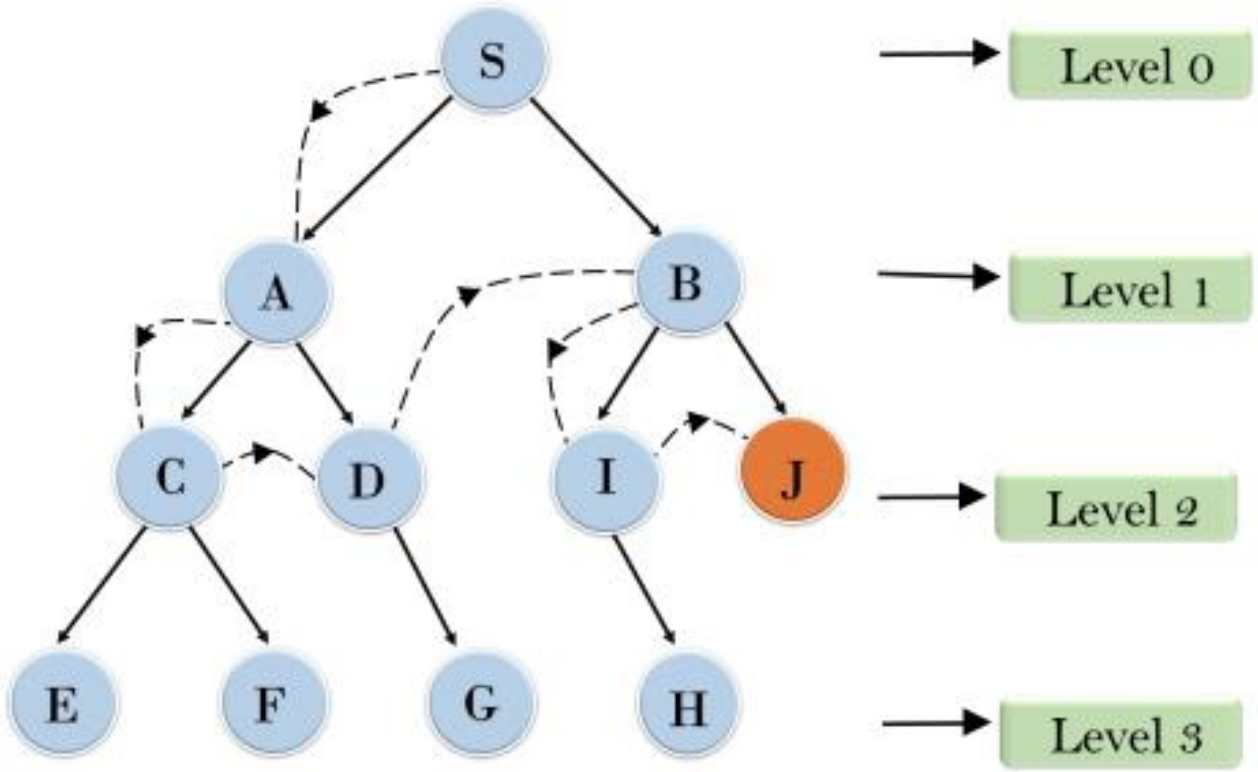
Depth-First Search

- **Completeness:** DFS search algorithm is complete within finite state space as it will expand every node within a limited search tree.
- **Time Complexity:** Time complexity of DFS will be equivalent to the node traversed by the algorithm. It is given by:
- $T(n)= 1+ n^2+ n^3 +.....+ n^m=O(n^m)$
- Where, m = maximum depth of any node and this can be much larger than d (Shallowest solution depth)
- **Space Complexity:** DFS algorithm needs to store only single path from the root node, hence space complexity of DFS is equivalent to the size of the fringe set, which is O(bm).
- **Optimal:** DFS search algorithm is non-optimal, as it may generate a large number of steps or high cost to reach to the goal node.

Depth-Limited Search Algorithm

- A depth-limited search algorithm is similar to depth-first search with a predetermined limit. Depth-limited search can solve the drawback of the infinite path in the Depth-first search. In this algorithm, the node at the depth limit will treat as it has no successor nodes further.
- Depth-limited search can be terminated with two Conditions of failure:
- **Standard failure value:** It indicates that problem does not have any solution.
- **Cutoff failure value:** It defines no solution for the problem within a given depth limit.
- **Advantages:** Depth-limited search is Memory efficient.
- **Disadvantages:**
  - ✓ Depth-limited search also has a disadvantage of incompleteness.
  - ✓ It may not be optimal if the problem has more than one solution.

Depth-Limited Search Algorithm



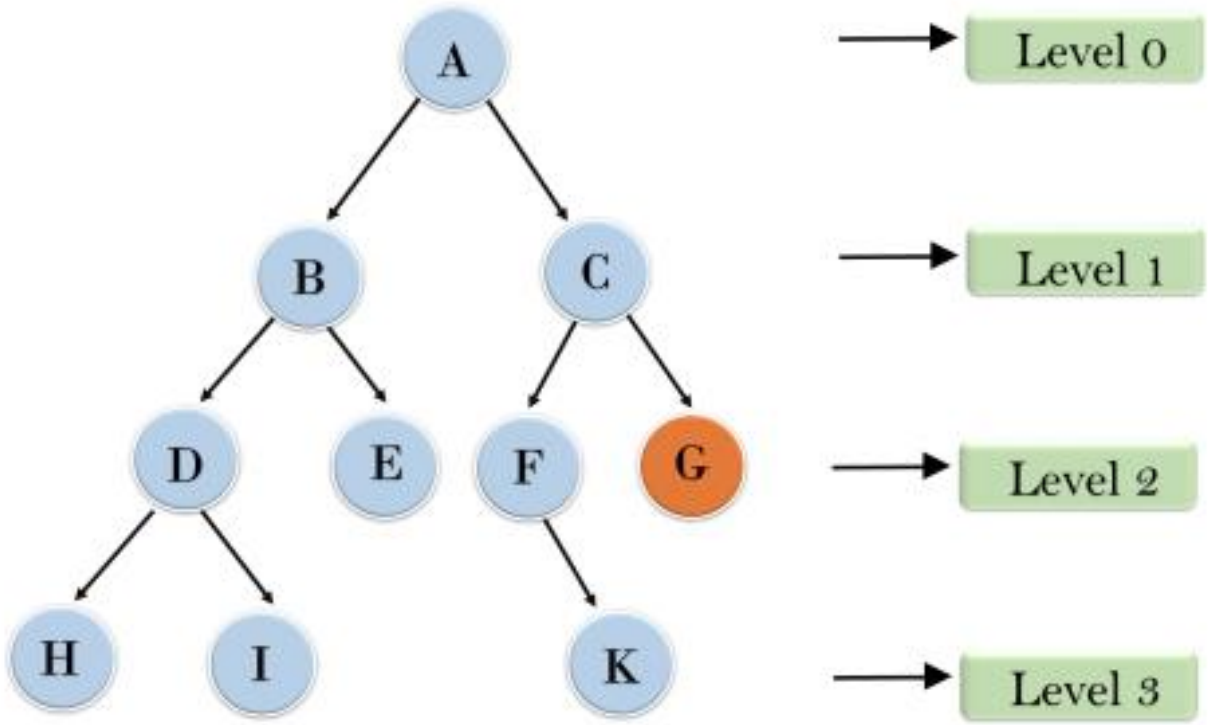
Depth-Limited Search Algorithm

- **Completeness:** DLS search algorithm is complete if the solution is above the depth-limit.
- **Time Complexity:** Time complexity of DLS algorithm is $O(b^{\ell})$.
- **Space Complexity:** Space complexity of DLS algorithm is $O(b \times \ell)$.
- **Optimal:** Depth-limited search can be viewed as a special case of DFS, and it is also not optimal even if $\ell > d$.

Iterative Deepening Depth-First Search

- 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.
- 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.

Iterative Deepening Depth-First Search



Iterative Deepening Depth-First Search

- 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 if the 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.

Thank You