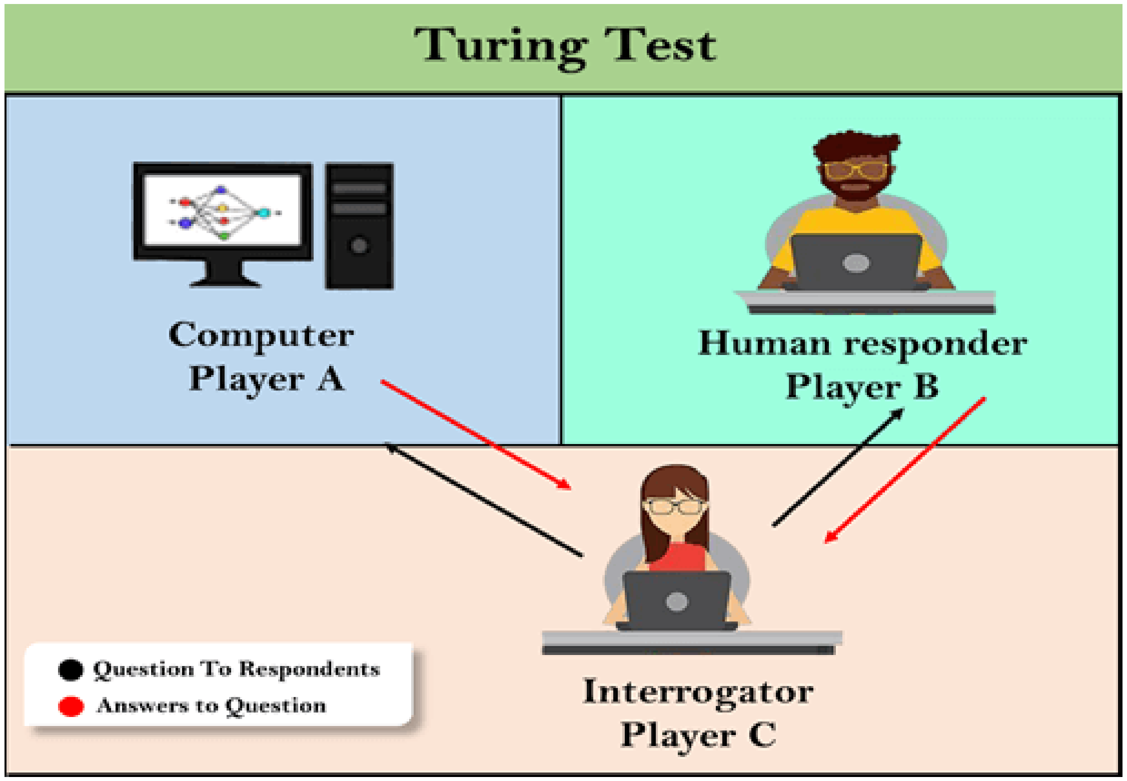
Artificial Intelligence

Dr. Mubashir Ahmad (Ph.D.)

Turing Test in AI

- In 1950, Alan Turing introduced a test to check whether a machine can think like a human or not, this test is known as the Turing Test. In this test, Turing proposed that the computer can be said to be an intelligent if it can mimic human response under specific conditions.
- Turing Test was introduced by Turing in his 1950 paper, "Computing Machinery and Intelligence," which considered the question, "Can Machine think?"

Turing Test in AI



If the machine’s responses are indistinguishable from those of the human, it passes the Turing Test.

Turing Test in AI

Ques:- what do you mean by Turing Test? How it can be performed?

Coined by Alan Turing in 1950.

"Turing test is used to determine "whether or not machines can think intelligently like humans".

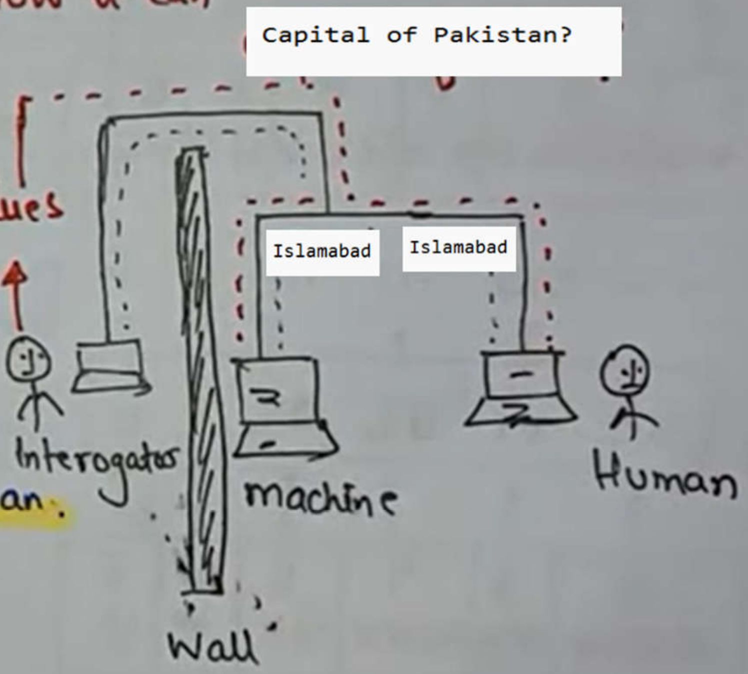
Basic Configuration:-

↳ There will be a human Interrogator on one side of wall and other side a machine and human.

↳ Machine Intelligent ⇒ when human interrogator can't distinguish response given by machine and human.

↳ Knowledge Base.

→ Examiner



Chatbots to attempt the Turing test:

- **ELIZA:** ELIZA was a Natural language processing computer program created by Joseph Weizenbaum. It was created to demonstrate the ability of communication between machine and humans. It was one of the first chattbots, which has attempted the Turing Test.
- **Parry:** Parry was a chatterbot created by Kenneth Colby in 1972. Parry was designed to simulate a person with **Paranoid schizophrenia**(most common chronic mental disorder). Parry was described as "ELIZA with attitude." Parry was tested using a variation of the Turing Test in the early 1970s.
- **Eugene Goostman:** Eugene Goostman was a chatbot developed in Saint Petersburg in 2001. This bot has competed in the various number of Turing Test. In June 2012, at an event, Goostman won the competition promoted as largest-ever Turing test content, in which it has convinced 29% of judges that it was a human.

Features required for a machine to pass the Turing test:

- **Natural language processing:** NLP is required to communicate with Interrogator in general human language like English.
- **Knowledge representation:** To store and retrieve information during the test.
- **Automated reasoning:** To use the previously stored information for answering the questions.
- **Machine learning:** To adapt new changes and can detect generalized patterns.
- **Vision (For total Turing test):** To recognize the interrogator actions and other objects during a test.
- **Motor Control (For total Turing test):** To act upon objects if requested.

For General Knowledge

What is the Turing Award?

Turing Award is an annual prize given by the Association for Computing Machinery (ACM) to "an individual selected for contributions of a technical nature made to the computing community". The Turing Award is recognized as the "highest distinction in computer science" and "Nobel Prize of Computing".

Problem Solving Techniques

- In artificial intelligence, problems can be solved by using searching algorithms, evolutionary computations, knowledge representations, etc.
- In general, searching is referred to as finding information one needs.

Properties of search algorithms

- **Completeness**
- A search algorithm is said to be complete when it gives a solution or returns any solution for a given random input.
- **Optimality**
- If a solution found is best (lowest path cost) among all the solutions identified, then that solution is said to be an optimal one.
- **Time complexity**
- The time taken by an algorithm to complete its task is called time complexity. If the algorithm completes a task in a lesser amount of time, then it is an efficient one.
- **Space complexity**
- It is the maximum storage or memory taken by the algorithm at any time while searching.
- These properties are also used to compare the efficiency of the different types of searching algorithms.

Types of search algorithms

- Based on the search problems, we can classify the search algorithm as
- Uninformed search
- Informed search

Uninformed search algorithms

- The uninformed search algorithm does not have any domain knowledge such as closeness, location of the goal state, etc. it behaves in a **brute-force way**. It only knows the information about how to traverse the given tree and how to find the goal state. This algorithm is also known as the Blind search algorithm or Brute -Force algorithm.

Brute Force

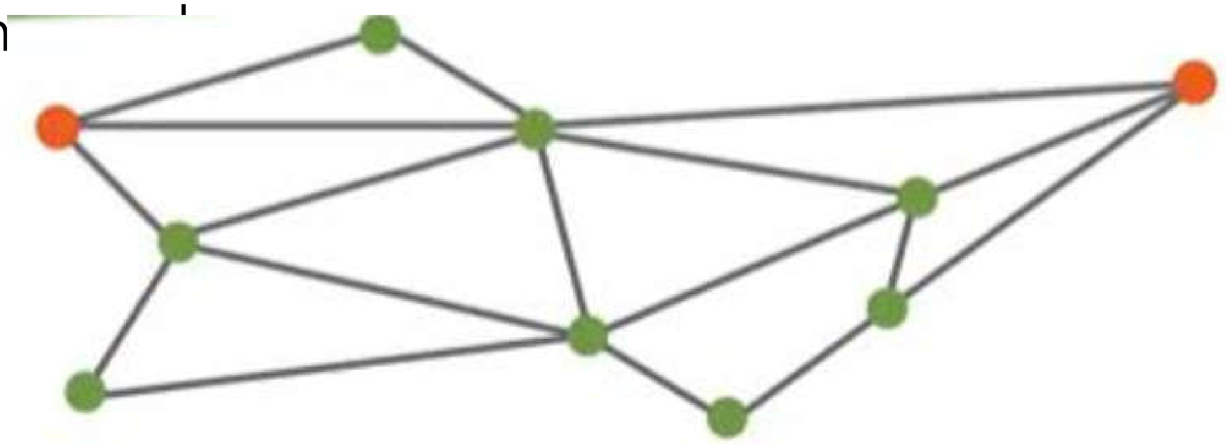
- Brute force is a straightforward approach to solve a problem based on the problem’s statement. It is considered as one of the easiest approach to apply and is useful for solving small-size instances of a problem.
- Brute Force Algorithms refers to a programming style that does not include any shortcuts to improve performance, but instead relies on sheer computing power to try all possibilities until the solution to a problem is found.

Brute Force

- A classic example is the traveling salesman problem (TSP). Suppose a salesman needs to visit 10 cities across the country.
- How does one determine the order in which cities should be visited such that the total distance traveled is minimized? The brute force solution is simply to calculate the total distance for every possible route and then select the shortest one.

Brute Force

- Path



Brute Force

```
int sum(int n)
  {
   int sum =0;
   for(int i=1; i<=n; i++)
  {
   sum += i;
   }
   return(sum);
}
```

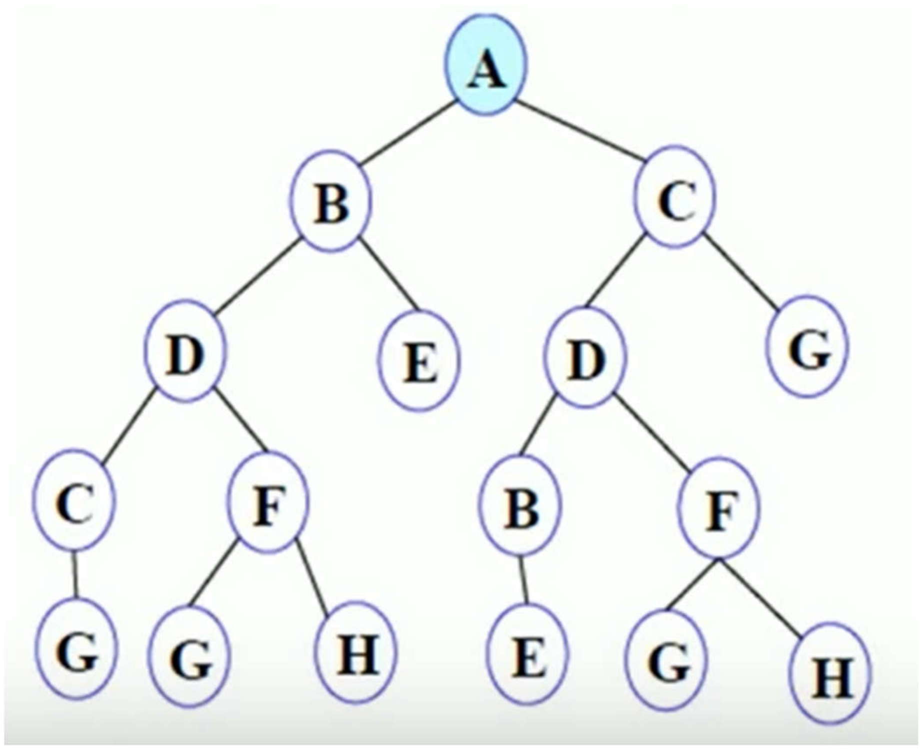
Uninformed search algorithms

- The uninformed search strategies are as follows.
- Breadth-first search
- Depth-first search

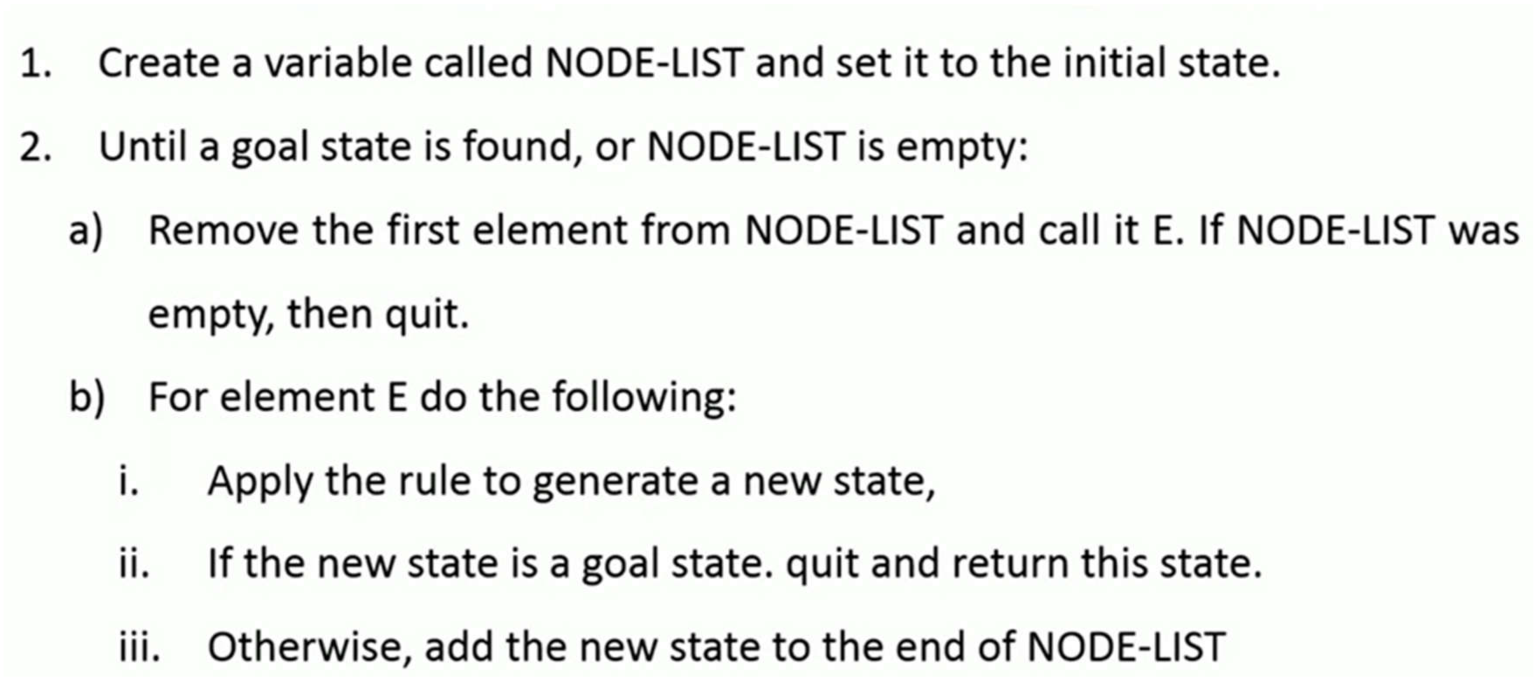
Breadth-First Search (BFS)

- This is a graph search algorithm in AI that traverses breadthwise to search for the goal in a tree. It begins searching from the root node and expands the successor node before expanding further along breadthwise and traversing those nodes rather than searching depth-wise.

Breadth-First Search (BFS) Example



Breadth-First Search (BFS) Example

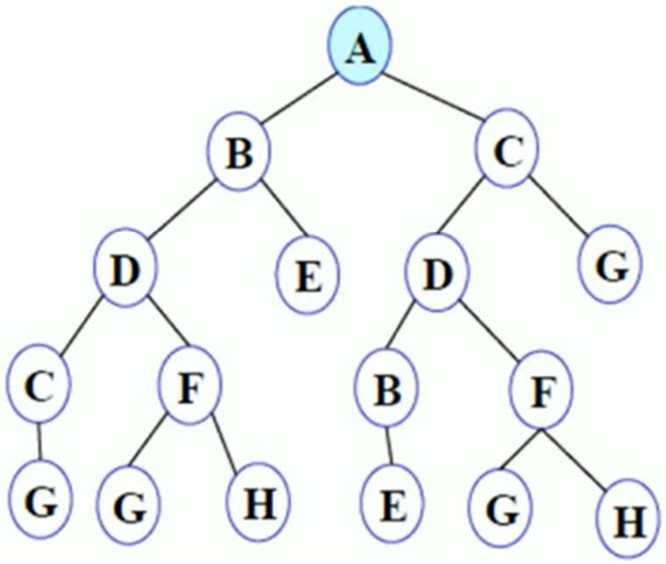


1. Create a variable called NODE-LIST and set it to the initial state.
2. Until a goal state is found, or NODE-LIST is empty:
   a) Remove the first element from NODE-LIST and call it E. If NODE-LIST was empty, then quit.
   b) For element E do the following:
      i. Apply the rule to generate a new state,
      ii. If the new state is a goal state. quit and return this state.
      iii. Otherwise, add the new state to the end of NODE-LIST

Breadth-First Search (BFS) Example

- **Step 1:** Initially NODE-LIST contains only one node corresponding to the source state A.

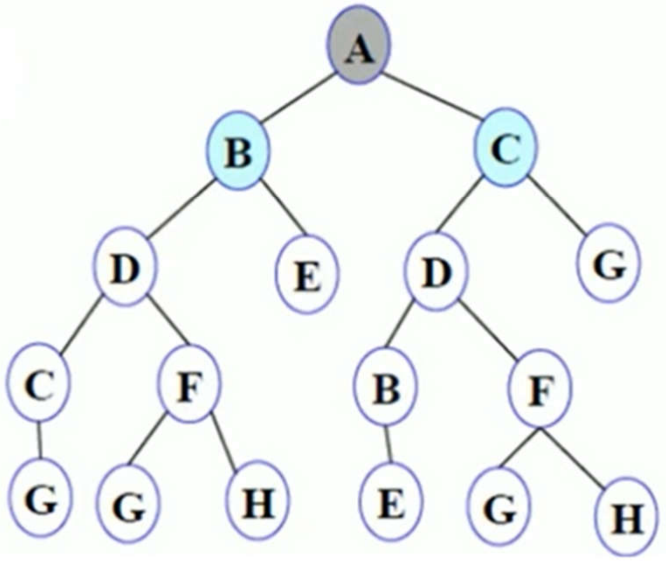
- **NODE-LIST: A**



Breadth-First Search (BFS) Example

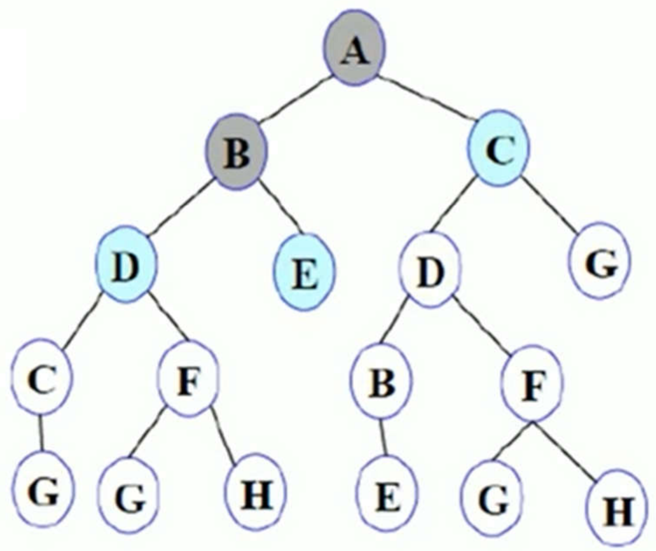
- **Step 2:** A is removed from NODE-LIST. The node is expanded, and its children B and C are generated. They are placed at the back of NODE-LIST.

- **NODE-LIST: B C**



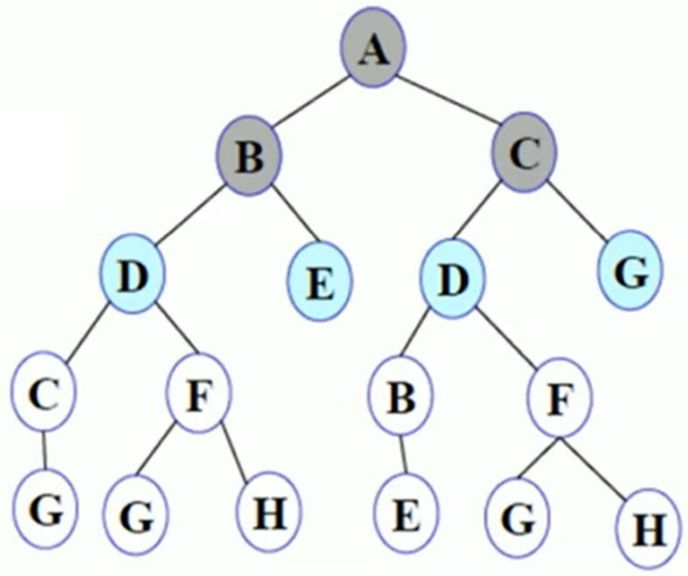
Breadth-First Search (BFS) Example

- **Step 3:** Node B is removed from NODE-LIST and is expanded. Its children D, E are generated and put at the back of NODE-LIST.
- NODE-LIST: C D E



Breadth-First Search (BFS) Example

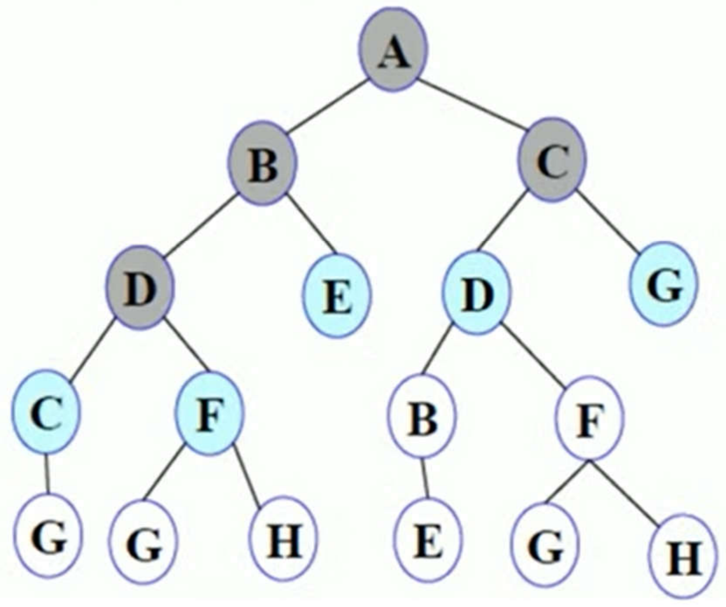
- **Step 4:** Node C is removed from NODE-LIST and is expanded. Its children D and G are added to the back of NODE-LIST.
- **NODE-LIST: D E D G**



Breadth-First Search (BFS) Example

- **Step 5**: Node D is removed from NODE-LIST. Its children C and F are generated and added to the back of NODE-LIST.

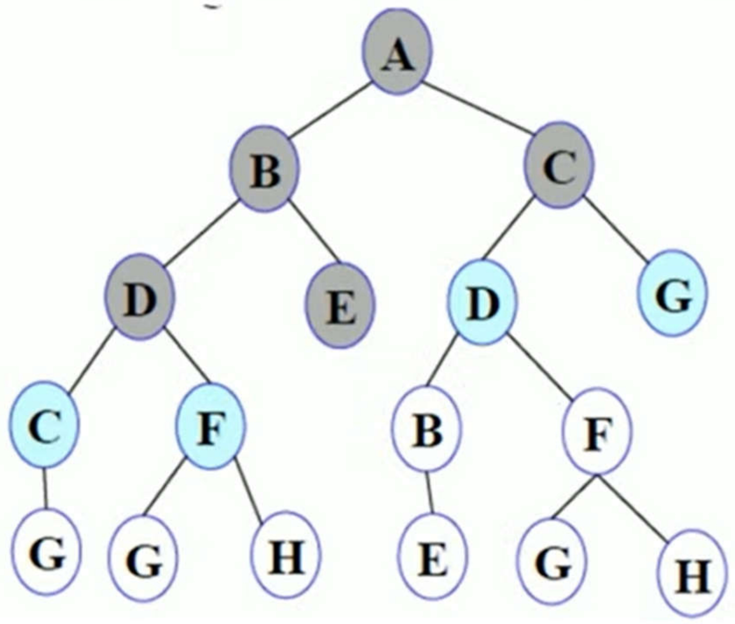
- **NODE-LIST:**
- E D G C F



Breadth-First Search (BFS) Example

- **Step 6**: Node E is removed from NODE-LIST. It has no children.

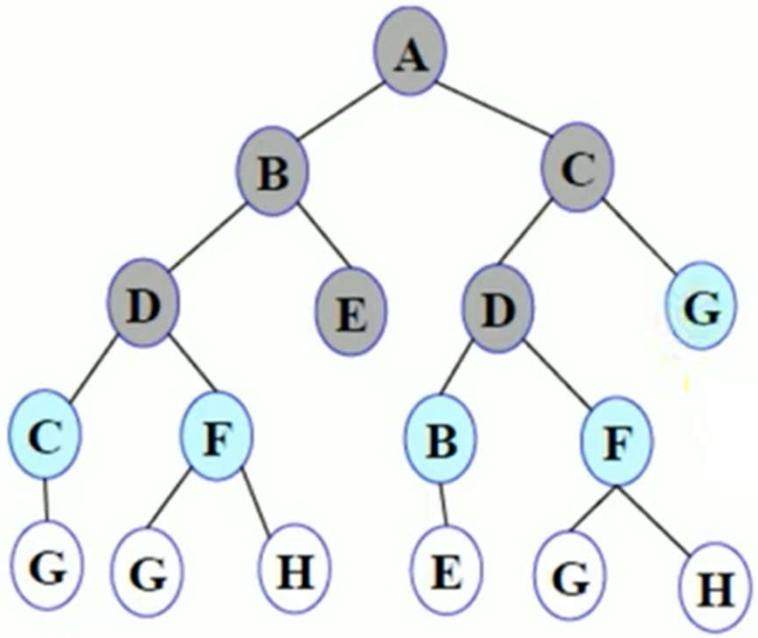
- **NODE-LIST:**
- **D G C F**



Breadth-First Search (BFS) Example

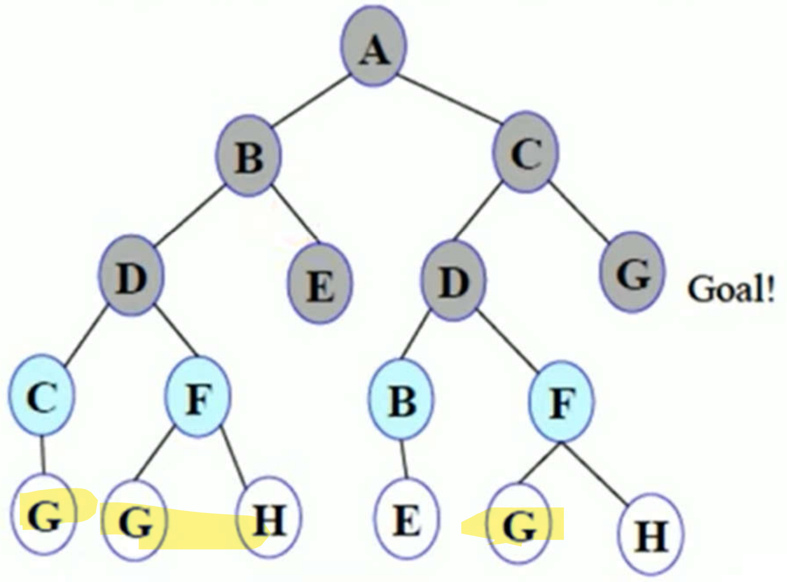
- **Step 7**: D is expanded; B and F are put in OPEN.

- **NODE-LIST:**
- **G C F B F**

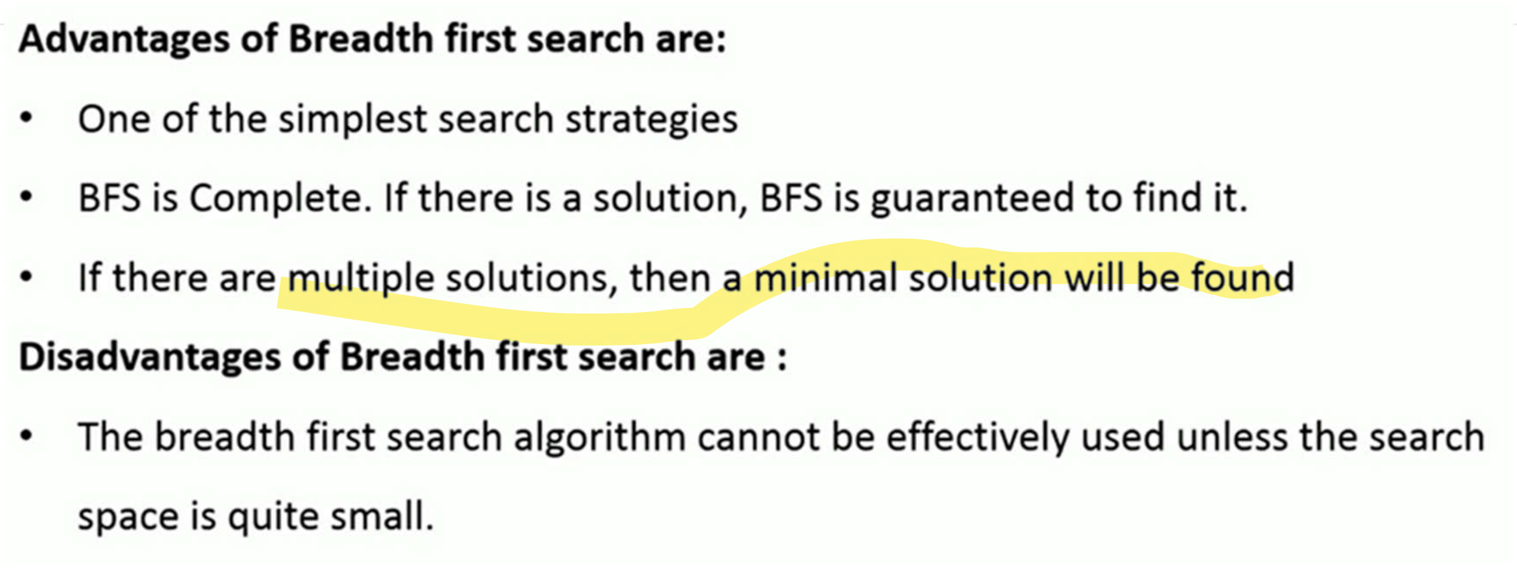


Breadth-First Search (BFS) Example

- **Step 8**: G is selected for expansion. It is found to be a goal node.
- Hence the algorithm returns the path A - C - G by following the parent pointers of the node corresponding to G.



Breadth-First Search (BFS) Example



Advantages of Breadth first search are:

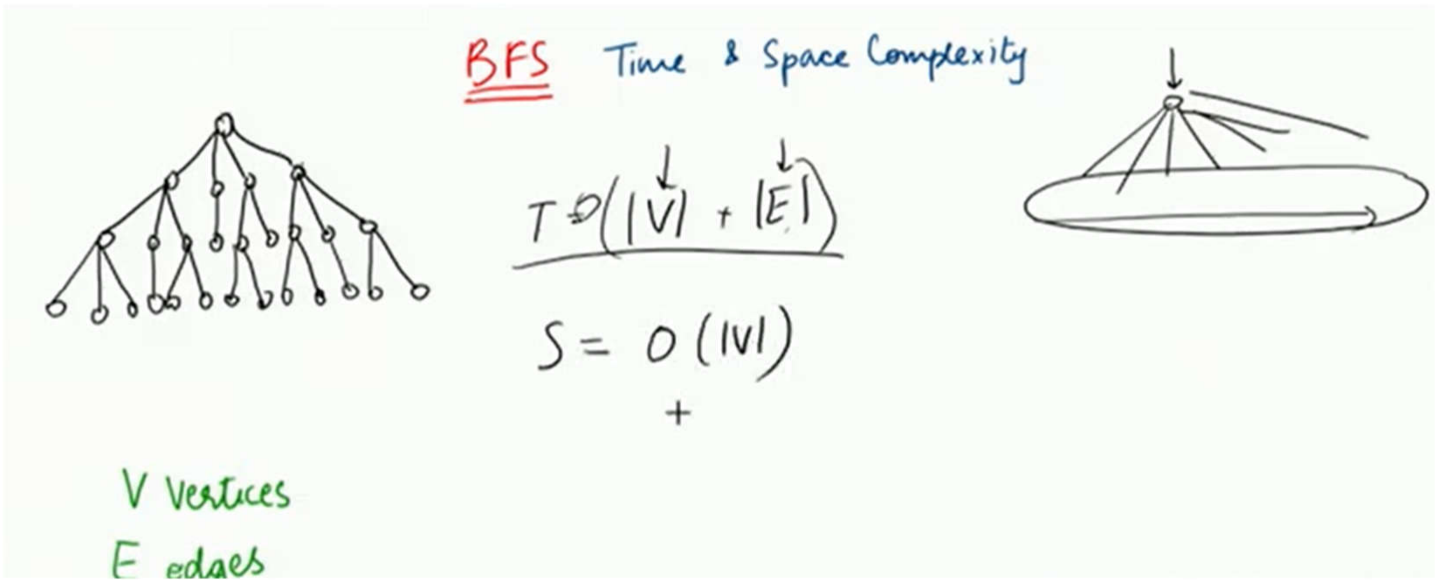
- One of the simplest search strategies
- BFS is Complete. If there is a solution, BFS is guaranteed to find it.
- If there are multiple solutions, then a minimal solution will be found

Disadvantages of Breadth first search are :

- The breadth first search algorithm cannot be effectively used unless the search space is quite small.

Breadth-First Search (BFS)

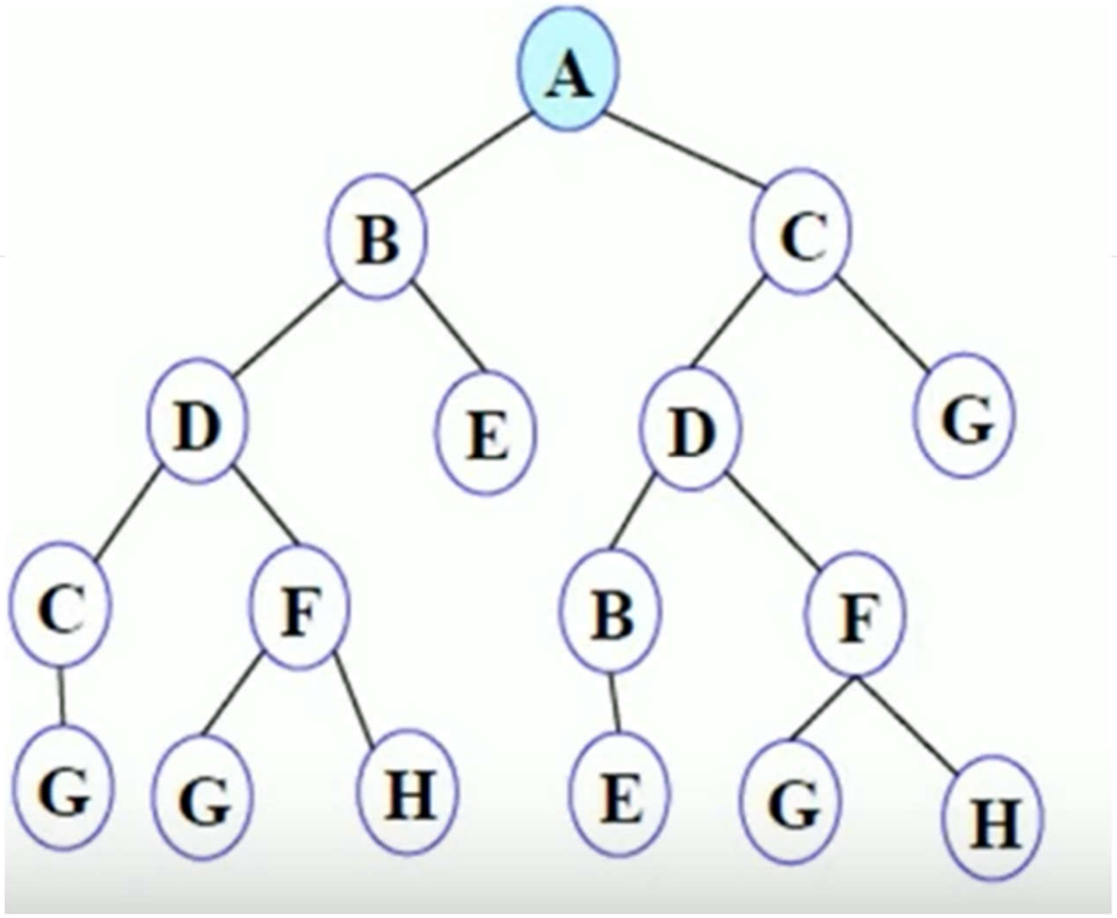
- The time complexity can be expressed as O (| V | + | E |) , since every vertex and every edge will be explored in the worst case. | V | is the number of vertices and | E | is the edges.



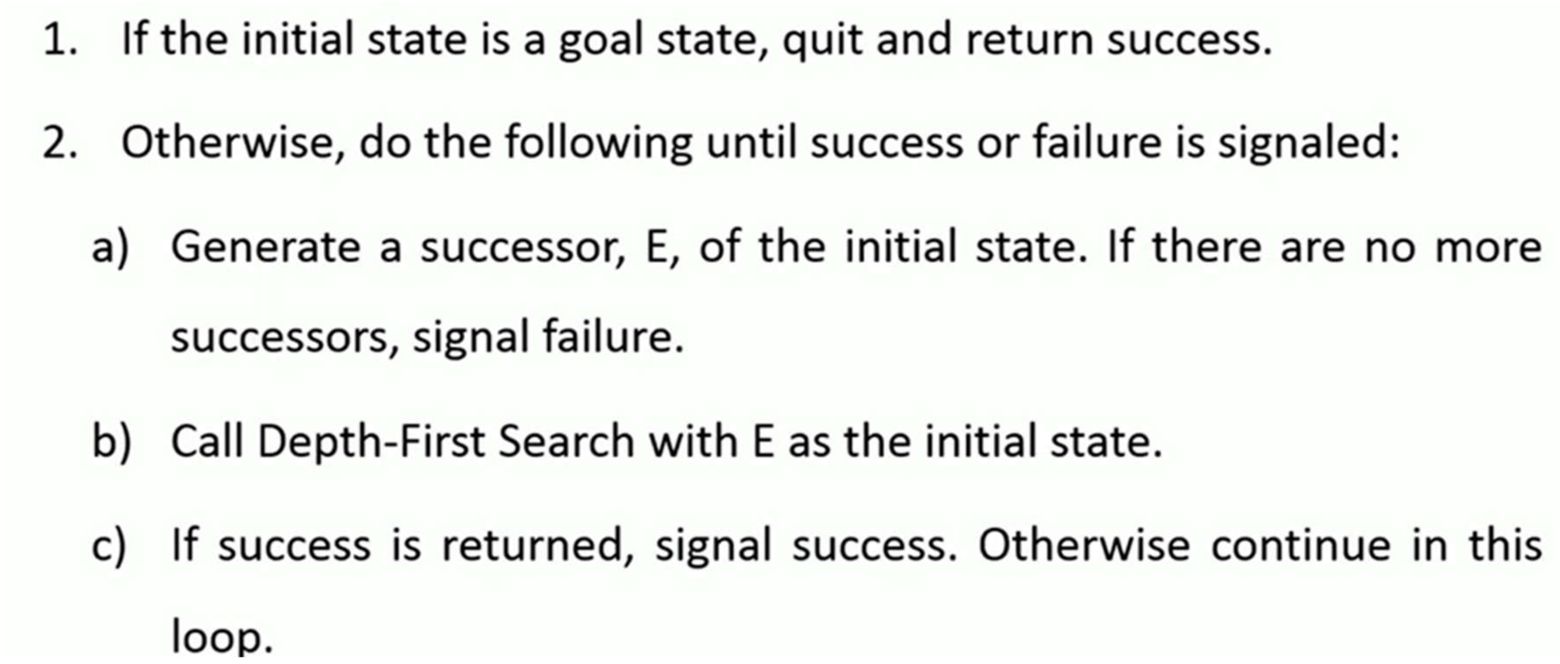
Depth First Search (DFS)

- It is a search algorithm where the search tree will be traversed from the root node. It will be traversing, searching for a key at the leaf of a particular branch. If the key is not found, the searcher retraces its steps back (backtracking) to the point from where the other branch was left unexplored, and the same procedure is repeated for that other branch.

Depth First Search (DFS)



Depth First Search (DFS)

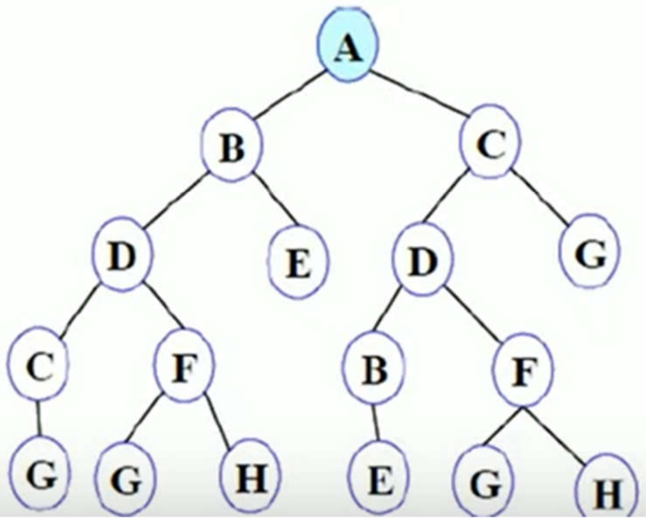


1. If the initial state is a goal state, quit and return success.
2. Otherwise, do the following until success or failure is signaled:
    a) Generate a successor, E, of the initial state. If there are no more successors, signal failure.
    b) Call Depth-First Search with E as the initial state.
    c) If success is returned, signal success. Otherwise continue in this loop.

Depth First Search (DFS)

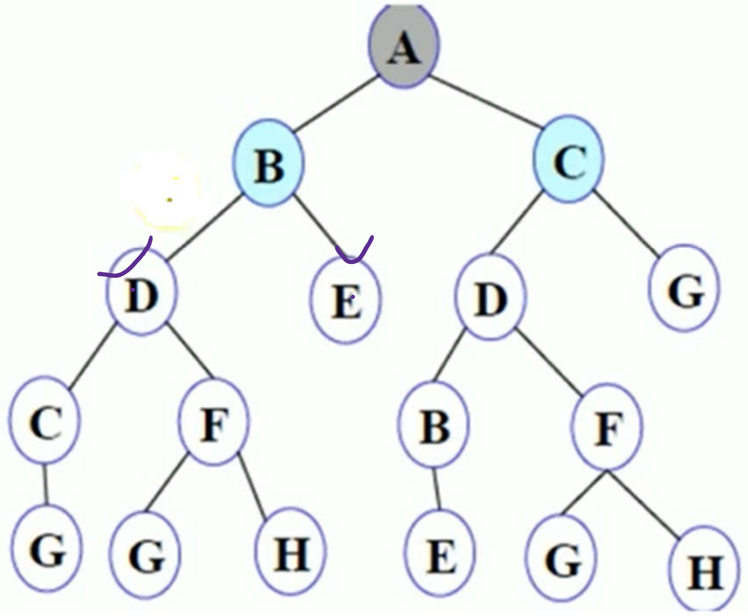
- **Step 1:** Initially NODE-LIST contains only one node corresponding to the source state A.

- **NODE-LIST: A**



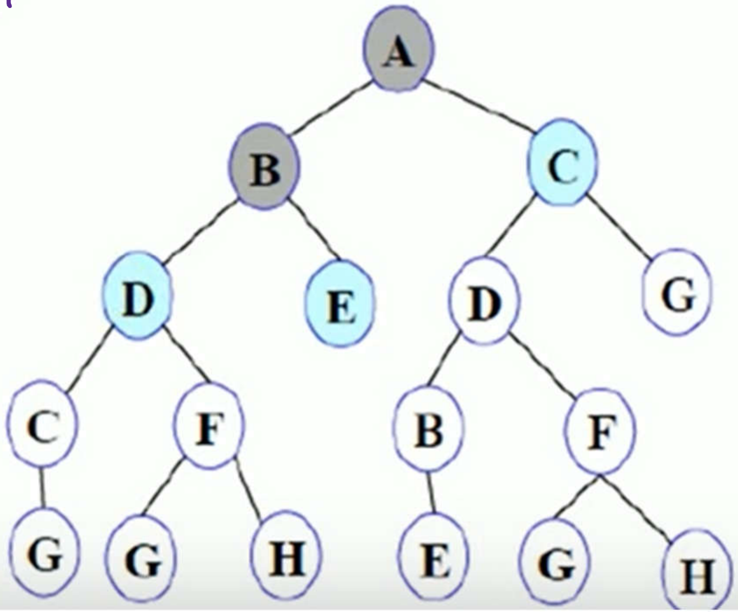
Depth First Search (DFS)

- **Step 2:** A is removed from NODE-LIST . A is expanded and its children B and C are put in front of NODE-LIST .
- **NODE-LIST: B C**



Depth First Search (DFS)

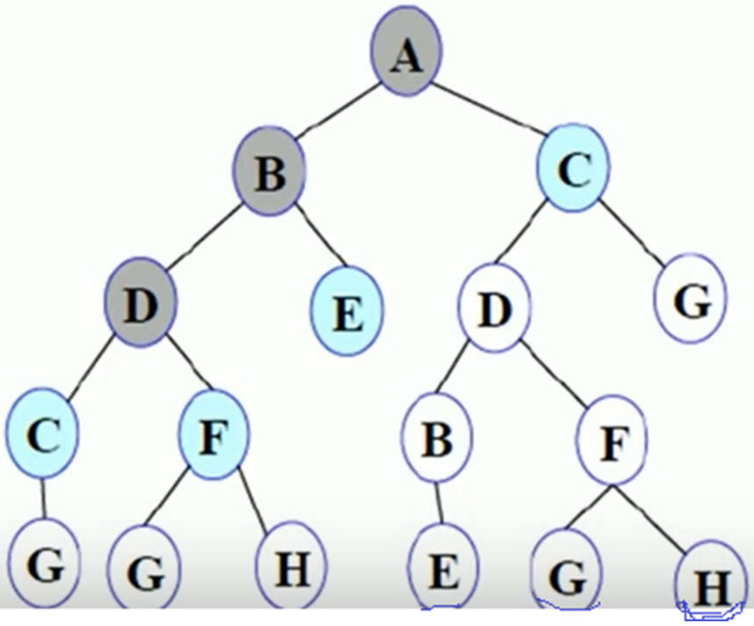
- **Step 3:** Node B is removed from NODE-LIST , and its children D and E are pushed in front of NODE-LIST
- NODE-LIST: D E C



Depth First Search (DFS)

- **Step 4:** Node D is removed from NODE-LIST . C and F are pushed in front of NODE-LIST .

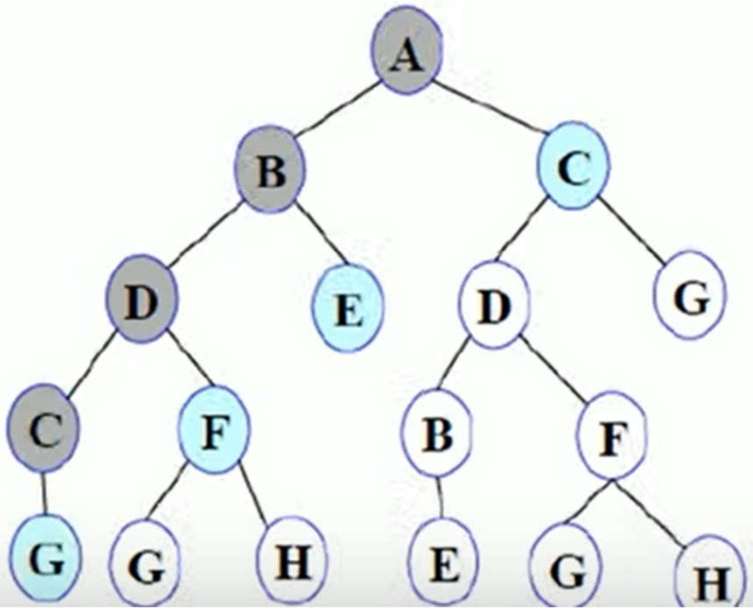
- **NODE-LIST:**
- **C F E C**



Depth First Search (DFS)

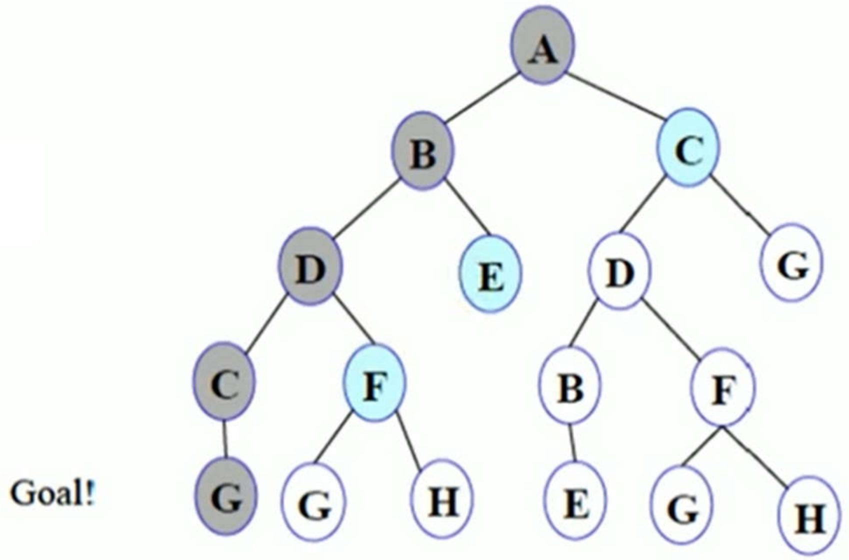
- **Step 5:** Node C is removed from NODE-LIST . Its child G is pushed in front of NODE-LIST .

- **NODE-LIST:**
- **G F E C**



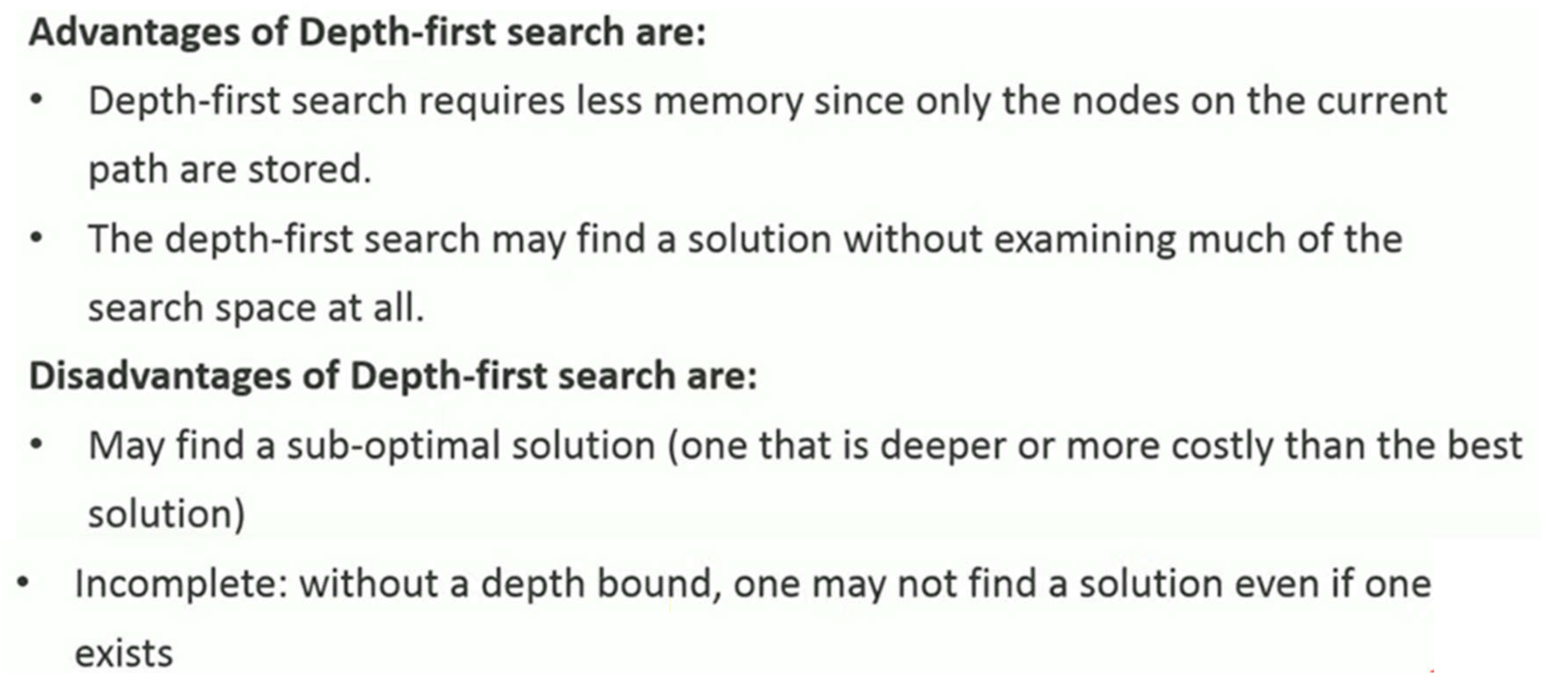
Depth First Search (DFS)

- **Step 6:** Node G is expanded and found to be a goal node.
- **NODE-LIST:** ***G*** **F E C**



- The solution path **A-B-D-C-G** is returned and the algorithm terminates.

Depth First Search (DFS)



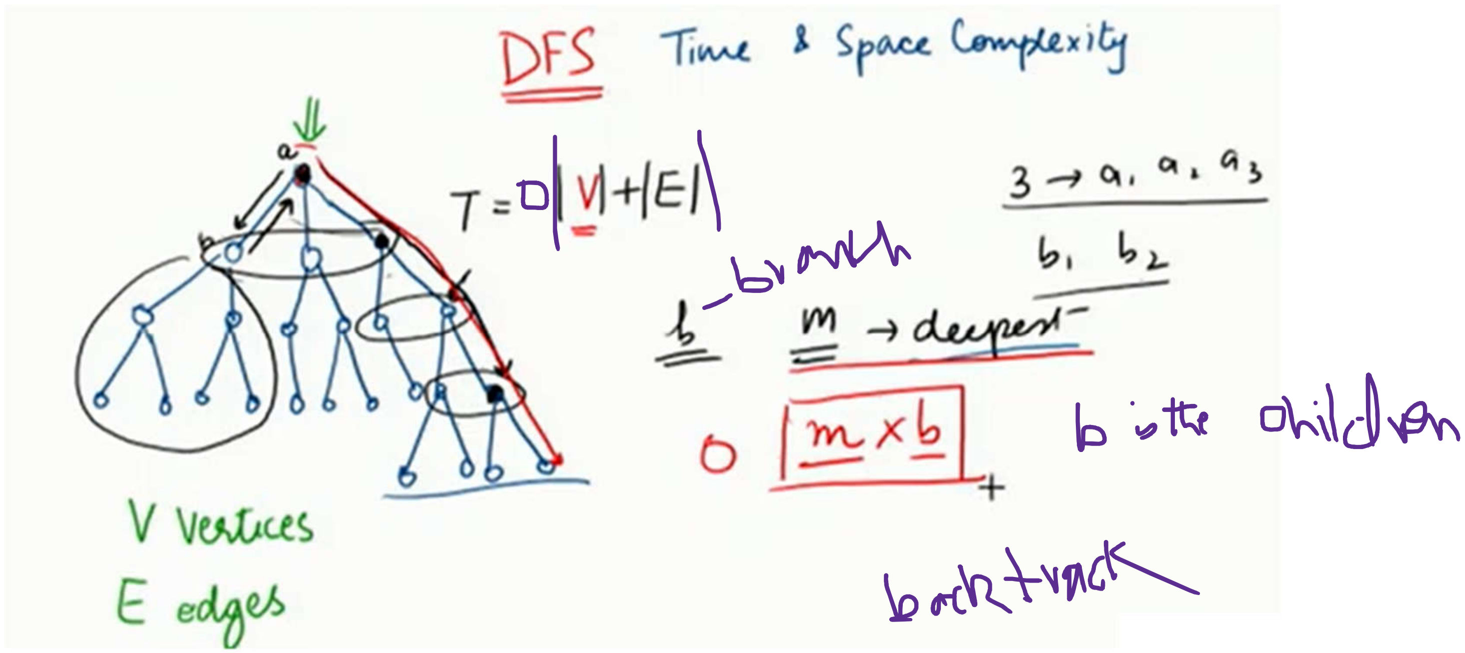
Advantages of Depth-first search are:

- Depth-first search requires less memory since only the nodes on the current path are stored.
- The depth-first search may find a solution without examining much of the search space at all.

Disadvantages of Depth-first search are:

- May find a sub-optimal solution (one that is deeper or more costly than the best solution)
- Incomplete: without a depth bound, one may not find a solution even if one exists

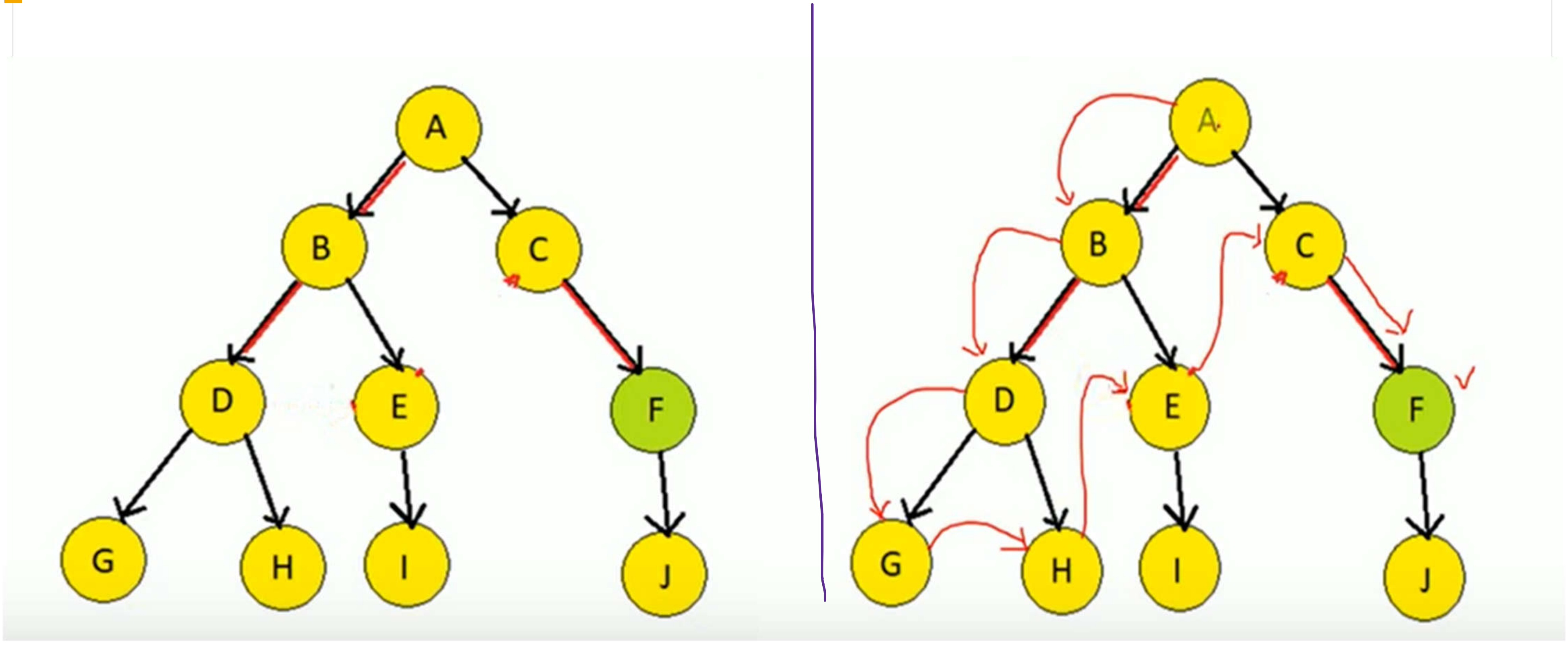
Depth First Search (DFS)



Depth Limited Search Algorithm

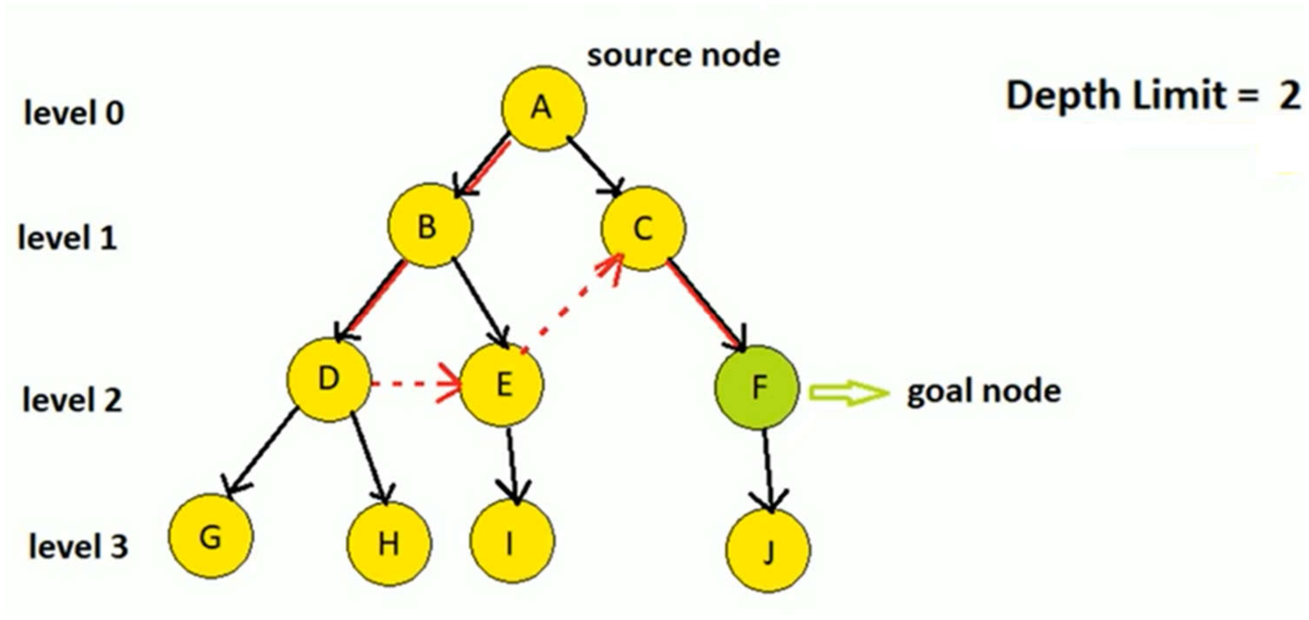
- It’s a modification of Depth First Search Algorithm.
- A depth-limited search algorithm is similar to depth-first search with a predetermined limit. Depth-limited search can solve the drawback of the infinite path in the Depth-first search. In this algorithm, the node at the depth limit will treat as it has no successor nodes further.

Problem in Depth Search Algorithm Example

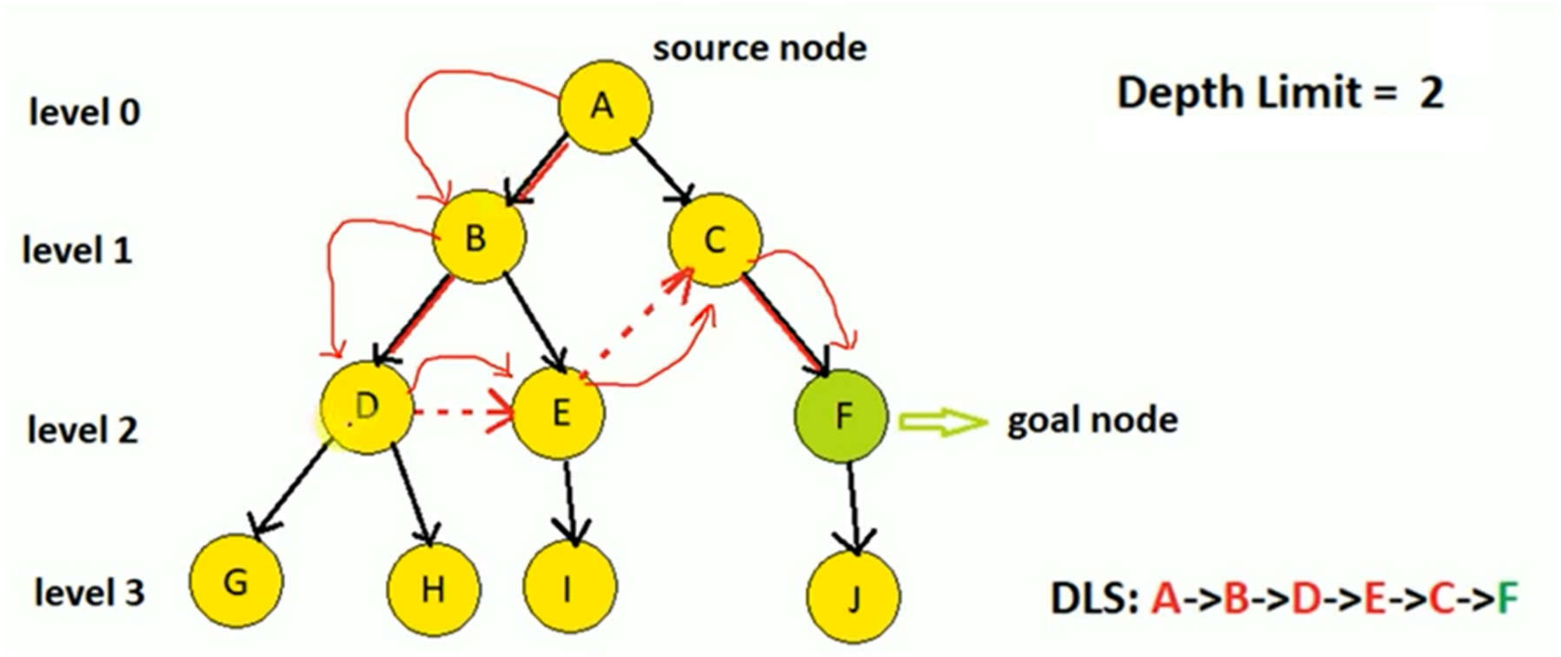


DFS: A->B->D->G->H->E->I->C->F

Depth Limited Search Algorithm



Depth Limited Search Algorithm



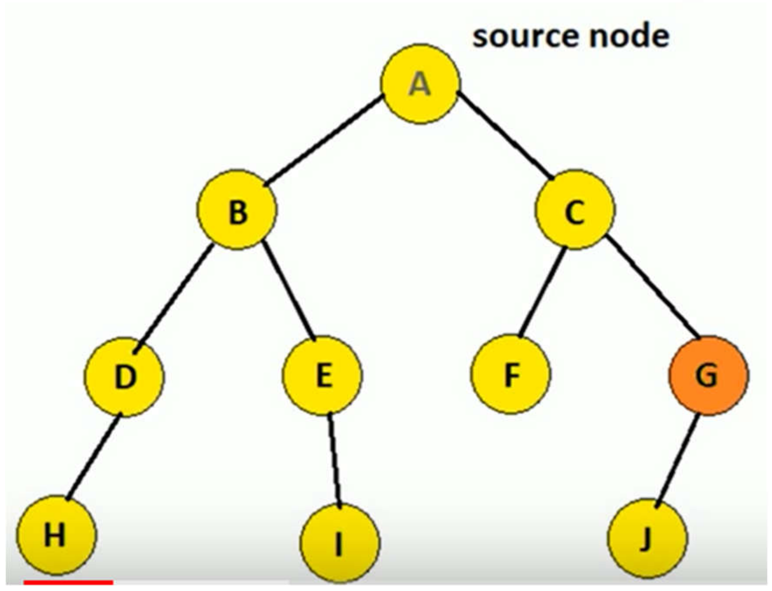
Depth Limited Search Algorithm

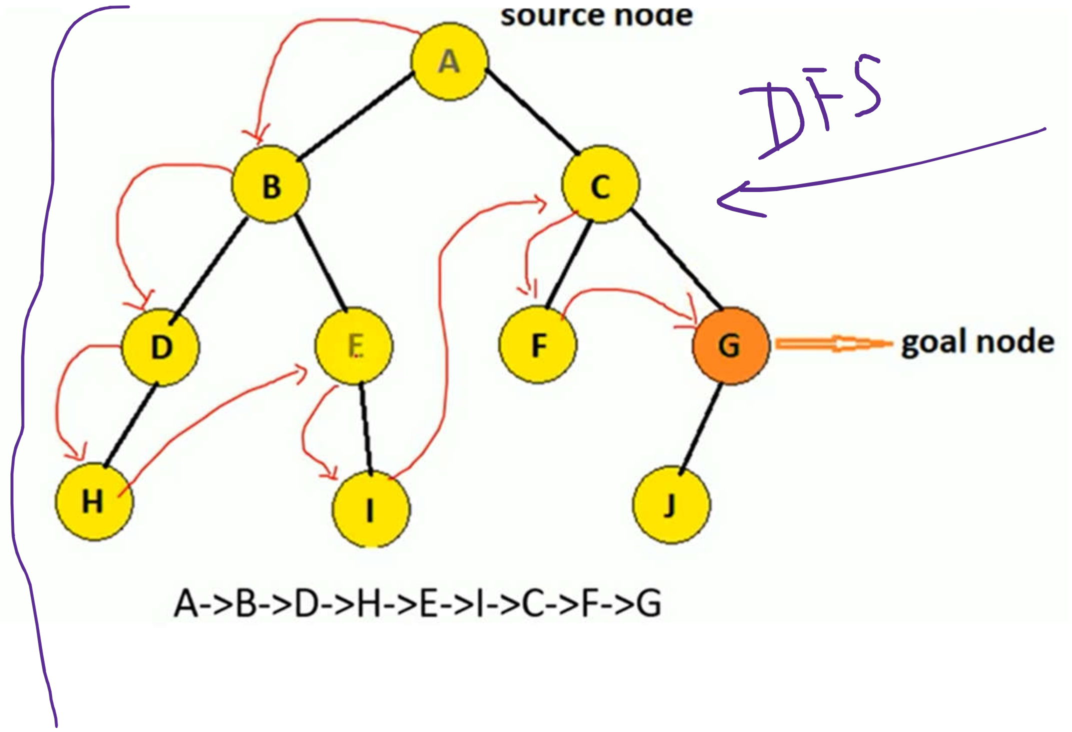
- **Advantages:**
- Depth-limited search is Memory efficient.
- **Disadvantages:**
- Depth-limited search also has a disadvantage of incompleteness.
- It may not be optimal if the problem has more than one solution.

Iterative Deepening DFS

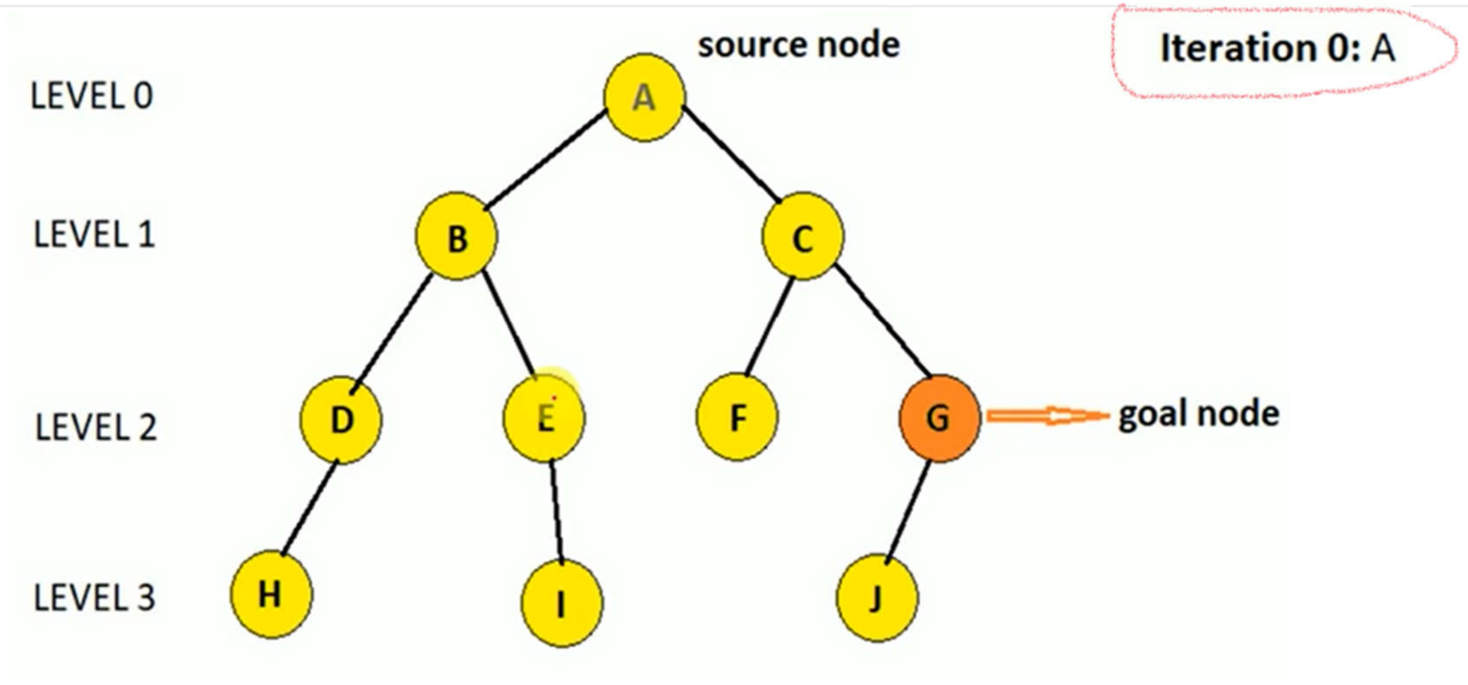
- The iterative deepening algorithm is a combination of DFS and BFS algorithms. This search algorithm finds out the best depth limit and does it by gradually increasing the limit until a goal is found.
- This algorithm performs depth-first search up to a certain "depth limit", and it keeps increasing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

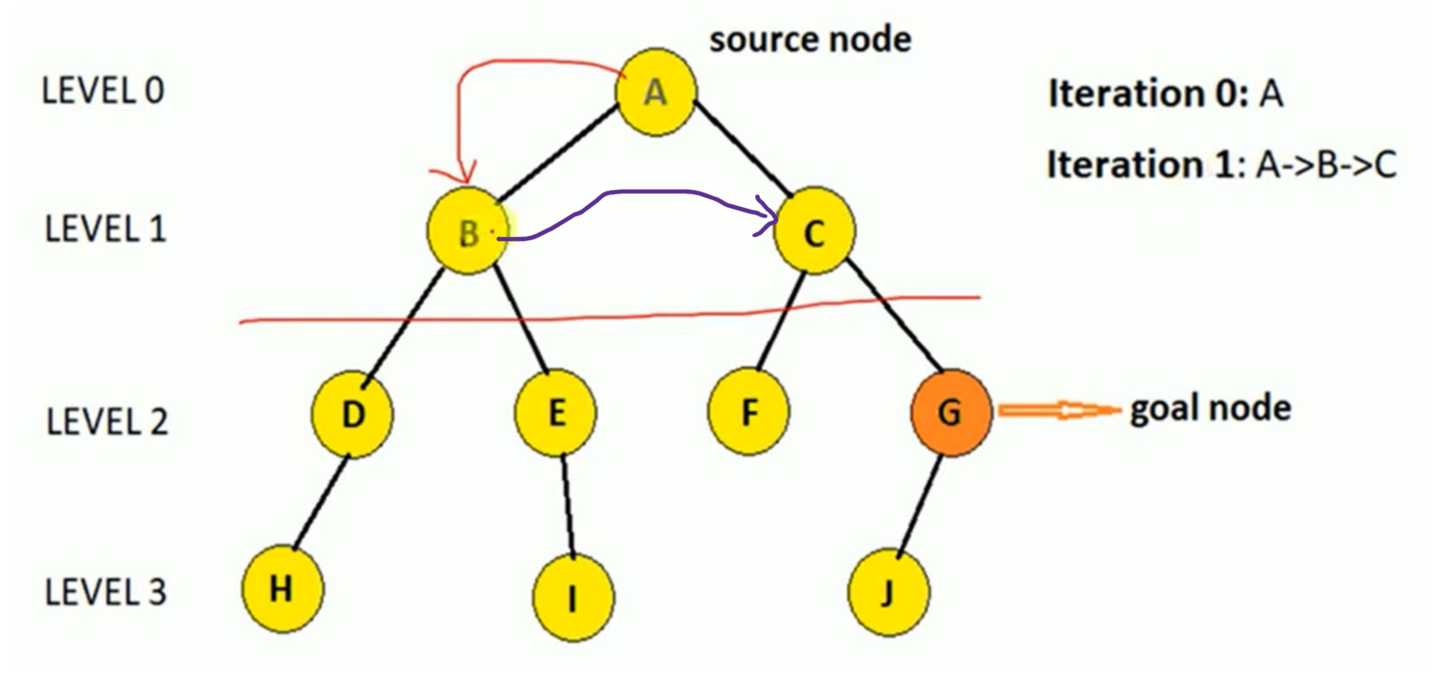




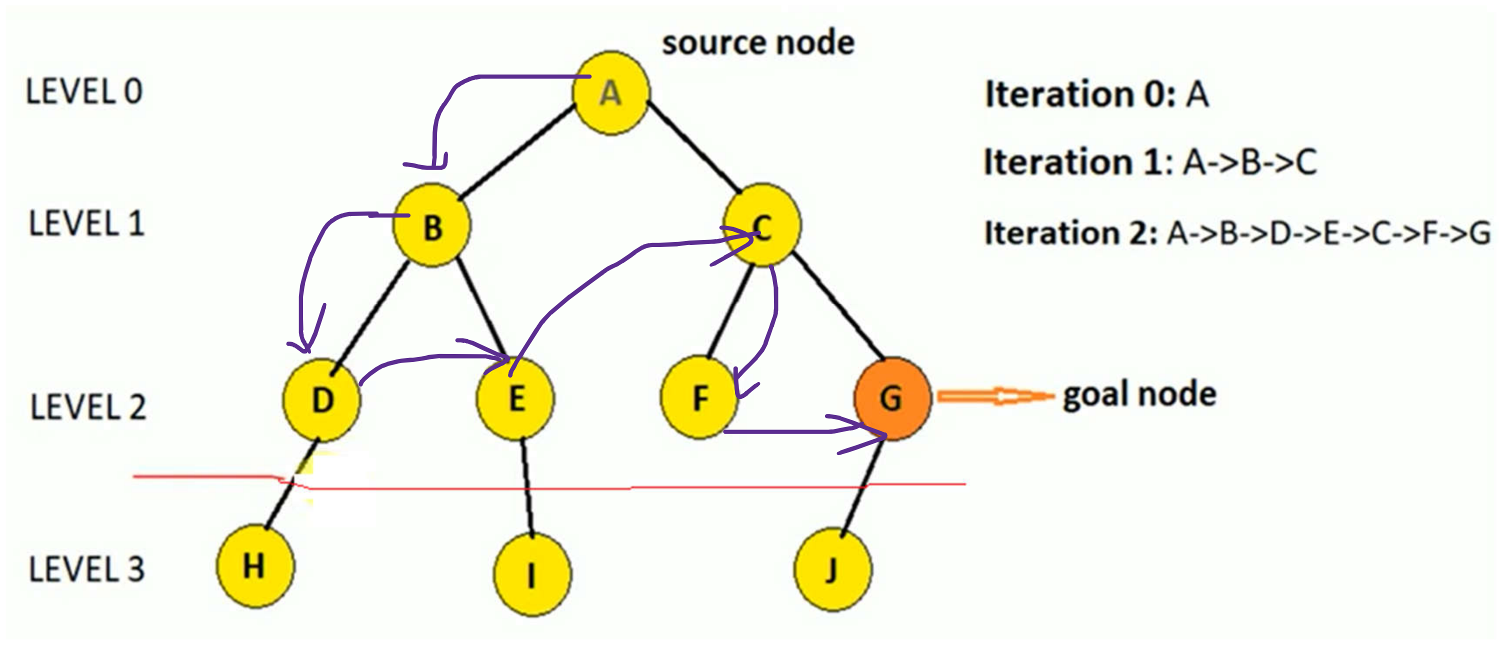
Iterative Deepening DFS Example



Iterative Deepening DFS Example



Iterative Deepening DFS Example



Iterative Deepening DFS

- **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.