IARE
INSTITUTE OF
AERONAUTICAL ENGINEERING

Course Title - Logic Programming for Artificial Intelligence

Topic Title – Problem Solving Methods & Introduction to Search Strategies

Presenter's Name – Ms. Bidyutlata Sahoo
Presenter's ID – IARE11028
Department Name – CSE (AI & ML)
Lecture Number - 01
Presentation Date – 21/09/2024

Course Outcome

At the end of the course, students should be able to:

CO3: Interpret uninformed and informed search strategies, and select the appropriate approach for different AI problems.

Topic Learning Outcome

1. **Make use of** Problem-solving methods in AI to find solutions for complex issues.
2. **Identify** the search strategies in AI which provide systematic methods for exploring possible solutions to problems.

Problem Solving Methods

Problem Solving Methods

- The method of solving problem through AI ***involves the process of defining the search space, deciding start and goal states then finding the path from start to goal state through search space.***
- The movement from start state to goal state is guided by set of rules specifically designed for that particular problem.

Problem

- It is the question which is to be solved. For solving a problem it needs to be ***precisely defined.***
- The definition means, ***defining the start state, goal state, other valid states and transitions.***
- ***Finding the Solution***
- After representation of the problem and related knowledge in the suitable format, the ***appropriate methodology is chosen*** which uses the knowledge and transforms the start state to goal state.
- The Techniques of finding the solution are called ***search techniques.***
- Various search techniques are developed for this purpose.

Problem Solving Agents

- In Artificial Intelligence, ***Search techniques*** are universal problem-solving methods.
- ***Rational agents or Problem-solving agents*** in AI mostly used these search strategies or algorithms to solve a specific problem and provide the best result.
- Problem-solving agents are the ***goal-based agents*** and use atomic representation. So here, we will discuss various problem-solving search algorithms.

Search Algorithm Terminologies:

Search: Searching is a step by step procedure to solve a search-problem in a given search space. A search problem can have three main factors:

- ***Search Space***: Search space represents *a set of possible solutions*, which a system may have.
- ***Start State***: It is a *state from which begins the search.*
- ***Goal test***: It is a function *where agent observe the current state and returns whether the goal state is achieved or not*.

Search Algorithm Terminologies: (Contd..)

- ***Search tree***: A tree representation of search problem is called Search tree. The *root of the search tree is the root node* which is corresponding to the initial state.
- ***Actions***: It gives the description of *all the available actions* to the agent.
- ***Transition model***: A description of *what each action do*, can be represented as a transition model.

Search Algorithm Terminologies: (Contd..)

- ***Path Cost***: It is a function which *assigns a numeric cost to each path.*
- ***Solution***: It is an *action sequence* which leads from the start node to the goal node.
- ***Optimal Solution***: If a solution *has the lowest cost* among all solutions.

Properties of Search Algorithms:

- **Completeness**: A search algorithm is said to be completed *if it guarantees to return a solution if at least any solution exists for any random input.*
- **Optimality**: If a solution found for an algorithm is guaranteed to be the best solution (*lowest path cost*) among all other solutions, then such a solution for is said to be an optimal solution.
- **Time Complexity**: Time complexity is a *measure of time for an algorithm to complete its task.*
- **Space Complexity**: It is the *maximum storage space required at any point during the search*, as the complexity of the problem.

Search Strategies / Control Strategies

Search Strategies

- It is one of the most important component of problem solving that ***describes the order of application of the rules to the current state.*** That means:
  - Helps us to decide ***which rule to apply next.***
  - What to do ***when there are more than 1 matching rules?***

[NOTE: This question usually arises when we have more than one rule (and sometimes fewer than one rule) that will match with the left side of the current state.]

Requirements of Search Strategies

There are mainly **<u>two requirements to choose a control strategy.</u>**

- The first requirement of a good control strategy is that ***it should cause motion towards solution.***
- The second requirement of a good control strategy is that ***it should explore the solution space in a systematic manner.***

Requirements of Search Strategies (Contd..)

Example:

Consider the water jug problem. On each cycle, choose at random from among the applicable rules. This strategy is better than the first. It causes motion. It will lead to a solution eventually. But we are ***likely to arrive at the same state several times during the process and to use many more steps than are necessary***. Because the ***control strategy is not systematic***, we may explore a particular useless sequence of operators several times before we finally find a solution.

Search Directions

To solve real world problems, effective control strategy must be needed.

There are ***2 directions in which search could proceed.***

1. **Data driven search (forward chaining)** [start state → goal state]
2. **Goal driven search (backward chaining)** [goal state → start state]

Search Directions

Forward Chaining

- Forward chaining ***starts from known facts and applies inference rules to extract more data until it reaches to the goal.***
- It is a ***bottom-up approach.***
- It is known as ***data driven inference technique*** as we reach to the goal using available data.
- It applies breadth first search strategy.

Backward Chaining

- It ***starts from the goal and works backward through inference rules to find the required facts that support the goal.***
- It is a ***top down approach***.
- It is known as ***goal driven technique*** as we start from the goal and divide into subgoals to extract the facts.
- It applies depth first search strategy.

Search Directions

Forward Chaining

- Forward chaining tests for all available rules.
- It is suitable for planning, monitoring, control and interpretation application.

Backward Chaining

- Backward chaining only tests for few required rules.
- It is suitable for diagnostic, prescription and debugging applications.

Types of Search Strategies

Types of Search Strategies

Based on the search problems, there are mainly ***<u>two types of search strategies:</u>***

1. **Uninformed search (**Blind search / Exhaustive search / Brute force search)
2. **Informed search (**Heuristic search / Intelligent search)

1. Uninformed / Blind / Exhaustive search

- The uninformed search ***does not contain any domain knowledge*** such as closeness, the location of the goal.
- It ***operates in a brute-force way*** as *it only includes information about how to traverse the tree and how to identify leaf and goal nodes.*
- Uninformed search applies a way in which search tree is searched *without any information about the search space like initial state operators and test for the goal*, so it is ***also called blind search***.
- It ***examines each node of the tree until it achieves the goal node.***
- ***May not be very efficient***.

1.Uninformed / Blind / Exhaustive search (Contd..)

Following are the various types of uninformed search algorithms:

1. ***Breadth-first search***
2. ***Depth-first search***
3. ***Depth limited search***
4. ***Iterative deepening depth-first search***
5. ***Uniform cost search***
6. ***Bidirectional Search***

2. Informed / Intelligent Search

- Informed search algorithms ***use domain knowledge***. In an informed search, ***problem information is available which can guide the search.***

- Informed search strategies ***can find a solution more efficiently than an uninformed search*** strategy. Informed search is also called a ***Heuristic search.***

- A **heuristic** is a way which might not always be guaranteed for best solutions but ***guaranteed to find a good solution*** in reasonable time.

- Informed search can ***solve much complex problem*** which could not be solved in another way.

2. Informed / Intelligent Search (Contd..)

- It uses ***Heuristic function***, which maps each state to a numerical value to depict the goodness of a node.

- Represented as:

H(n)=value

(Where ,H(n) is a heuristic function and ‘n’ is the current state.)

2. Informed / Intelligent Search (Contd..)

Following are the various types of informed search algorithms:

1. ***Hill Climbing***
2. ***Best-First Search***
3. ***A* Algorithm***
4. ***AO* Algorithm***
5. ***Beam Search***
6. ***Constraint Satisfaction***

References

- Stuart Russell and Peter Norvig, “Artificial Intelligence”, 2nd edition, Pearson Education, 2003.
- Saroj Koushik, “Artificial intelligence”.
- NPTEL

IARE
INSTITUTE OF
AERONAUTICAL ENGINEERING

Thank You

IARE
INSTITUTE OF
AERONAUTICAL ENGINEERING

Course Title - Logic Programming for Artificial Intelligence

Topic Title – Uninformed Search (BFS, DFS)

Presenter's Name – Ms. Bidyutlata Sahoo
Presenter's ID – IARE11028
Department Name – CSE (AI & ML)
Lecture Number - 02
Presentation Date – 21/09/2024

Course Outcome

At the end of the course, students should be able to:

CO3: Interpret uninformed and informed search strategies, and select the appropriate approach for different AI problems.

Topic Learning Outcome

Understand the concept of uninformed search where little or no domain-specific information (heuristics) is available to guide the search process.

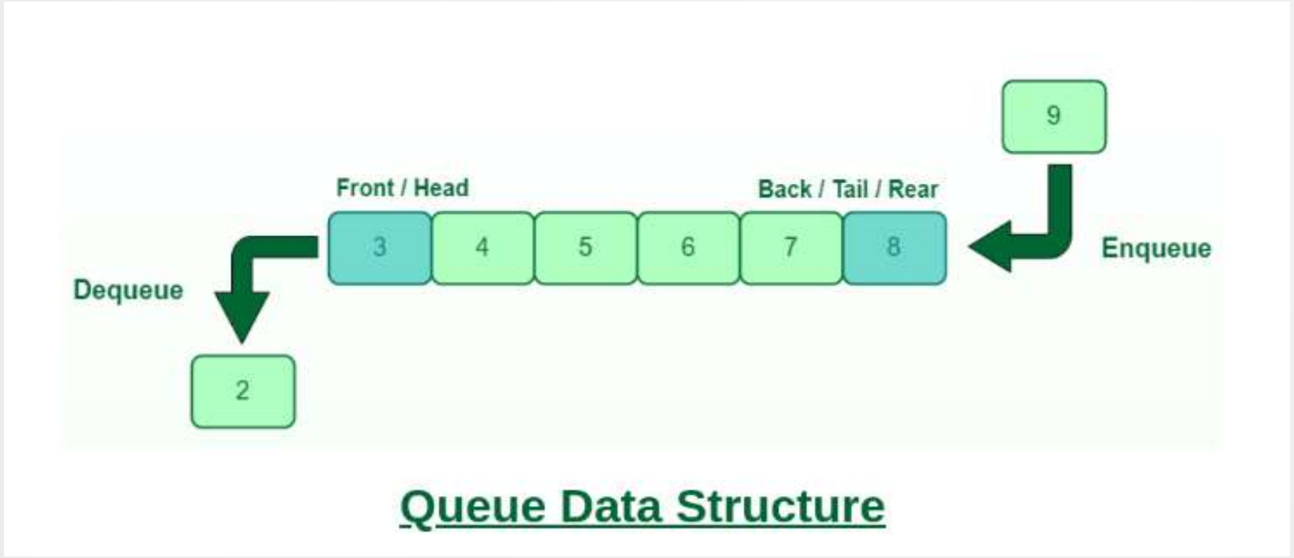
Uninformed / Exhaustive / Blind Search (Breadth First Search)

Breadth First Search (BFS)

- Breadth-first search is the most common search strategy for traversing a tree or graph. This algorithm ***searches breadth wise*** in a tree or graph, so it is called breadth-first search.
- BFS algorithm ***explores all the nodes at a given depth*** before proceeding to the next level.
- The breadth-first search algorithm is an example of a ***general-graph search algorithm.***
- BFS is ***used where the given problem is very small*** and ***space complexity is not considered.***

Breadth First Search (BFS) [Contd..]

- Breadth-first search ***implemented using FIFO queue data structure.***



Breadth First Search (BFS) [Contd..]

Algorithm:

- **S1:** Enter the starting node on Queue.
- **S2**: If Queue is empty, then return fail and stop.
- **S3**: If first element on Queue is the Goal node, then return success and stop.
- **S4**: Remove and expand first element from the Queue and place children at end of the Queue.
- **S5**: Goto S2.

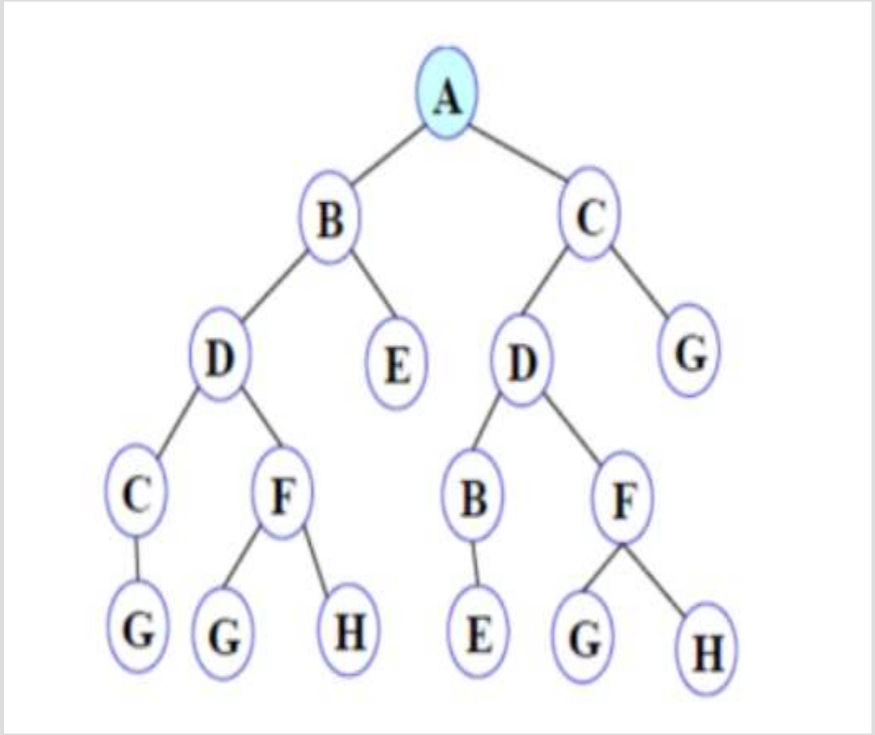
Example-1(BFS)

Consider the following State Space to be searched:

Here node **'A'** is the source or start or initial node and node **'G'** is the goal node.

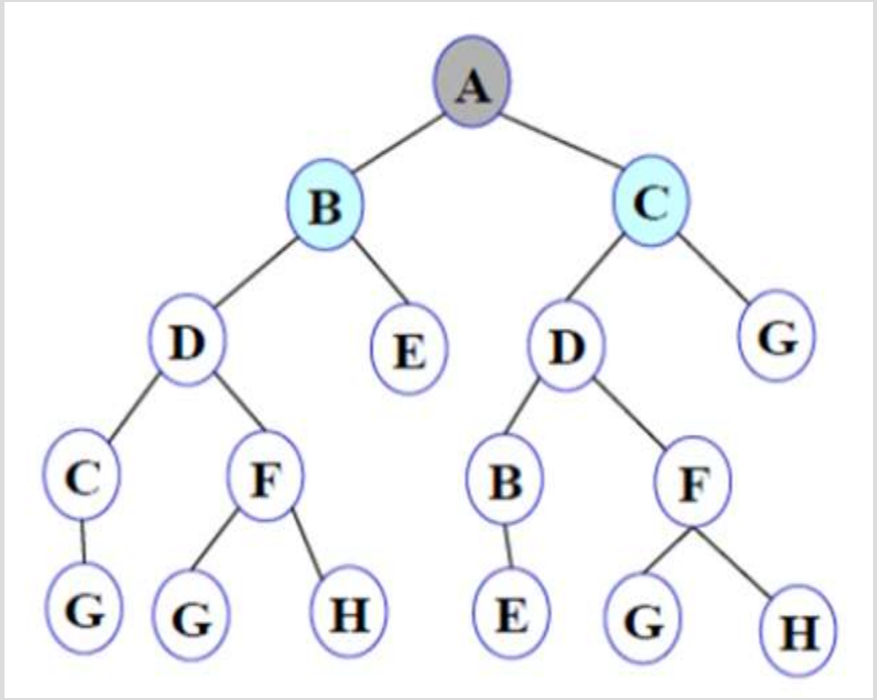
- **Step 1:** Initially Queue contains only one node corresponding to the source state A.

  i.e., Queue : A



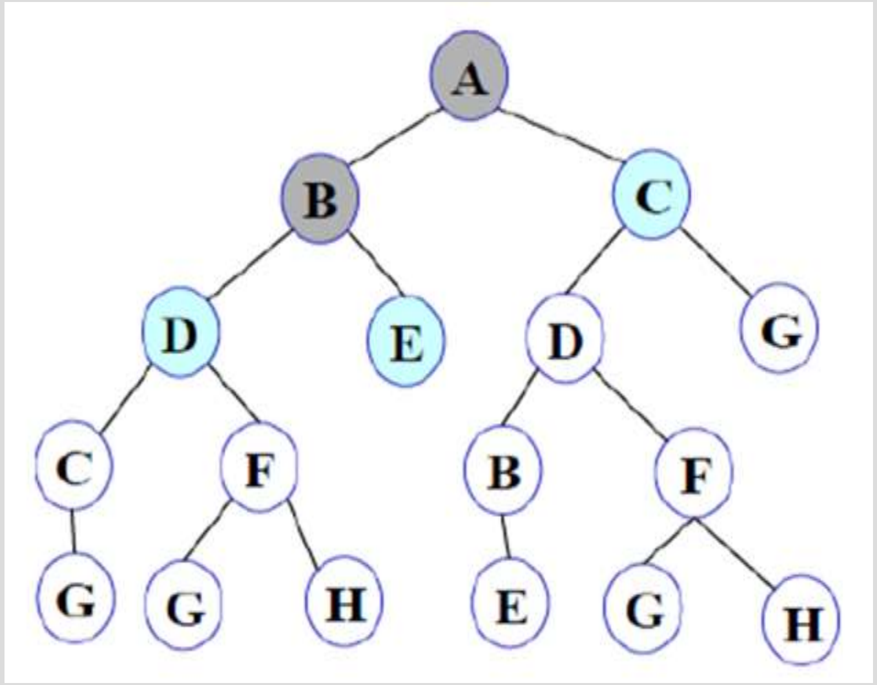
- **Step 2**: Node A is removed from Queue. The node is expanded, and its children B and C are generated. They are placed at the back of Queue.

i.e., Queue : B C



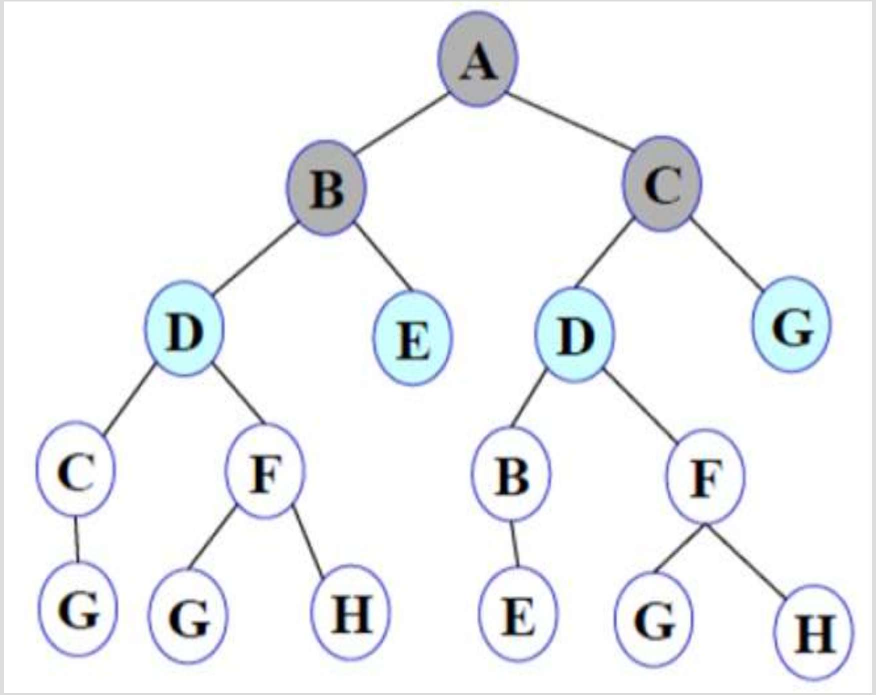
- **Step 3:** Node B is removed from Queue and is expanded. Its children D, E are generated and put at the back of Queue.

  i.e., Queue : C D E



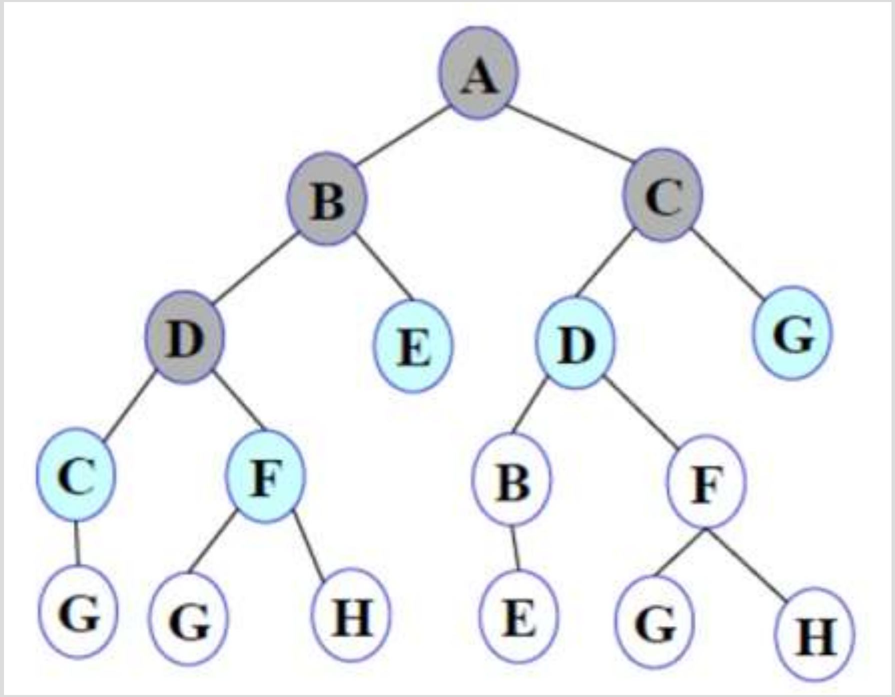
- **Step 4:** Node C is removed from Queue and is expanded. Its children D and G are added to the back of Queue.

  i.e., Queue: D E D G

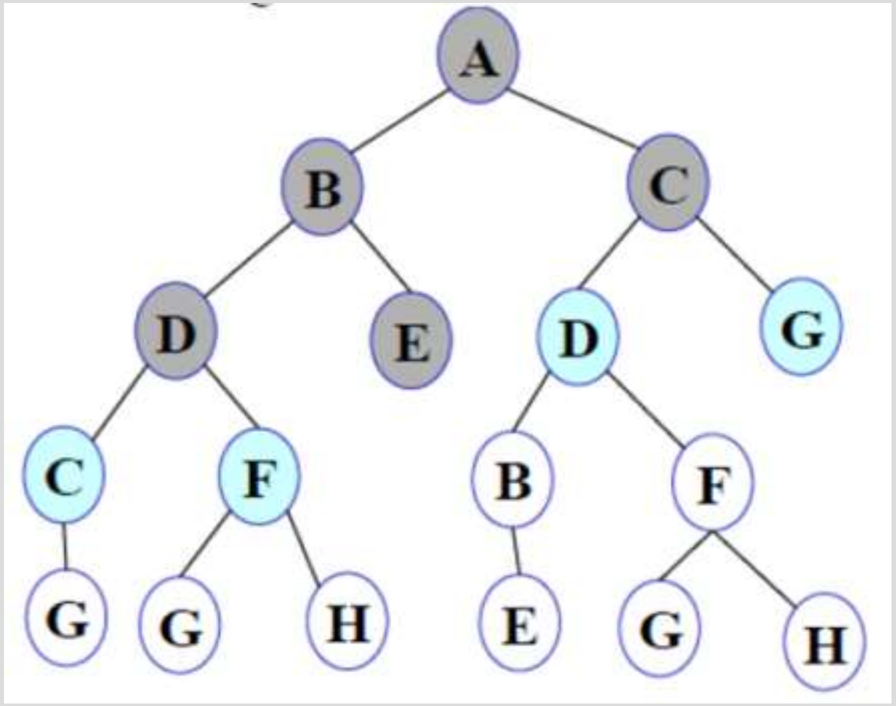


- **Step 5:** Node D is removed from Queue. Its children C and F are generated and added to the back of Queue.

  i.e., Queue: E D G C F

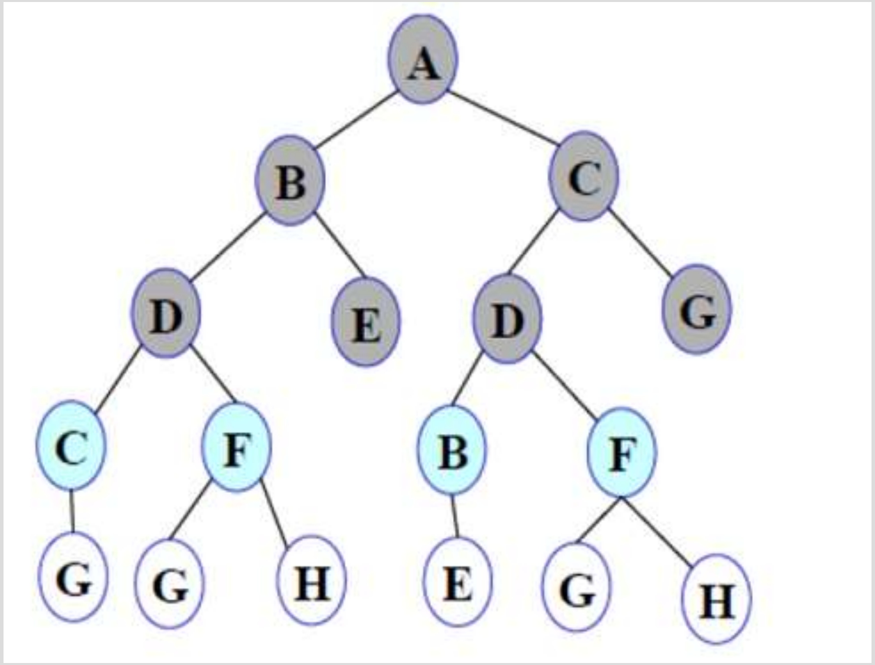


- **Step 6:** Node E is removed from Queue. It has no children.
- i.e., Queue: D G C F

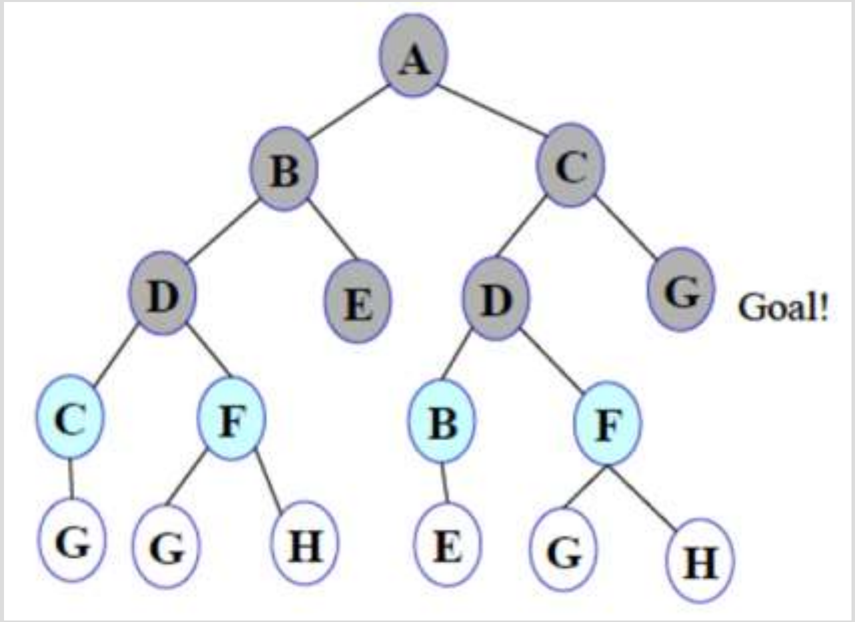


- **Step 7:** D is expanded; B and F are put in OPEN.

  i.e., Queue: G C F B F



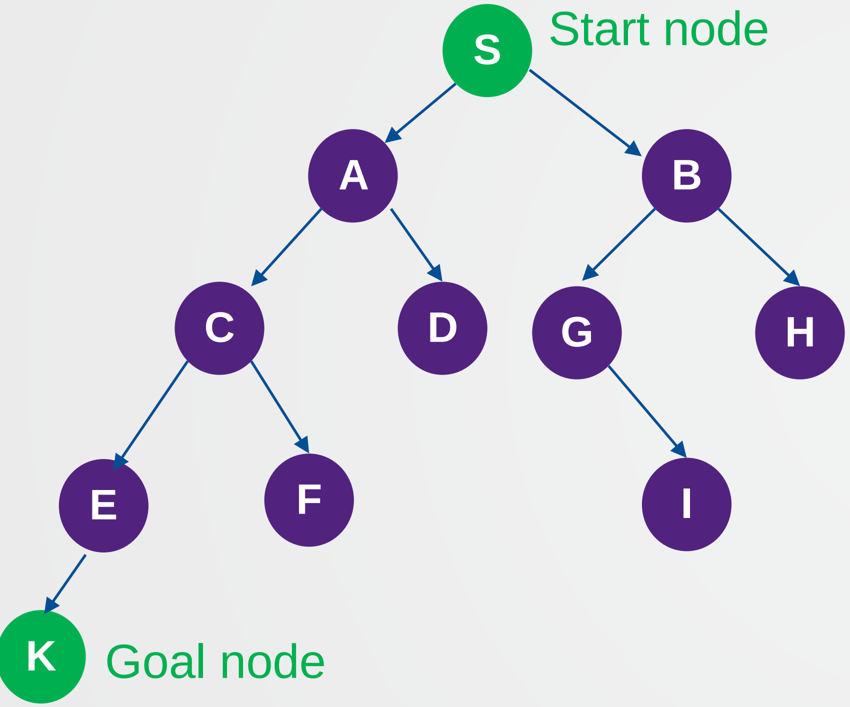
- **<u>Step 8</u>: G is selected for expansion. It is found to be a goal node**. So the algorithm returns the path **A B C D E D G** by following the parent pointers of the node corresponding to G. The algorithm terminates.



GOAL NODE FOUND!!

TRAVERSAL ORDER: A B C D E D G

Example-2 (BFS)



Queue: S (S is not goal node, remove and expand)

Queue: A B (A is not goal node, remove and expand)

Queue: B C D (B is not goal node, remove and expand)

Queue: C D G H (C is not goal node, remove and expand)

Queue: D G H E F (D is not goal node, no expansion, G is not the goal node, remove and expand)

Queue: H E F I (H is not goal node, no expansion, E is not the goal node, remove and expand)

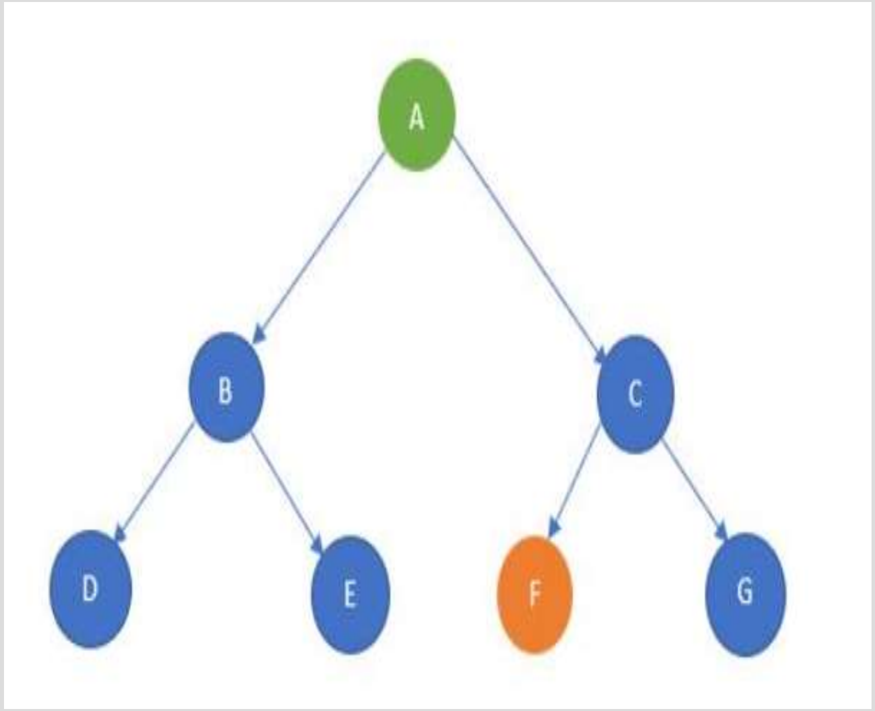
Queue: F I K (Remove F, I as no expansion) K goal node found.

TRAVERSAL ORDER: S→A→B→C→D→G→H→E→F→I→K

Example-3 (BFS)

Consider the following State Space to be searched:

Let A be the start state and F be the final or goal state to be searched.



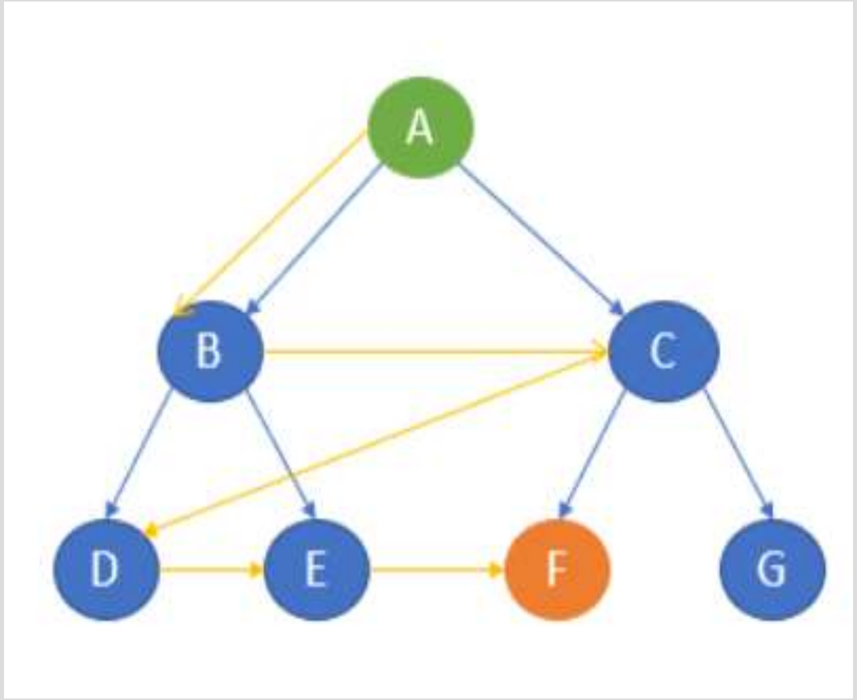
Shortcut method for BFS Traversal:

It starts from A and then travel to the next level and visits B and C and then

travel to the next level and visits D, E, F and G. Here, the goal state is defined as F. So, the traversal will stop at F.

The path of traversal is:

A —-> B —-> C —-> D —-> E —-> F



Breadth First Search (BFS) - Advantages

- One of the ***simplest search strategies.***
- 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. (refer example-2)

Breadth First Search (BFS) - 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.***

Breadth First Search (BFS) - Analysis

- **Time Complexity**: $O(b^d)$
- **Space Complexity**: $O(b^d)$

  [where b→ branching factor and d→ depth]
- **Solution** : Optimal.

Breadth First Search (BFS) – What is the need?

Some of the most vital aspects that make this algorithm your first choice are:

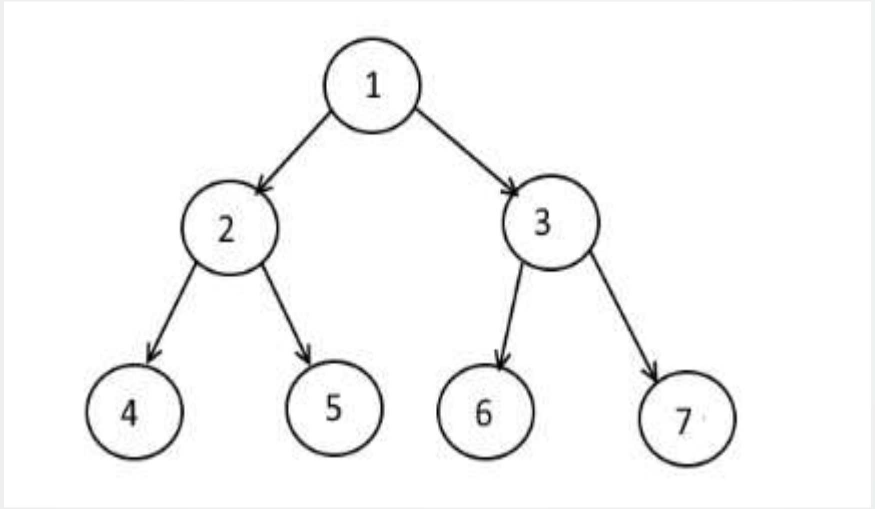
- BFS can ***traverse through a graph in the smallest number of iterations***.
- The architecture of the BFS algorithm is ***simple and robust.***
- The result of the BFS algorithm ***holds a high level of accuracy*** in comparison to other algorithms.
- ***No possibility of this algorithm getting caught up in an infinite loop problem.***

Breadth First Search (BFS) – Applications

- **Un-weighted Graphs:** BFS algorithm can easily create the shortest path and a minimum spanning tree to visit all the vertices of the graph in the shortest time possible with high accuracy.
- **P2P Networks:** BFS can be implemented to locate all the nearest or neighboring nodes in a peer to peer network. This will find the required data faster.
- **Navigation Systems:** BFS can help to find all the neighboring locations from the main or source location.
- **Network Broadcasting:** A broadcasted packet is guided by the BFS algorithm to find and reach all the nodes it has the address for.

Try

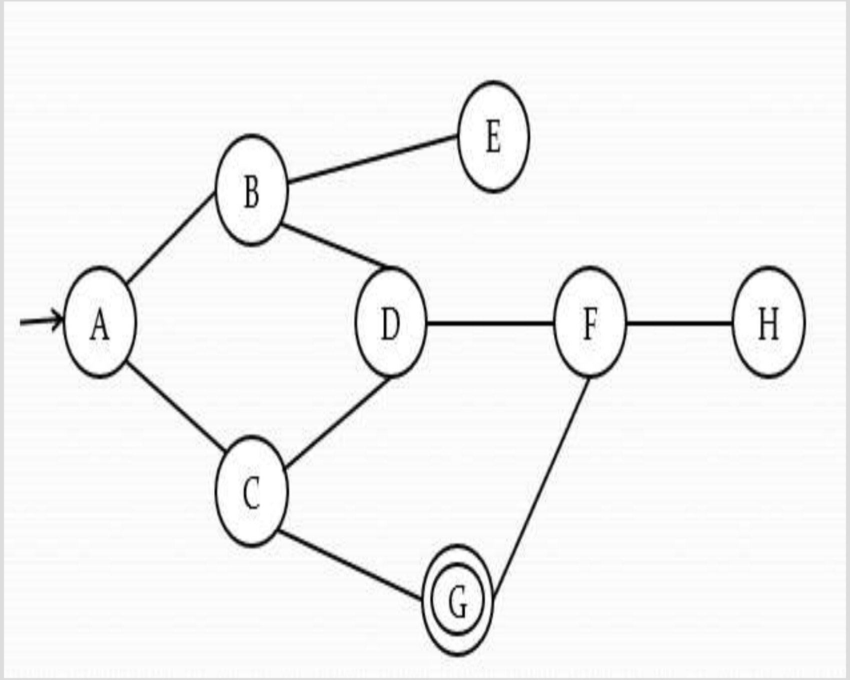
The graph is given with nodes from 1 to 7. Find the shortest path from 1 to 7 using BFS traversal.



Try

The graph is given with nodes from A to H. Find the shortest path from A to G using BFS traversal.

.



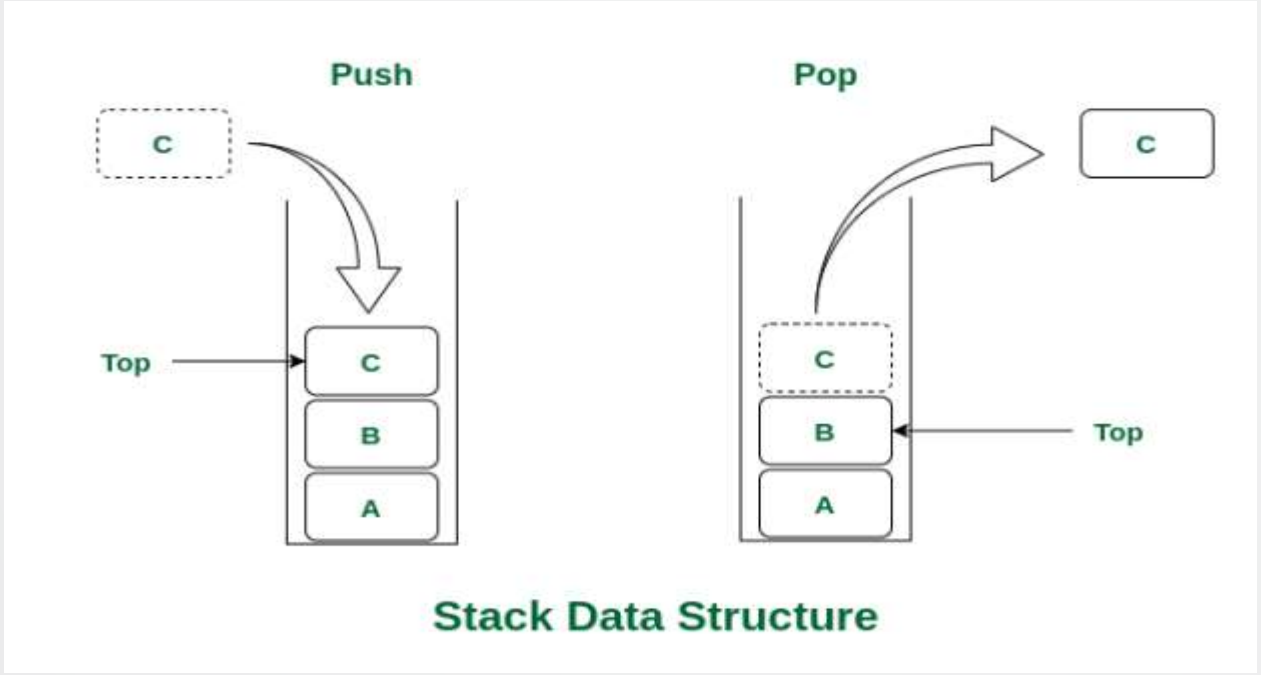
Uninformed / Exhaustive / Blind Search (Depth First Search)

Depth First Search (DFS)

- Depth-first search is a ***recursive algorithm*** for traversing a tree or graph data structure.
- It is called as 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 backtracking.***
- The process of the DFS algorithm is ***similar to the BFS algorithm.***

Depth First Search (DFS) [Contd..]

- DFS ***uses a stack data structure (LIFO)*** for its implementation.



Depth First Search (DFS) [Contd..]

Algorithm:

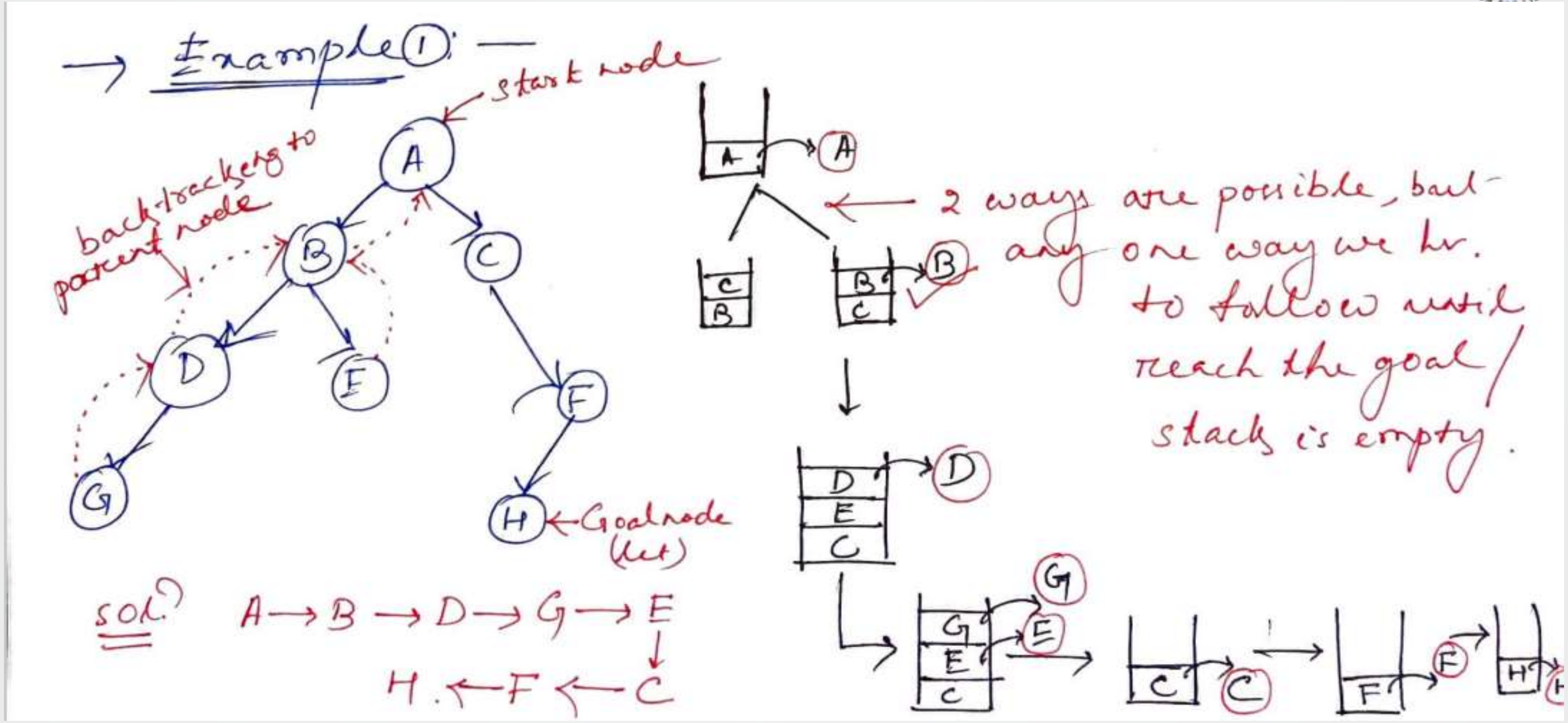
- **S1**: If initial state is a goal state, quit and return success.
- **S2**: Otherwise do the following until success or failure is reported:

  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 obtained, return success, otherwise continue in this loop.

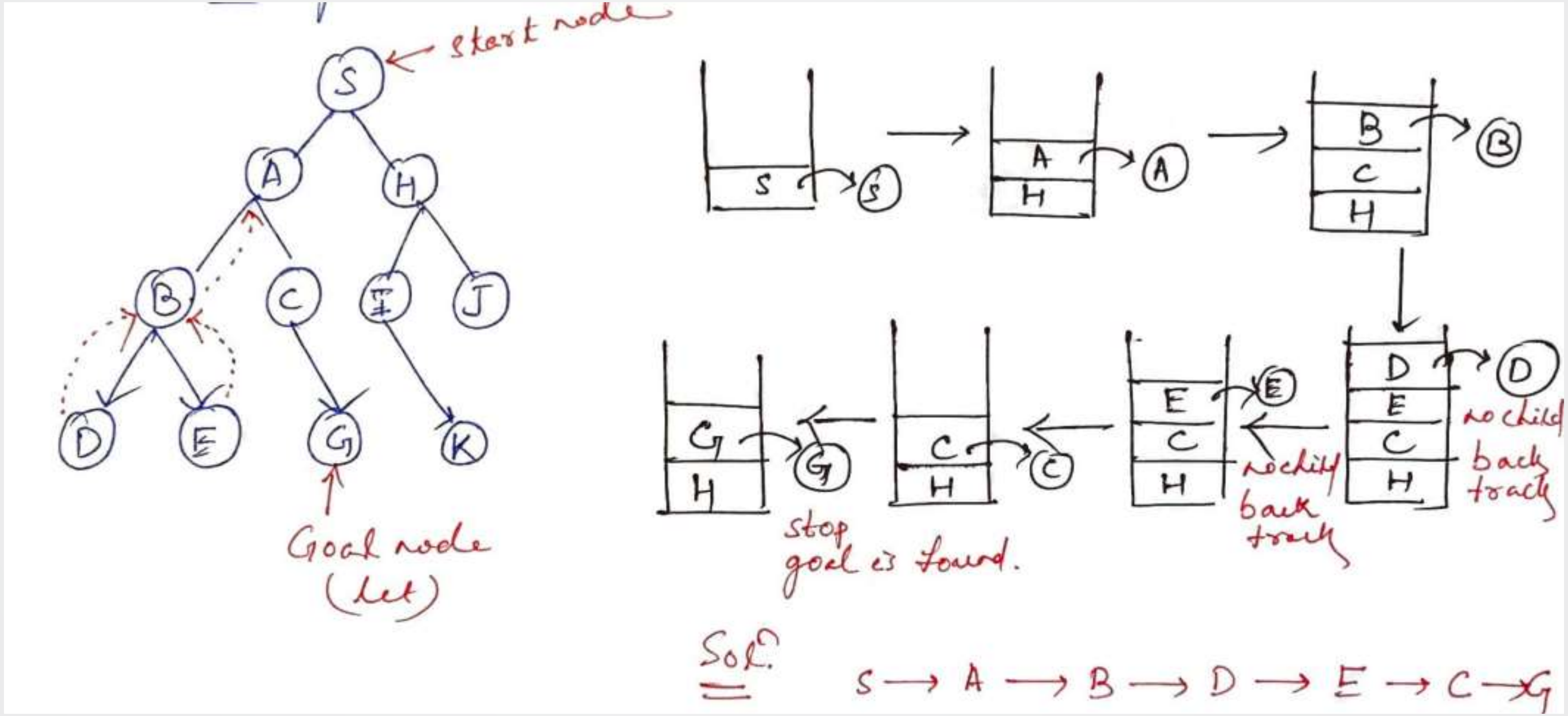
Example-1 (DFS)



→ Example ①: —
start node
back-tracking to parent node
A
B
C
D
E
F
G
H
Goal node (let)
2 ways are possible, but any one way we hv. to follow until reach the goal / stack is empty.

sol: A → B → D → G → E → C → F → H.

Example-2 (DFS)

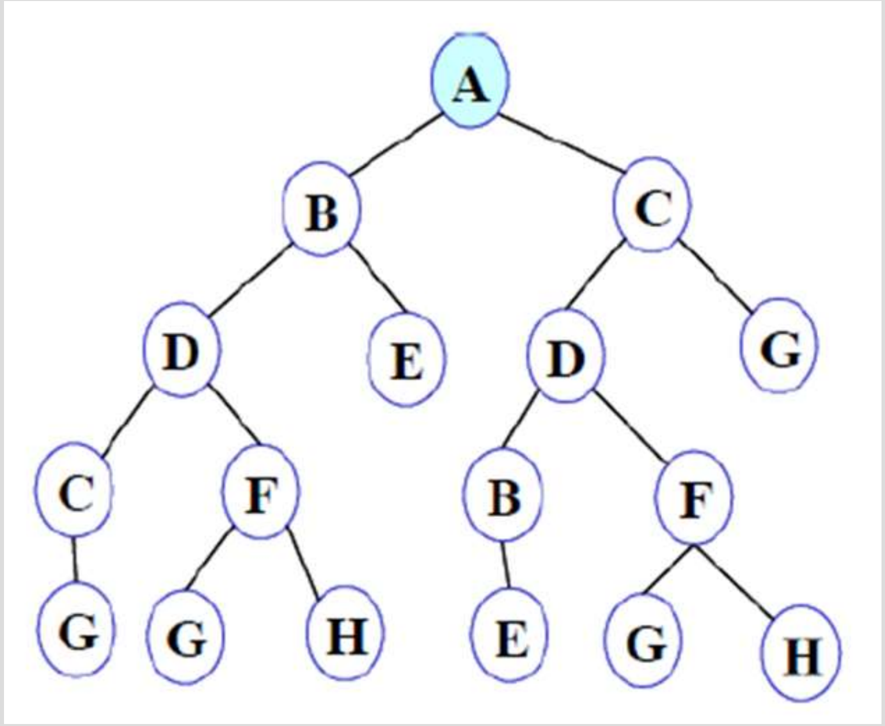


Example-3 (DFS)

Let us consider the following tree. Here node ‘A’ is the source or start or initial node and node ‘G’ is the goal node.

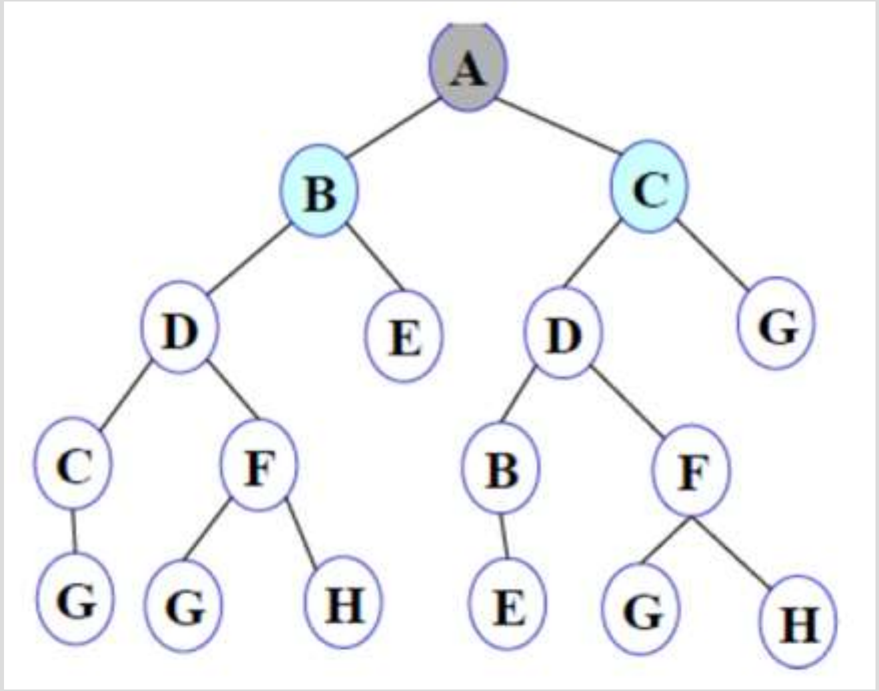
Step 1: Initially Stack contains only one node corresponding to the source state A.

i.e., Stack : A



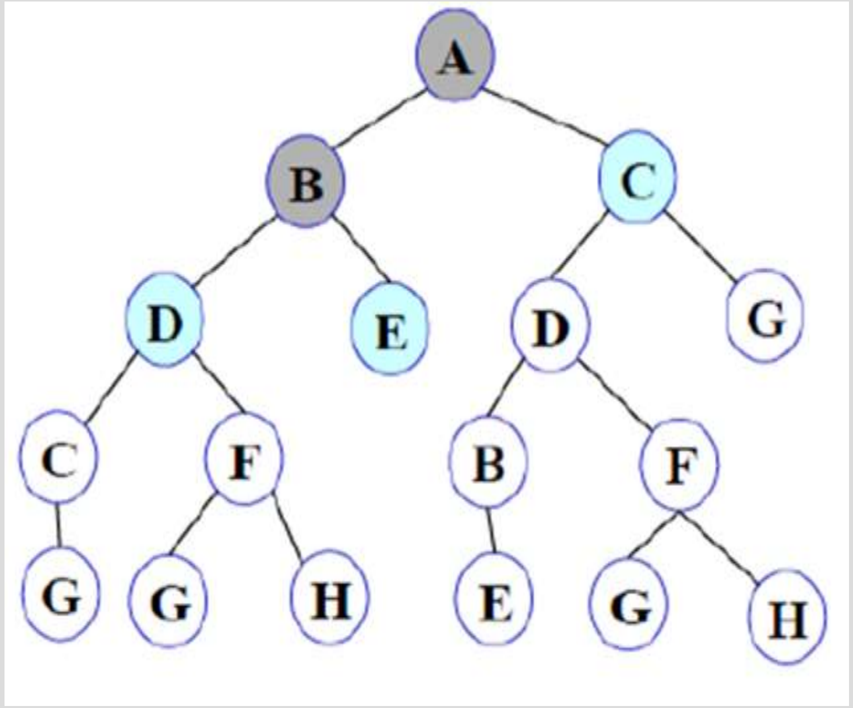
Step 2: Node A is removed from Stack. A is expanded and its children B and C are put on the top of the Stack.

i.e., Stack : B C



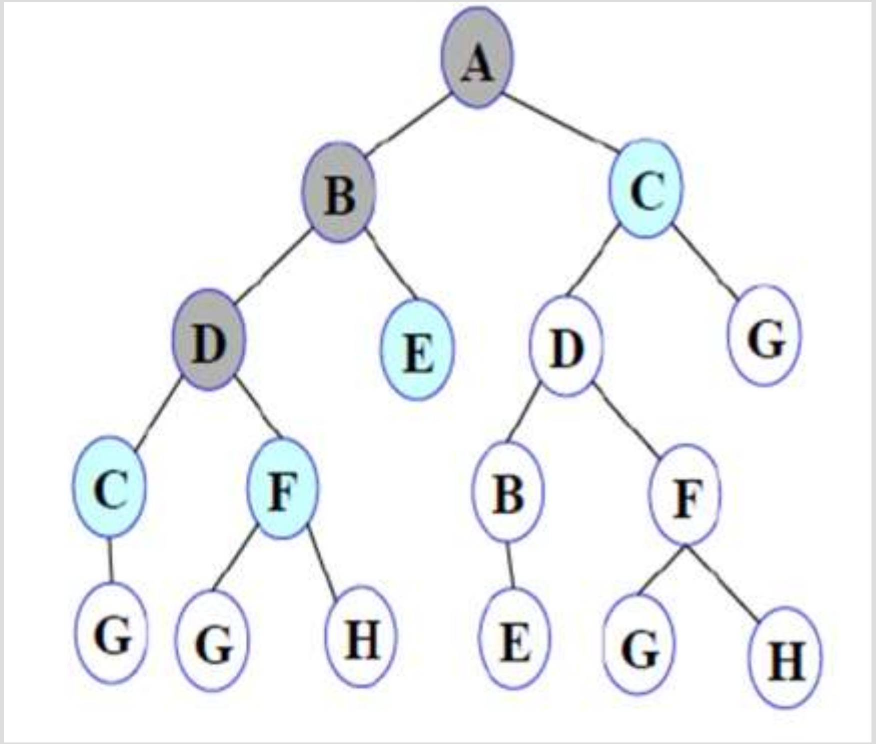
Step 3: B is not the goal node. So it is removed from the stack and expanded.

i.e., Stack : D E C



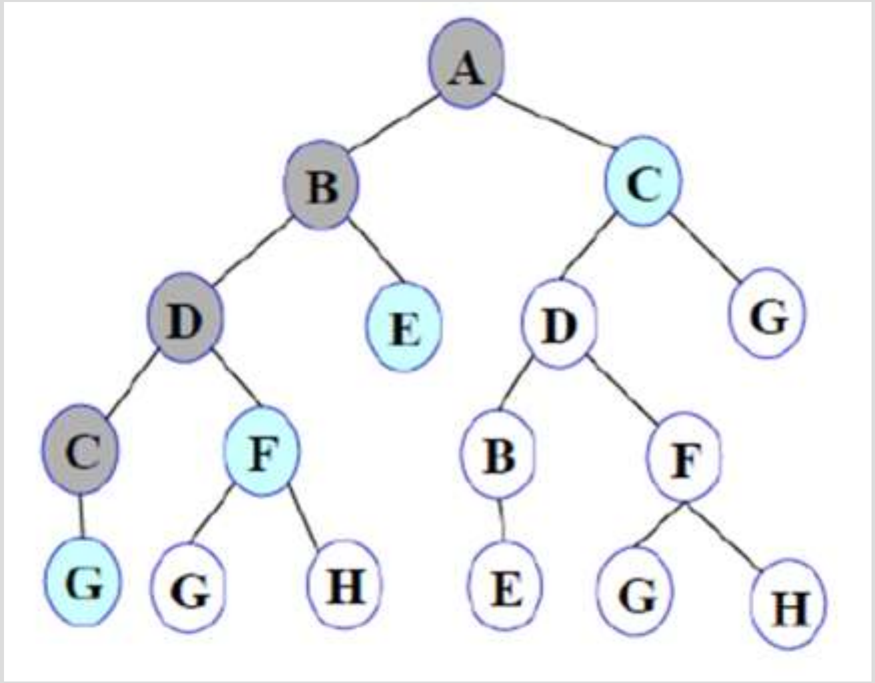
Step 4: D is not the goal node. So it is removed from the stack and expanded.

i.e., Stack : C F E C



Step 5: C is not the goal node. So it is removed from the stack and expanded.

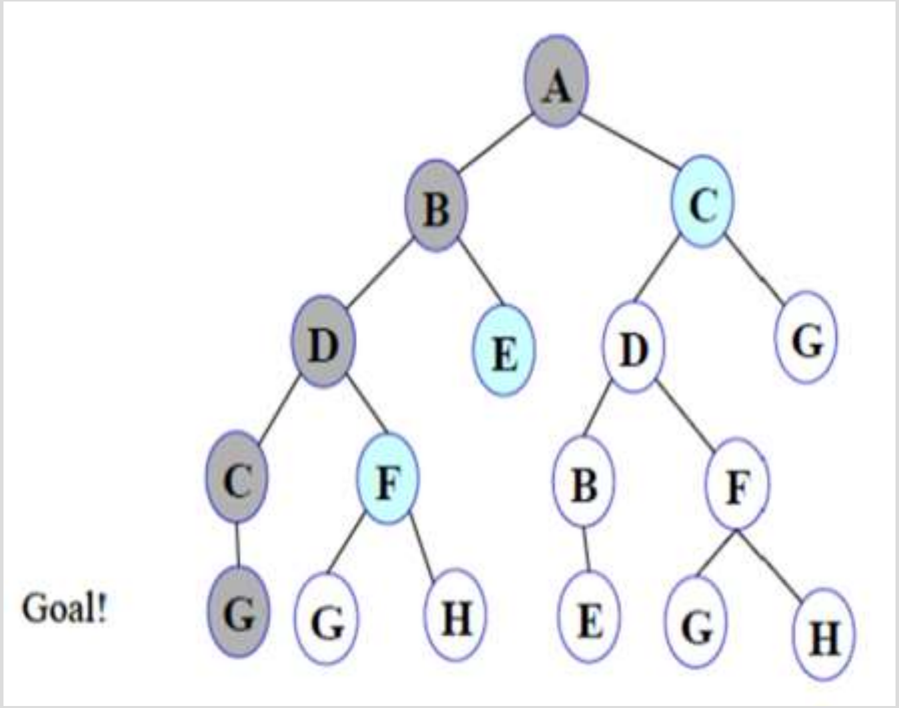
i.e., Stack : G F E C



Step 6: Node G is expanded and found to be a goal node.

i.e., Stack : G F E C

TRAVERSAL ORDER: A-B-D-C-G

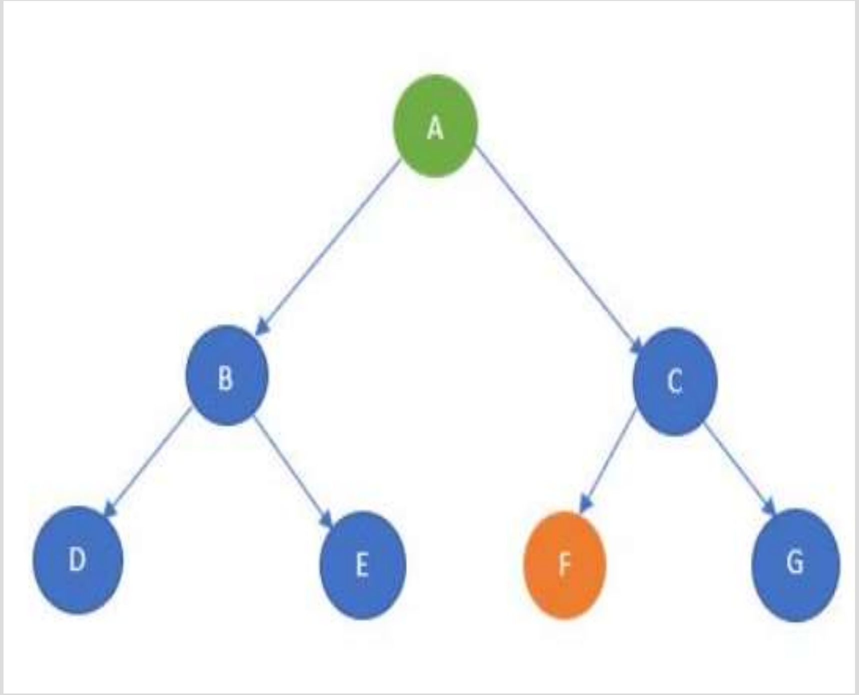


GOAL NODE FOUND!!

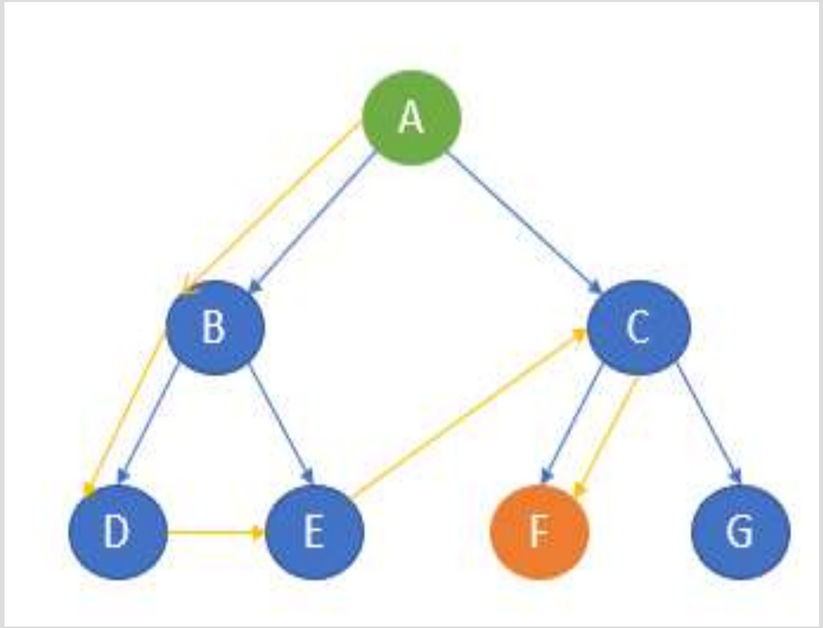
Example-4 (DFS)

Consider the following State Space to be searched:

Let A be the start state and F be the final or goal state to be searched.



Here, it starts from the start state A and then travels to B and then it goes to D. After reaching D, it backtracks to B. B is already visited, hence it goes to the next depth E and then backtracks to B. as it is already visited, it goes back to A. A is already visited. So, it goes to C and then to F. F is our goal state and it stops there.



The path of traversal is:

A —-> B —-> D —-> E —-> C —-> F

Depth First Search (DFS) - Advantages

- DFS ***requires very less memory (as compared to BFS)*** 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).
- DFS does not require much more search.

Depth First Search (DFS) - Disadvantages

- 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 (DFS) - Analysis

- **Time Complexity**: $O(b^d)$
- **Space Complexity**: $O(bd)$

  [where b→ branching factor and d→ depth]
- **Solution** : No particular solution. (not optimal).

Depth First Search (DFS) - Applications

- For finding the path.
- To test if the graph is bipartite.
- For finding the strongly connected components of a graph.
- For detecting cycles in a graph.

Difference Between BFS & DFS

BFS

- BFS stands for ***Breadth First Search.***
- BFS(Breadth First Search) uses ***Queue data structure*** for finding the shortest path.
- BFS can be used to find single source shortest path in an unweighted graph, because in BFS, we ***reach a vertex with minimum number of edges*** from a source vertex.

DFS

- DFS stands for ***Depth First Search.***
- DFS(Depth First Search) uses ***Stack data structure.***
- In DFS, we might ***traverse through more edges*** to reach a destination vertex from a source.

Difference Between BFS & DFS

BFS

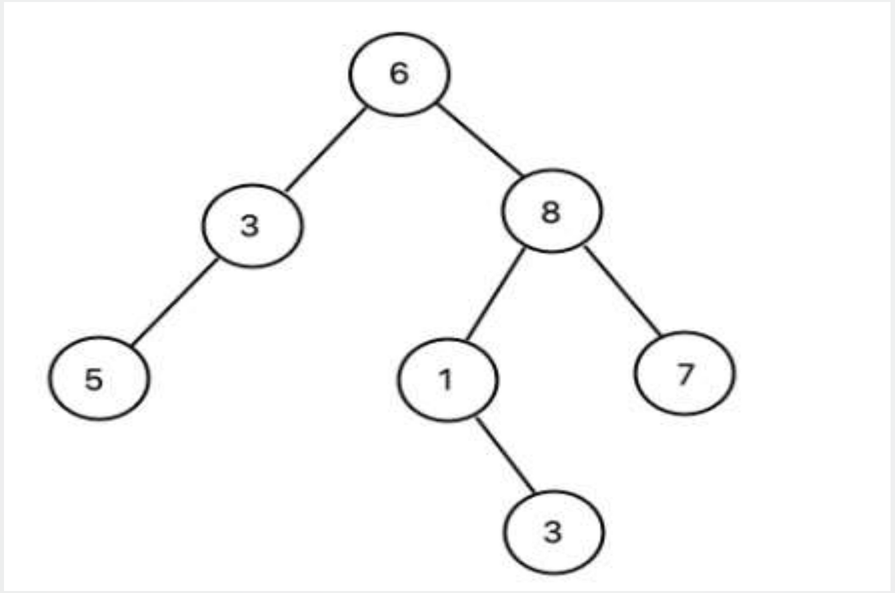
- BFS is more ***suitable for searching vertices which are closer to the given source.***
- BFS considers all neighbours first and therefore ***not suitable for decision making trees used in games or puzzles.***

DFS

- DFS is more ***suitable when there are solutions away from source.***
- DFS is more ***suitable for game or puzzle problems.*** We make a decision, then explore all paths through this decision. And if this decision leads to win situation, we stop.

Try

Perform the DFS traversal to find the shortest path from 6 to 7.



References

- Stuart Russell and Peter Norvig, “Artificial Intelligence”, 2nd edition, Pearson Education, 2003.
- Saroj Koushik, “Artificial intelligence”.
- NPTEL

IARE

INSTITUTE OF AERONAUTICAL ENGINEERING

Thank You

IARE
INSTITUTE OF
AERONAUTICAL ENGINEERING

Course Title - Logic Programming for Artificial Intelligence

Topic Title – Uninformed Search (DLS, IDDFS, UCS, BS)

Presenter's Name – Ms. Bidyutlata Sahoo
Presenter's ID – IARE11028
Department Name – CSE (AI & ML)
Lecture Number - 03
Presentation Date – 25/09/2024

Course Outcome

At the end of the course, students should be able to:

CO3: Interpret uninformed and informed search strategies, and select the appropriate approach for different AI problems.

Topic Learning Outcome

Understand the concept of uninformed search where little or no domain-specific information (heuristics) is available to guide the search process.

Uninformed / Exhaustive / Blind Search (Depth Limited Search(DLS / DLDFS))

Shortcomings of DFS

- 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 Limited Search

- 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

Depth-limited search can be terminated with two Conditions of failure:

1. **Standard failure value:** It indicates that problem does not have any solution.
2. **Cutoff failure value:** It defines no solution for the problem within a given depth limit.

Depth Limited Search-Example

Here, Start node = S, Goal node = J

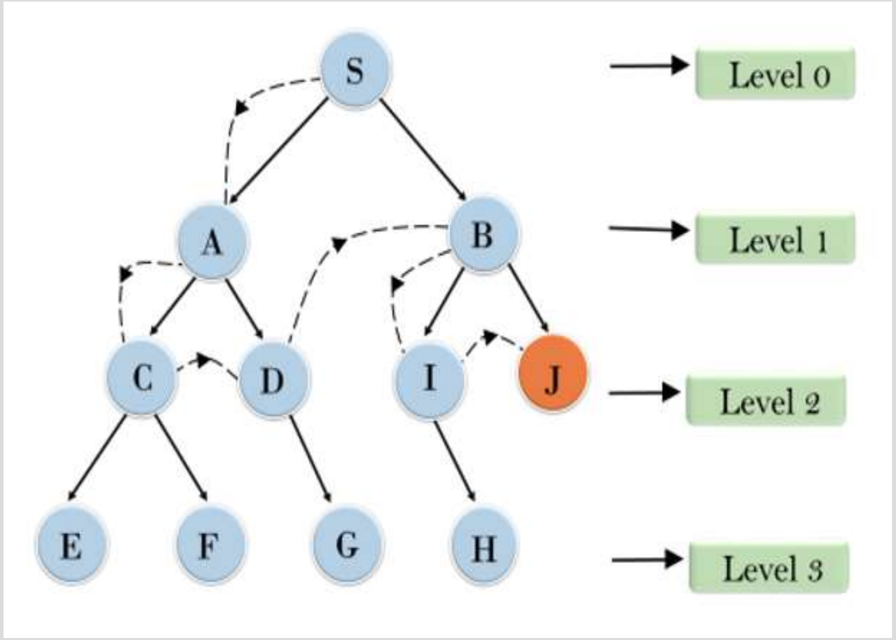
Let Depth (d) = 2

Then path : S→A→C→D→B→I→J

[NOTE:

If we define **Depth (d) = 2 and Goal node = H**

In this case the result will have **no solution.(**as per 2nd failure criteria**)]**



Depth Limited Search - Advantages

Depth-limited search is ***Memory efficient.***

Depth Limited Search - Disadvantages

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

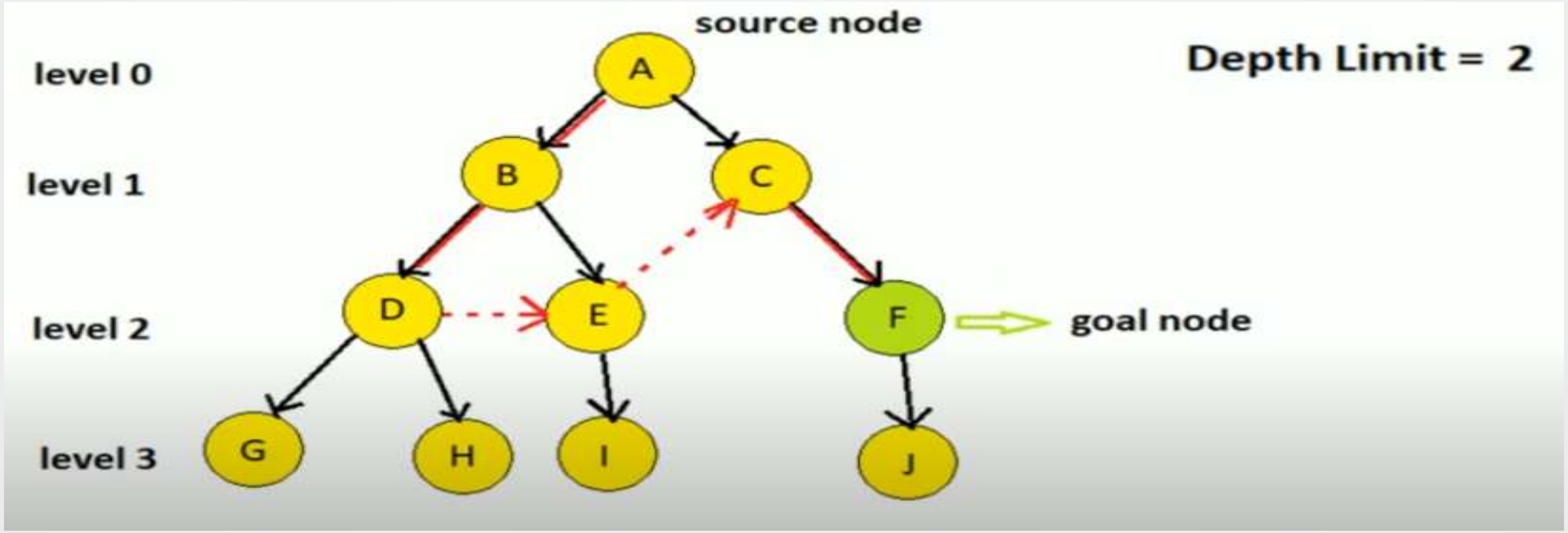
Depth Limited Search - Analysis

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

Try

Here, Start node = A, Goal node = F, Depth Limit = 2

Perform the DLS traversal to find the shortest path between **A and F.**



Uninformed / Exhaustive / Blind Search
(Iterative Deepening Depth First Search (IDDFS))

Iterative Deepening DFS (IDDFS)

- This is the ***modified version of Depth Limited Search algorithm.***
- This is type of Uninformed search technique also called as ***blind search.***
- This algorithm ***Works on only present value.***
- This algorithm is a ***combination of DFS & BFS***. (This Search algorithm combines the benefits of <u>Breadth-first search's</u> ***fast search*** and <u>depth-first search's</u> ***memory efficiency***.)
- It ***uses STACK (LIFO)*** to perform search operations.

Iterative Deepening DFS (IDDFS)

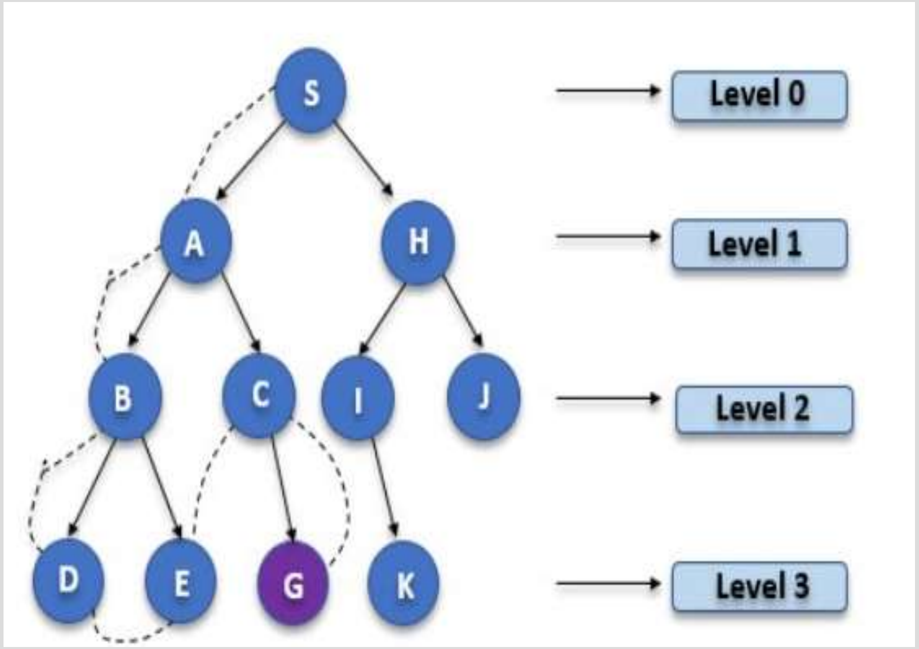
- Here, the best depth limit is found out by gradually increasing depth limit on each iterations.
- ***Initially depth limit is =0***
- ***On each iteration depth increased by exactly one.***
- The iterative search algorithm is ***useful when search space is large, and depth of goal node is unknown.***

Iterative Deepening DFS (IDDFS) – Example - 1

Here, Start node = S, Goal node = G

1’st Iteration → S (d=0)
2’nd Iteration → S, A, H (d=0+1)
3’rd Iteration → S, A, B, C, H, I, J (d=1+1)
4’th Iteration → S, A, B, D, E, C, G (d=2+1)

In the fourth iteration, the algorithm will find the goal node.



Then path : S→A→B→D→E→C→G

Iterative Deepening DFS (IDDFS) - Advantages

- Combine the ***advantage of both DFS and BFS***.
- It is ***Complete & Optimal***. (definitely find optimal solution)
- Will not go in infinite loop.
- ***Fast searching***.
- Less memory requirement. (i.e. ***memory efficient***)

Iterative Deepening DFS (IDDFS) - Disadvantages

- The main drawback of IDDFS is that ***it repeats all the work of the previous phase.***

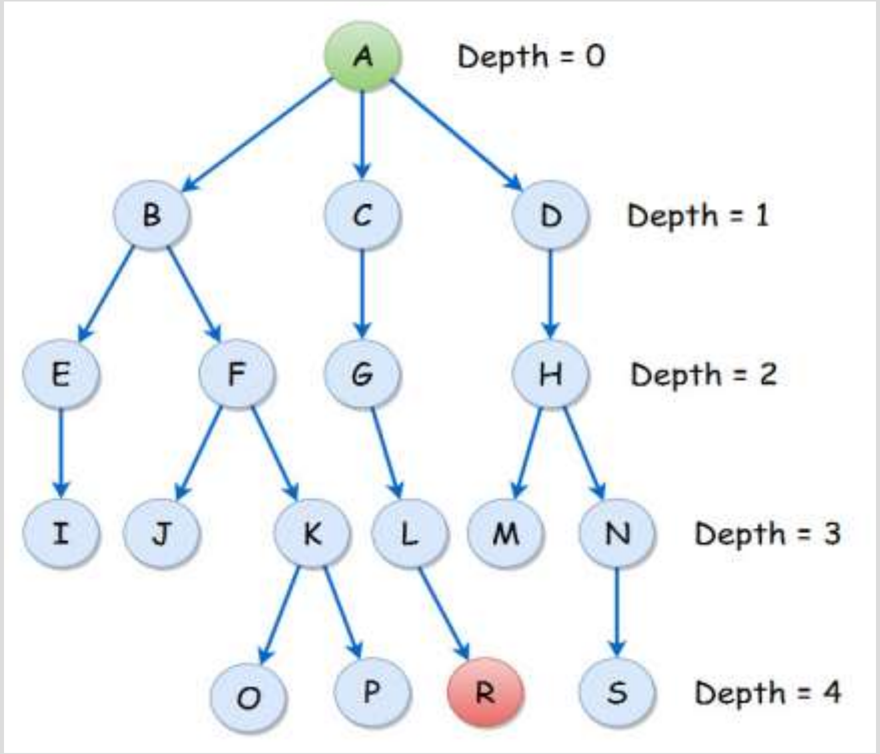
Iterative Deepening DFS (IDDFS) - Analysis

- **Completeness:** This algorithm is complete 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 **$O(d)$.**
- **Optimal:** IDDFS algorithm is optimal.

Try

Here, Start node = A, Goal node = R

Perform the IDDFS traversal to find the shortest path between A and R.



Uninformed / Exhaustive / Blind Search (Uniform Cost Search (UCS))

Uniform Cost Search (UCS)

- Uniform-cost search is a searching algorithm ***used for traversing a weighted tree or graph.***
- This algorithm comes into play ***when a different cost is available for each edge.***
- The primary ***goal*** of the uniform-cost search is ***to find a path to the goal node which has the lowest cumulative cost***. Uniform-cost search expands nodes according to their path costs form the root node.

Uniform Cost Search (UCS)

- It can be ***used to solve any graph/tree where the optimal cost is in demand.***
- A uniform-cost search algorithm ***is implemented by the priority queue.***
- It ***gives maximum priority to the lowest cumulative cost***. Uniform cost search is ***equivalent to BFS algorithm if the path cost of all edges are same.***

Uniform Cost Search (UCS) - Algorithm

- **S1**: Insert the root node into the priority queue.
- **S2**: Remove the element with the highest priority.
- **S3**: If the removed node is the goal node,

  Print total cost and stop the algorithm.

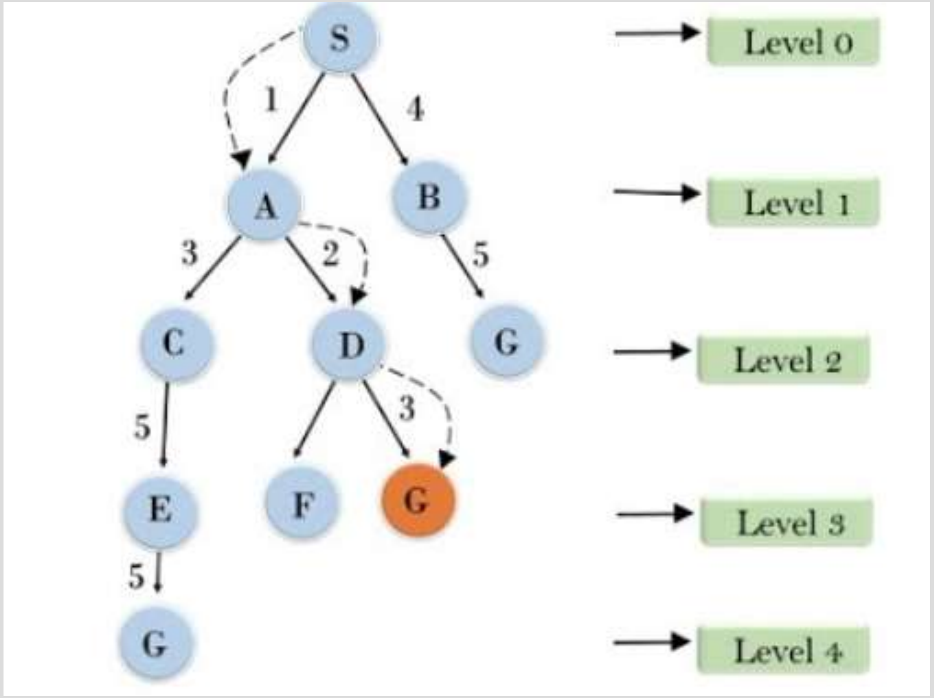
  else

  Enqueue all the children of the current node to the priority queue, with their cumulative cost from the root as priority and the current node to the visited list.

Uniform Cost Search- Example

Here, Start node = S, Goal node = G

From node S we look for a node to expand, and we have nodes A and B, but since it's a uniform cost search, it's expanding the node with the lowest step cost, so node A becomes the successor rather than B. From A we look at its children nodes C and D. Since D has the lowest step cost, it traverses through node D. Then we look at the successors of D, i.e. G. Since D has only one child G (with cost) which is our required goal state we finally reach the goal state G by implementing UCS Algorithm.



Then path : S→A→D→G (Path cost=6)

Uniform Cost Search (UCS) - Advantages

- Uniform cost search is ***an optimal search method*** because at every state, the path with the least cost is chosen.

Uniform Cost Search (UCS) - Disadvantages

- It ***does not care about the number of steps*** or finding the shortest path involved in the search problem, and it is only ***concerned about path cost***.
- This algorithm ***may be stuck in an infinite loop***.

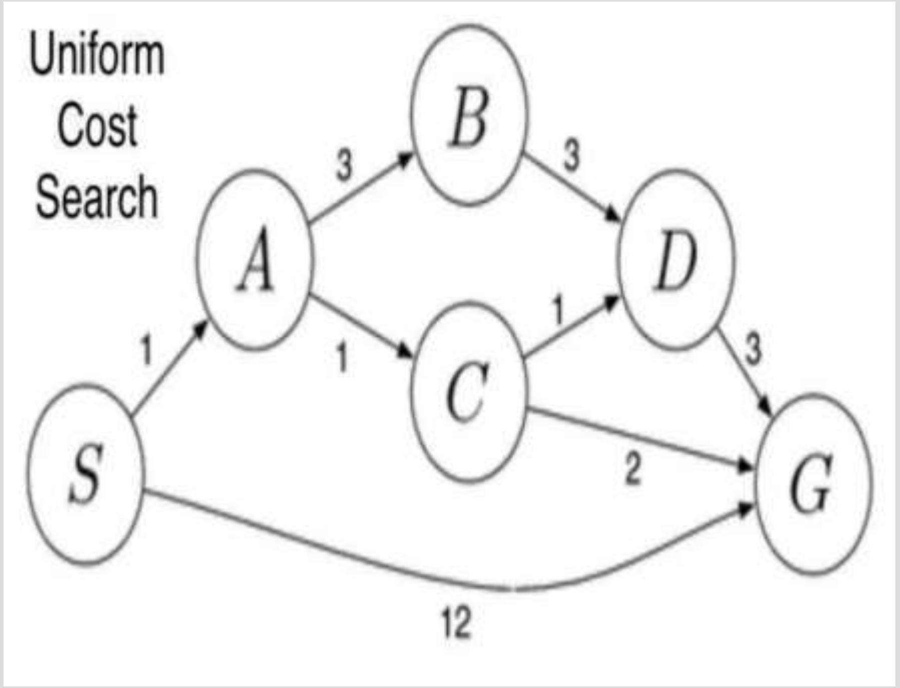
Uniform Cost Search (UCS) - Analysis

- **Complete:** Yes (if b is finite)
- **Time Complexity:** O(b(c/ϵ)) [*where, ϵ -> is the lowest cost, c -> optimal cost]*
- **Space complexity**: O(b(c/ϵ))
- **Optimal:** Yes (even for non-even cost)

Try

Here, Start node = S, Goal node = G

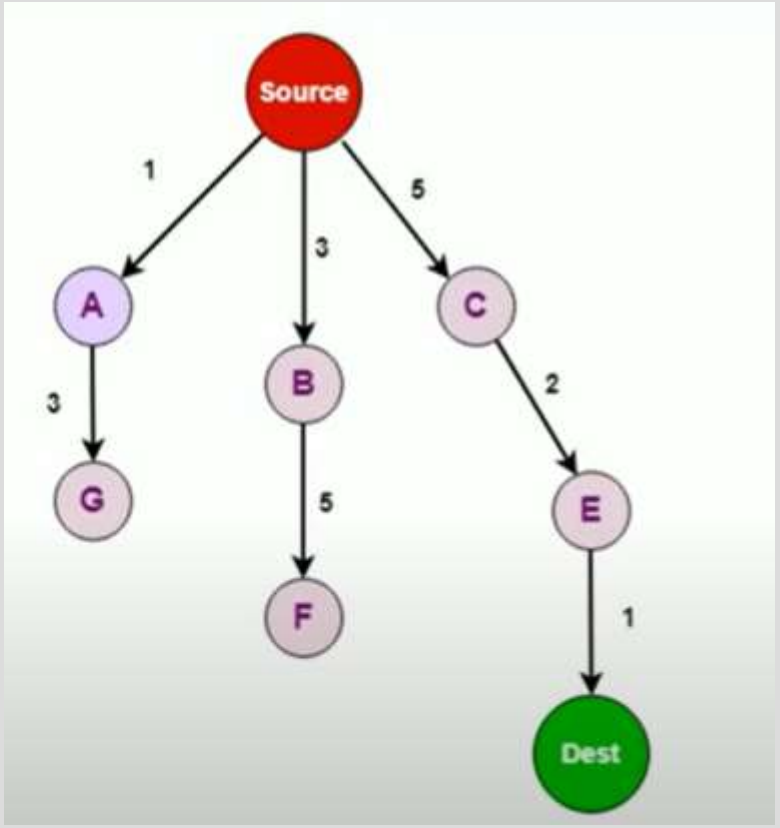
Traverse the tree from S to G using Uniform Cost Search algorithm and find the path cost for the same.



Try

Here, Start node = Source, Goal node = Dest

Traverse the tree from **Source** to **Dest** using Uniform Cost Search algorithm and find the path cost for the same.

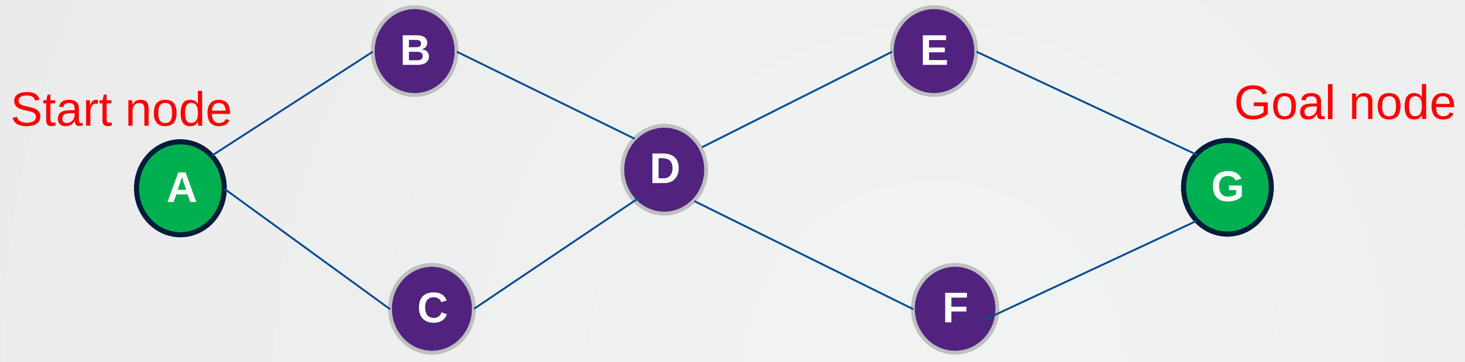


Uninformed / Exhaustive / Blind Search (Bidirectional Search(BS))

Bidirectional Search (BS)

- Bidirectional search algorithm *runs two simultaneous searches*.
  - **forward-search** : Looking from start node → end node
  - **backward-search**: Looking from end node → start 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.
- **<u>Stopping Criteria</u>**: ***The search stops when these two graphs intersect each other.***
- Bidirectional search ***can use search techniques such as BFS, DFS, DLS, etc.***

Bidirectional Search (BS) – Example - 1



<u>Forward BFS</u>

A

A→B→C

A→B→C→D

<u>Backward BFS</u>

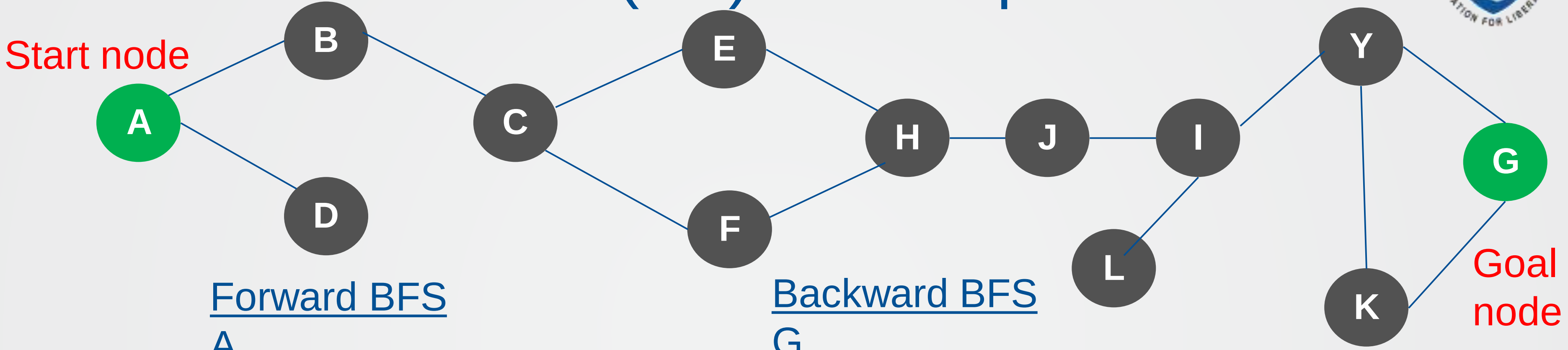
G

G→E→F

G→E→F→D

Hence after combining both the searches path will be: A→B→C→D→F→E→G

Bidirectional Search (BS) – Example - 2



Forward BFS
A
A→B→D
A→B→D→C
A→B→D→C→E→F
A→B→D→C→E→F→H

Backward BFS
G
G→K→Y
G→K→Y→I
G→K→Y→I→L→J
G→K→Y→I→L→J→H

After combining both searches

Then path : A→B→D→C→E→F→H→J→L→I→Y→K→G

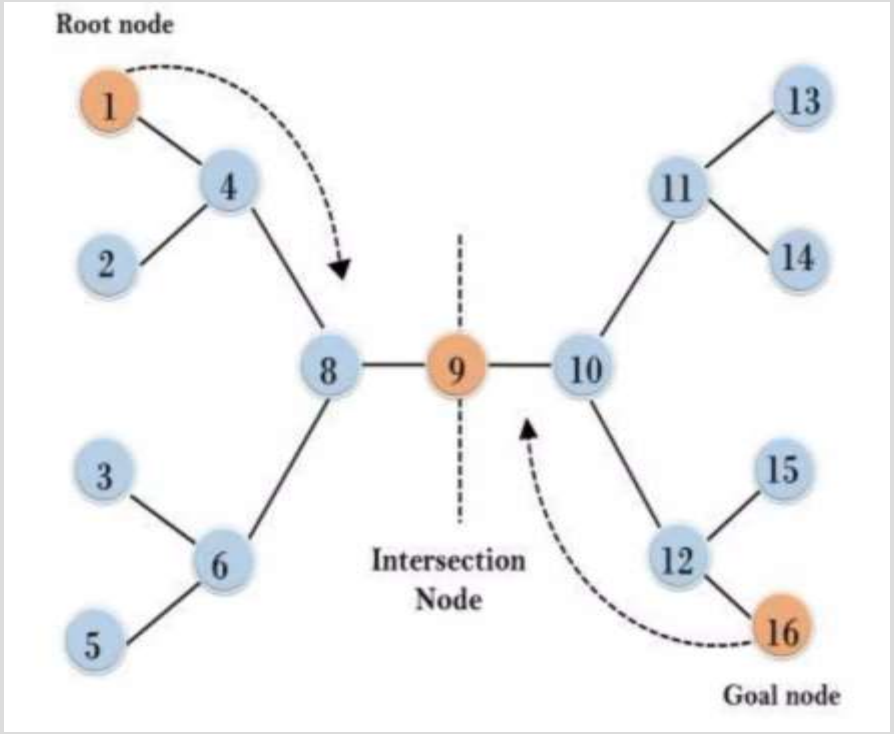
Bidirectional Search (BS) – Example - 3

Here, Start node = 1, Goal node = 16

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.

Then path : 1→4→8→9→10→12→16



Bidirectional Search (BS) - Advantages

- Bidirectional search ***is fast.***
- Bidirectional search ***requires less memory.***

Bidirectional Search (BS) - Disadvantages

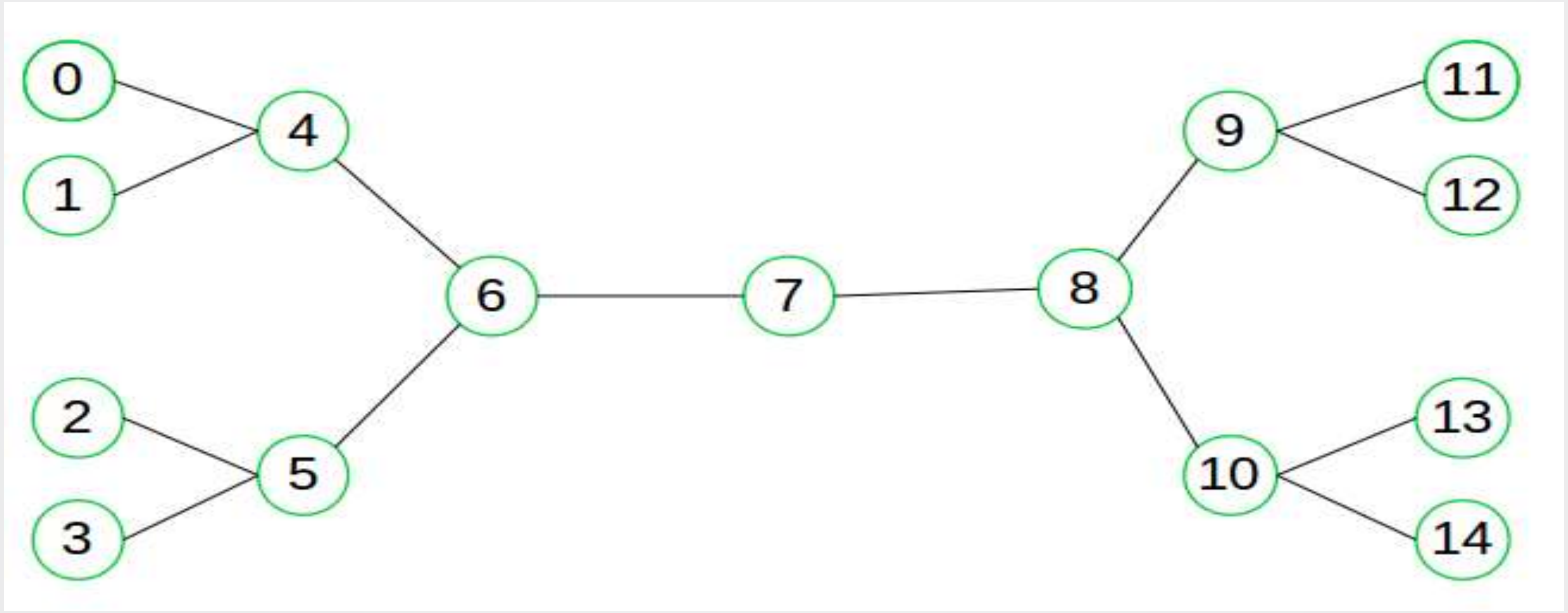
- ***Implementation*** of the bidirectional search tree ***is difficult.***
- In bidirectional search, ***one should know the goal state in advance***.

Bidirectional Search (BS) - Analysis

- **Completeness**: Bidirectional Search *is complete* if we use BFS in both searches. (No particular solution.)
- **Time Complexity**: Time complexity of bidirectional search using BFS is $O(b^{d/2})$.
- **Space Complexity**: Space complexity of bidirectional search is $O(b^{d/2})$.
- **Optimal**: Bidirectional search is Optimal.

Try

Perform Bidirectional search from 0 to 14.



References

- Stuart Russell and Peter Norvig, “Artificial Intelligence”, 2nd edition, Pearson Education, 2003.
- Saroj Koushik, “Artificial intelligence”.
- NPTEL

IARE
INSTITUTE OF
AERONAUTICAL ENGINEERING

Thank You

IARE
INSTITUTE OF
AERONAUTICAL ENGINEERING

Course Title - Logic Programming for Artificial Intelligence

Topic Title – Informed Search / Heuristic / Intelligent Search

Presenter's Name – Ms. Bidyutlata Sahoo
Presenter's ID – IARE11028
Department Name – CSE (AI & ML)
Lecture Number - 04
Presentation Date – 28/09/2024

Course Outcome

At the end of the course, students should be able to:

CO3: Interpret uninformed and informed search strategies, and select the appropriate approach for different AI problems.

Topic Learning Outcome

Understand the concept of informed or heuristic search where additional information’s are used to find the best possible solution without exhaustively searching every possible state.

Informed / Heuristic / Intelligent Search

Informed / Heuristic / Intelligent Search

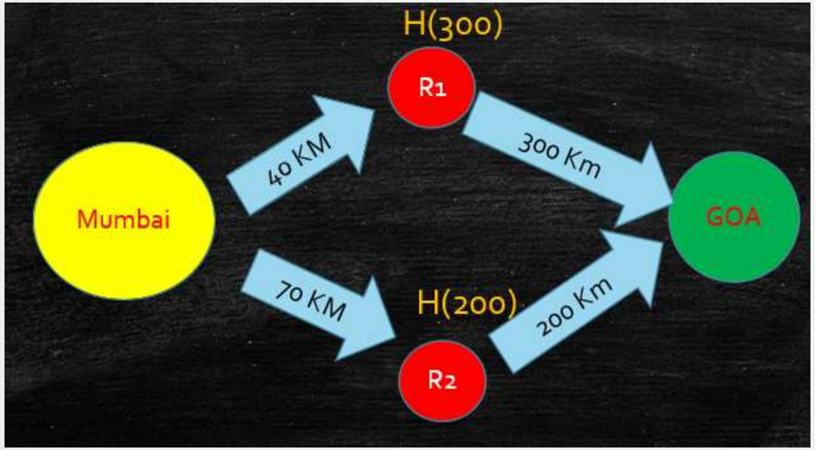
- A heuristic is a technique that is ***used to solve a problem faster than the classic methods***. These techniques are ***used to find the approximate solution of a problem*** when classical methods do not.
- Heuristics are said to be the ***problem-solving techniques that result in practical and quick solutions***.
- Heuristics are used in situations ***where there is the requirement of a short-term solution.*** On facing complex situations with limited resources and time, Heuristics can ***help the companies to make quick decisions by shortcuts and approximated calculations.***

Informed / Heuristic / Intelligent Search (Contd..)

- Heuristic search is a simple searching technique that tries to ***optimize*** a problem using ***Heuristic function.***
- ***Optimization*** means that we will ***try to solve a problem in minimum number of steps or cost.***
- This searching technique ***comes under Informed search.***

Heuristic Function : h(n)

- It is a function H(n) that gives an estimation on the cost of getting from node 'n' to the goal state.
- ***It helps in selecting optimal node for expansion.***
- ***Ex:***



R1= 340 Km

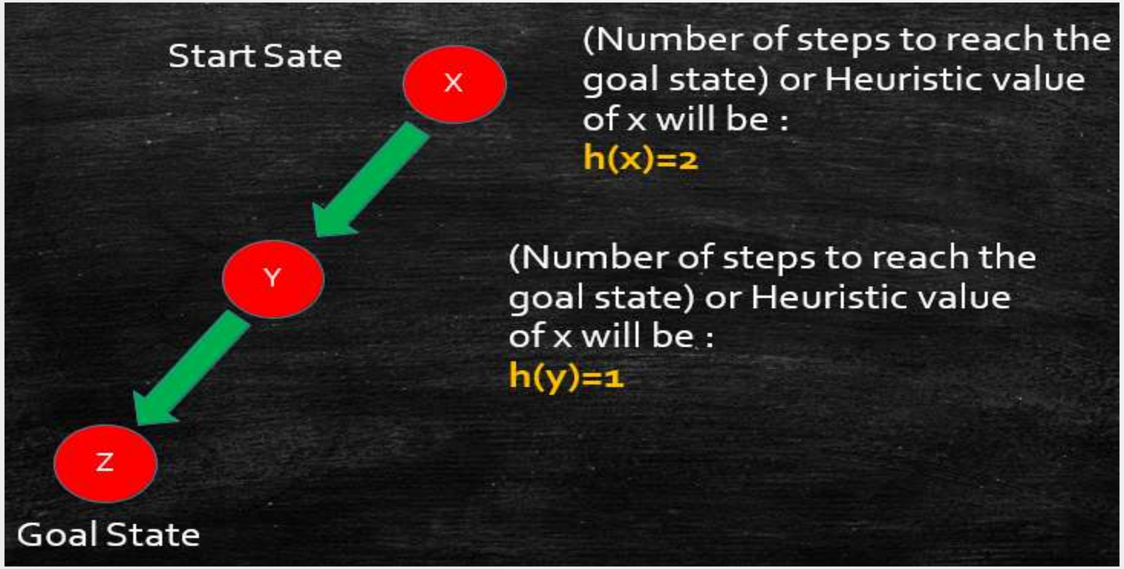
R2= 270Km

Normally, we select least route i.e., 40km from start node. But heuristic search finds the cost from each node till destination.

Here, **Start node**= Mumbai, **Goal node**= GOA

Heuristic Function : h(n)

- Heuristic search is an informed search ***as it is informing us that how far is our goal state.***
- So heuristic function can be denoted as:


Start Sate
X
(Number of steps to reach the goal state) or Heuristic value of x will be :
h(x)=2
Y
(Number of steps to reach the goal state) or Heuristic value of x will be :
h(y)=1
Z
Goal State

Types of Heuristic Function : h(n)

- There are ***two types of Heuristic Functions:***

1. **Admissible**
2. **Non-Admissible**

Admissible Heuristic Function : h(n)

- A heuristic function is admissible if it ***never overestimates the cost of reaching the goal .***
- **i.e.,** it ***underestimates the path cost.***

$$h(n) <= h^*(n)$$

Here h(n) is heuristic cost, and h*(n) is the estimated cost / actual cost.

So ***heuristic cost should be less than or equal to the estimated cost.***

Non-Admissible Heuristic Function : h(n)

- A non-admissible heuristic ***may overestimate the cost of reaching the goal.***

- $h(n) > h^*(n)$

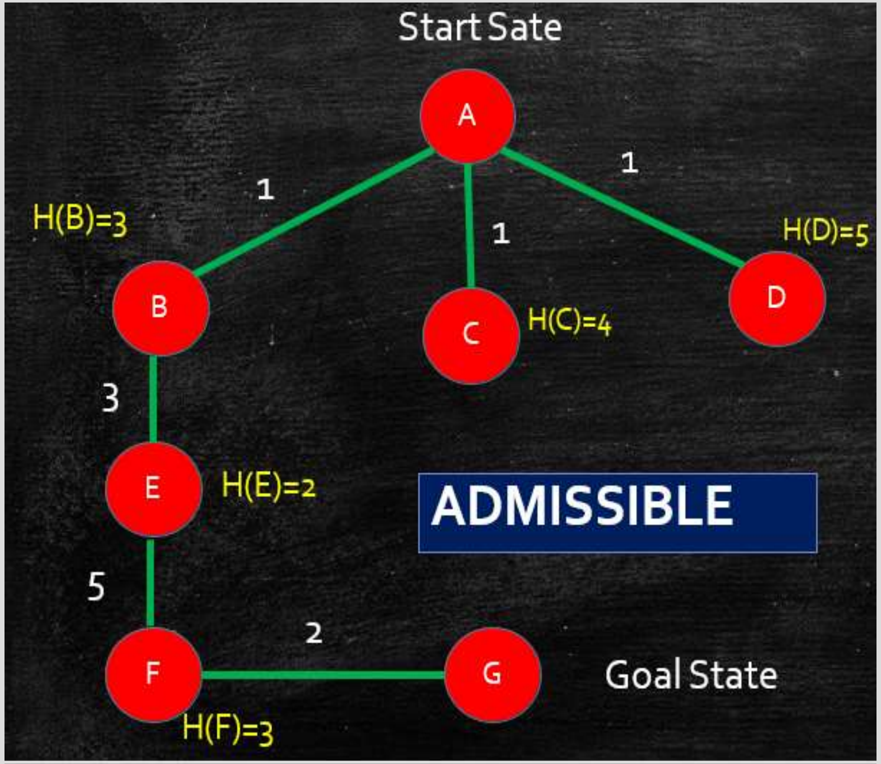
  Here h(n) is heuristic cost, and h*(n) is the estimated cost / actual cost.
  So ***heuristic cost may be greater than to the estimated cost.***

  Total Cost = Heuristic cost + Actual cost
  F(n) = G(n) + H(n)

Admissible Heuristic Function : h(n) : Example

Heuristic Cost given:
H(B)=3, H(C)=4, H(D)=5

F(n) = G(n) + H(n)
Cost = Actual cost + Heuristic cost
B= 1+3=4 (least cost)
C= 1+4=5
D= 1+5=6
Actual cost from A to G = 1+3+5+2= 11
H(B)=3 i.e., **h(n) =3 and h*(n)=11**
i.e., h(n)<=h*(n) so 3<=11
So **search is admissible.**
Same for node E, 2<=11 and node F, 3<=11



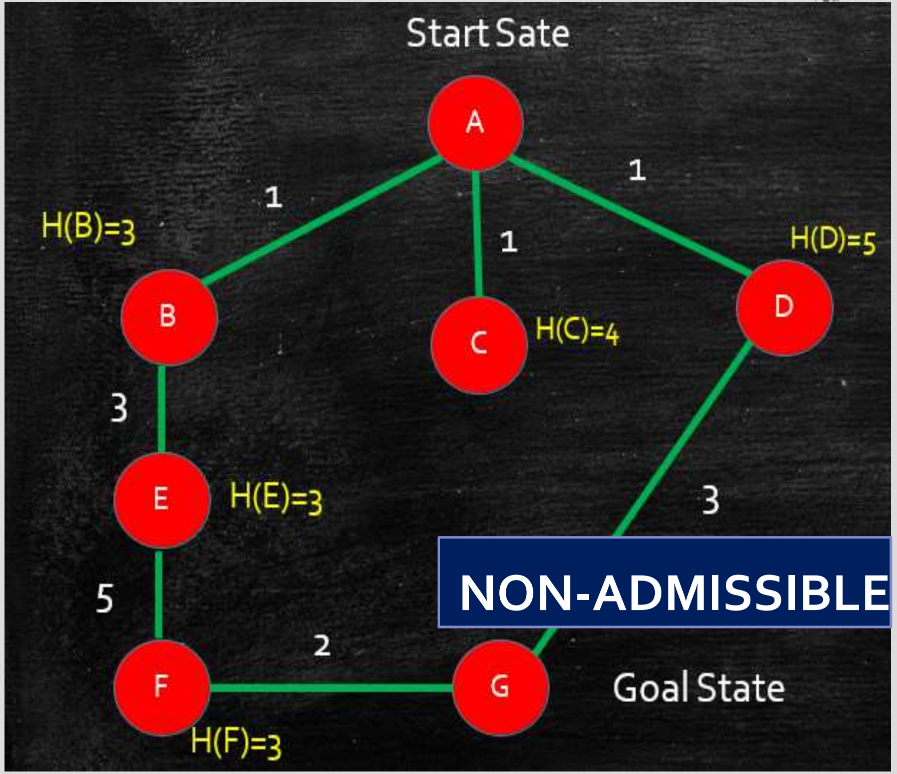
Non-admissible Heuristic Function : h(n) : Example

If we choose the node D,
Heuristic Cost : H(D)=5

F(n) = G(n) + H(n)
Cost = Actual cost + Heuristic cost

Actual cost from A to G(via D) = 1+3= 4
H(D)=5 i.e., **h(n) =5 and h*(n)=4**

i.e., h(n)>h*(n)
5<=4
So **search is non-admissible.**



Heuristic Search Techniques - Categories

1. Direct Heuristic Search techniques
2. Weak Heuristic Search techniques

1. Direct Heuristic Search techniques

- It includes ***Blind Search, Uninformed Search, and Blind control strategy.***
- These search techniques ***are not always possible as they require much memory and time.***
- These techniques *search* ***the complete space*** for a solution and ***use the arbitrary ordering of operations.***
- The **<u>examples</u>** of Direct Heuristic search techniques include Breadth-First Search ***(BFS)*** and Depth First Search ***(DFS).***

2. Weak Heuristic Search techniques

- It includes ***Informed Search, Heuristic Search, and Heuristic control strategy.***
- These techniques are helpful when they are applied properly to the right types of tasks. They usually ***require domain-specific information.***
- The **examples** of Weak Heuristic search techniques include Best First Search ***(BFS) and A*.***

Use of Heuristic Evaluation Functions

- The following algorithms make use of heuristic evaluation functions:
  - ***Hill Climbing***
  - ***Generate-and-Test***
  - ***Best-First Search***
  - ***A* Algorithm***
  - ***AO* Algorithm***
  - ***Beam Search***
  - ***Constraint Satisfaction***

References

- Stuart Russell and Peter Norvig, “Artificial Intelligence”, 2nd edition, Pearson Education, 2003.
- Saroj Koushik, “Artificial intelligence”.
- NPTEL

IARE
INSTITUTE OF
AERONAUTICAL ENGINEERING

Thank You

IARE
INSTITUTE OF
AERONAUTICAL ENGINEERING

Course Title - Logic Programming for Artificial Intelligence

Topic Title – Informed Search (Generate-and-Test, Best First Search, A* Algorithm)

Presenter’s Name – Ms. Bidyutlata Sahoo
Presenter’s ID – IARE11028
Department Name – CSE (AI & ML)
Lecture Number - 05
Presentation Date – 28/09/2024

Course Outcome

At the end of the course, students should be able to:

CO3: Interpret uninformed and informed search strategies, and select the appropriate approach for different AI problems.

Topic Learning Outcome

Understand the concept of informed or heuristic search where additional information’s are used to find the best possible solution without exhaustively searching every possible state.

What is Global Search?

Global search techniques ***explore the entire solution space*** to find the optimal / satisfactory solution.

Global Search Techniques:

1. **Generate and Test**
2. **Best First Search(OR graph)**

Where not only the current branch of the search space but all the so far explored nodes/states in the search space are considered in determining the next best state/node.

3. **A* Algorithm**

Which is improvised version of Best first Search.

4. **Problem Reduction and And-Or Graphs.**

AO* Algorithm.

5. **Constraint Satisfaction Problem (CSP)**

Graph Colouring Problem and Crypt Arithmetic Problems.

6. **Mean End Analysis (MEA)**

Informed / Heuristic / Intelligent Search
(1. Generate-and-Test)

Generate-and-Test

- It is a ***heuristic / informed search***.
- In this technique ***all the solutions are generated and tested for best solution.***
- It ensures that the ***best solution is checked from all possible generated solutions.***
- This approach is what is ***known as the British Museum algorithm***: finding an object in the British Museum by wandering randomly.
- It is ***like depth-first search with backtracking***, requires that complete solutions be generated for testing.
- If the solution is found, then quit.

Generate-and-Test

- The generate-and-test s***trategy is the simplest of all the approaches***. It consists of the following steps:

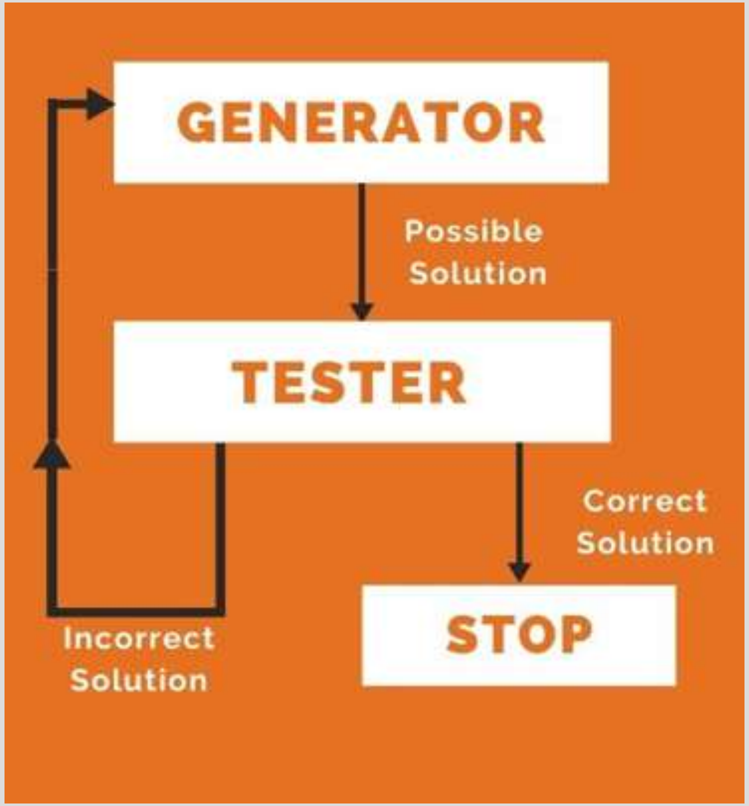
- **Algorithm**

1. Generate all possible solutions.
2. Select one solution among the possible solutions.
3. If a solution has been found and acceptable, quit. Otherwise return to step 1.

Generate-and-Test-Diagram

Two approaches are followed while generating solutions:

- Generating complete solutions &
- Generating random solutions



[Generate and Test Heuristic Search Algorithm]

Generate-and-Test – Example:

Travelling Salesman Problem (TSP)

A salesman has a list of cities, each of which he must visit exactly once. There are direct roads between each pair of cities on the list. Find the route the salesman should follow for the shortest possible round trip that both starts and finishes at any one of the cities.

Generate-and-Test – Example:

Travelling Salesman Problem (TSP)

You are given-

- A set of some cities.
- Distance between every pair of cities.

Travelling Salesman Problem states-

- A salesman has to visit every city exactly once.
- He has to come back to the city from where he starts his journey.
- What is the shortest possible route that the salesman must follow to complete his tour?

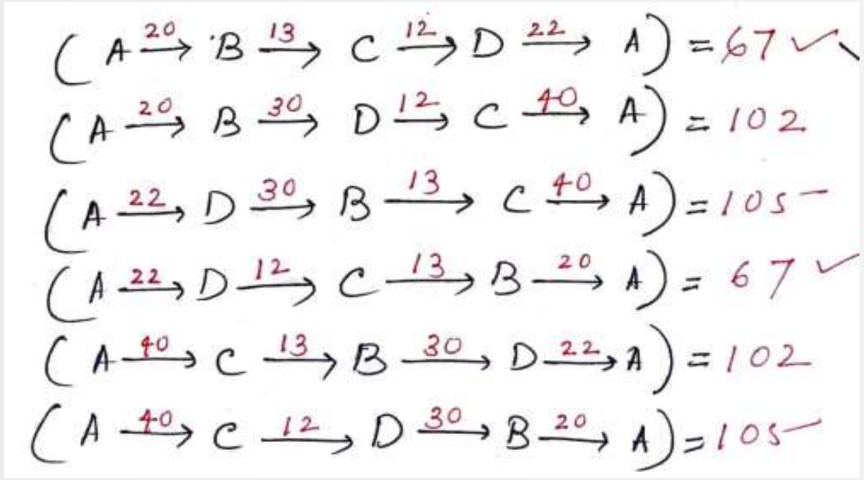
Travelling Salesman Problem (TSP)

Let the **initial state be A.**

So, all possible states from A will be:

{(A→B),(A→D),(A→C)}

i.e.,



$(A \xrightarrow{20} B \xrightarrow{13} C \xrightarrow{12} D \xrightarrow{22} A) = 67$ ✓

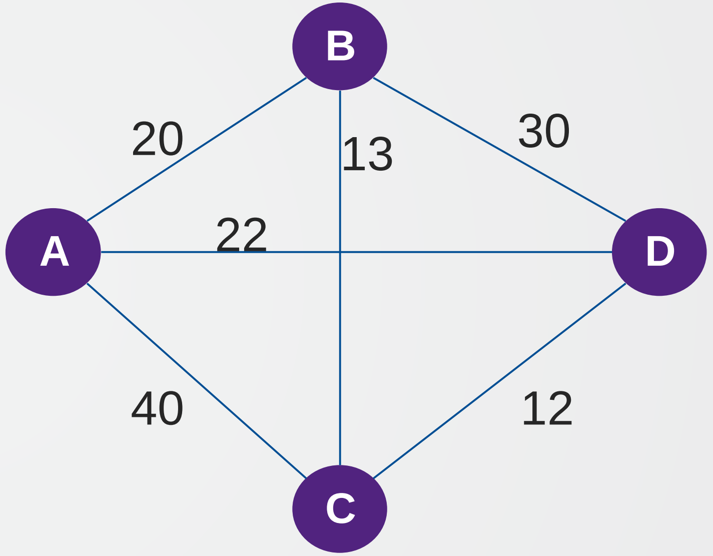
$(A \xrightarrow{20} B \xrightarrow{30} D \xrightarrow{12} C \xrightarrow{40} A) = 102$

$(A \xrightarrow{22} D \xrightarrow{30} B \xrightarrow{13} C \xrightarrow{40} A) = 105$ —

$(A \xrightarrow{22} D \xrightarrow{12} C \xrightarrow{13} B \xrightarrow{20} A) = 67$ ✓

$(A \xrightarrow{40} C \xrightarrow{13} B \xrightarrow{30} D \xrightarrow{22} A) = 102$

$(A \xrightarrow{40} C \xrightarrow{12} D \xrightarrow{30} B \xrightarrow{20} A) = 105$ —



67 is the shortest path cost if A is considered as the initial point.

Travelling Salesman Problem (TSP)

Similarly, let the **initial state be B.**
So, all possible states from B will be:
{(B→A),(B→C),(B→D)}

Similarly, let the **initial state be C.**
So, all possible states from C will be:
{(C→A),(C→B),(C→D)}

Similarly, let the **initial state be D.**
So, all possible states from D will be:
{(D→B),(D→C),(D→A)}

Find all possible routes taking into consideration as initial point as B, C, D and observe the shortest path among all. Finally, select the path whose length is less..

Informed / Heuristic / Intelligent Search
(2. Best First Search(BFS/Greedy Search/OR graph))

Best First Search (BFS)

- This is an ***informed search*** technique(Greedy Search) also called as ***HEURISTIC search.***
- This algorithm works using heuristic value.
- ***Every node*** in the search space has an ***Evaluation function*** (***heuristic function***)associated with it.
- Evaluation value= cost/distance of current node from goal node.
- For goal node evaluation function value=0
- ***Evaluation function==heuristic cost function*** (in case of minimization problem) OR ***objective function***(in case of maximization).

Best First Search (BFS)

- This algorithm uses ***evaluation function*** to decide ***which adjacent node is most promising and then explore.***
- ***i.e., the node which is having lowest evaluation is selected for the expansion because the evaluation measures distance to the goal.***
- ***Priority queue*** is used to store cost of function.
- It serves as a ***combination of both BFS and DFS.***
- This is also called as ***Greedy Search*** as it quickly attack the most desirable path as soon as its heuristic weight becomes the most desirable.

Best First Search (BFS) – maintains two lists

- The implementation of Best First Search Algorithm involves *maintaining two lists*- **OPEN and CLOSED**.
- **OPEN** list contains those nodes *that have been evaluated by the heuristic function but have not been expanded* into successors yet.
- **CLOSED** list contains those nodes *that have already been visited.*

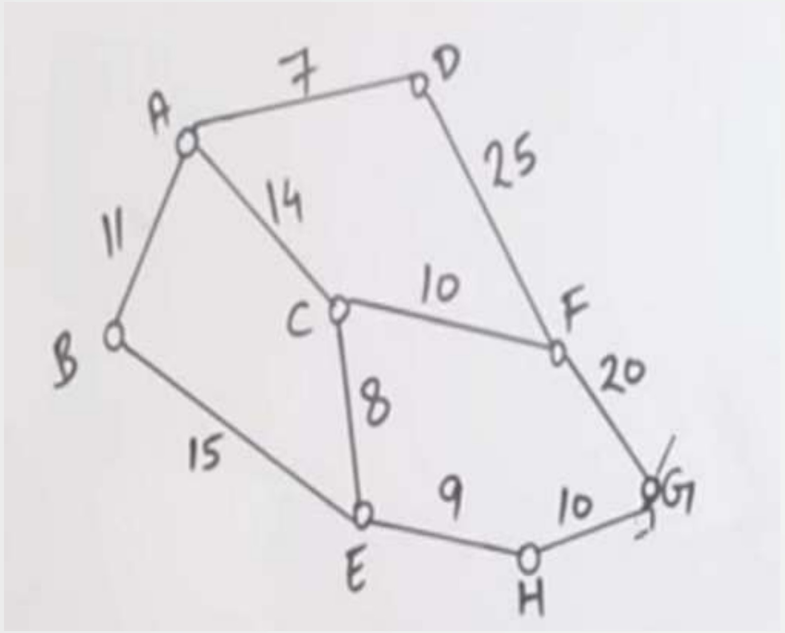
Best First Search (BFS) - Concept

- **S1**: Traverse the root node.
- **S2**: Traverse any neighbour of the root node, that is maintaining a least distance from the root node and insert them in ascending order into the queue.
- **S3**: Traverse any neighbour of neighbour of the root node, that is maintaining a least distance from the root node and insert them in ascending order into the queue.
- **S4:** The process will continue until we are getting the goal node.

Best First Search (BFS) - Algorithm

```
Priority queue 'PQ' containing initial states.
Loop
    If PQ=Empty Return Fail
Else
    Insert Node into PQ(Open-List)
    Remove First(PQ) ->NODE(Close-List)
    If NODE=GOAL
    Return path from initial state to NODE
Else
    Generate all successor of NODE and insert newly generated NODE into 'PQ'
    according to cost value.
End Loop
```

Best First Search (BFS) – Example



Here, A= Root node, G=Goal node.

Straight-line Distance / f(n)

A→G =40
B→G =32
C→G =25
D→G =35
E→G =19
F→G =17
G→G =0
H→G =10

OPEN	CLOSED
[A]	[ ]
[C,B,D]	[A]
[B,D]	[A,C]
[F,E,B,D]	[A,C]
[G,E,B,D]	[A,C,F]
[E,B,D]	[A,C,F,G]

Path = A→C→F→G, Cost = 44

Best First Search (BFS) - Advantages

- ***Memory efficient as compared to DFS & BFS*.**
- It is ***Complete.***
- ***Time Complexity is much lesser than BFS.***

Best First Search (BFS) - Disadvantages

- It gives good solution but ***not optimal solution.***
- In worst case it may behave like unguided DFS i.e., ***sometimes it covers more distance than our consideration.***

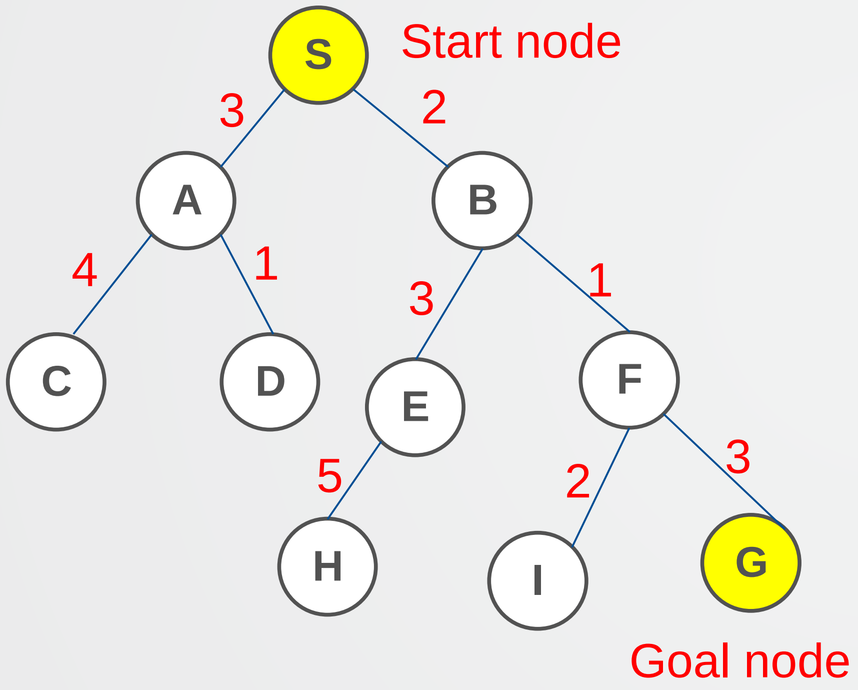
Best First Search (BFS) - Analysis

- ***Space Complexity***: $O(b^d)$
- ***Time Complexity***: $O(b^d)$

  [where, b→ branching factor, d→ depth]
- ***Solution:*** not optimal.

Try

Find the shortest path from S to G using Best First Search.



Straight-line Distance / f(n)

Node(n)	**f(n)**
A→G	=12
B→G	=4
C→G	=7
D→G	=3
E→G	=8
F→G	=2
G→G	=0
H→G	=4
I→G	=9
S→G	=13

Informed / Heuristic / Intelligent Search

(3. A* Algorithm)

A* Algorithm

- This is an informed search technique also called as ***HEURISTIC search.***
- It is essentially the ***extension of best first search algorithm.***
- This algorithm works using heuristic value.
- **A* uses**
    - **h(n)->**Heuristic function (cost from current node to goal node)
    - **g(n)**->Actual cost (cost from start node to current node)

A* Algorithm (Contd..)

- It ***finds shortest path*** though search space.
- It ***provides fast and optimal result.***
- A* requires the heuristic function to evaluate the cost of the path that passes through the particular state. It can be defined by the following formula.

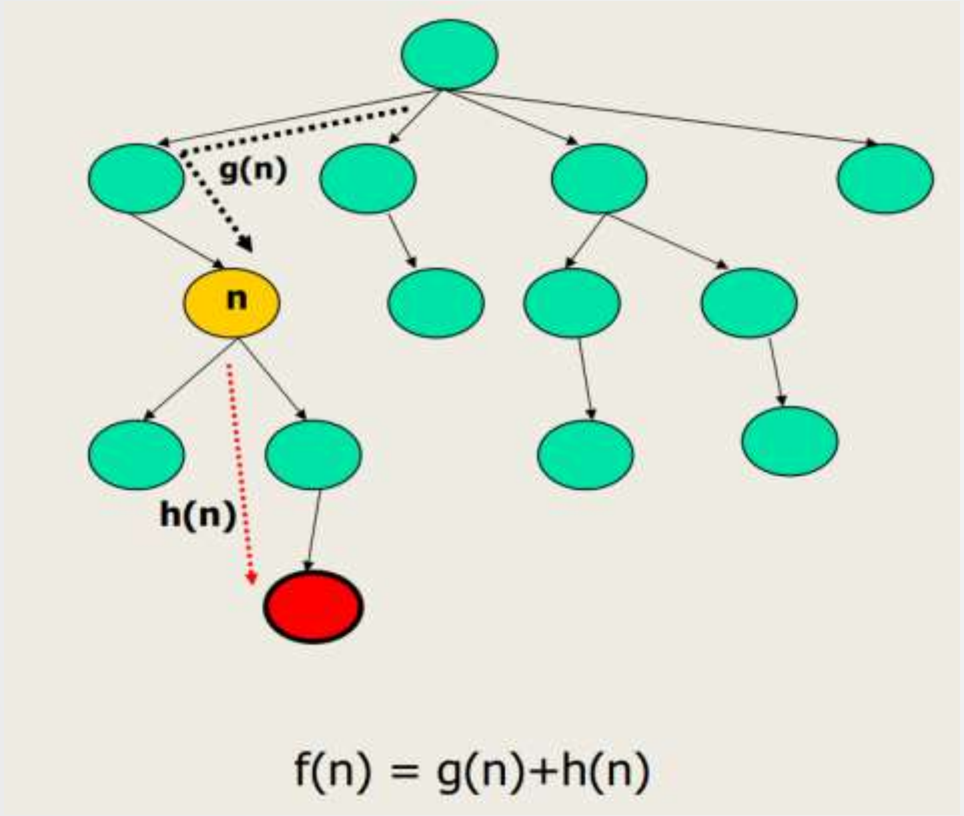
Estimated Cost : **$f(n)=g(n)+h(n)$**

[Where,f(n) = estimated cost of the node

g(n) = is the cost of the path from start node to node ‘n’

h(n) = heuristic value(*of child node always considered)*]

A* Algorithm (Contd..)



A* Algorithm (Contd..) – maintains two lists

- The implementation of A* Algorithm involves maintaining two lists- ***OPEN and CLOSED.***

1. **OPEN** list contains those nodes that *have been evaluated by the heuristic function but have not been expanded* into successors yet.
2. **CLOSED** list contains those nodes *that have already been visited.*

A* Algorithm (Contd..) – Algorithm

- **Step-01:**
  - Enter starting node in OPEN list.
- **Step-02:**
  - If OPEN list is empty, return fail and exit.
- **Step-03:**

```
Select the node from OPEN list which has smallest value(g+h).
        If node=Goal,
               return SUCCESS.
```

A* Algorithm (Contd..) – Algorithm

- **Step-04:**
  - Expand node ‘n’ and generate all successors. Compute (g+h) for each successor node.
- **Step-05:**
  - If node ‘n’ is already in OPEN / CLOSED, attach to backpointer.
- **Step-06:**

  Go back to Step-02.

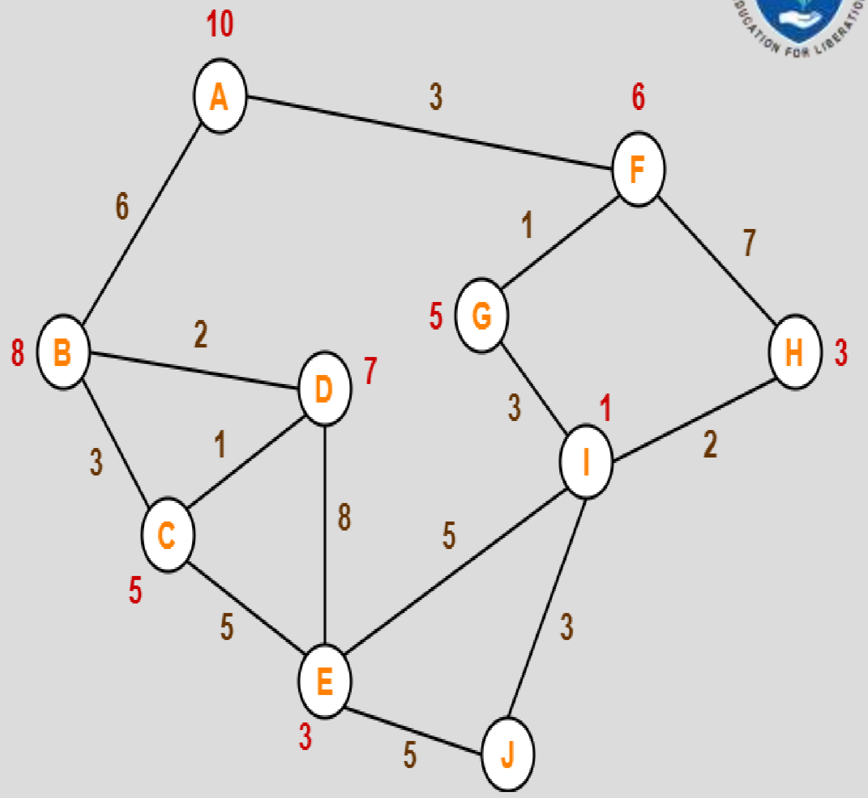
A* Algorithm (Contd..) – Example-1

Consider the following graph-

❖The numbers ***written on edges*** represent the distance between the nodes.

❖The numbers ***written on nodes*** represent the heuristic value.

❖Find the most cost-effective path to reach from ***start state A*** to ***final state J*** using A* Algorithm.

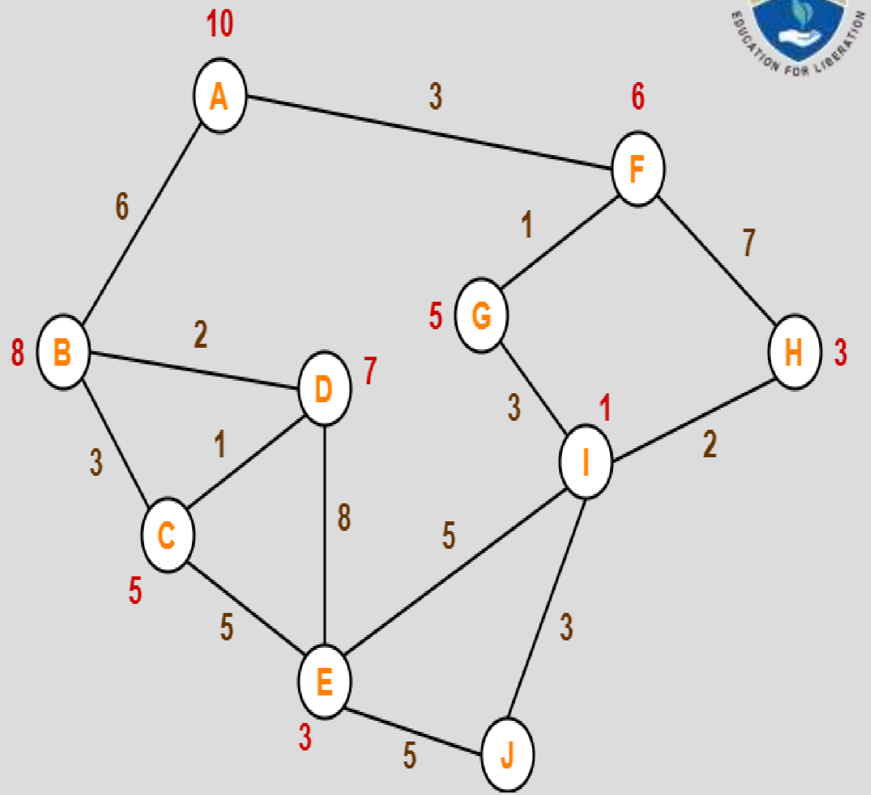


A* Algorithm (Contd..) – Example-1

Sloution:

❖**Step-01:**

- We start with node A. Node B and Node F can be reached from node A.
- A* Algorithm calculates f(B) and f(F).
- Estimated Cost **f(n)=g(n)+h(n)**
  - f(B) = g(B) + h(B) = 6 + 8 = 14
  - f(F) = g(F) + h(F) = 3 + 6 = 9
- Since f(F) < f(B), so it decides to go to node F.



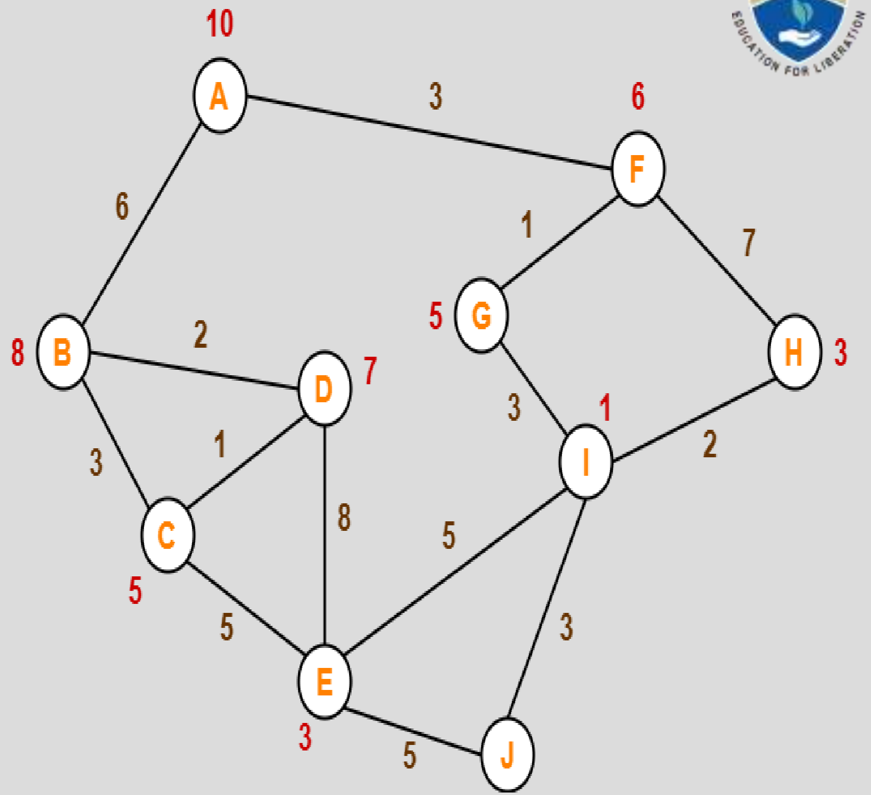
Closed list(F)
Path: A → F

A* Algorithm (Contd..) – Example-1

Sloution:

❖ **Step-02:**

- Node G and Node H can be reached from node F.
- A* Algorithm calculates f(G) and f(H).
  - f(G) = g(G) + h(G) = (3+1) + 5 = 9
  - f(H) = g(H) + h(H) = (3+7) + 3 = 13
- Since f(G) < f(H), so it decides to go to node G.



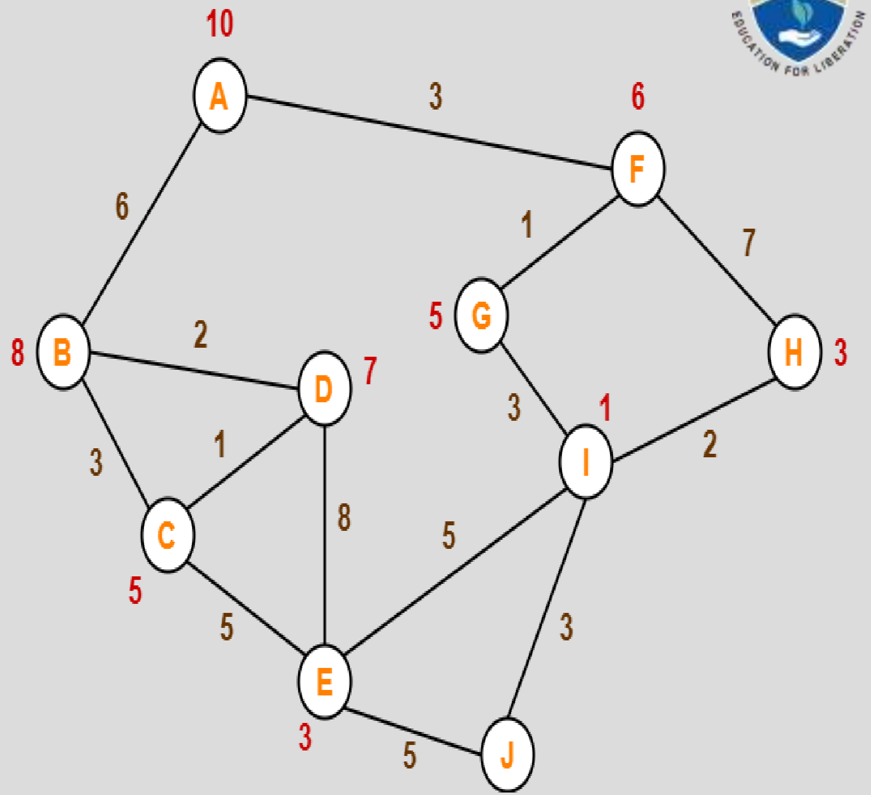
Closed list(G)
Path: A → F → G

A* Algorithm (Contd..) – Example-1

Sloution:

❖**Step-03:**

- Node I can be reached from node G.
- A* Algorithm calculates f(I).
  - f(I) = g(I) + h(I) = (3+1+3) + 1 = 8
- It decides to go to node I.



Closed list(I)

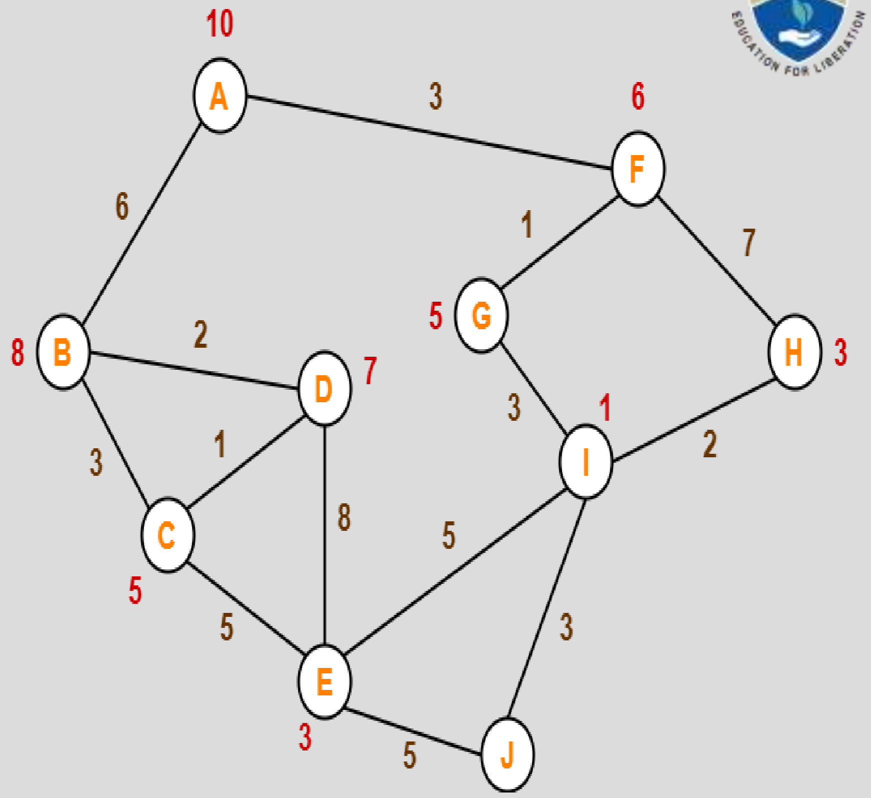
Path: A → F → G → I

A* Algorithm (Contd..) – Example-1

Sloution:

❖**Step-04:**

- Node E, Node H and Node J can be reached from node I.
- A* Algorithm calculates f(E), f(H) and f(J).
  - f(E) = g(E) + h(E) = (3+1+3+5) + 3 = 15
  - f(H) = g(H) + h(H) = (3+1+3+2) + 3 = 12
  - f(J) = g(J) + h(J) = (3+1+3+3) + 0 = 10
- Since f(J) is least, so it decides to go to node J.



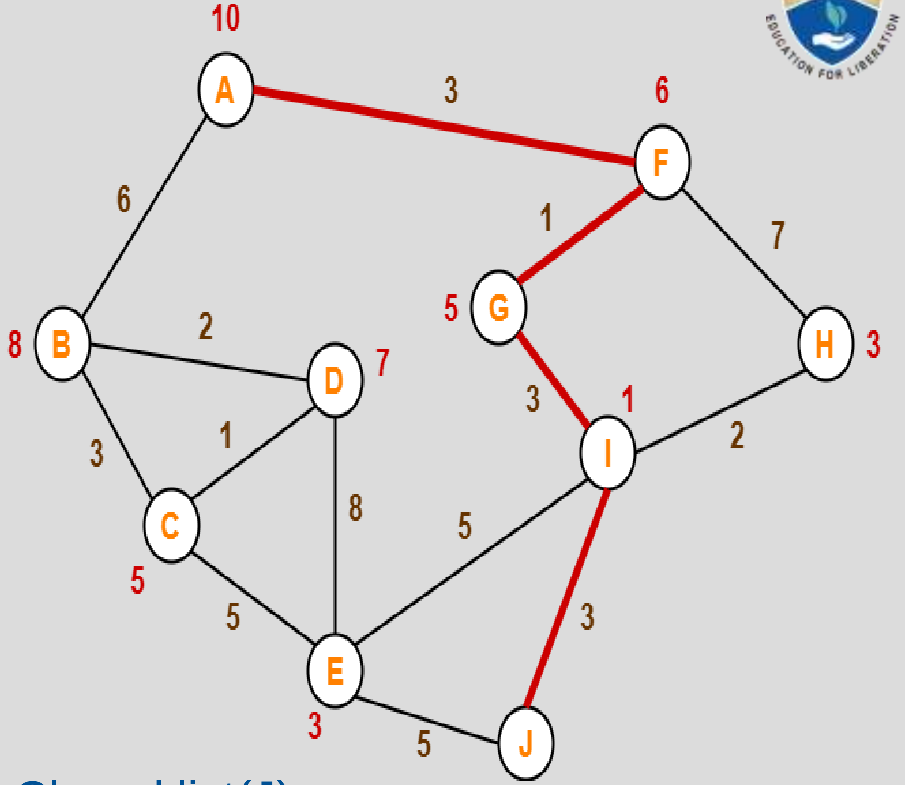
Closed list(J)
Path: A → F → G → I → J

A* Algorithm (Contd..) – Example-1

This is the required shortest path from node A to node J.

OPEN	CLOSED
[A]	[ ]
[F,B]	[A]
[G,H,B]	[A,F]
[I,H,B]	[A,F,G]
[J,H,B,E]	[A,F,G,I]
[H,B,E]	[A,F,G,I,J]

Goal node is found.

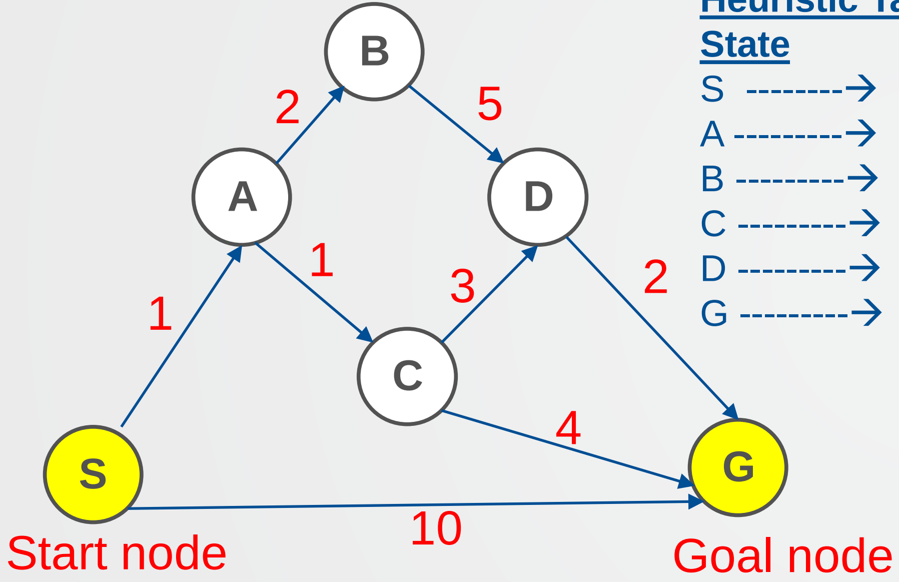


Closed list(J)

Path: A → F → G → I → J

A* Algorithm (Contd..) – Example-2

Find the shortest path from S to G using A* Algorithm by referring the given heuristic table.



Heuristic Table

State	h(n)
S --------→	5
A ---------→	3
B ---------→	4
C ---------→	2
D ---------→	6
G ---------→	0

SOLUTION:

S→A=1+3=4
S→G=10+0=10------(route1)

S→A→B=(1+2)+4=7
S→A→C=(1+1)+2=4

S→A→C→D=(1+1+3)+6=11
S→A→C→G=(1+1+4)+0=6------(route2)

S→A→B→D=(1+2+5)+6=14
S→A→C→D→G=(1+1+3+2)+0=7-----(route3)

S→A→B→D→G=(1+2+5+2)+0=10--(route4)

Path: S→A→C→G, cost=6(least cost)

A* Algorithm (Contd..) – Advantages

- A* Algorithm is ***one of the best path finding algorithms.***
- It is ***Complete & Optimal.***
- Used to ***solve complex problems***.

A* Algorithm (Contd..) – Disadvantages

- Doesn’t always produce shortest path.
- It has complexity issues.
- Requires more memory.

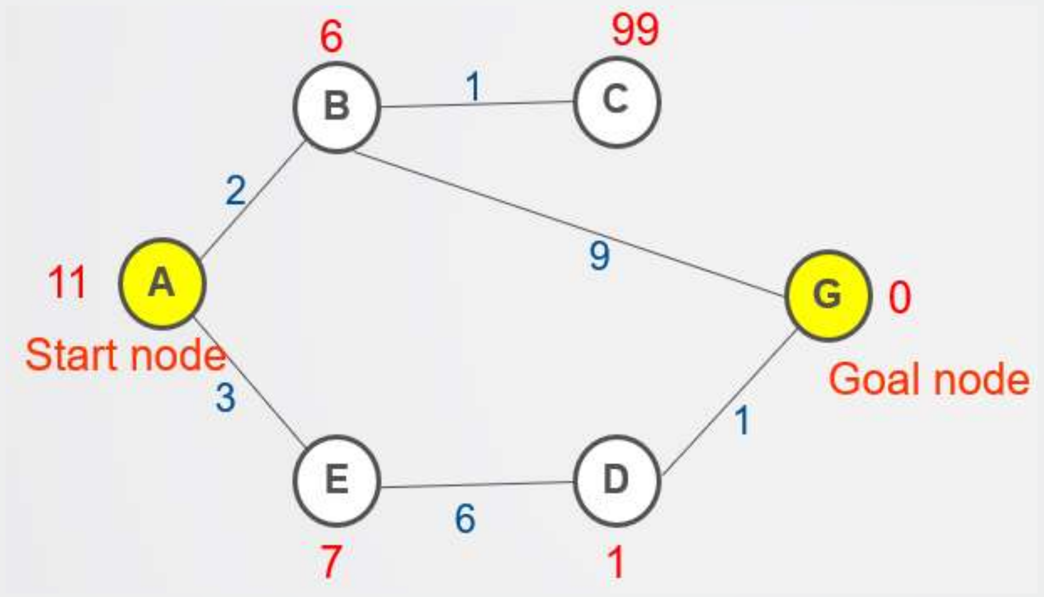
A* Algorithm (Contd..) – Analysis

- ***Space Complexity***: O(b^d)
- ***Time Complexity***: O(b^d) [depends on the heuristic.]

  [where, b→ branching factor, d→ depth]
- ***Solution:*** Optimal. [guaranteed to find a shortest path if one exists.]

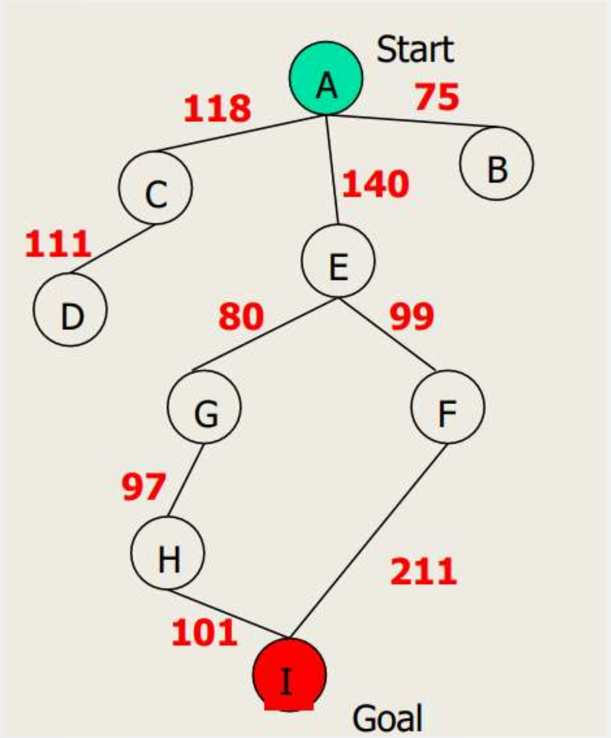
Try

- For the given graph, find the cost effective path from A to G using A* algorithm.



Try

- For the given graph, find the cost effective path from A to I using A* algorithm.



State	Heuristic: h(n)
A	366
B	374
C	329
D	244
E	253
F	178
G	193
H	98
I	0

References

- Stuart Russell and Peter Norvig, “Artificial Intelligence”, 2nd edition, Pearson Education, 2003.
- Saroj Koushik, “Artificial intelligence”.
- NPTEL

IARE
INSTITUTE OF
AERONAUTICAL ENGINEERING

Thank You