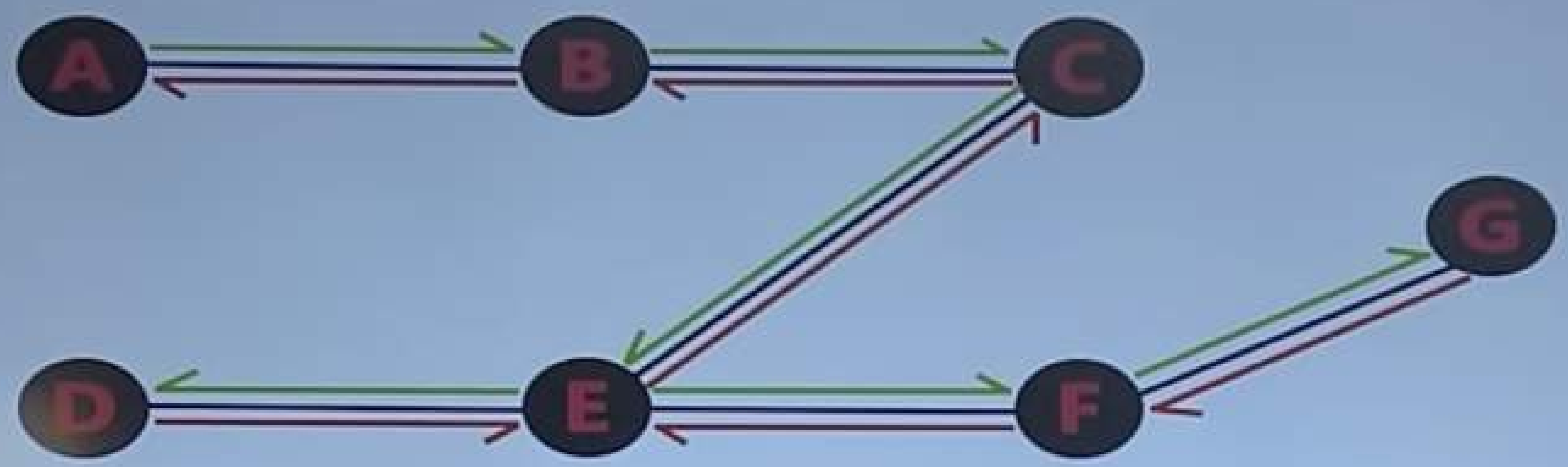
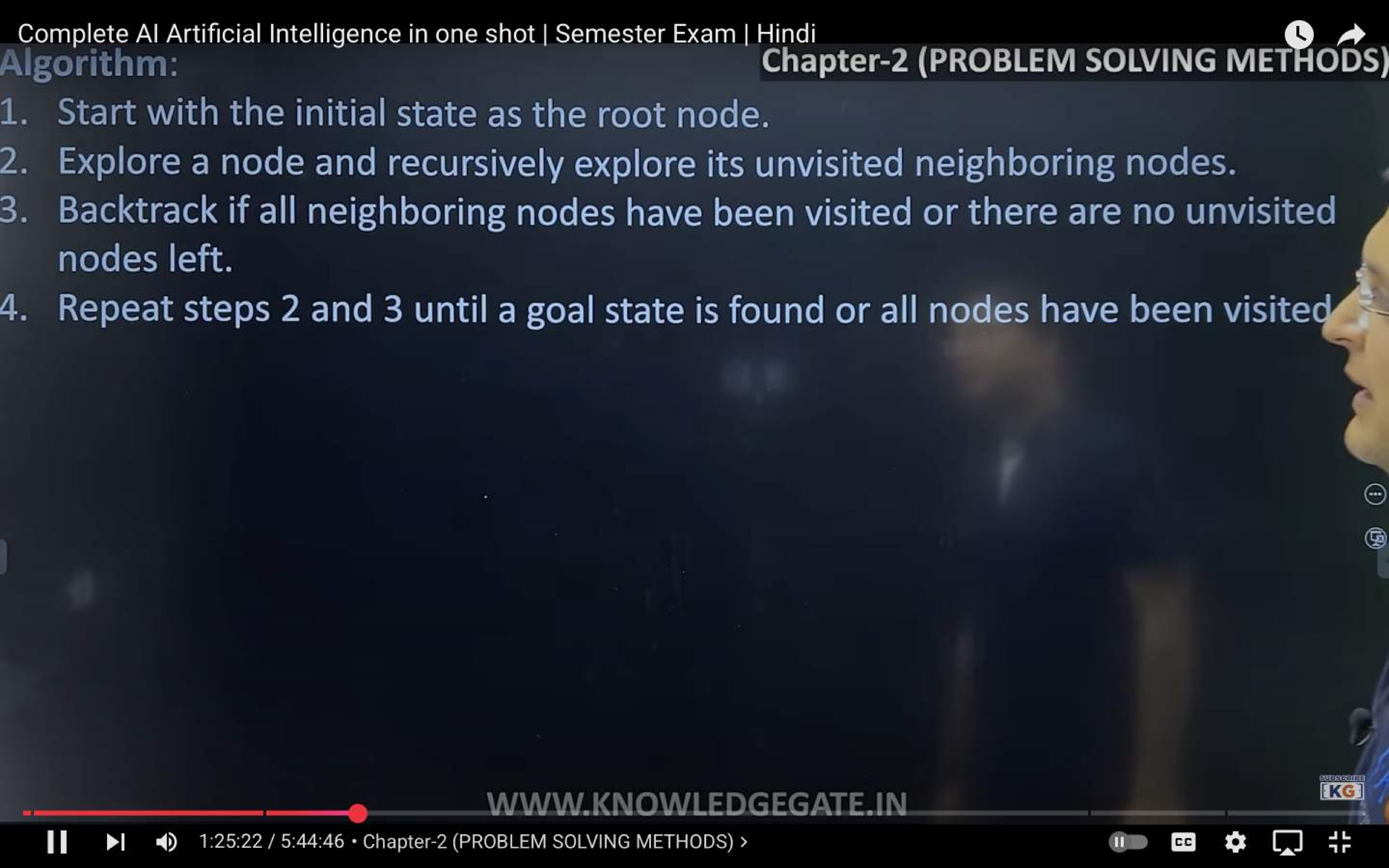
Complete AI Artificial Intelligence in one shot | Semester Exam | Hindi

- Stack became Empty. So stop DFS Treversal.
- Final result of DFS traversal is following spanning tree.



1:25:18 / 5:44:46 • Chapter-2 (PROBLEM SOLVING METHODS) >


Complete AI Artificial Intelligence in one shot | Semester Exam | Hindi
1:25:22 / 5:44:46 • Chapter-2 (PROBLEM SOLVING METHODS) ›

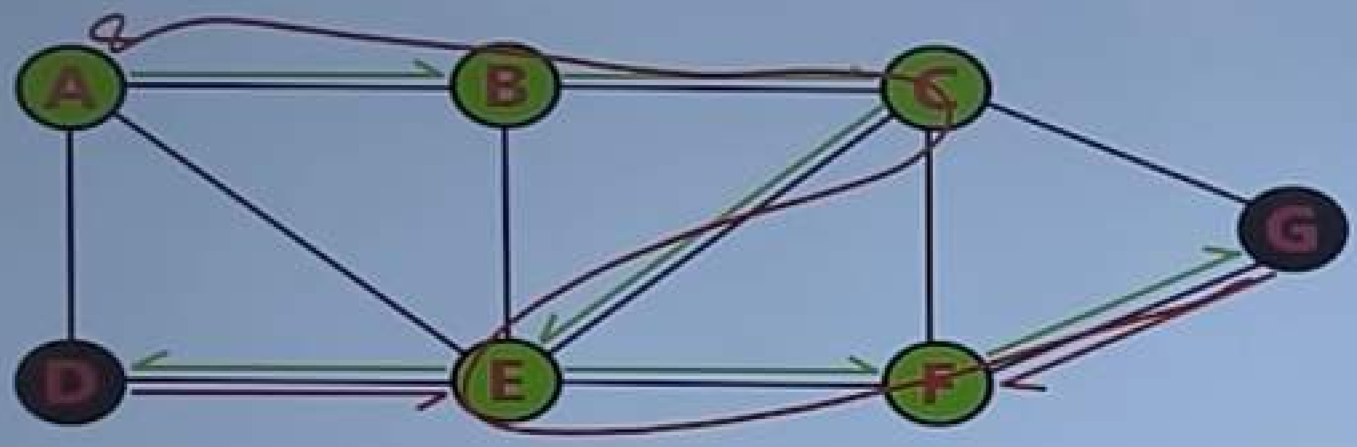
Algorithm:

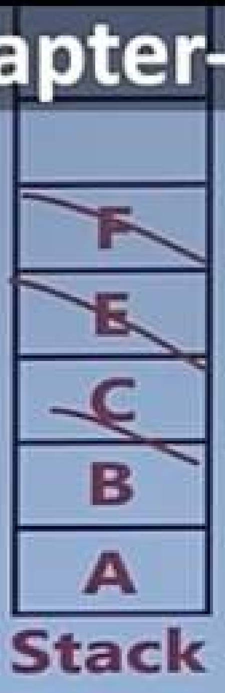
Chapter-2 (PROBLEM SOLVING METHODS)

1. Start with the initial state as the root node.
2. Explore a node and recursively explore its unvisited neighboring nodes.
3. Backtrack if all neighboring nodes have been visited or there are no unvisited nodes left.
4. Repeat steps 2 and 3 until a goal state is found or all nodes have been visited

Step 9:

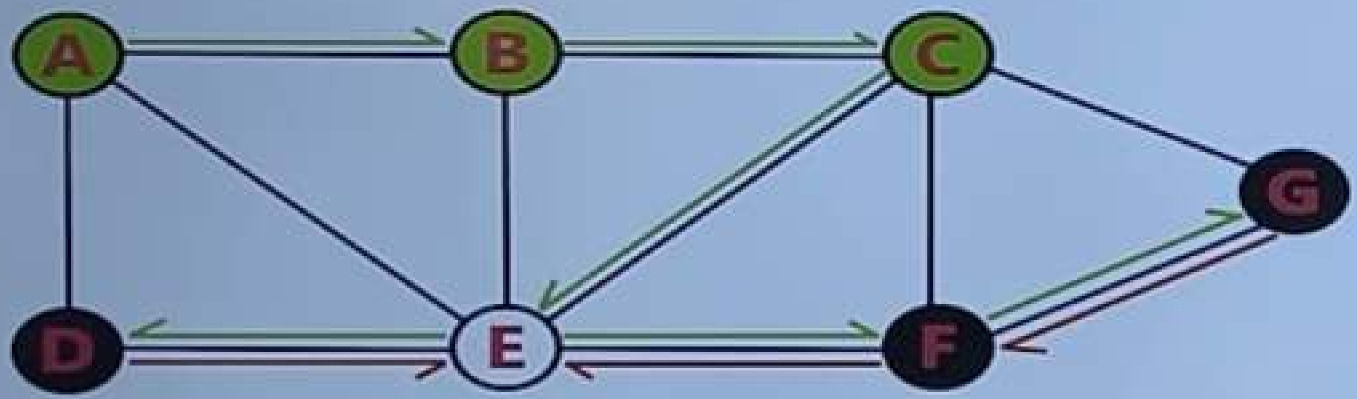
- There is no new vertiex to be visited from G. So use back track.
- Pop G from the Stack.

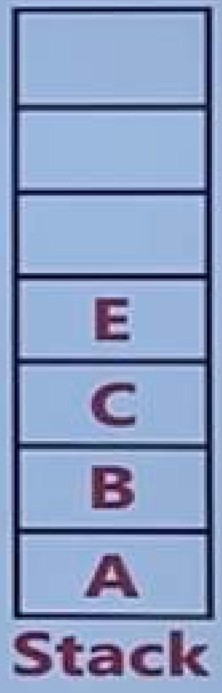




Step 10:

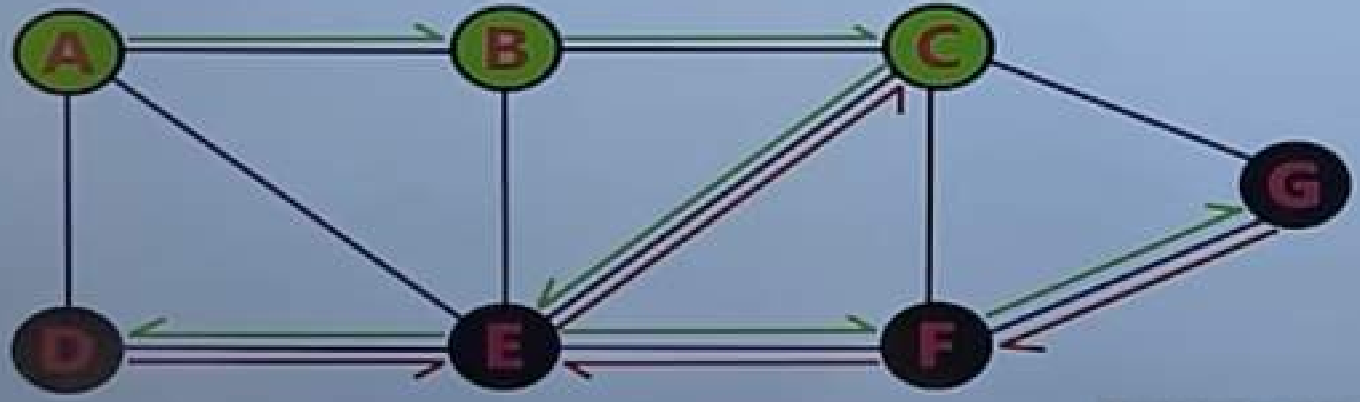
- There is no new vertiex to be visited from F. So use back track.
- Pop F from the Stack.

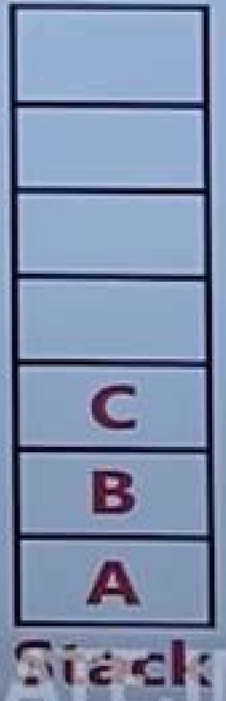




Step 11:

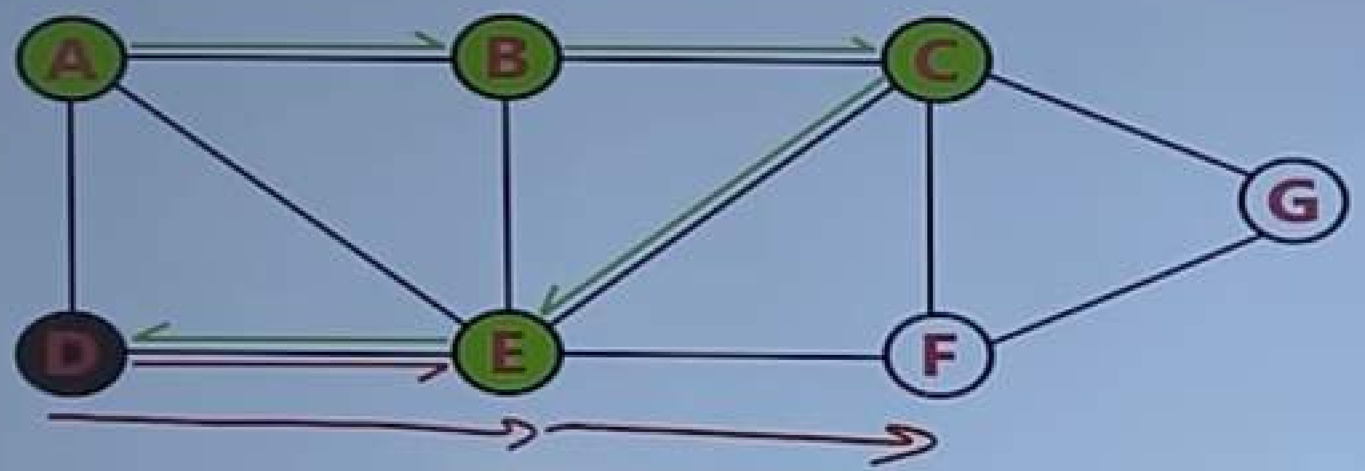
- There is no new vertiex to be visited from E. So use back track.
- Pop E from the Stack.

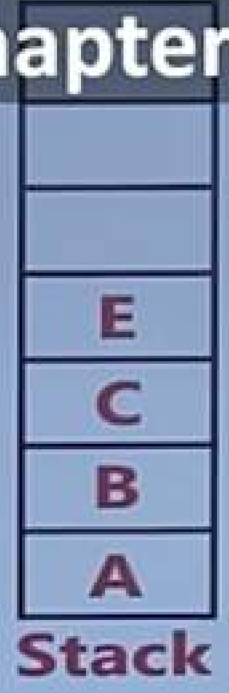




Step 6:

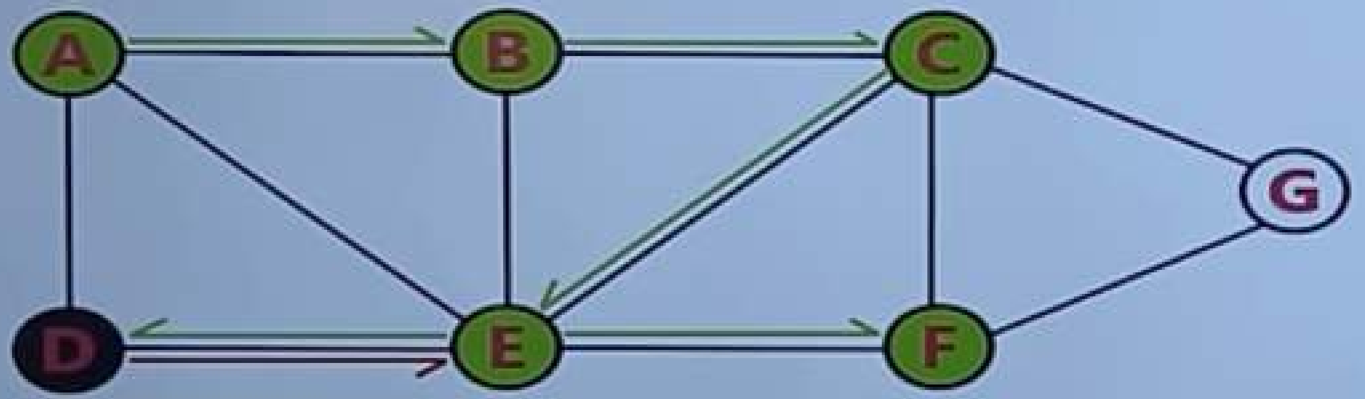
- There is no new vertiex to be visited from D. So use back track.
- Pop D from the Stack.

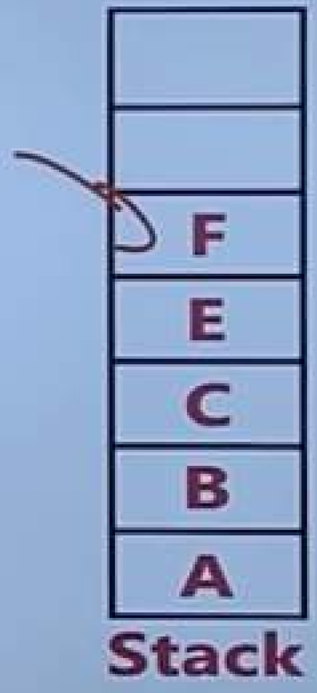




Step 7:

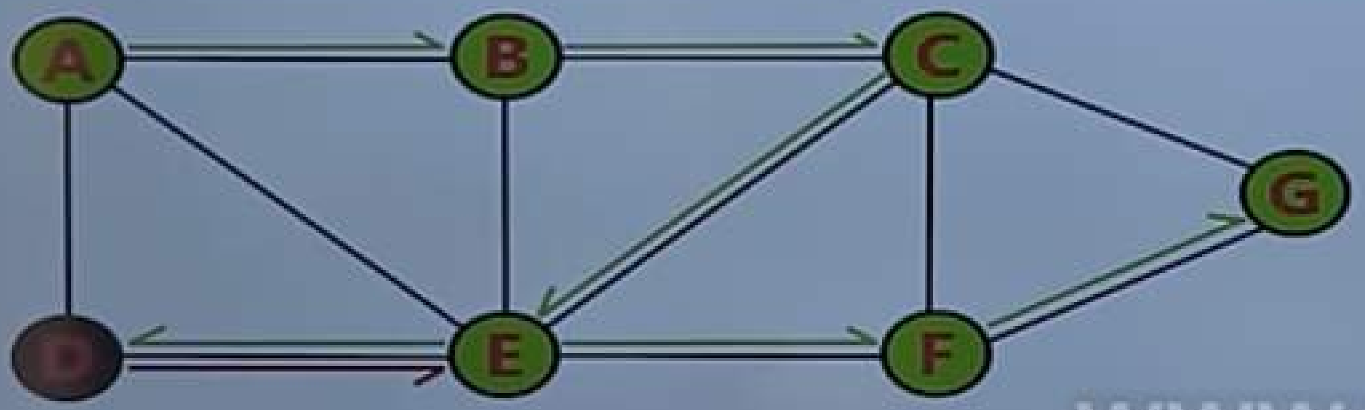
- Visit any adjacent vertex of **E** which is not visited (**F**).
- Push **F** on to the Stack.

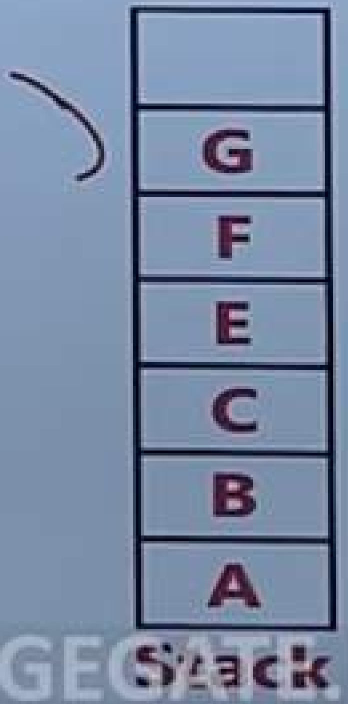




Step 8:

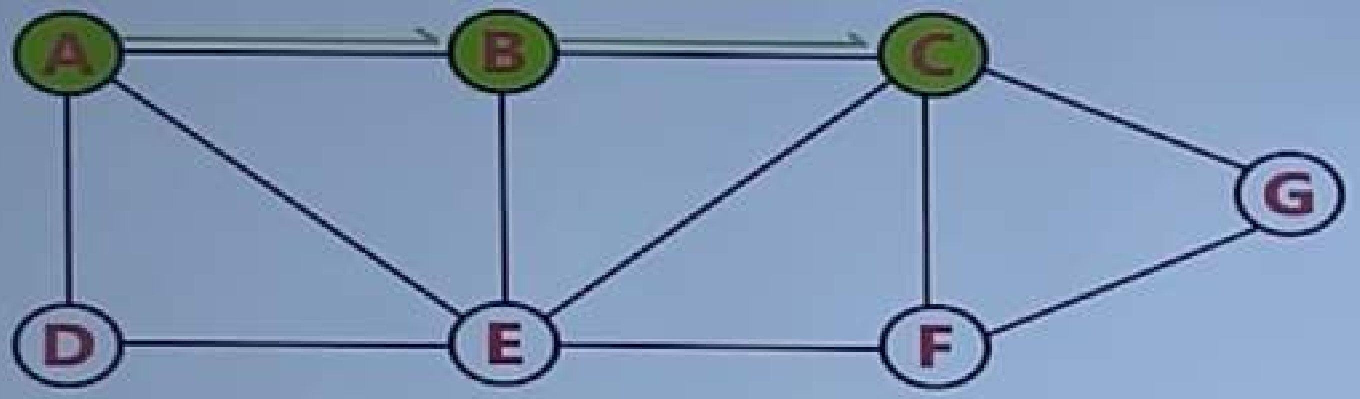
- Visit any adjacent vertex of **F** which is not visited (**G**).
- Push **G** on to the Stack.

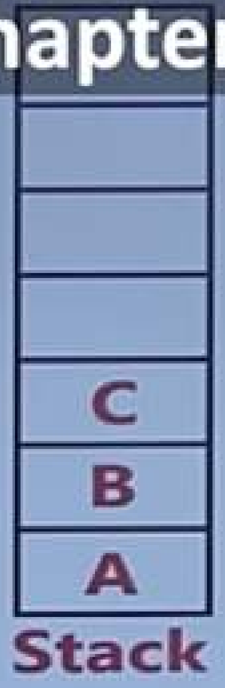




Step 3:

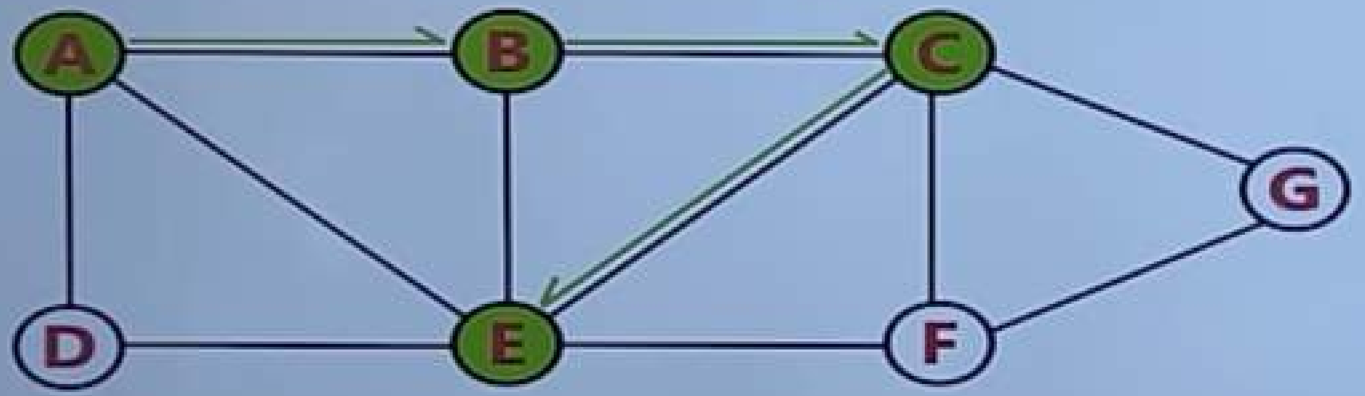
- Visit any adjacent vertext of **B** which is not visited (**C**).
- Push C on to the Stack.

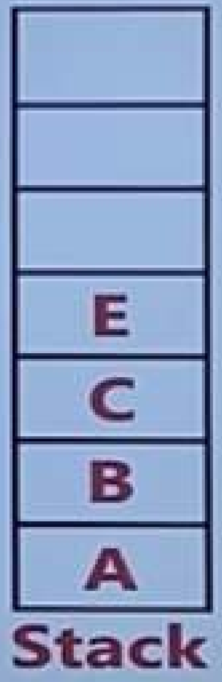




Step 4:

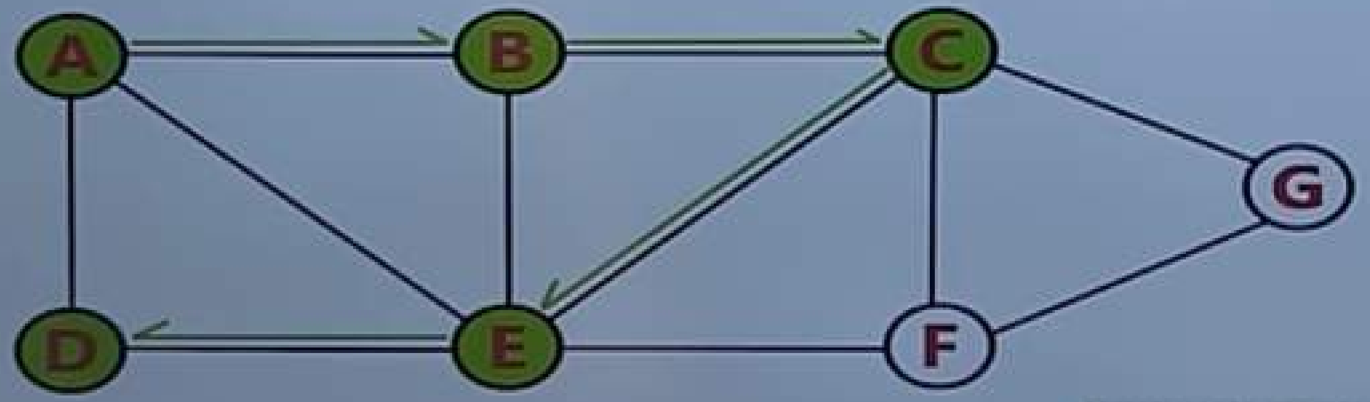
- Visit any adjacent vertext of **C** which is not visited (**E**).
- Push E on to the Stack

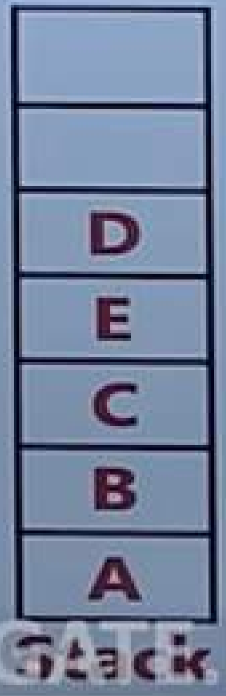




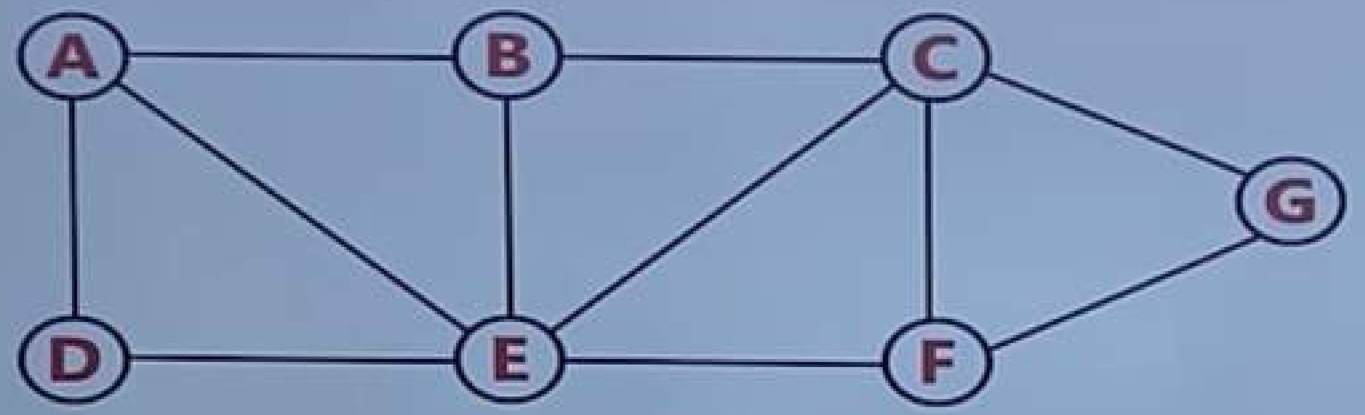
Step 5:

- Visit any adjacent vertext of **E** which is not visited (**D**).
- Push D on to the Stack



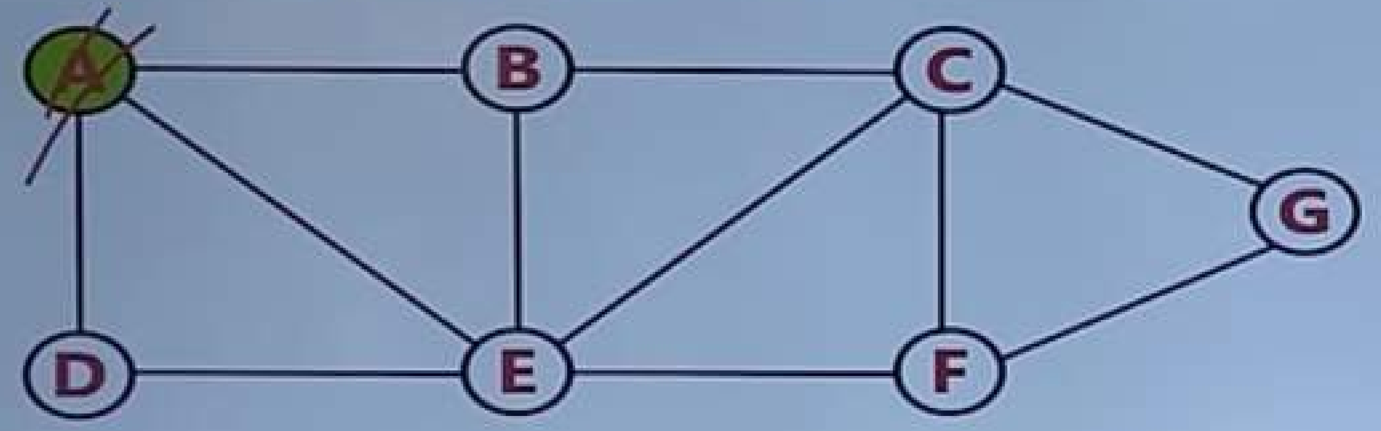


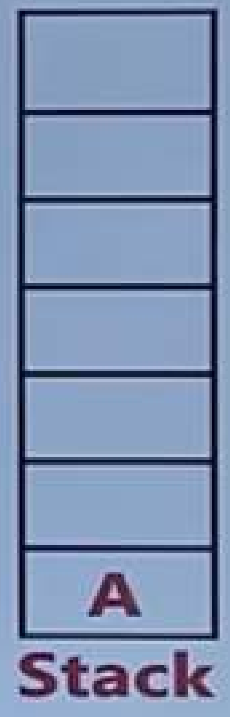
Consider the following example graph to perform DFS travers



Step 1:

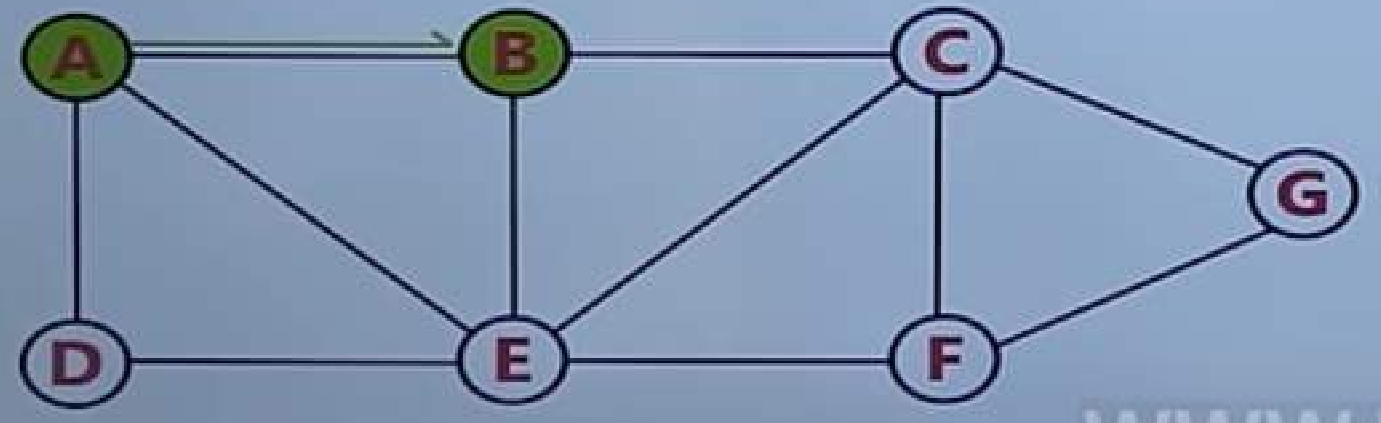
- Select the vertex **A** as starting point (visit **A**).
- Push **A** on to the Stack.

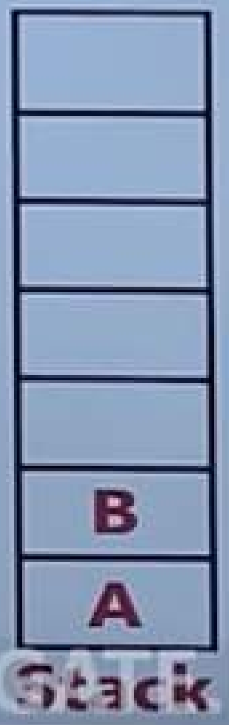




Step 2:

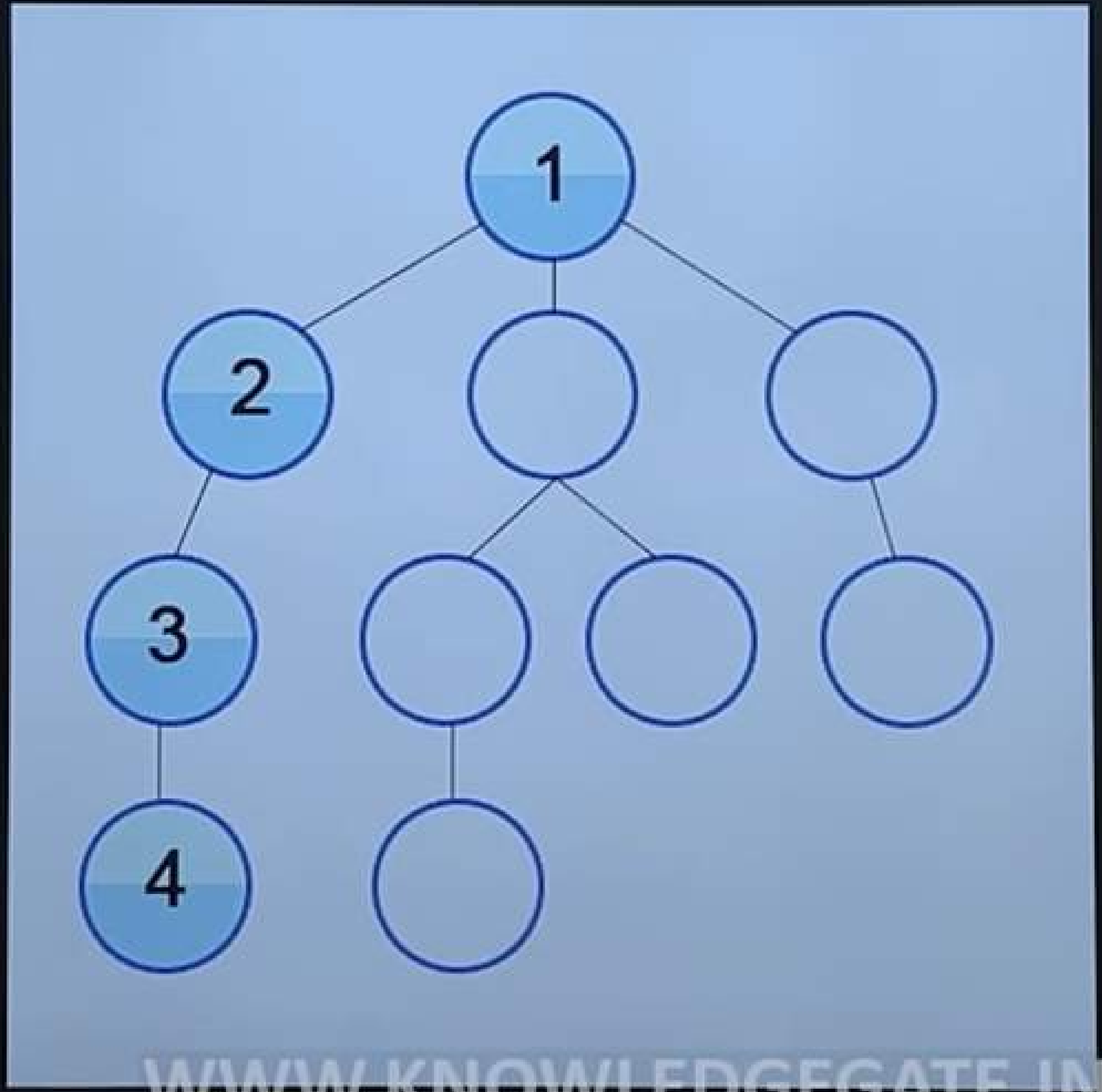
- Visit any adjacent vertex of **A** which is not visited (**B**).
- Push newly visited vertex B on to the Stack.





Depth-first search

- Depth-First Search is an uninformed search strategy that explores as far as possible along each branch before backtracking. It traverses the depth of a search tree or graph before exploring the neighboring nodes.



- **Completeness**:
  - Breadth-First Search is complete if the search space is finite or if there is a solution within a finite depth.
- **Optimality**:
  - Breadth-First Search is optimal in terms of finding the shortest path in an unweighted or uniformly weighted graph.
- **Time Complexity**:
  - The time complexity of Breadth-First Search is $O(b^d)$, where b is the branching factor and d is the depth of the shallowest goal node.
- **Space Complexity**:
  - The space complexity of Breadth-First Search is $O(b^d)$, as it stores all the nodes at each level in the search tree in memory.

- **Advantages**:
  - Guarantees finding the shortest path if one exists in an unweighted graph.
  - Closer nodes are discovered before moving to farther ones.

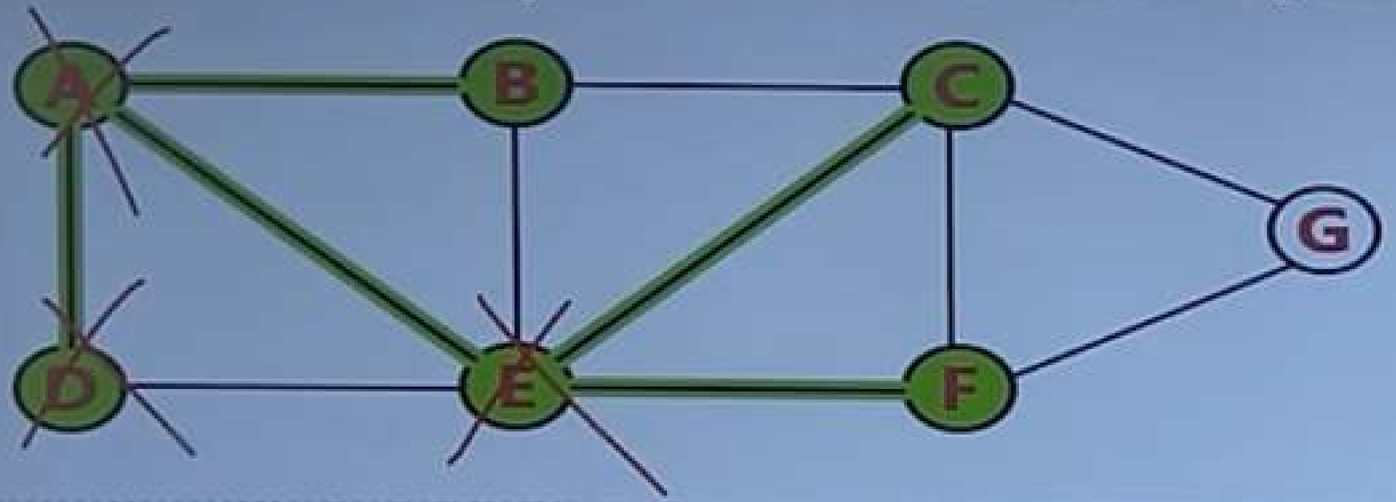
- **Disadvantages**:
  - Inefficient in terms of time and space for large search spaces.
  - Not suitable for scenarios where the cost or weight of edges is not uniform.

Algorithm:

1. Start with the initial state as the root node.
2. Enqueue the root node into a queue.
3. While the queue is not empty, do the following:
    1. Dequeue a node from the front of the queue.
    2. Expand the node and enqueue its unvisited neighboring nodes.
    3. Mark the dequeued node as visited.
4. Repeat steps 3 until the queue is empty or a goal state is found.

Step 4:

- Visit all adjacent vertices of **E** which are not visited (**C**, **F**).
- Insert newly visited vertices into the Queue and delete E from the Queue.

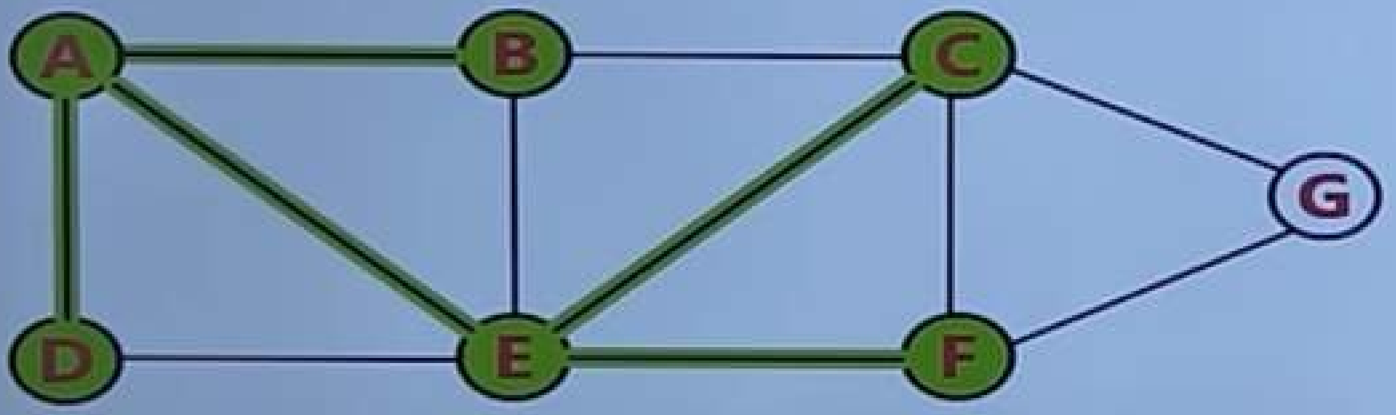


Queue

			~~B~~	C	F	

Step 5:

- Visit all adjacent vertices of **B** which are not visited (**there is no vertex**).
- Delete **B** from the Queue.

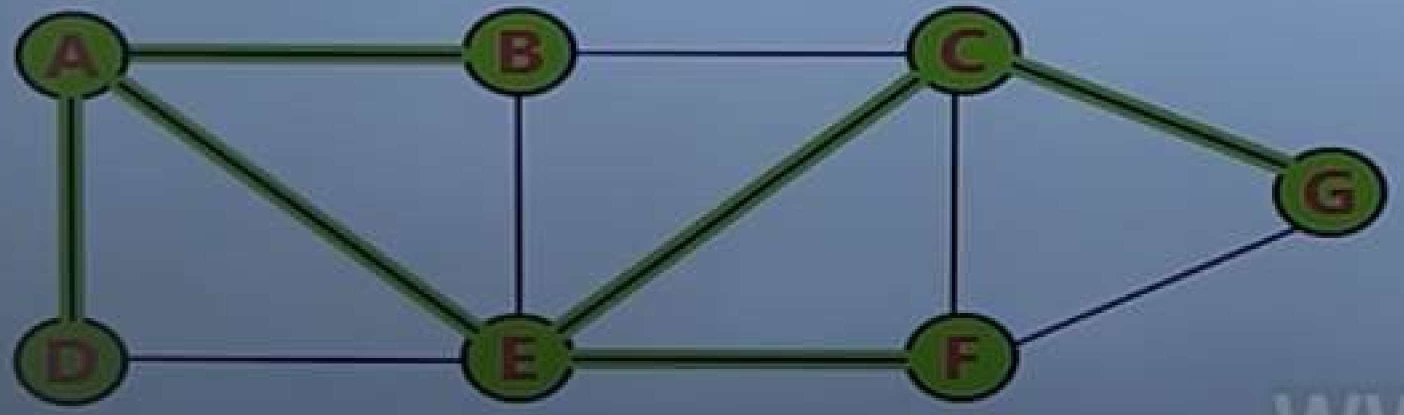


Queue

				~~C~~	F	

Step 6:

- Visit all adjacent vertices of **C** which are not visited (**G**).
- Insert newly visited vertex into the Queue and delete **C** from the Queue.

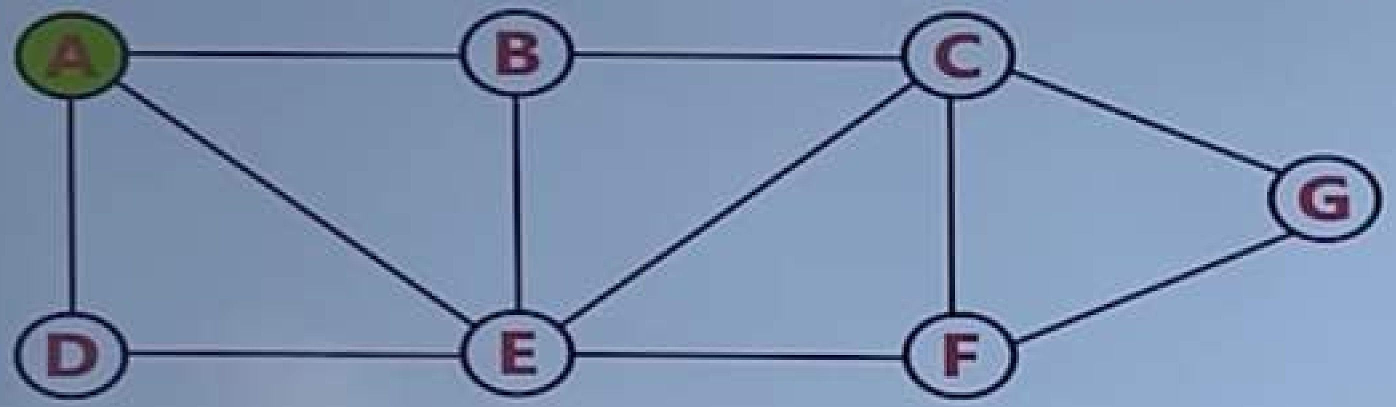


Queue

					F	G

Step 1:

- Select the vertex **A** as starting point (visit **A**).
- Insert **A** into the Queue.

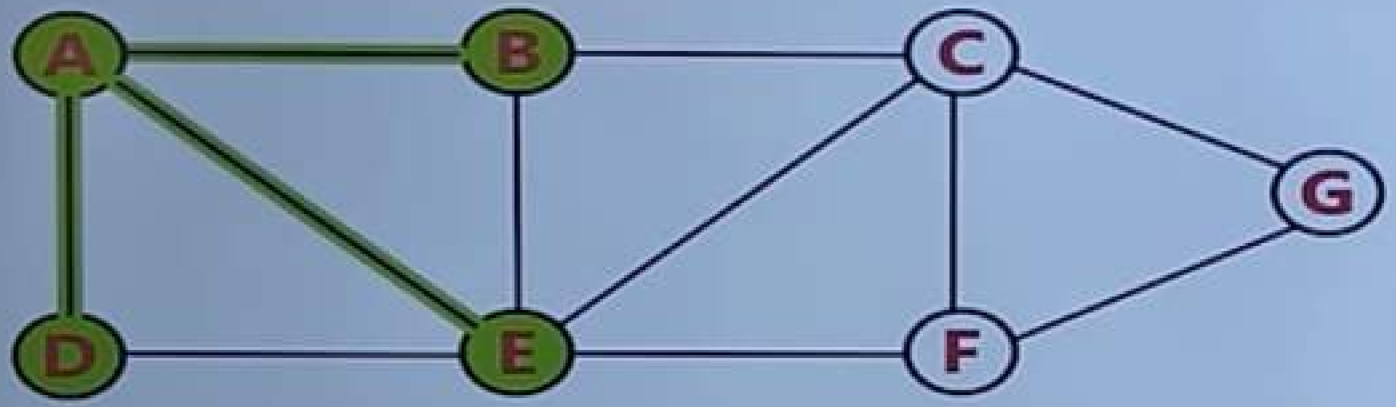


Queue

A						

Step 2:

- Visit all adjacent vertices of **A** which are not visited (**D**, **E**, **B**).
- Insert newly visited vertices into the Queue and delete A from the Queue..

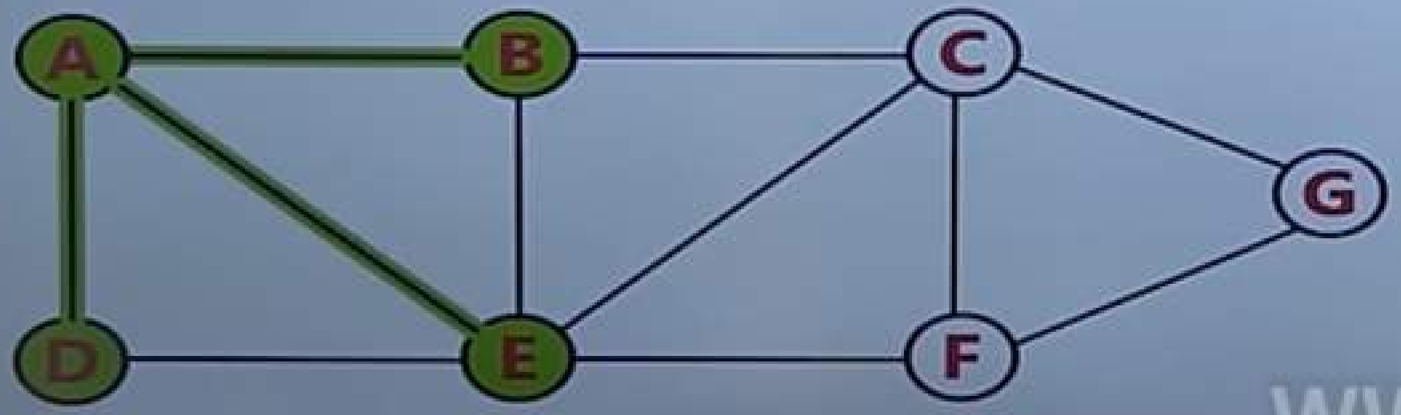


Queue

	D	E	B			

Step 3:

- Visit all adjacent vertices of **D** which are not visited (there is no vertex).
- Delete D from the Queue.

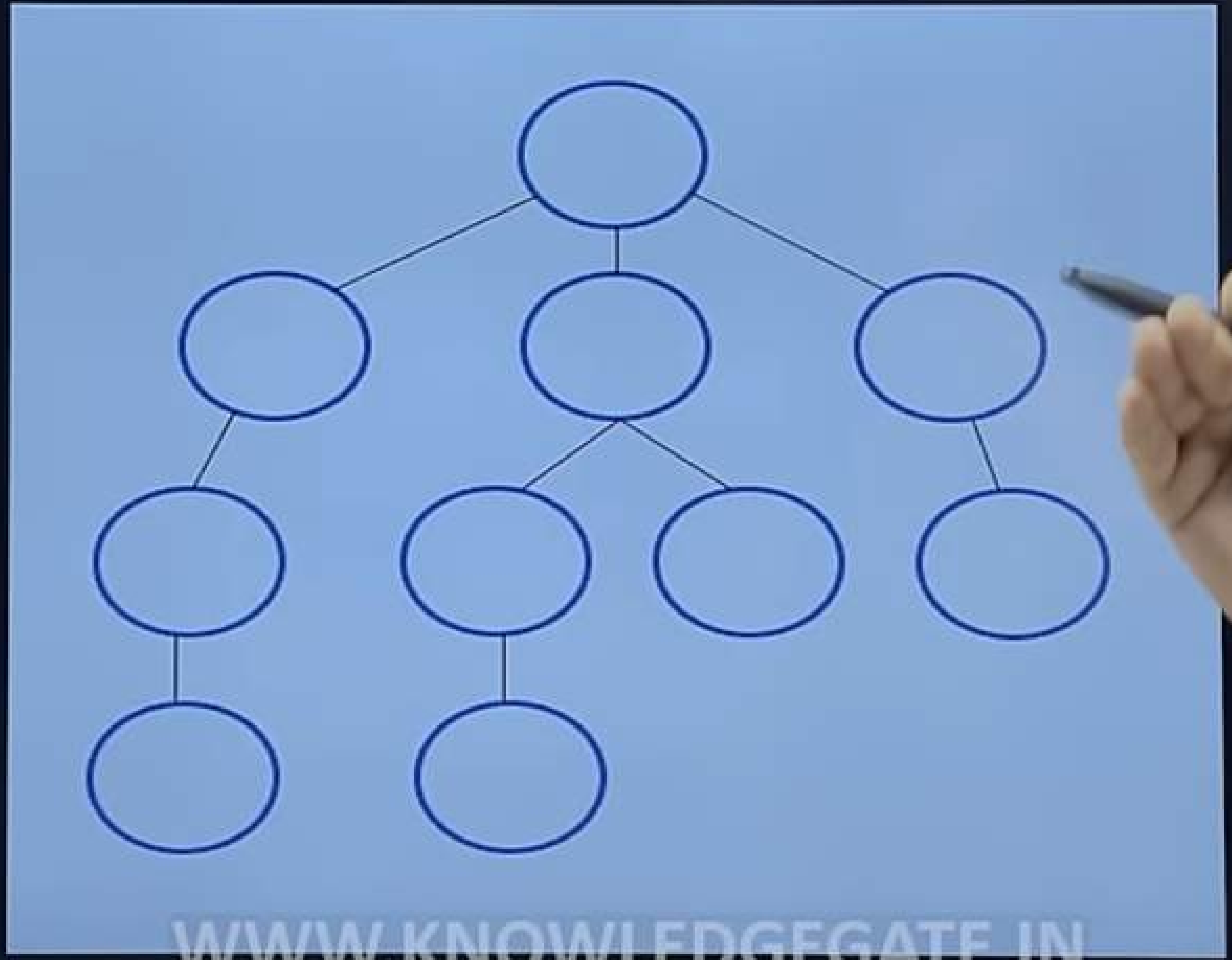


Queue

		E	B			

BFS

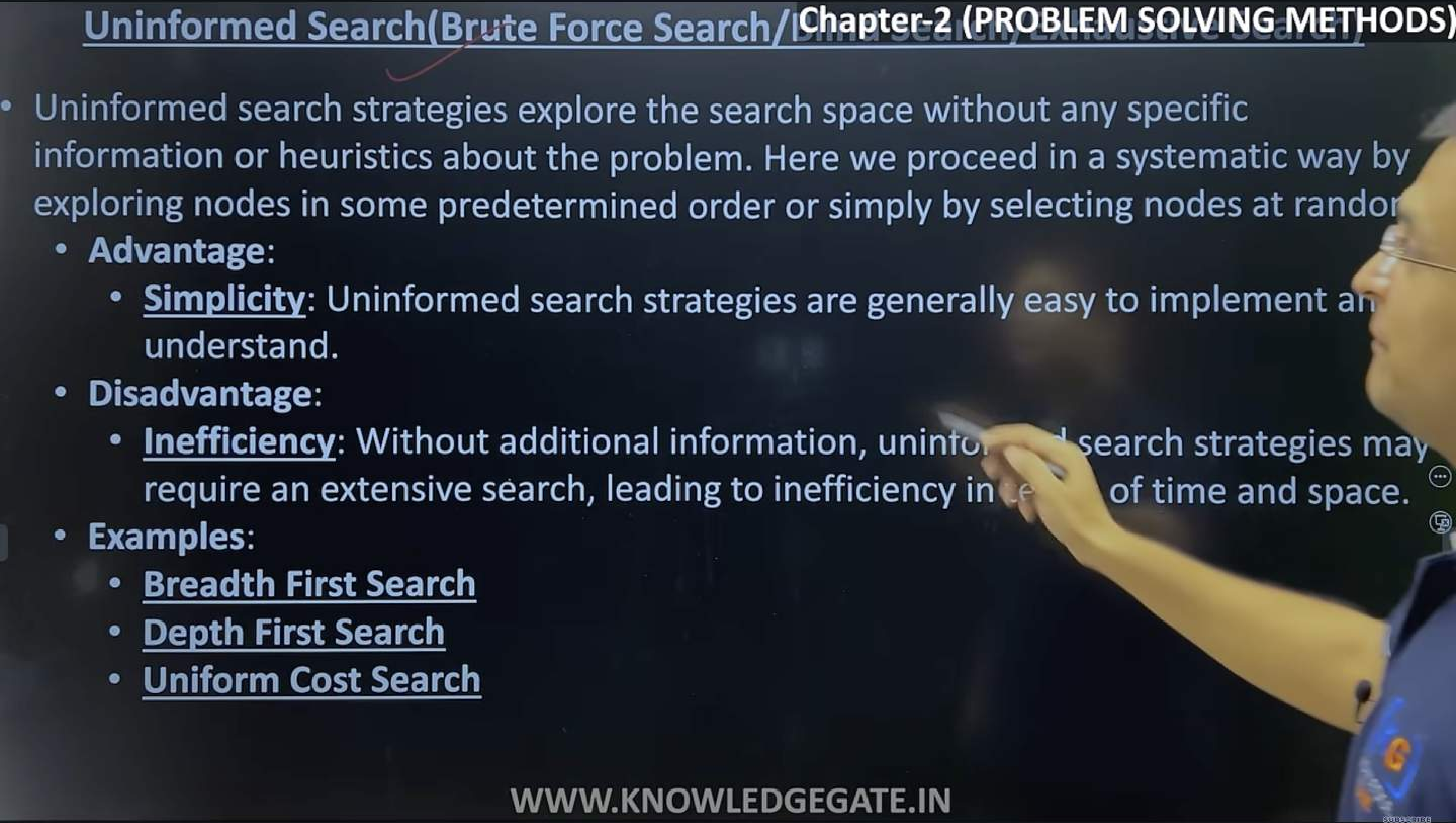
- Breadth-First Search is an uninformed search strategy that explores all neighboring nodes at the current level before moving to the next level starting from the root.
- This is achieved very simply by using a FIFO queue. New nodes (which are always than their parents) go to the back of the queue, and old nodes, which are shallow new nodes, get expanded first.



Aspect	Uninformed Search	Informed Search
Knowledge	Searches without any prior knowledge.	Uses knowledge, such as heuristics, to guide the search.
Time Efficiency	Generally more time-consuming.	Finds solutions quicker by prioritizing certain paths.
Complexity	Higher complexity due to lack of information, affecting time and space complexity.	Reduced complexity and typically more efficient in both time and space due to informed decisions.
Example Techniques	Depth-First Search (DFS), Breadth-First Search (BFS).	A* Search, Heuristic Depth-First Search, Best-First Search.
Solution Approach	Explores paths blindly.	Explores paths strategically towards the goal.

Informed Search(Heuristic Search)

- Informed search strategies utilize domain-specific information or heuristics to guide the search towards more promising paths. Here we have knowledge such as how far we are from the goal, path cost, how to reach to goal node etc. This knowledge helps agents to explore less to the search space and find more efficiently the goal node.
- **Advantage**:
  - **Efficiency**: By leveraging domain knowledge, informed search strategies can make informed decisions and focus the search on more relevant areas, leading to faster convergence to a solution.
- **Disadvantage**:
  - **Heuristic Accuracy**: The effectiveness of informed search strategies heavily relies on the quality and accuracy of the chosen heuristic function. An inaccurate or misleading heuristic can lead to suboptimal or incorrect solutions.
- **Example**:
  - **Hill Climbing**
  - **Best First Search**
  - **A* Algorithm**

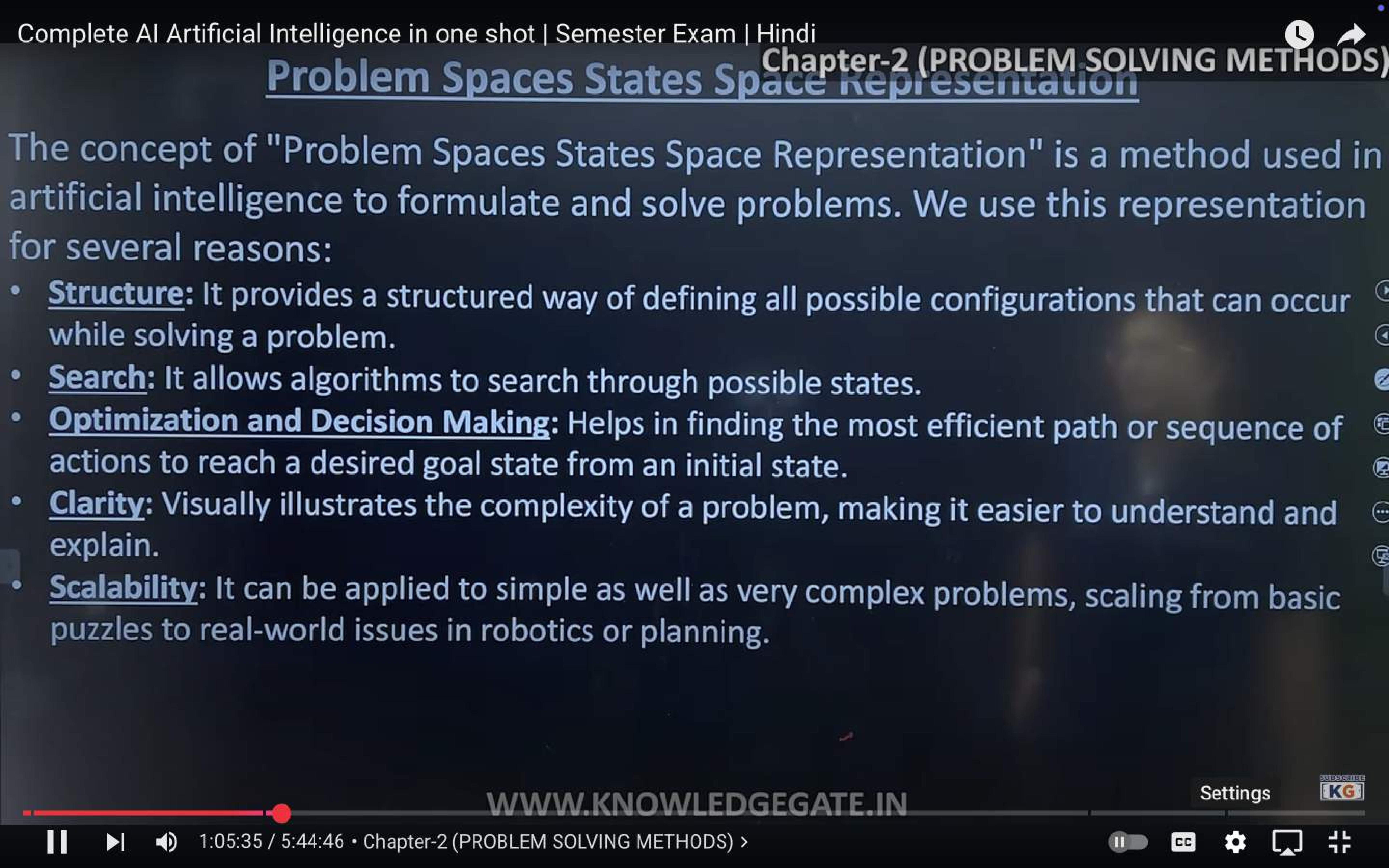


Uninformed Search(Brute Force Search/

- Uninformed search strategies explore the search space without any specific information or heuristics about the problem. Here we proceed in a systematic way by exploring nodes in some predetermined order or simply by selecting nodes at randor
  - **Advantage**:
    - **Simplicity**: Uninformed search strategies are generally easy to implement ar understand.
  - **Disadvantage**:
    - **Inefficiency**: Without additional information, uninfor search strategies may require an extensive search, leading to inefficiency in of time and space.
  - **Examples**:
    - **Breadth First Search**
    - **Depth First Search**
    - **Uniform Cost Search**

Search Strategies

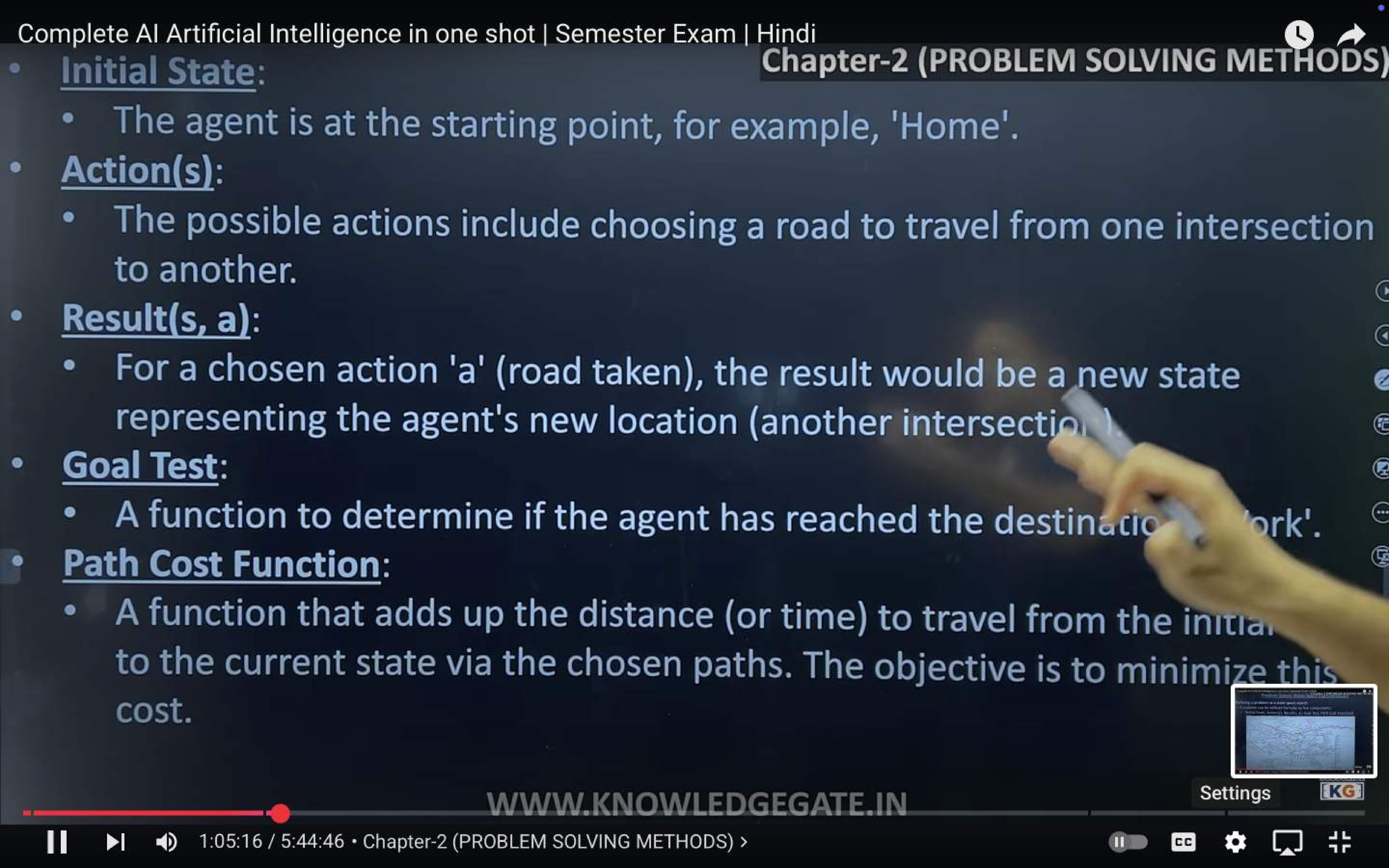
- Search strategies refer to systematic methods and approaches used to explore and find relevant information or solutions within a given search space or dataset. Parameters for Evaluating Search Technique Performance:
  - **Completeness**: Determines if the search technique guarantees finding a solution if one exists within the search space.
  - **Time and Space Complexity**: Evaluates the efficiency of the search technique in terms of the time and space required to find a solution.
  - **Optimality**: Determines whether the search technique can find the best or optimal solution among all possible solutions.


Complete AI Artificial Intelligence in one shot | Semester Exam | Hindi
Chapter-2 (PROBLEM SOLVING METHODS)
Settings
1:05:35 / 5:44:46 • Chapter-2 (PROBLEM SOLVING METHODS) ›

Problem Spaces States Space Representation

The concept of "Problem Spaces States Space Representation" is a method used in artificial intelligence to formulate and solve problems. We use this representation for several reasons:

- **Structure**: It provides a structured way of defining all possible configurations that can occur while solving a problem.
- **Search**: It allows algorithms to search through possible states.
- **Optimization and Decision Making**: Helps in finding the most efficient path or sequence of actions to reach a desired goal state from an initial state.
- **Clarity**: Visually illustrates the complexity of a problem, making it easier to understand and explain.
- **Scalability**: It can be applied to simple as well as very complex problems, scaling from basic puzzles to real-world issues in robotics or planning.



Complete AI Artificial Intelligence in one shot | Semester Exam | Hindi

- **Initial State**:
  - The agent is at the starting point, for example, 'Home'.
- **Action(s)**:
  - The possible actions include choosing a road to travel from one intersection to another.
- **Result(s, a)**:
  - For a chosen action 'a' (road taken), the result would be a new state representing the agent's new location (another intersectio[illegible]).
- **Goal Test**:
  - A function to determine if the agent has reached the destinatio[illegible]ork'.
- **Path Cost Function**:
  - A function that adds up the distance (or time) to travel from the initia[illegible] to the current state via the chosen paths. The objective is to minimize this cost.

Complete AI Artificial Intelligence in one shot | Semester Exam | Hindi

Chapter-2 (PROBLEM SOLVING METHODS)

Problem Spaces States Space Representation

Defining a problem as a state space search

- A problem can be defined formally by five components:
  - {Initial State, Action(s), Result(s, a), Goal Test, Path Cost Function}

Bahadurgarh City
Pandit Shree Ram Sharma
Tikri Border
Tikri Kalan
Ghevra Metro Station
Mundka Industrial Area
Rithala
Rohini(W)
Rohini(E)
Pitampura
Kohat Enclave
NSP
Rohini Sector 18,19
Haiderpur Badli Mor
Jahangirpuri
Adarsh Nagar
Azadpur
Shalimar Bagh
Majlis Park
Model Town
GTB Nagar
Vishwavidyalaya
Vidhan Sabha
Civil Lines
Kashmere Gate
Tis Hazari
Chandni Chowk
Chawri Bazar
New Delhi
Rajiv Chowk
Patel Chowk
Central Secretariat
Udyog Bhawan
Lok Kalyan Marg
Jor Bagh
INA
AIIMS
Green Park
Hauz Khas
Malviya Nagar
Saket
Qutab Minar
Chhatarpur
Sultanpur
Keshav Puram
Kanhaiya Nagar
Inderlok
Shastri Nagar
Pratap Nagar
Pulbangash
Shakur Pur
Punjabi Bagh
Ashok Park Main
Satguru Ram Singh Marg
Shadipur
Patel Nagar
Rajendra Place
Karol Bagh
Jhandewalan
RK Ashram Marg
Shivaji Stadium
Dhaula Kuan
Delhi Cantt.
Naraina Vihar
Mayapuri
Rajouri Garden
Tagore Garden
Subhash Nagar
Tilak Nagar
Janakpuri(E)
Janakpuri(W)
Uttam Nagar(E)
Uttam Nagar(W)
Nawada
Dwarka Mor
Dwarka
Dwarka Sector 14
Dwarka Sector 13
Dwarka Sector 12
Dwarka Sector 11
Dwarka Sector 10
Dwarka Sector 9
Dwarka Sector 8
Dwarka Sector 21
Najafgarh
Dabri Mor Janakpuri South
Dashrath Puri
Palam
Sadar Bazar Cantt.
Terminal-1 IGI Airport
Delhi Aerocity
Indira Gandhi International Airport
Shankar Vihar
Vasant Vihar
Munirka
R.K. Puram
IIT
South Campus
Moti Bagh
Bhikaji Cama Place
Sarojini Nagar
Madipur
Shivaji Park
Punjabi Bagh West
ESI Hospital
Kirti Nagar
Moti Nagar
Ramesh Nagar
Shiv Vihar
Johri Enclave
Gokulpuri
Maujpur
Jaffrabad
Seelampur
Welcome
Shahdara
Mansarovar Park
Jhilmil
Dilshad Garden
Shahid Nagar
Raj Bagh
Rajendra Nagar
Shyam Park
Mohan Nagar
Arthala
New Bus Adda
Shastri Park
East Azad Nagar
Krishna Nagar
Karkarduma Court
Karkarduma
Anand Vihar ISBT
Kaushambi
Vaishali
Red Fort
Jama Masjid
Delhi Gate
ITO
Mandi House
Barakhamba Rd.
Pragati Maidan
Indraprastha
Preet Vihar
Nirman Vihar
Laxmi Nagar
Yamuna Bank
Akshardham
Mayur Vihar-I
Mayur Vihar-I Ext.
New Ashok Nagar
Noida Sec. 15
Noida Sec. 16
Noida Sec. 18
Botanical Garden
Golf Course
Noida City Centre
IP Extn.
Vinod Ngr.
Vinod Ngr. East
Trilokpuri (Sanjay Lake)
Mayur Vihar Pkt. -1
Noida Electronic City
Noida Sec-62
Noida Sec-59
Noida Sec-61
Noida Sec-52
Noida Sec-34
Sec-51
Sec-50
Sec-76
Sec-101
Sec-81
NSEZ
Sec-83
Sec-137
Sec-142
Sec-143
Sec-144
Sec-145
Sec-146
Sec-147
Sec-148
Pari Chowk
Alpha 1
Delta 1
GNIDA
Janpath
Khan Market
JLN Stadium
Jangpura
Nizamuddin
Ashram
Vinoba Puri
Lajpat Nagar
Moolchand
Kailash Colony
South Extn.
Nehru Place
Kalkaji Mandir
Govindpuri
Harkesh Nagar Okhla
Jasola
Sarita Vihar
Mohan Estate
Tughlakabad
Badarpur Border
Sarai
NHPC Chowk
Okhla NSIC
Sukhdev Vihar
Jamia Millia Islamia
Okhla Vihar
Jasola Vihar Shaheen Bagh
Okhla Bird Sanctuary
Kalindi Kunj
Panchsheel Park
Chirag Delhi
Greater Kailash
Nehru Enclave
IndusInd Bank Cyber City
Micromax Moulsari Avenue
Vodafone Belvedere Towers Station
Phase-3
Phase-2
Phase-1
IFFCO Chowk
Huda City Centre
MG Road
Sikanderpur
Guru Dronacharya
Arjangarh
Ghitorni
Sector 42-43

Settings

1:04:10 / 5:44:46 • Chapter-2 (PROBLEM SOLVING METHODS) ›