

GATE



Artificial Intelligence

Un-Informed search

Lecture No.- 01

By- Aditya sir



Topics to be Covered



Topic

-Intro

Topic

Syllabus: AI

Topic

Searching Algos





About Aditya Jain sir



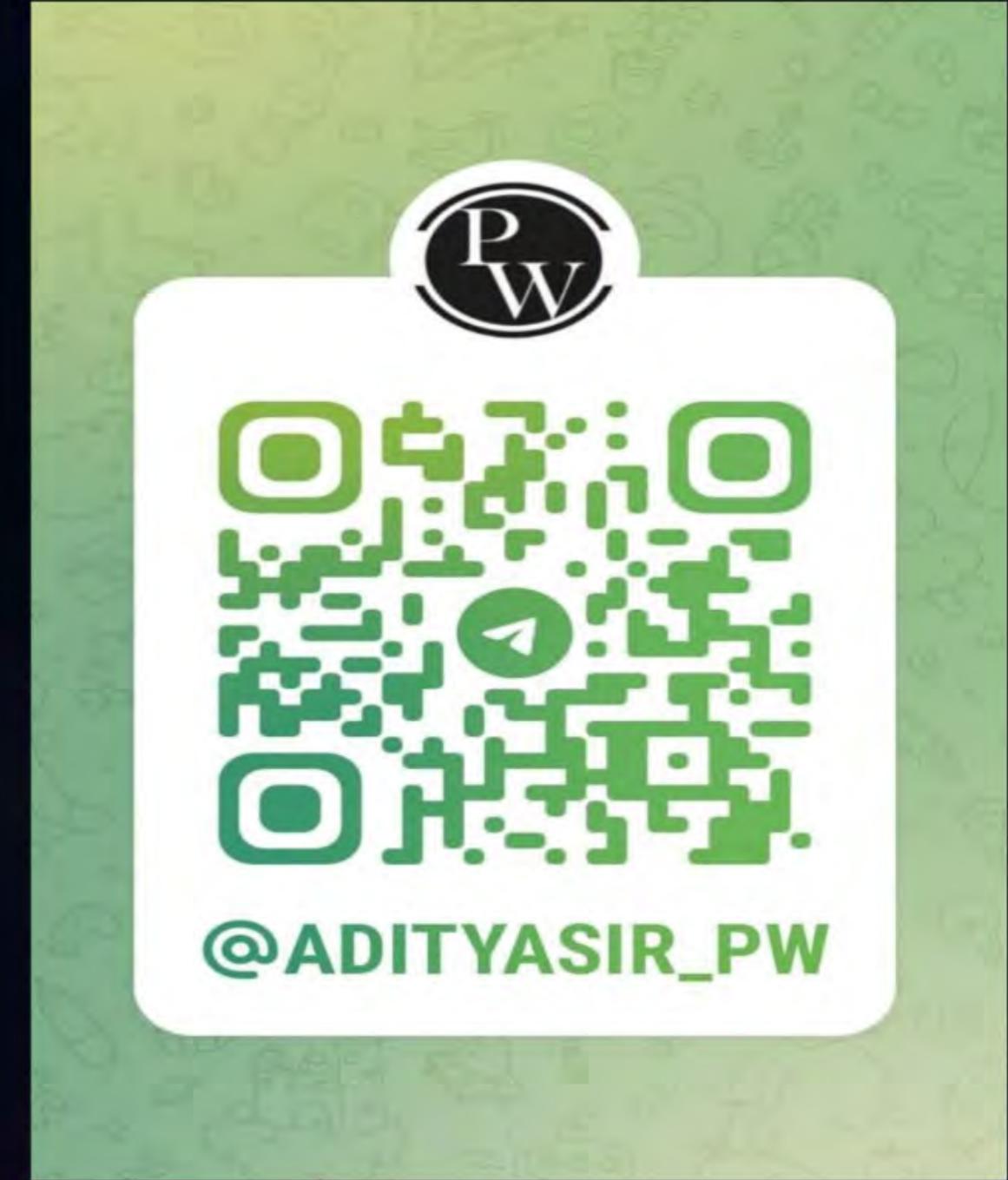
77.33/100

-0.66

1. Appeared for GATE during BTech and secured AIR 60 in GATE in very first attempt - City topper
2. Represented college as the first Google DSC Ambassador.
3. The only student from the batch to secure an internship at Amazon. (9+ CGPA)
4. Had offer from IIT Bombay and IISc Bangalore to join the Masters program
5. Joined IIT Bombay for my 2 year Masters program, specialization in Data Science
6. Published multiple research papers in well known conferences along with the team
7. Received the prestigious excellence in Research award from IIT Bombay for my Masters thesis in ML
8. Completed my Masters with an overall GPA of 9.36/10 ✨
9. Joined Dream11 as a Data Scientist
10. Have mentored 15,000+ students & working professionals in field of Data Science and Analytics (AI)
11. Have been mentoring & teaching GATE aspirants to secure a great rank in limited time
12. Have got around 27.5K followers on LinkedIn where I share my insights and guide students and professionals.



Telegram



Telegram Link for Aditya Jain sir:
https://t.me/AdityaSir_PW

Tomorrow onwards : 11:30 AM - 1:30 PM



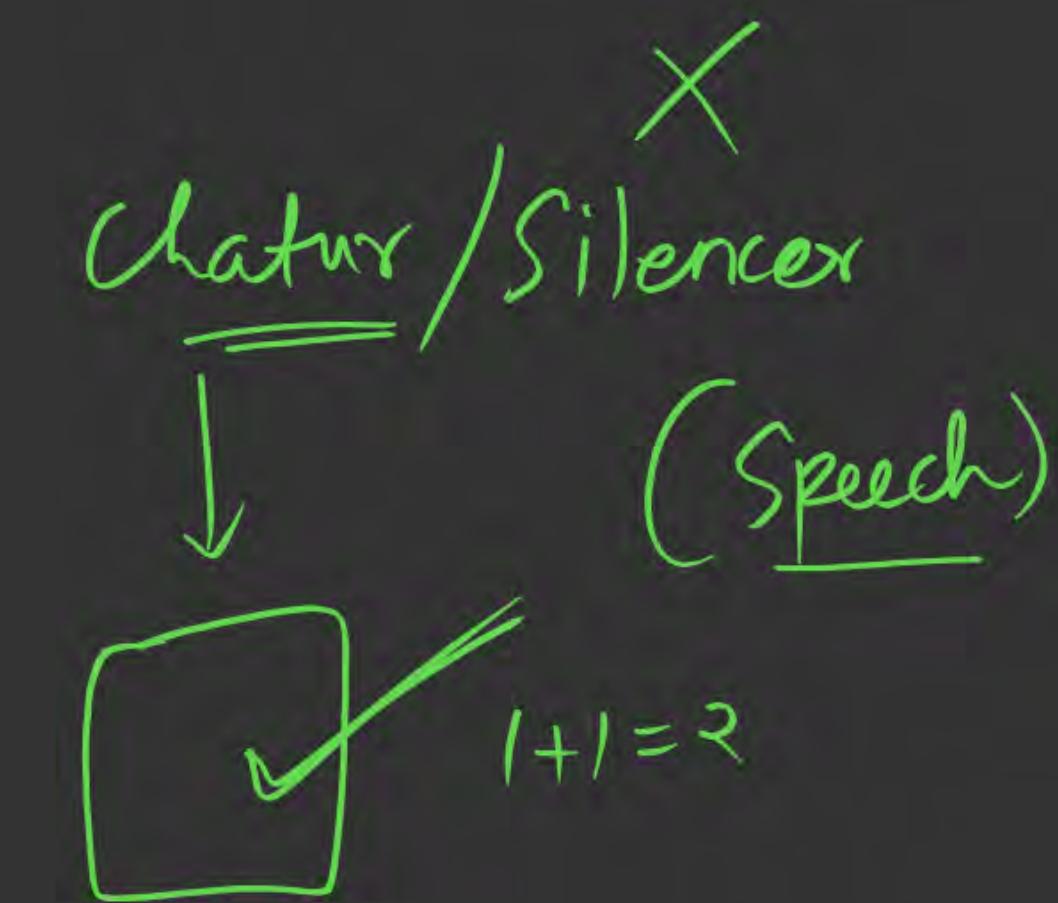
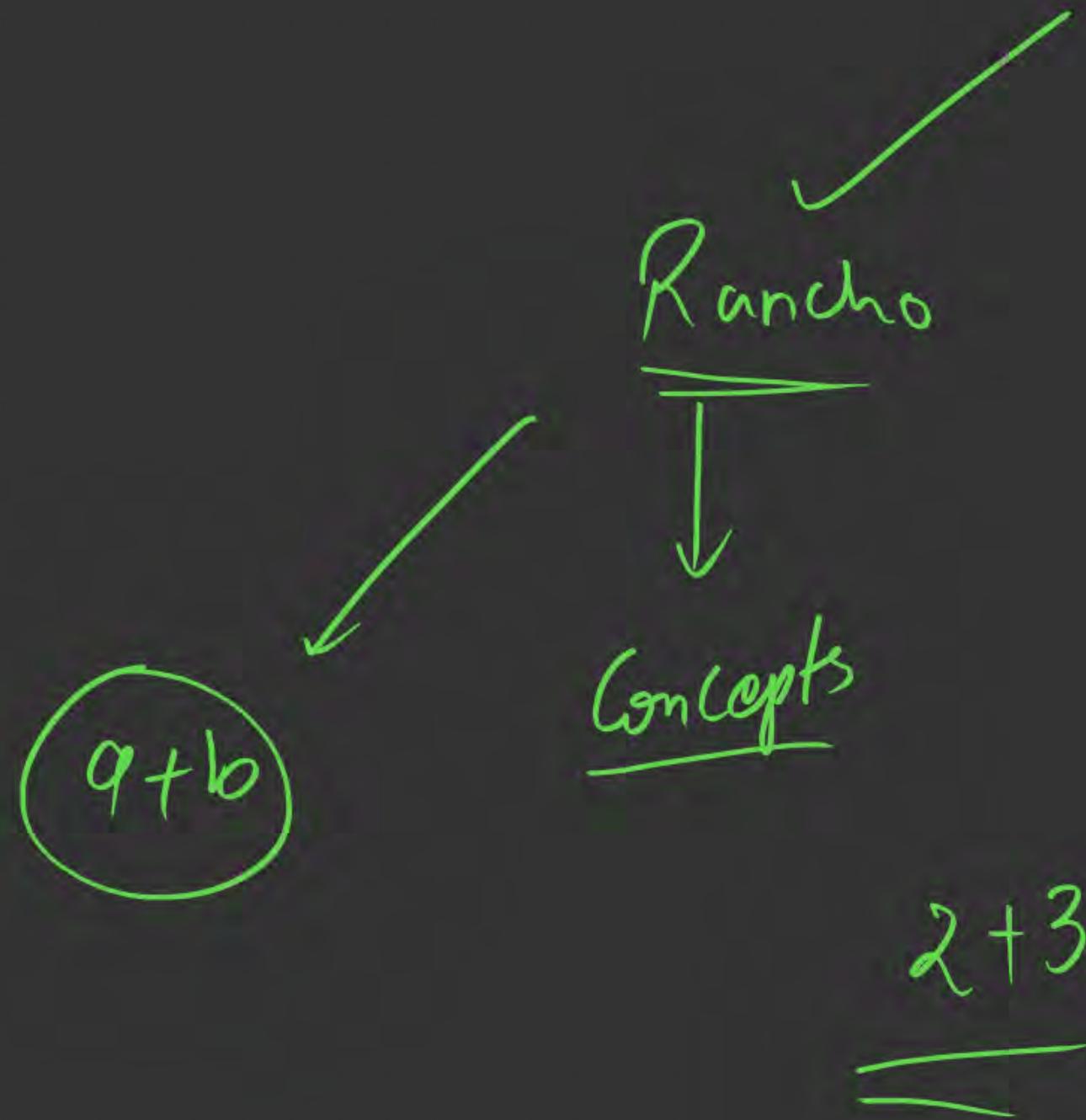
Topic : Uninformed Search

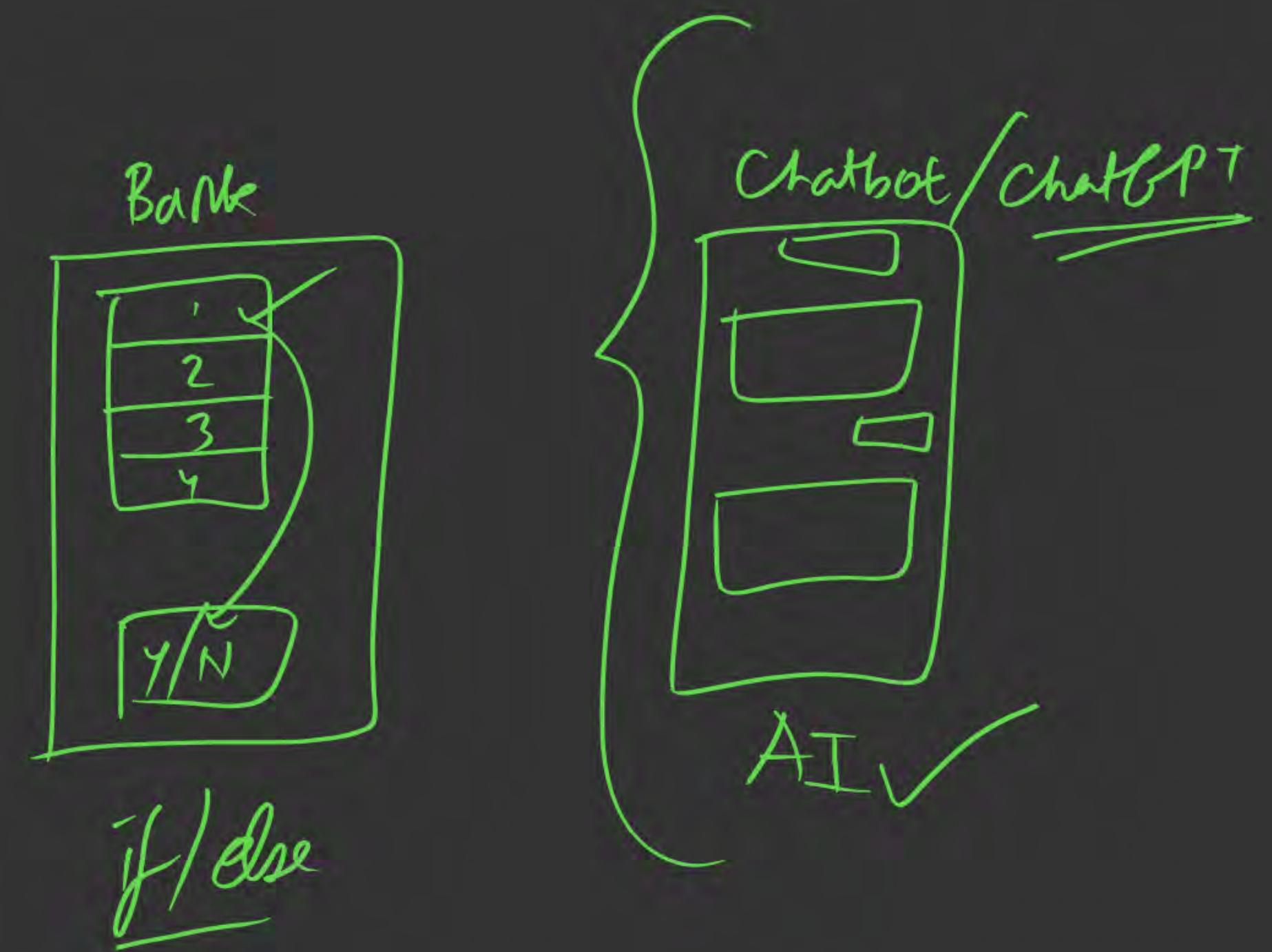
Syllabus



Search: Informed, Uniformed, adversarial, logic, propositional, predicate reasoning under uncertainty topic -Conditional independence representation, exact inference through variable elimination, and approximate inference through sampling.

What is AI?
↓
Artificial Intelligence







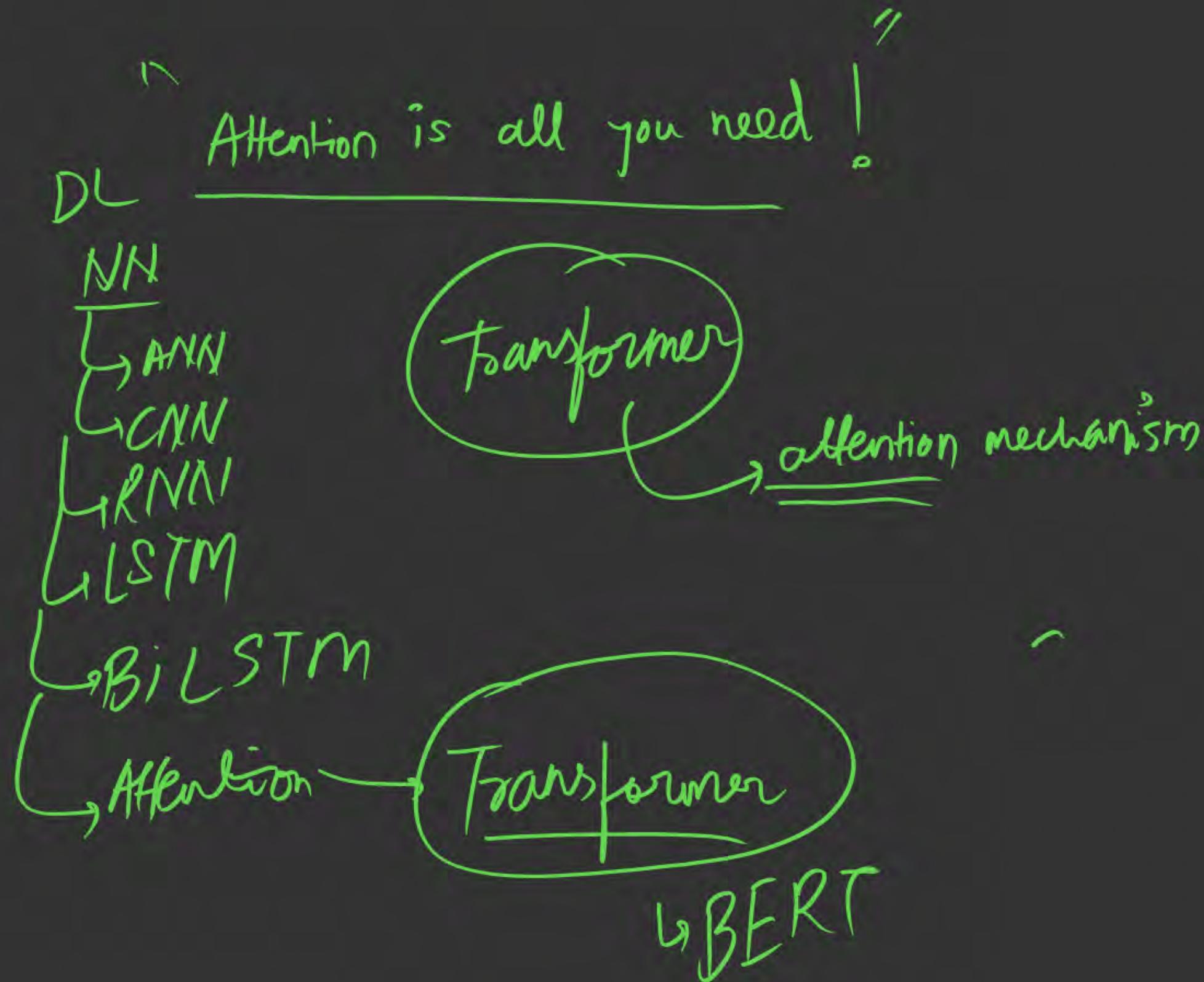
Topic : Uninformed Search

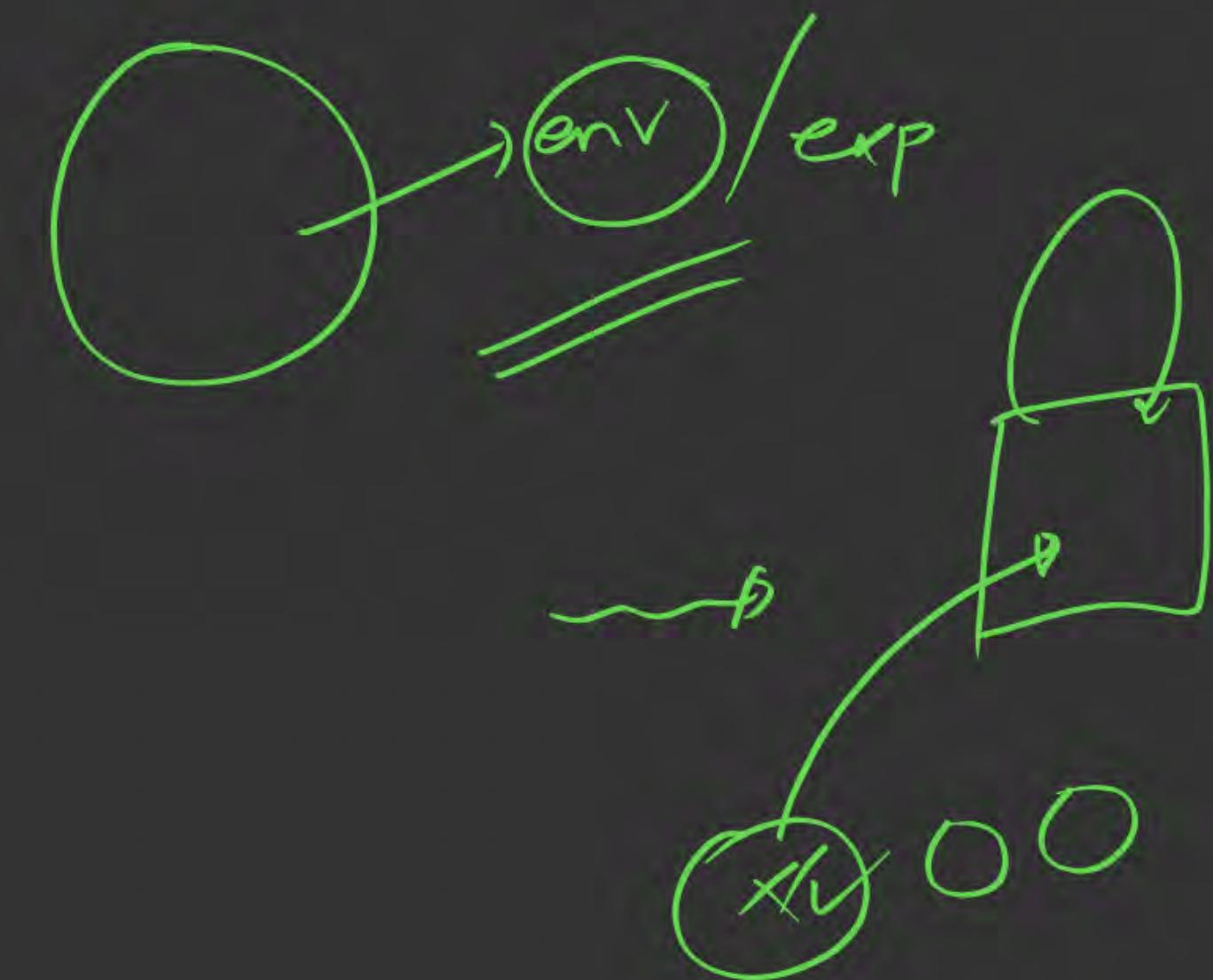


What is Artificial Intelligence ?

- Artificial Intelligence (AI) is a branch of computer science that focuses on creating machines and software capable of performing tasks that typically require human intelligence
- AI can cause a machine to work as human. 
- Artificial (man made) and Intelligence (Power of thinking)
- These tasks include learning, reasoning, problem-solving, perception, language understanding, and interaction.





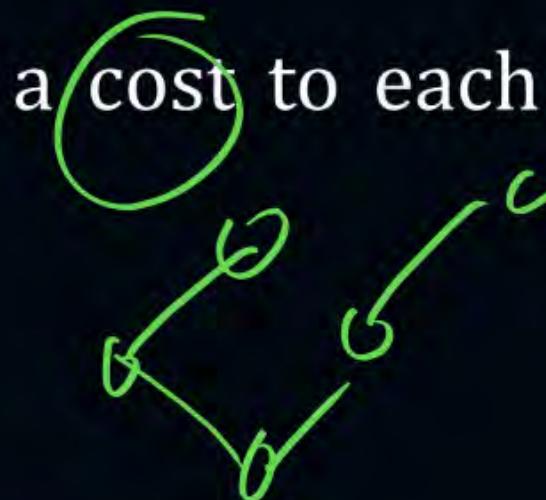




Topic : Uninformed Search

What is a state and state space...

- **Initial State:** The starting point of the search.
- **Goal State(s):** The desired state(s) that signify a solution.
- **Actions/Operators:** The set of possible actions that can be performed to move from one state to another.
- **Transition Model:** Describes the result of applying an action to a state, leading to a new state.
- **Path Cost:** A function that assigns a cost to each path, helping to find the most efficient solution.

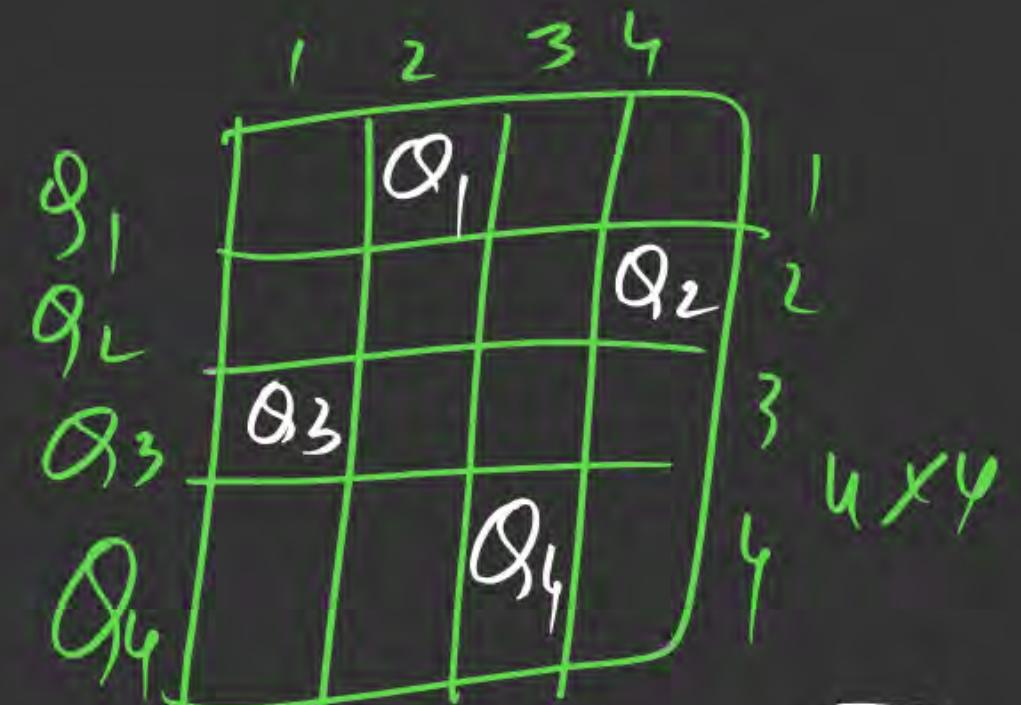


N-Queens Prob.

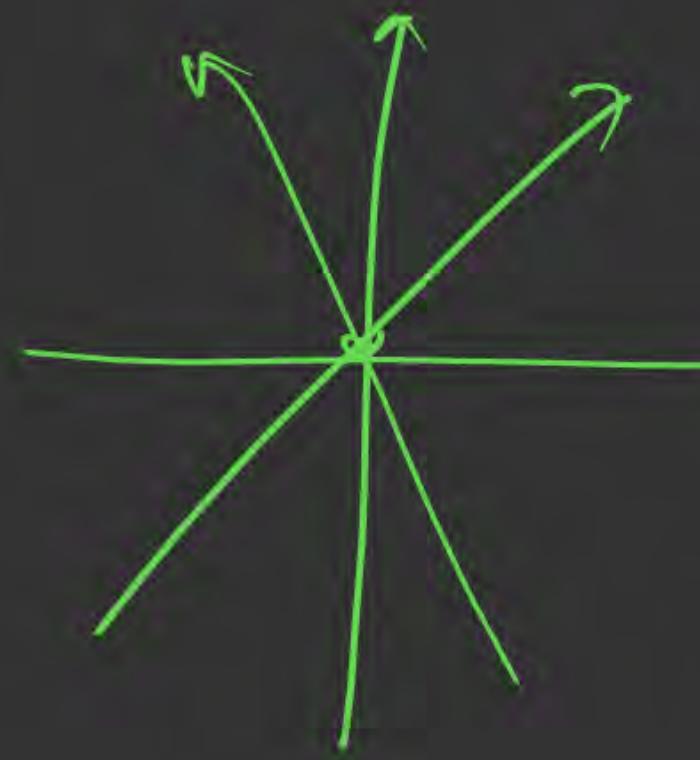
State space

$\begin{bmatrix} - & 1 & - & 1 & - \end{bmatrix}$

q!



$\begin{bmatrix} 2, 4, 1, 3 \end{bmatrix}$





Topic : Uninformed Search

What is a state and state space...

- **Example Eight tile puzzle:** Given a 3×3 board with 8 tiles (every tile has one number from 1 to 8) and one empty space. The objective is to place the numbers on tiles to match the final configuration using the empty space. We can slide four adjacent (left, right, above, and below) tiles into the empty space

actions

Initial Configuration

1	2	3
5	6	
4	8	4

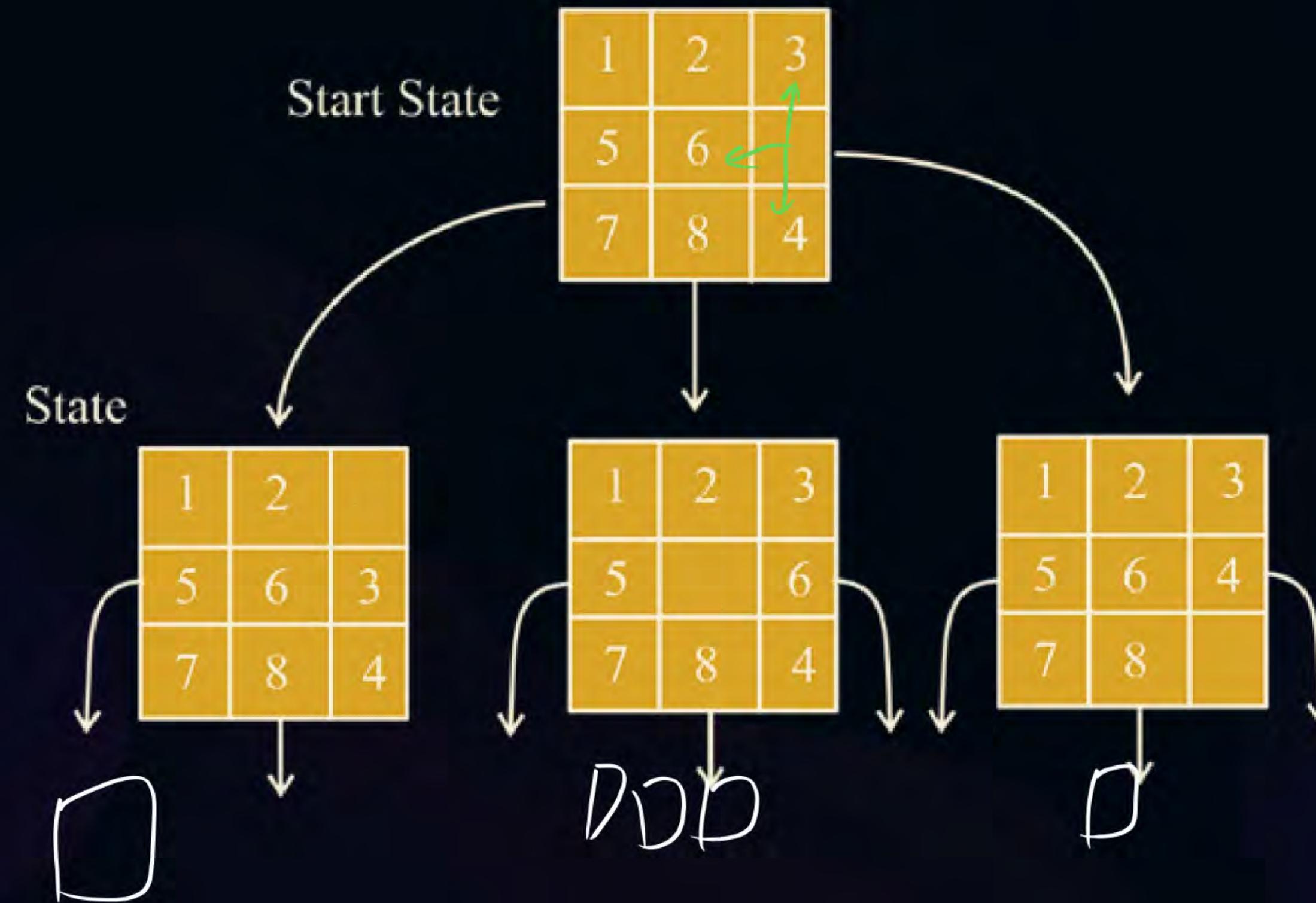
Final Configuration

1	2	3
5	8	6
	7	4





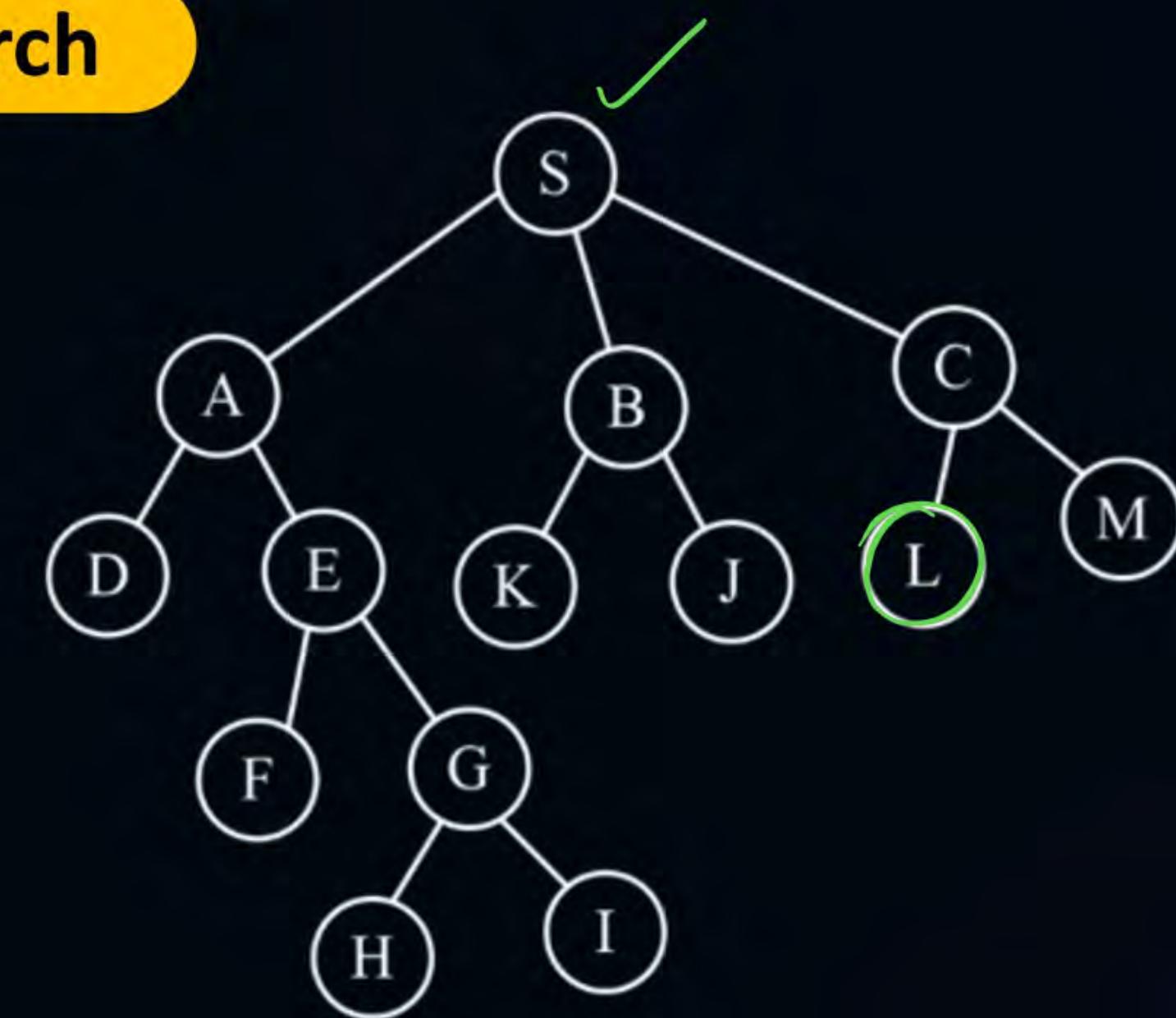
Topic : Uninformed Search





Topic : Uninformed Search

- These are states ✓
- State space ✓
- Starting state ✓
- Goal state ✓
- Search ??



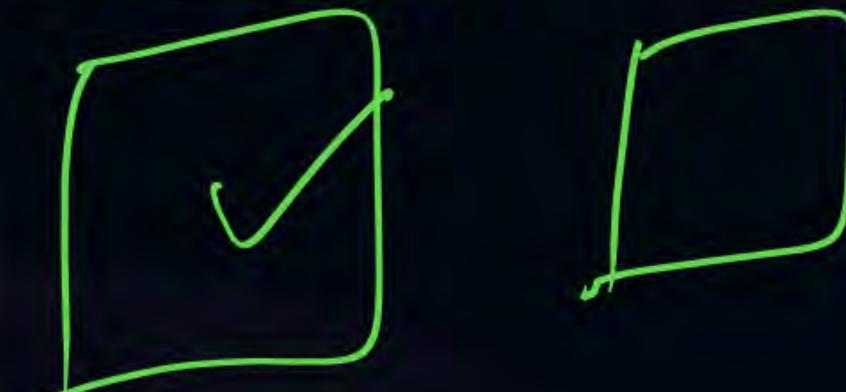


Topic : Uninformed Search



What is a state and state space...

- State: A specific configuration of the system at a particular point in time
- **State Space:** The entire set of possible states that the system can be in, including the initial state, goal state(s), and all intermediate states
- **State Space Search:** We are moving from start state and search for goal state in state space, Used in Problem Solving. It is a process used in A.I in which Successive Configurations or States of an instance are considered with intention of a GOAL State with desired property
- we are moving from start state and search for goal state in state space

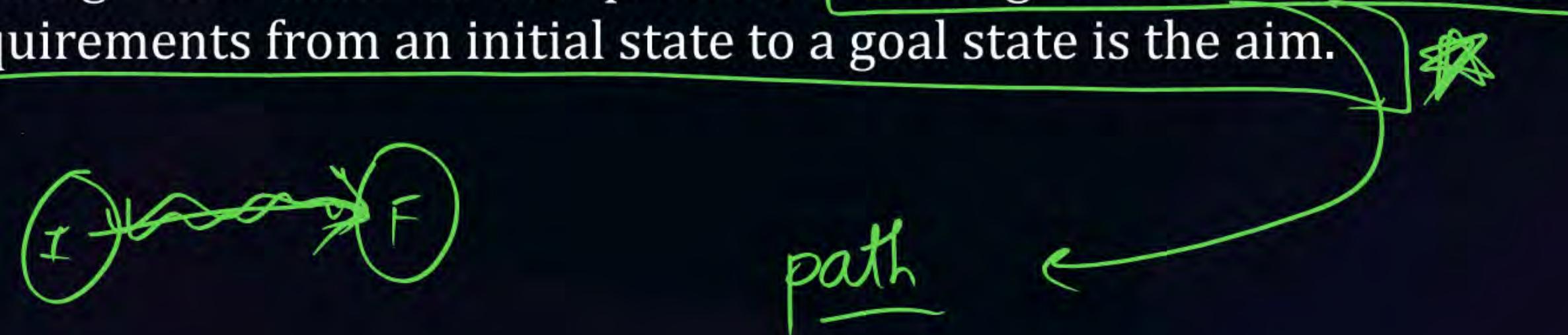


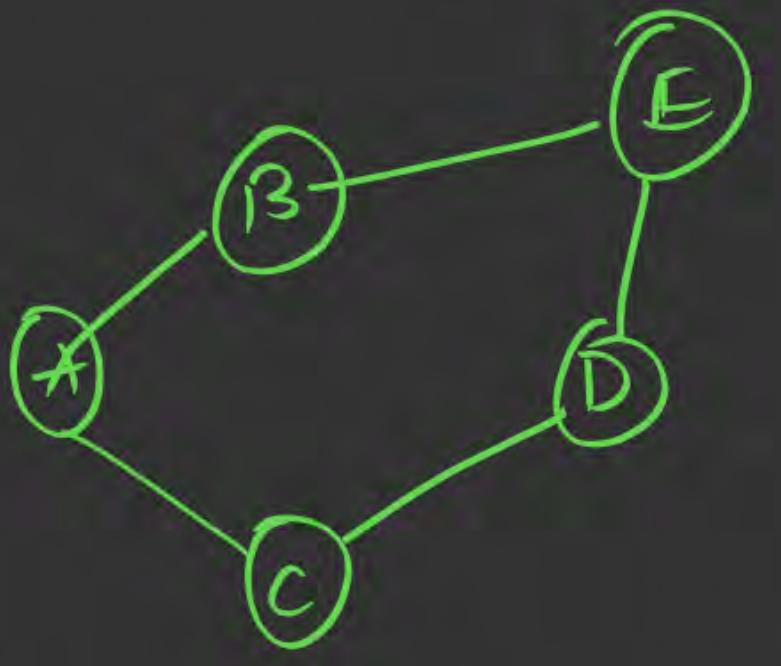


Topic : Uninformed Search

What is a state and state space...

- State space search is a fundamental concept in artificial intelligence (AI) that involves exploring possible states of a system to find a solution to a problem.
- An essential method in artificial intelligence is state space search, which looks for potential states and their transitions to solve issues.
- According to this method, the problem is modelled as a state space, with each state representing a possible configuration and transitions denoting actions or operations that change the state of the problem. Finding a route that meets predetermined requirements from an initial state to a goal state is the aim.





④ A

⑤ G
E

- 1) A - B - E
- 2) A - C - D - E

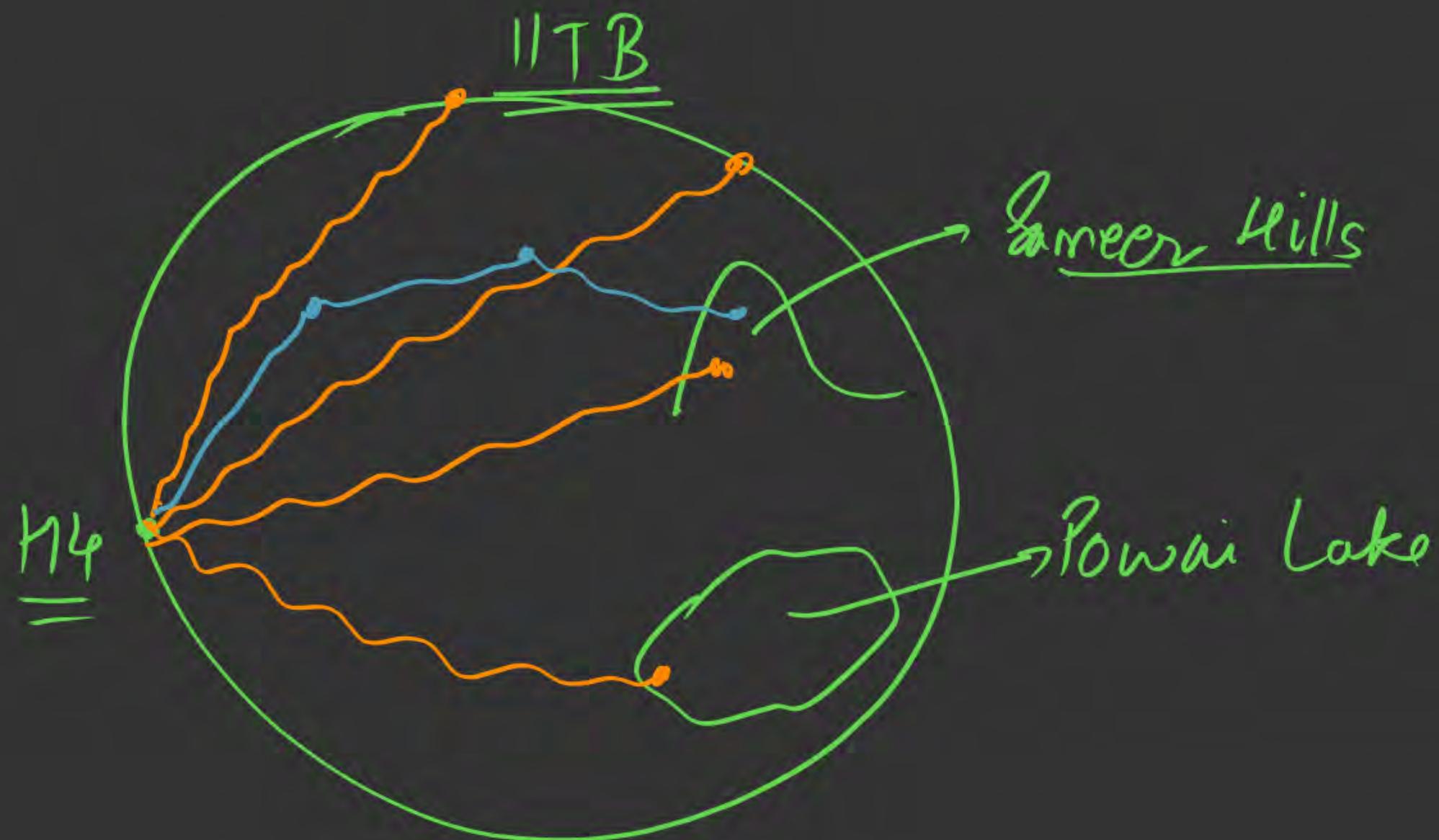


Topic : Uninformed Search

Types of search in AI

- In AI, search algorithms can be broadly categorized into three types:
- Uninformed (blind) search, ✓
- Informed (heuristic) search,
- And adversarial search.

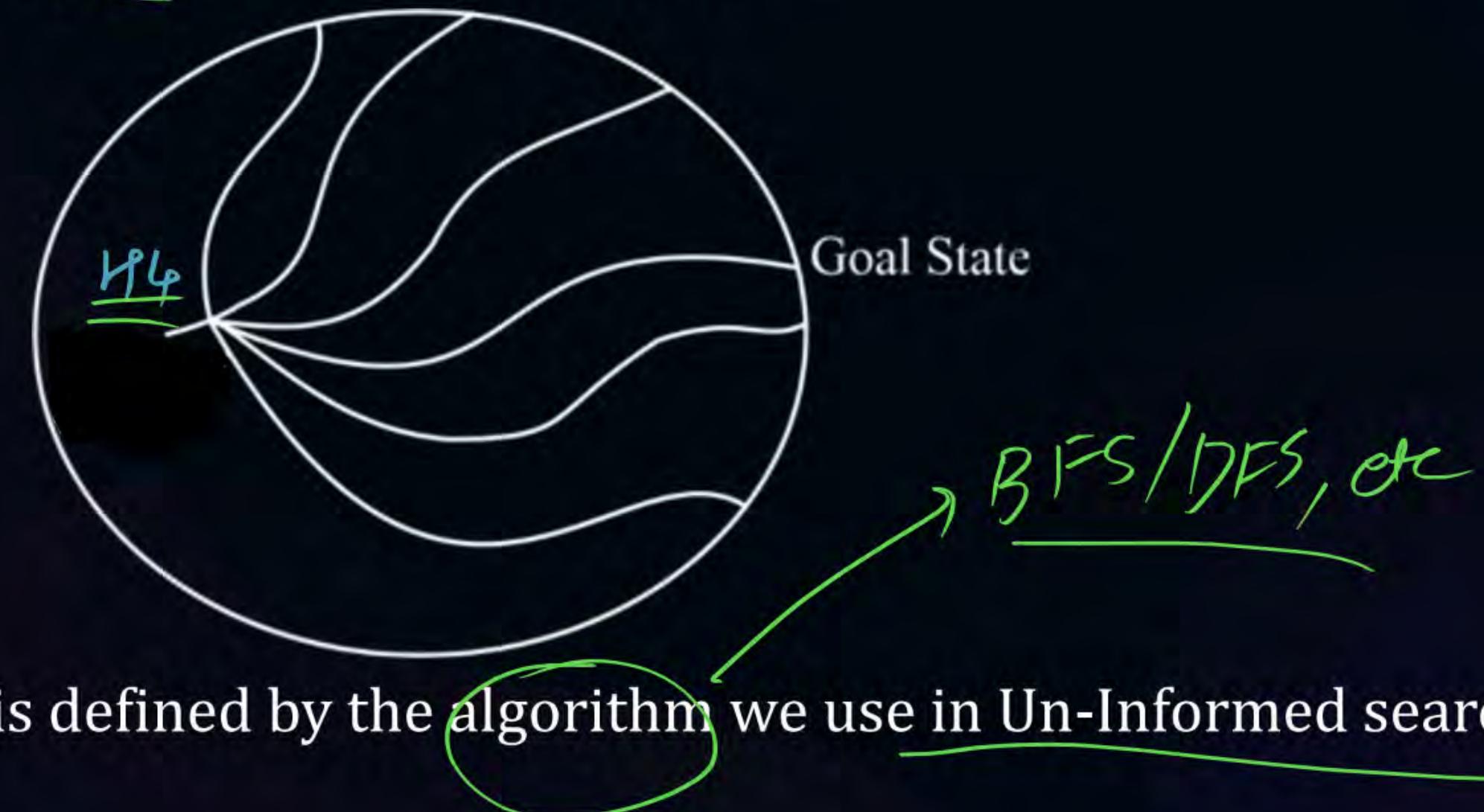
Chichase





Topic : Uninformed Search

- No info abt path
- Only goal state & Start state
- We try to follow all path until goal state is Reached.





Topic : Uninformed Search

- Uninformed search is a class of general-purpose search algorithms which operates in brute force-way. Uninformed search algorithms do not have additional information about state or search space other than how to traverse the tree, so it is also called blind search.
- The search algorithms in this section have no additional information on the goal node other than the one provided in the problem definition. The plans to reach the goal state from the start state differ only by the order and/or length of actions. Uninformed search is also called Blind search. These algorithms can only generate the successors and differentiate between the goal state and non-goal state.
- Uninformed search algorithms do not have any additional information about the goal state other than the information provided in the problem definition. They explore the search space blindly without any guidance or heuristics to indicate which path is likely to lead to the goal.



Topic : Uninformed Search

- The following uninformed search algorithms are discussed in this section.
- Depth First Search. (DFS)
- Breadth First Search (BFS)
- Uniform Cost Search (UCS)
- Each of these algorithms will have:
 - A problem graph, containing the start node S and the goal node G.
 - A strategy, describing the manner in which the graph will be traversed to get to G.
 - Store all the possible states (nodes) that you can go from the current states.
 - A solution plan, which the sequence of nodes from S to G.

~~path~~



4 Types of graphs



$A \rightarrow B$ | Directed

$A \leftrightarrow B$ | Undirected

	Weighted	Unweighted
DW	DW	D UW
UdW	UdW	Ud Uw



Topic : Uninformed Search

Breadth first search

- Breadth-first search (BFS) is an algorithm for traversing or searching tree or graph data structures.
- It starts at the tree root (or some arbitrary node of a graph, sometimes referred to as a 'search key'), and explores all of the neighbor nodes at the present depth prior to moving on to the nodes at the next depth level.
adjacent
- It is implemented using a queue (FIFO).
- Breadth First Search (BFS) is a graph traversal algorithm that explores all the vertices in a graph at the current depth before moving on to the vertices at the next depth level. It starts at a specified vertex and visits all its neighbors before moving on to the next level of neighbors. BFS is commonly used in algorithms for pathfinding, connected components, and shortest path problems in graphs.





Topic : Uninformed Search

Breadth first search

- The Breadth-First Search is a traversing algorithm used to satisfy a given property by searching the tree or graph data structure.
- It belongs to uninformed or blind search AI algorithms as It operates solely based on the connectivity of nodes and doesn't prioritize any particular path over another based on heuristic knowledge or domain-specific information.
- it doesn't incorporate any additional information beyond the structure of the search space. It is optimal for unweighted graphs and is particularly suitable when all actions have the same cost. Due to its systematic search strategy, BFS can efficiently explore even infinite state spaces.
- Breadth-First Search (BFS) is an algorithm for traversing or searching tree or graph data structures.



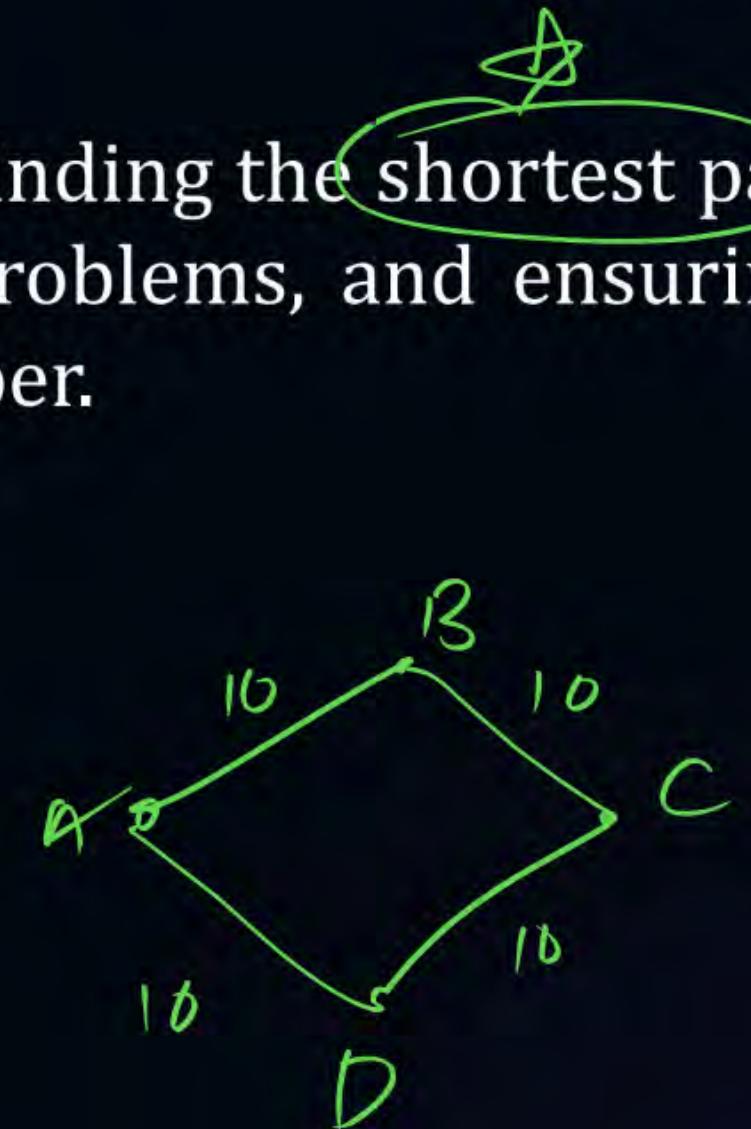
Topic : Uninformed Search

Breadth first search

- In artificial intelligence (AI), BFS is commonly used for finding the shortest path in unweighted graphs, exploring state spaces in search problems, and ensuring all nodes at a given depth are processed before moving deeper.

and
unweighted

11:30 AM





THANK - YOU



DS & AI ENGINEERING



Artificial Intelligence

Un-Informed search

Lecture No.- OR

By- Aditya sir



Recap of Previous Lecture



Topic

Topic

Topic

Topic

In/zo

Uninformed basics

Topics to be Covered



Topic

Topic

Topic

BFS *

DFS



About Aditya Jain sir

1. Appeared for GATE during BTech and secured AIR 60 in GATE in very first attempt - City topper
2. Represented college as the first Google DSC Ambassador.
3. The only student from the batch to secure an internship at Amazon. (9+ CGPA)
4. Had offer from IIT Bombay and IISc Bangalore to join the Masters program
5. Joined IIT Bombay for my 2 year Masters program, specialization in Data Science
6. Published multiple research papers in well known conferences along with the team
7. Received the prestigious excellence in Research award from IIT Bombay for my Masters thesis in ML
8. Completed my Masters with an overall GPA of 9.36/10
9. Joined Dream11 as a Data Scientist
10. Have mentored 15,000+ students & working professionals in field of Data Science and Analytics
11. Have been mentoring & teaching GATE aspirants to secure a great rank in limited time
12. Have got around 27.5K followers on LinkedIn where I share my insights and guide students and professionals.





Telegram



Telegram Link for Aditya Jain sir:

https://t.me/AdityaSir_PW



Topic : Uninformed Search



What is a state and state space...

- **Initial State:** The starting point of the search.
- **Goal State(s):** The desired state(s) that signify a solution.
- **Actions/Operators:** The set of possible actions that can be performed to move from one state to another.
- **Transition Model:** Describes the result of applying an action to a state, leading to a new state.
- **Path Cost:** A function that assigns a cost to each path, helping to find the most efficient solution.



Topic : Uninformed Search

Breadth first search

- Breadth-first search (BFS) is an algorithm for traversing or searching tree or graph data structures.
- It starts at the tree root (or some arbitrary node of a graph, sometimes referred to as a 'search key'), and explores all of the neighbor nodes at the present depth prior to moving on to the nodes at the next depth level.
- It is implemented using a queue (FIFO).
- Breadth First Search (BFS) is a graph traversal algorithm that explores all the vertices in a graph at the current depth before moving on to the vertices at the next depth level. It starts at a specified vertex and visits all its neighbors before moving on to the next level of neighbors. BFS is commonly used in algorithms for pathfinding, connected components, and shortest path problems in graphs.





Topic : Uninformed Search

Breadth first search

- The Breadth-First Search is a traversing algorithm used to satisfy a given property by searching the tree or graph data structure.
- It belongs to uninformed or blind search AI algorithms as It operates solely based on the connectivity of nodes and doesn't prioritize any particular path over another based on heuristic knowledge or domain-specific information.
- it doesn't incorporate any additional information beyond the structure of the search space. It is optimal for unweighted graphs and is particularly suitable when all actions have the same cost. Due to its systematic search strategy, BFS can efficiently explore even infinite state spaces. ↗
- Breadth-First Search (BFS) is an algorithm for traversing or searching tree or graph data structures.



Topic : Uninformed Search

Breadth first search

- In artificial intelligence (AI), BFS is commonly used for finding the shortest path in unweighted graphs, exploring state spaces in search problems, and ensuring all nodes at a given depth are processed before moving deeper.

,08 uni-weighted

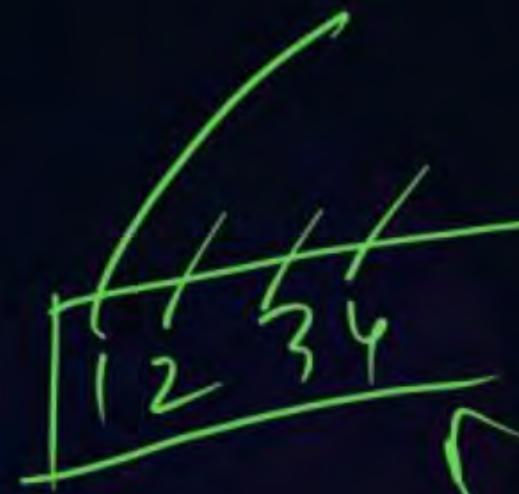


Topic : Uninformed Search

Breadth first search

Key Characteristics

- Traversal Method: BFS explores all nodes at the present depth level before moving on to nodes at the next depth level.
- FIFO Structure: Uses a queue (First-In-First-Out) data structure
- Completeness: BFS is complete, meaning it is guaranteed to find a solution if one exists.



Stack → LIFO
Queue → FIFO



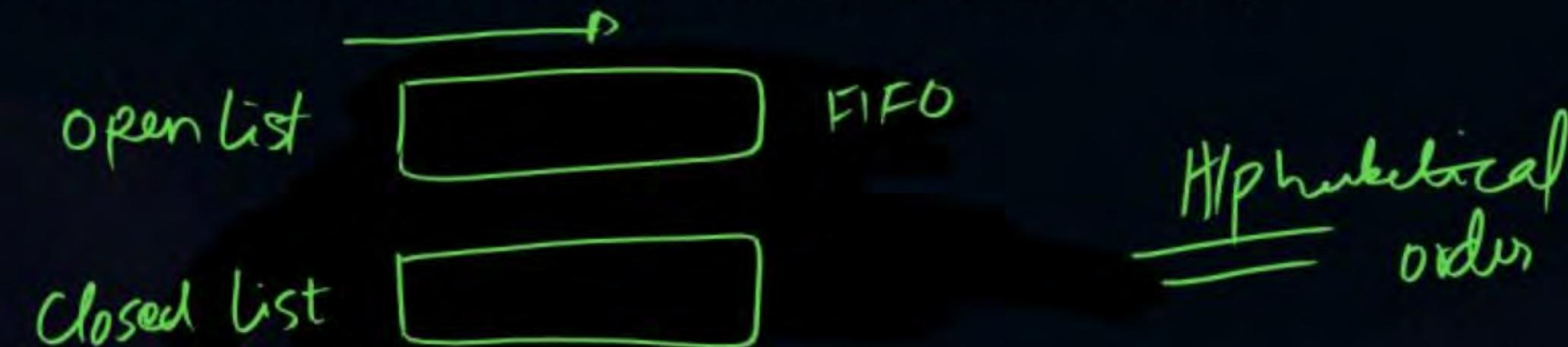


Topic : Uninformed Search

Breadth first search (Example)

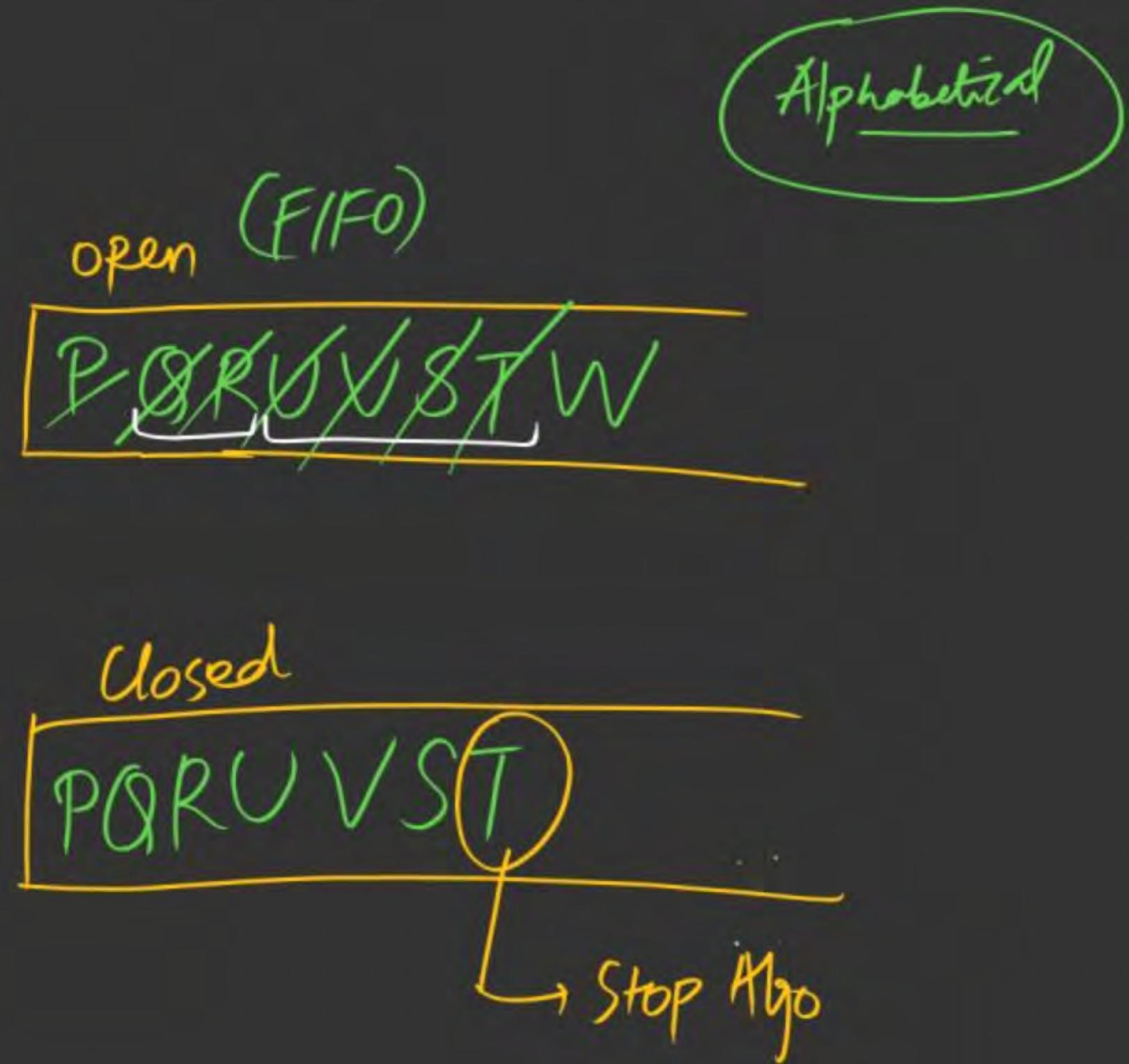
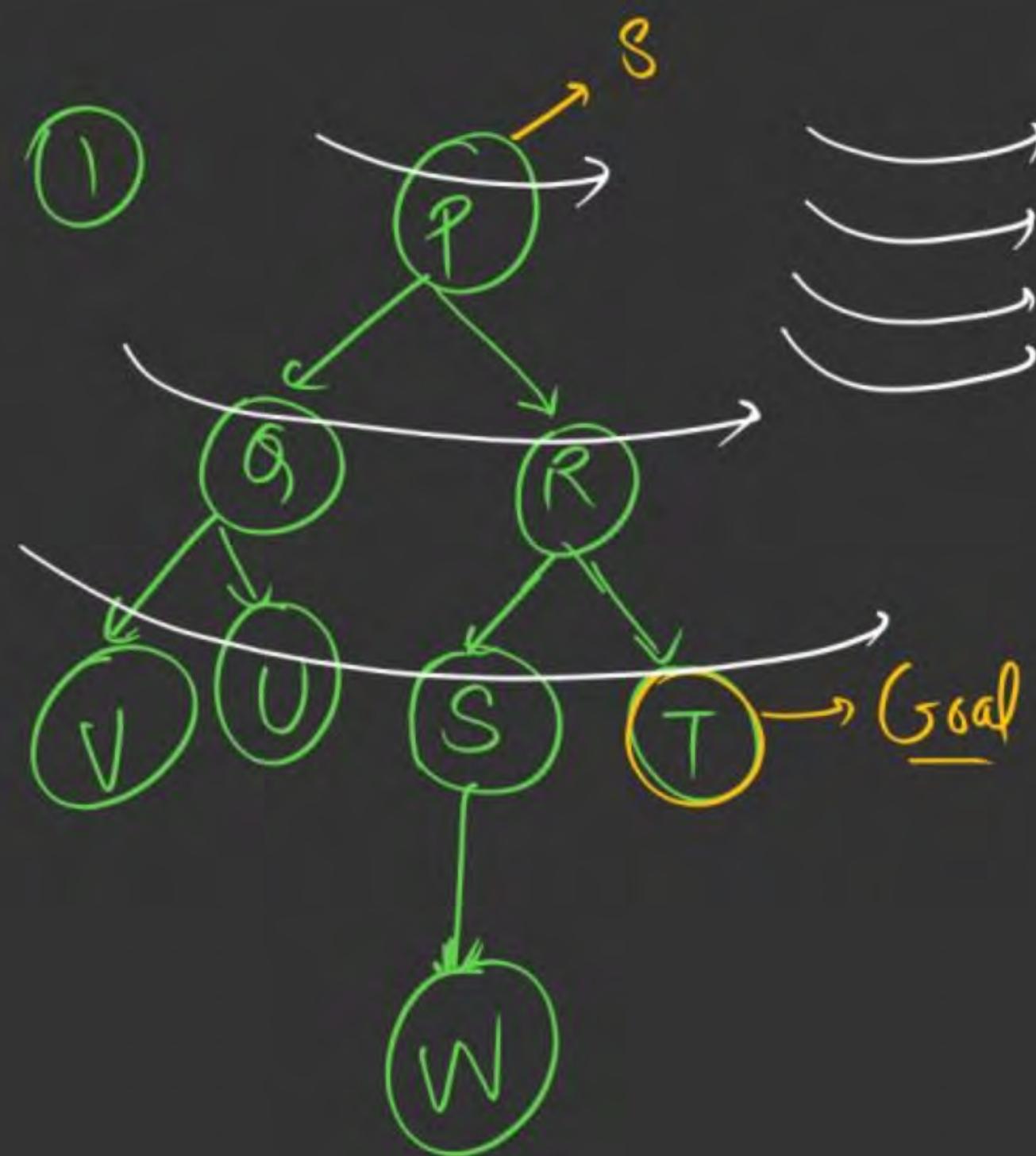
- BSF $\Rightarrow S \rightarrow G$, We create 2 List

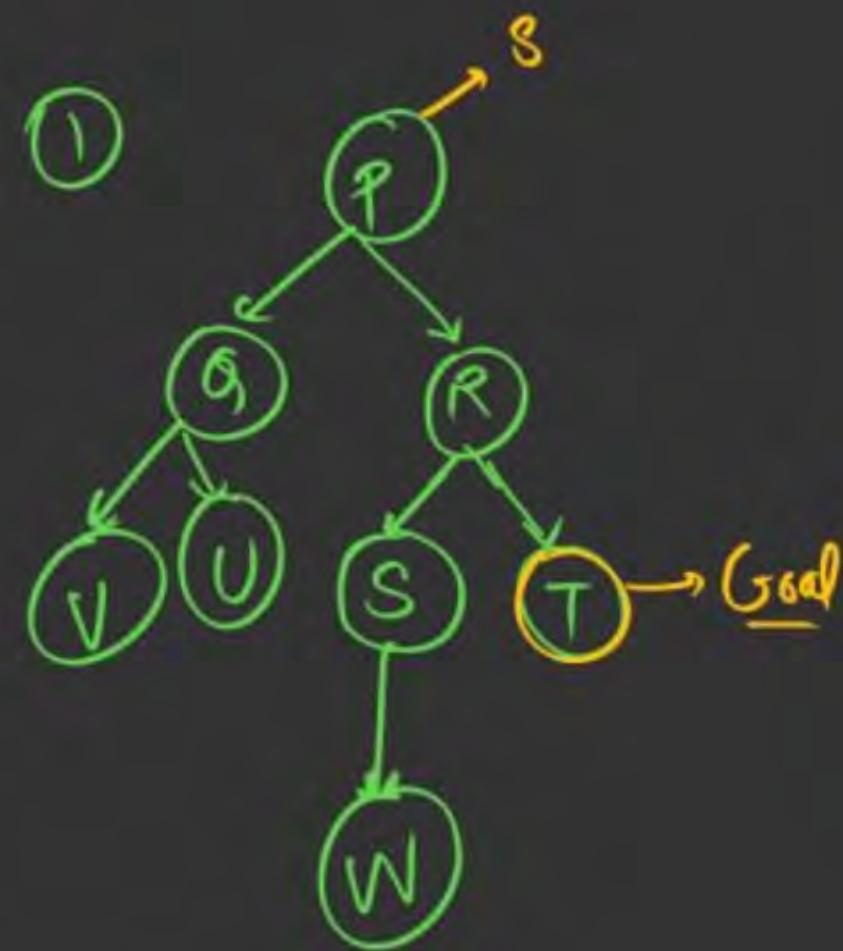
Step 1: We start with the start node. Enter the start node into Q.



Step 2: Now remove the first node in Q and inset it's immediate children(delete node from left of Q and insert from Right)

Step 3: Repeat step 2 until goal node Reach.

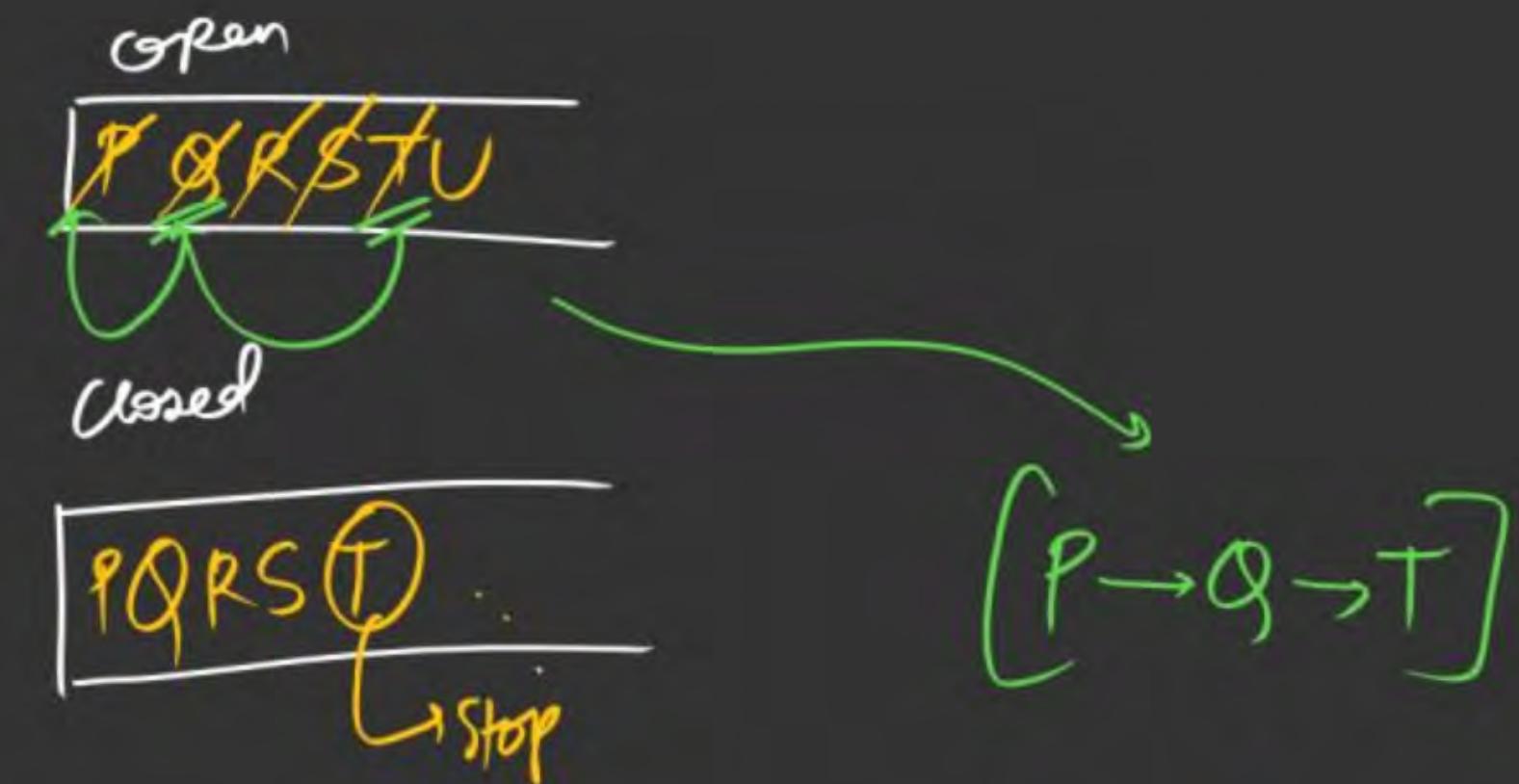
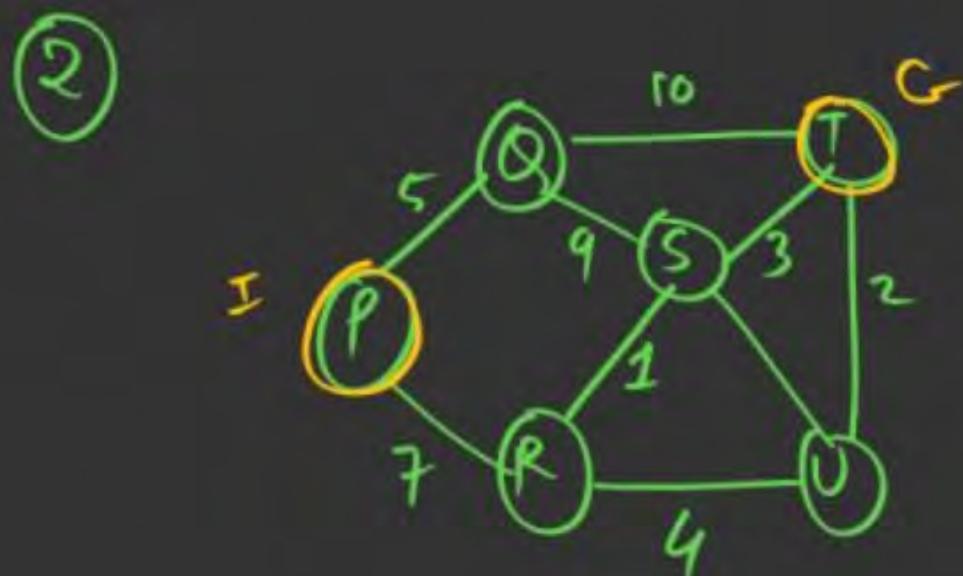




Shortest path: $P \xrightarrow{=} T$



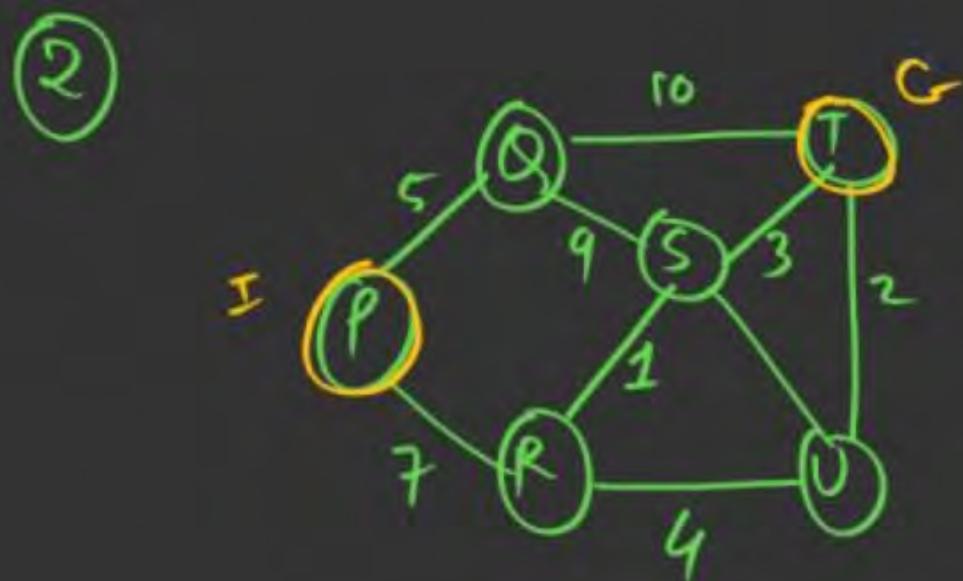
Path: $\underbrace{[P \rightarrow R \rightarrow T]}$



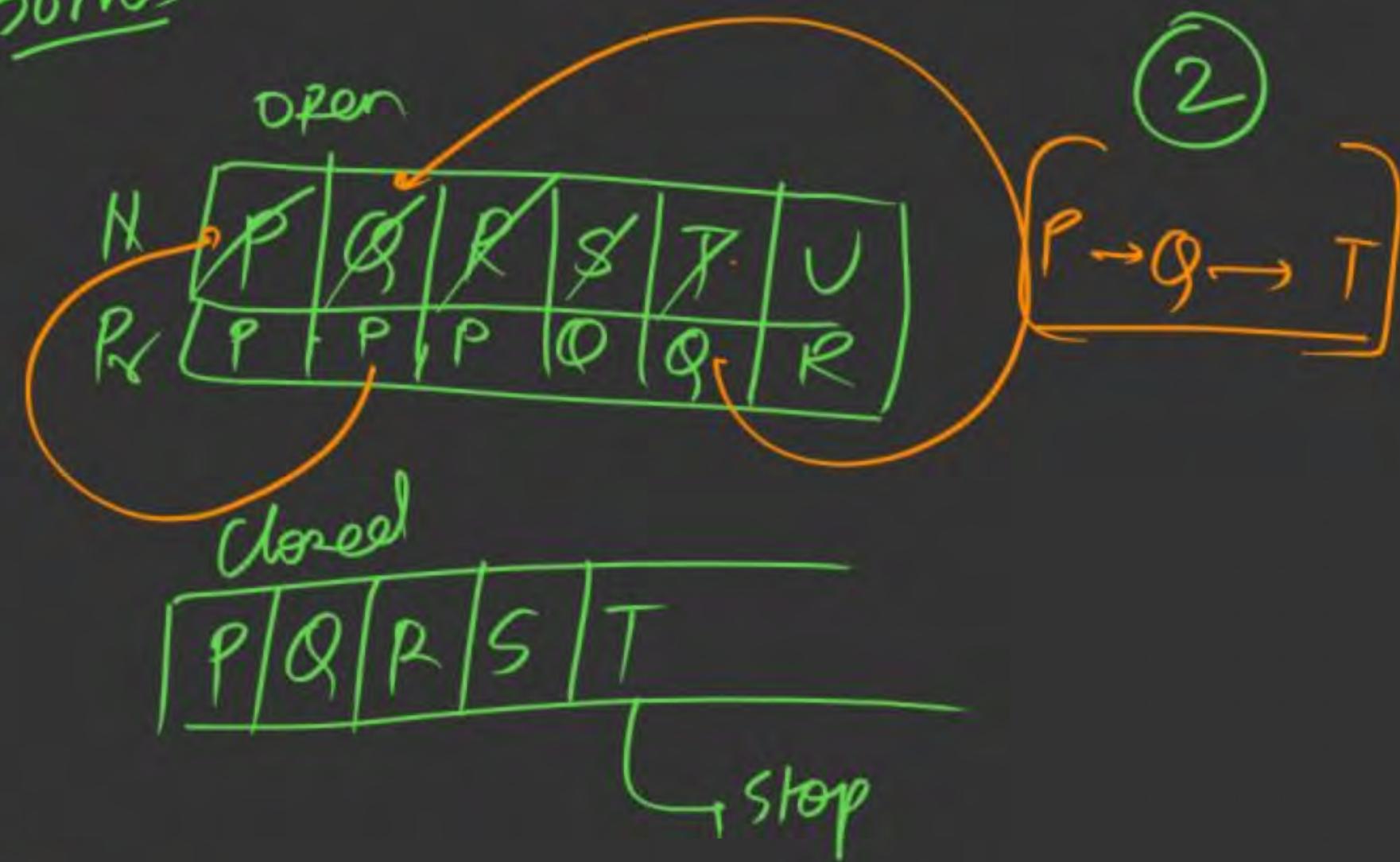
Shortest path: $P \rightarrow T$

- 1) P R S T
- 2) P Q T
- 3) P R U T

Shortest path: path with min
no. of edges



Borris



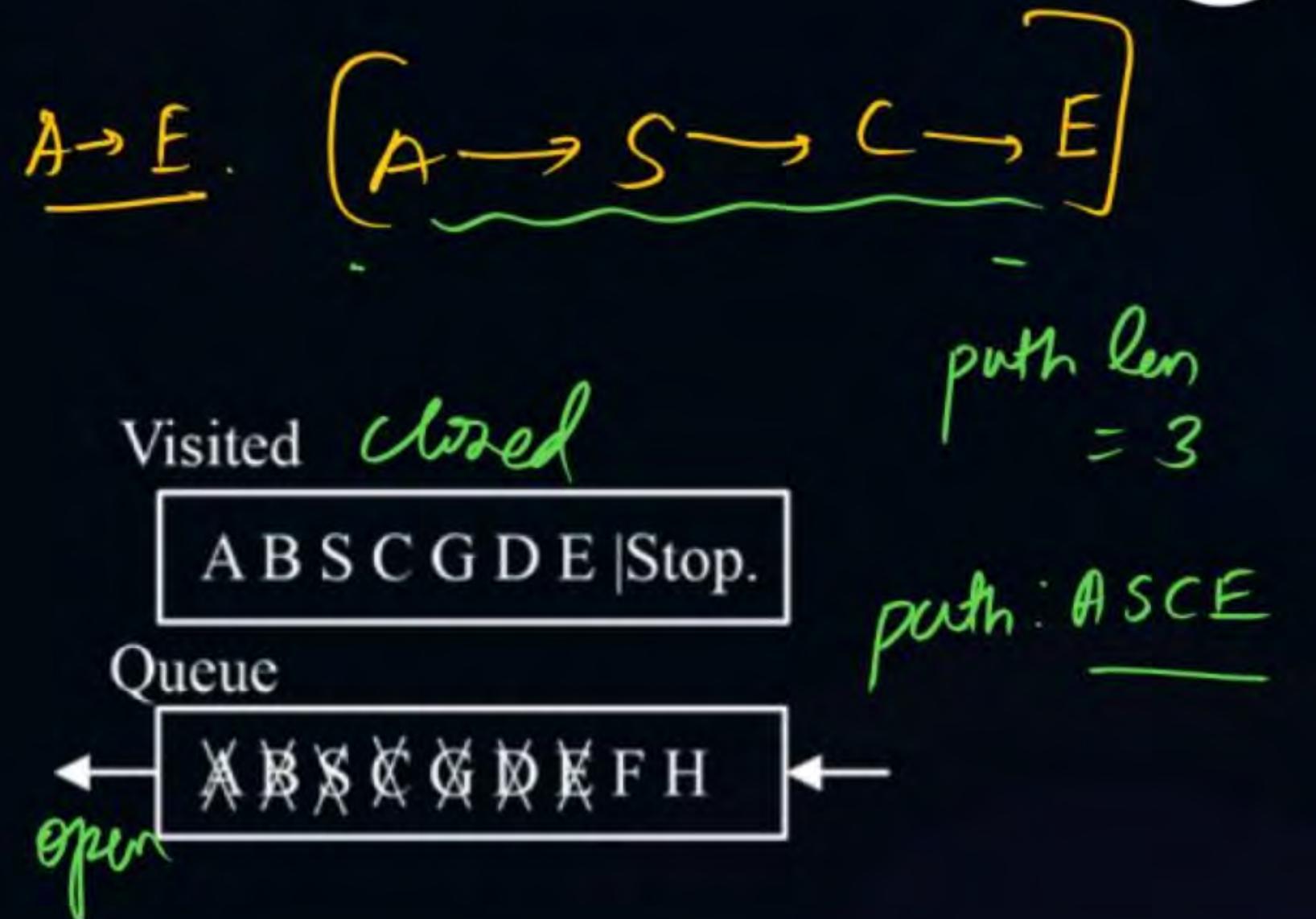
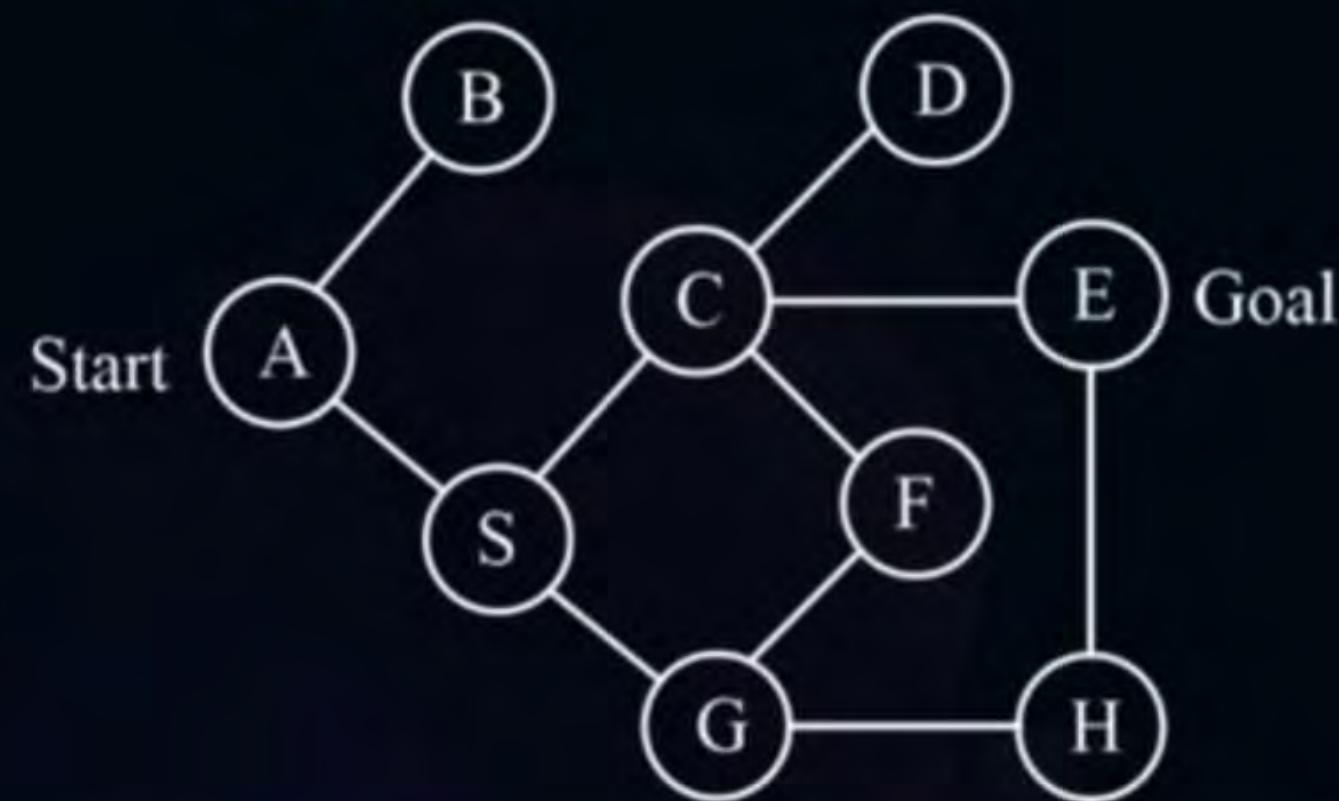
Shortest Path: $P \rightarrow T$

- 1) PR>T
- 2) PQT
- 3) PRUT



Topic : Uninformed Search

Let enter ~~nodes~~ in alphabetical order
nodes



So, if any node on state is present in any list no need to re-enter it



Topic : Uninformed Search

Breadth first Search

Algorithm Initialization: Start at the root node (or any arbitrary node in a graph). We create 2 list (visited and Queue)

- Mark the node as visited.
- Enqueue the node.
- Process: While the queue is not empty
- Dequeue a node from the queue.
- Process the node (e.g., check if it is the goal).
- For each unvisited adjacent node
- Mark it as visited
- Enqueue it.

Bonus

Closed → open



Topic : Uninformed Search

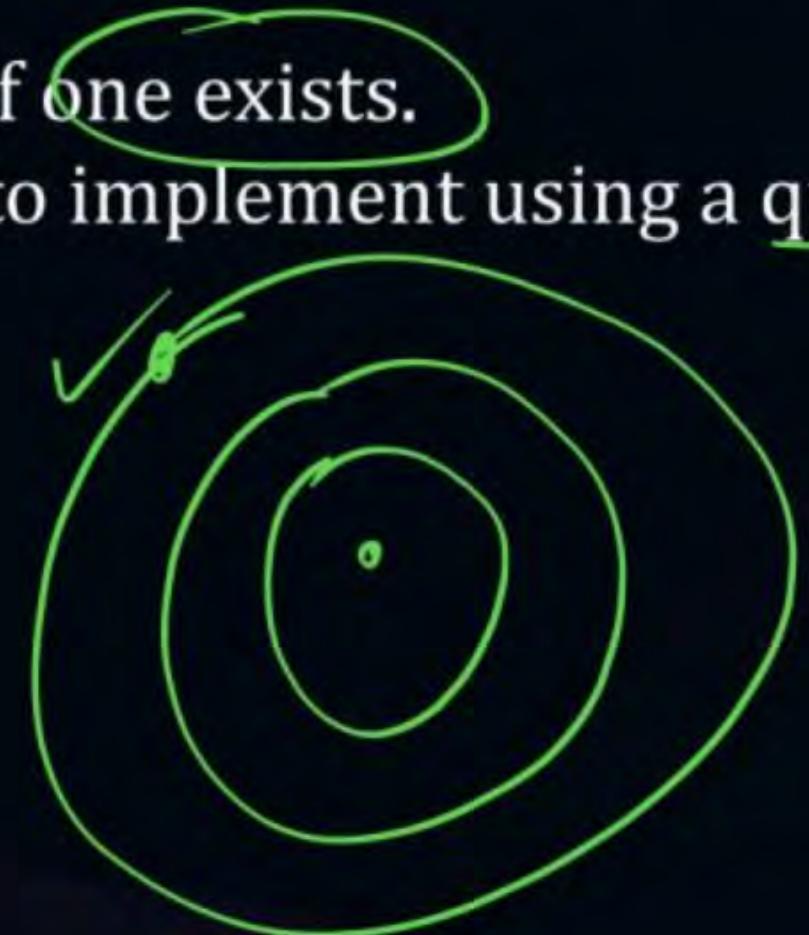
Breadth first Search

Advantages

- **Optimal for Unweighted Graphs:** BFS will find the shortest path in terms of the number of edges.
- **Complete:** Guaranteed to find a solution if one exists.
- Simple implementation: Straightforward to implement using a queue.

BFS 1) optimal

2) Complete





Topic : Uninformed Search

Breadth first Search

Disadvantages

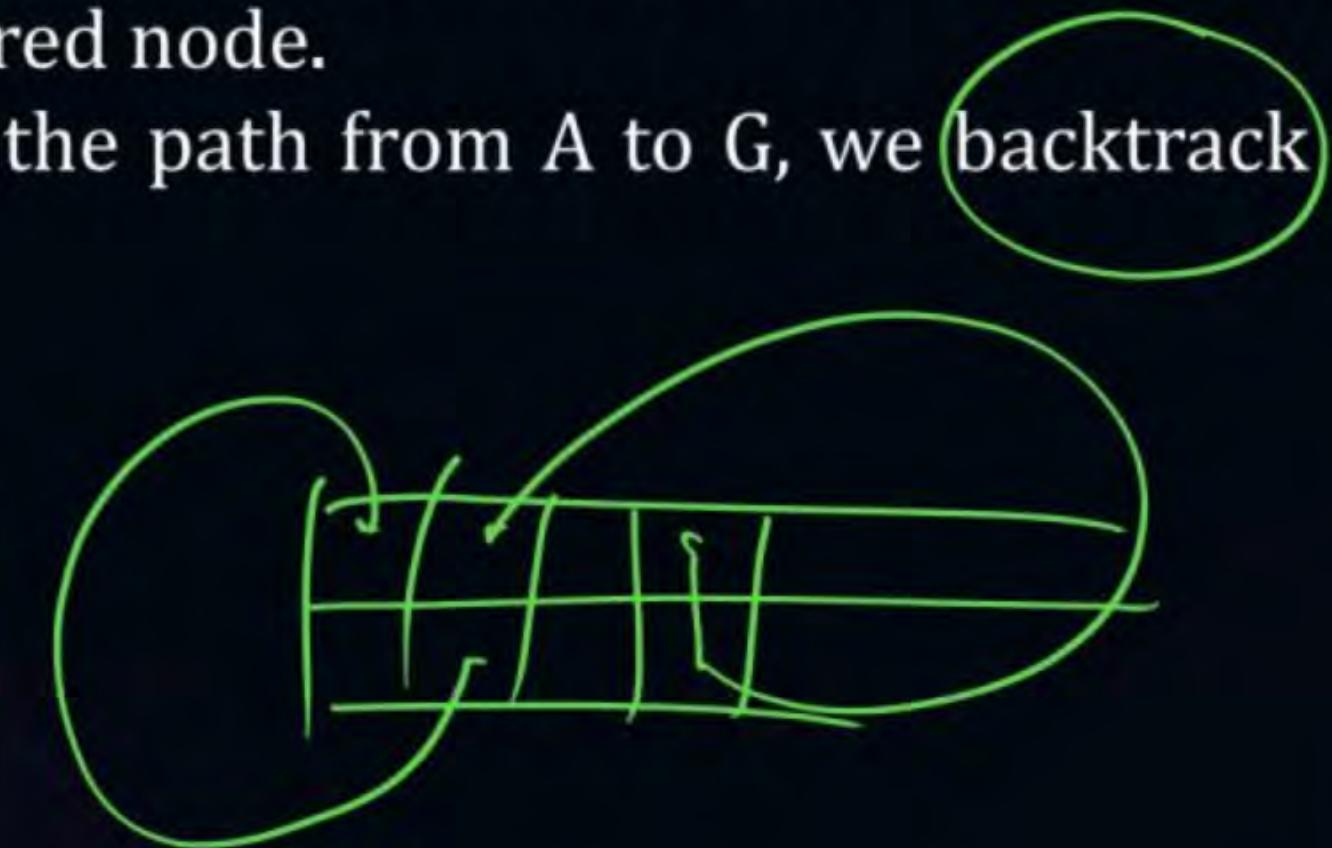
- Memory Usage: Can consume a lot of memory, as it needs to store all nodes at the current depth level.
- Not suitable for Weighted Graphs: BFS does not account for edge weights; Dijkstra's algorithm is more appropriate in such cases.





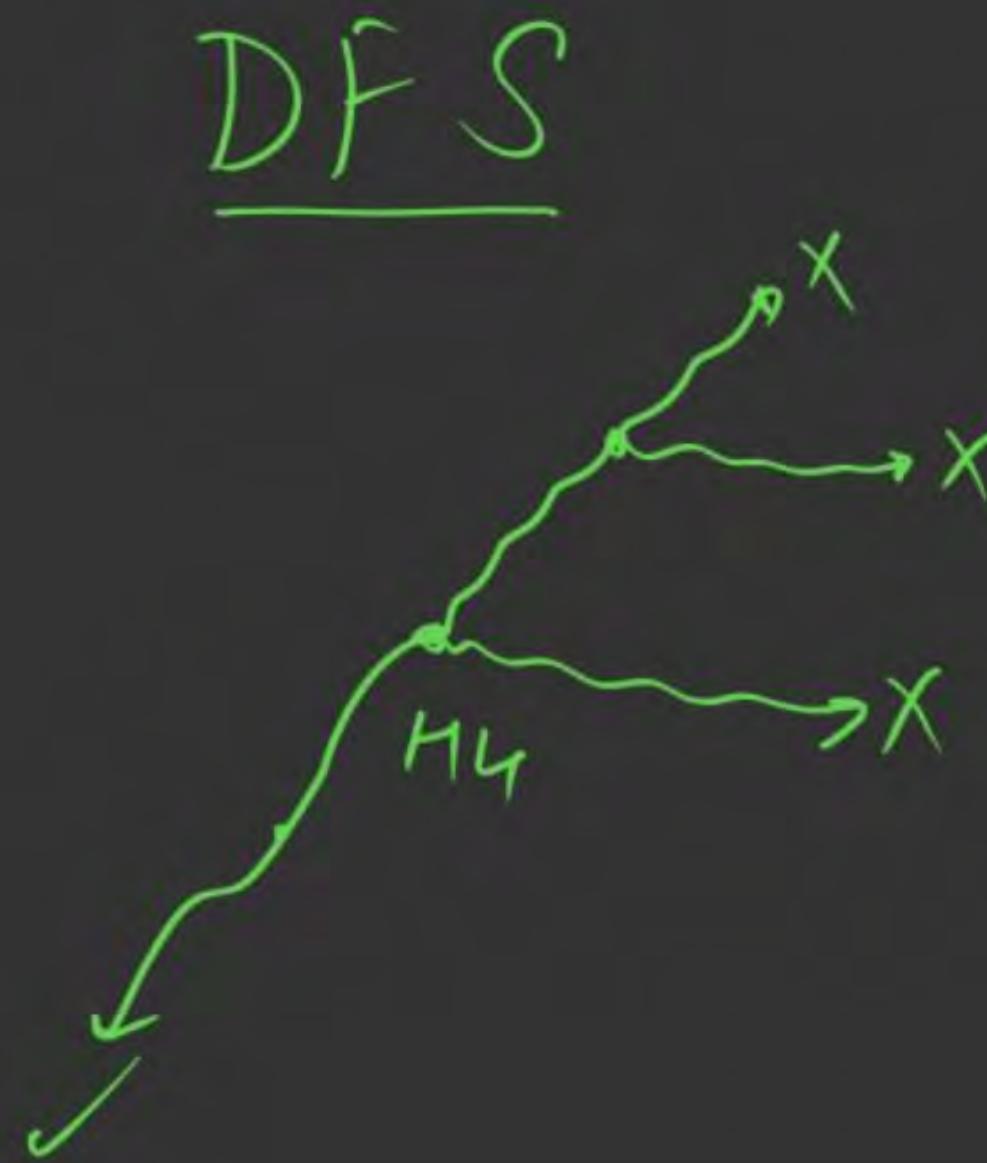
Topic : Uninformed Search

- This algorithm always give a result (if it exists)
- Because before removing the node form the queue, we check it with goal node
- And from root to the desired node, we can easily follow the parent - parent and get the path from root to the desired node.
- Path Reconstruction: To find the path from A to G, we backtrack using the parent information:

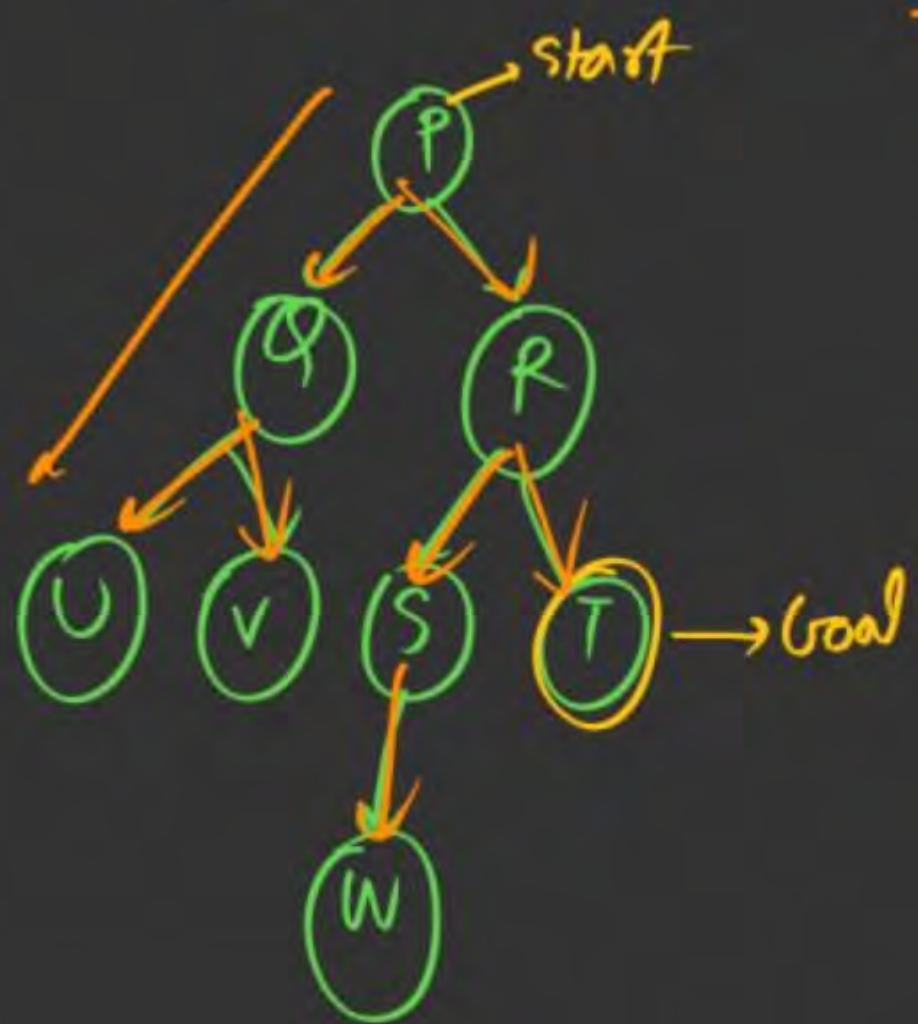


DFS

→ Depth First Search



DFS



Alphabetical order

open (LIFO)



closed



stop



THANK - YOU



DS & AI ENGINEERING

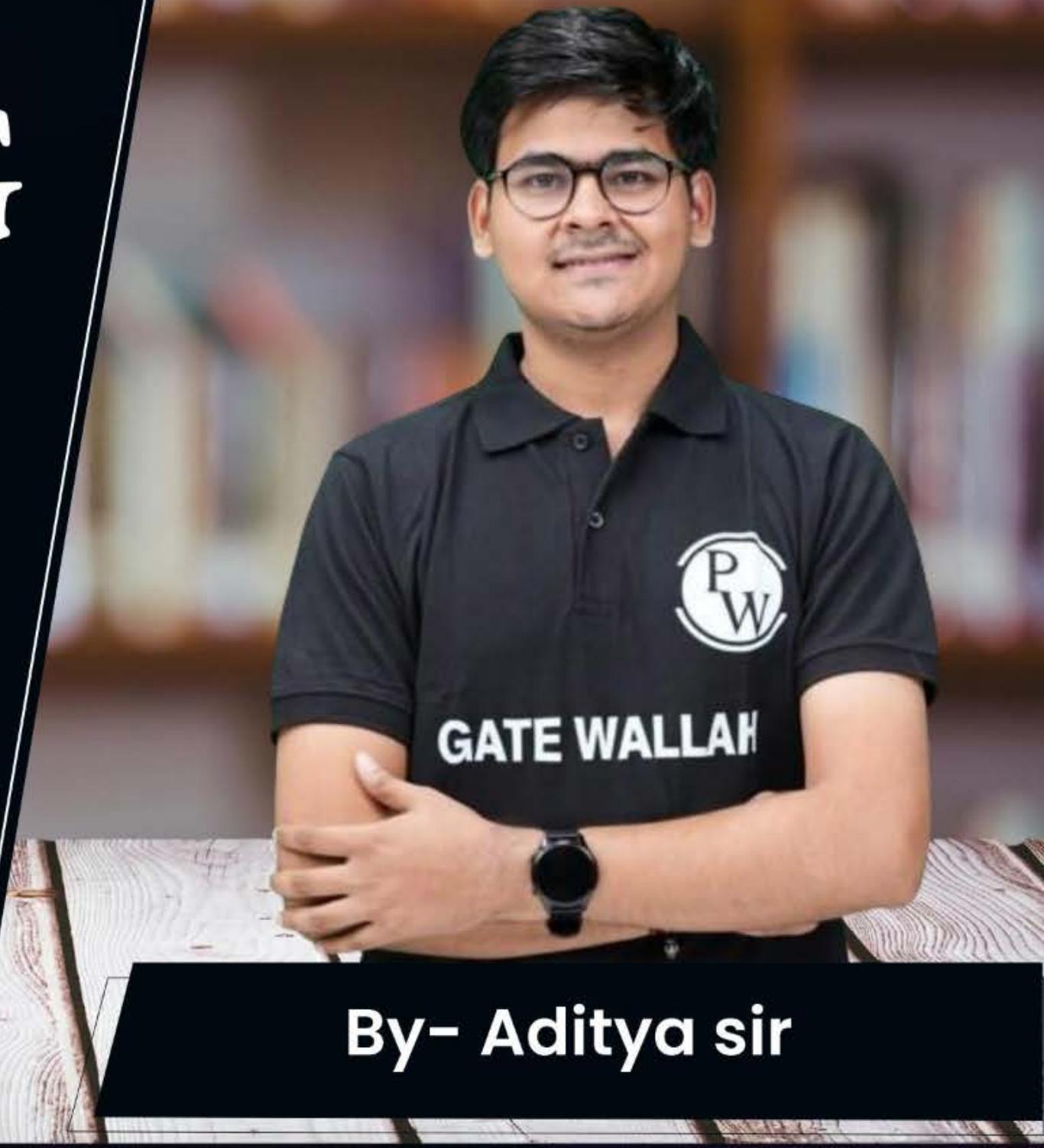


Artificial Intelligence

Un-Informed search

Lecture No.- 03

By- Aditya sir



Recap of Previous Lecture



Topic

Topic

Topic

Topic

BFS

DFS on Tree

Topics to be Covered

P
W



Topic

Topic

Topic

DFS on Graphs

Practice



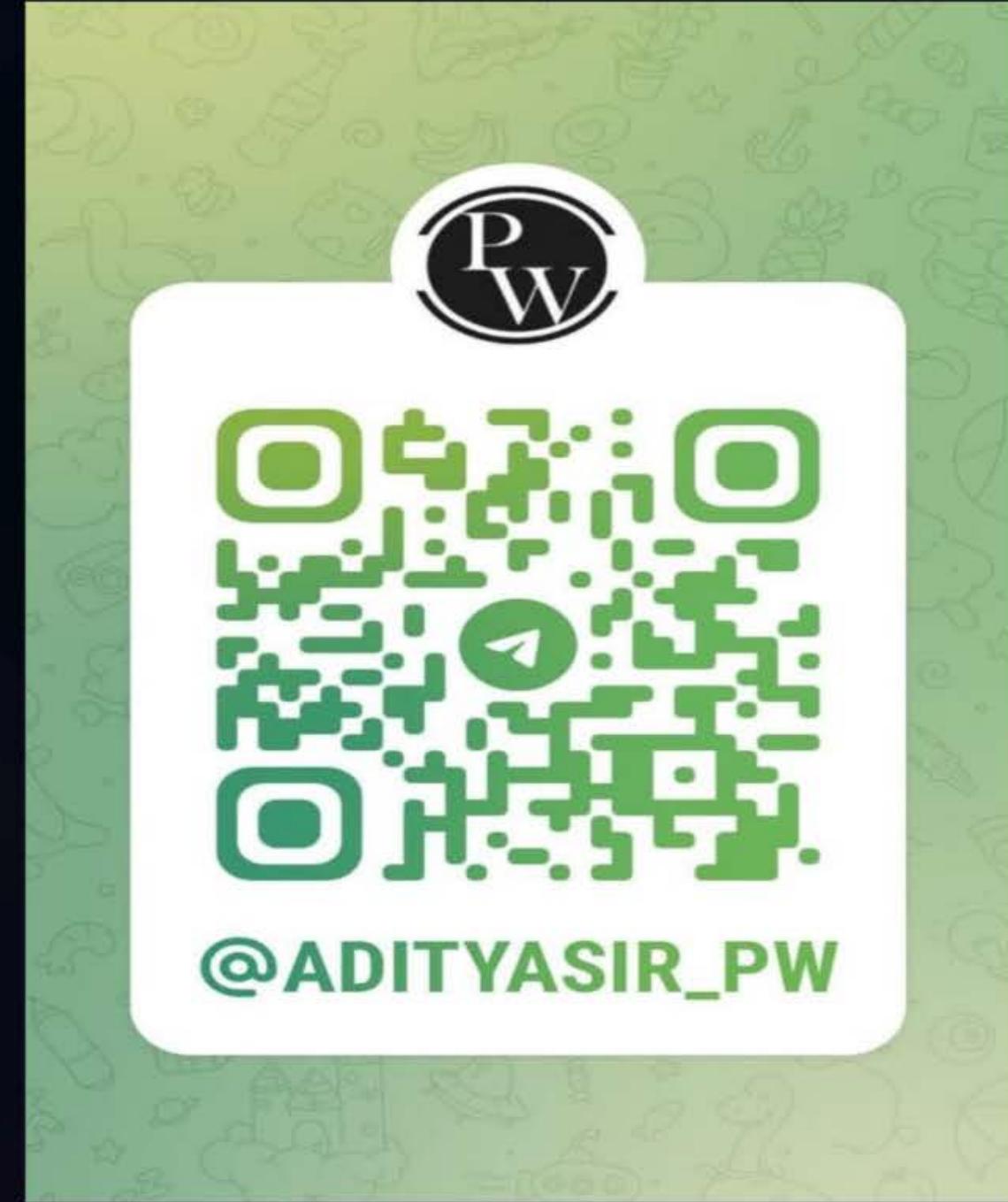
About Aditya Jain sir

1. Appeared for GATE during BTech and secured AIR 60 in GATE in very first attempt - City topper
2. Represented college as the first Google DSC Ambassador.
3. The only student from the batch to secure an internship at Amazon. (9+ CGPA)
4. Had offer from IIT Bombay and IISc Bangalore to join the Masters program
5. Joined IIT Bombay for my 2 year Masters program, specialization in Data Science
6. Published multiple research papers in well known conferences along with the team
7. Received the prestigious excellence in Research award from IIT Bombay for my Masters thesis in ML
8. Completed my Masters with an overall GPA of 9.36/10
9. Joined Dream11 as a Data Scientist
10. Have mentored 15,000+ students & working professionals in field of Data Science and Analytics
11. Have been mentoring & teaching GATE aspirants to secure a great rank in limited time
12. Have got around 27.5K followers on Linkedin where I share my insights and guide students and professionals.





Telegram



Telegram Link for Aditya Jain sir:
https://t.me/AdityaSir_PW



Topic : Analysis of Un-Informed Search



DFS on Graphs



Topic : Uninformed Search

BFS:

- Queue → FIFO.
- Scan complete level in on go.
- 2 List
 - Queue (*open*)
 - Visited nodes. (*Closed*)



Topic : Uninformed Search

Breath first search

- Breadth-first search (BFS) is an algorithm for traversing or searching tree or graph data structures.
- It starts at the tree root (or some arbitrary node of a graph, sometimes referred to as a 'search key'), and explores all of the neighbor nodes at the present depth prior to moving on to the nodes at the next depth level.
- It is implemented using a queue (FIFO).
- Breadth First Search (BFS) is a graph traversal algorithm that explores all the vertices in a graph at the current depth before moving on to the vertices at the next depth level. It starts at a specified vertex and visits all its neighbors before moving on to the next level of neighbors. BFS is commonly used in algorithms for pathfinding, connected components, and shortest path problems in graphs.



Topic : Uninformed Search



Breath first search

- The Breadth-First Search is a traversing algorithm used to satisfy a given property by searching the tree or graph data structure.
- It belongs to uninformed or blind search AI algorithms as It operates solely based on the connectivity of nodes and doesn't 'prioritize' any particular path over another based on heuristic knowledge or domain-specific information.
- It doesn't incorporate any additional information beyond the structure of the search space. It is optimal for unweighted graphs and is particularly suitable when all actions have the same cost. Due to its systematic search strategy, BFS can efficiently explore even infinite state spaces



Topic : Uninformed Search

BFS Algo:

- If the solution exist then BFS will surely find goal state in state space
- ~~Completeness~~ property of Algo.

BFS →
1) Complete
2) Optimal

DFS

a) Node is visited
(present in closed list)

b) Node not yet visited
but is present in open list

BFS

not in
open list
again

no need to
visit again
in open list

DFS

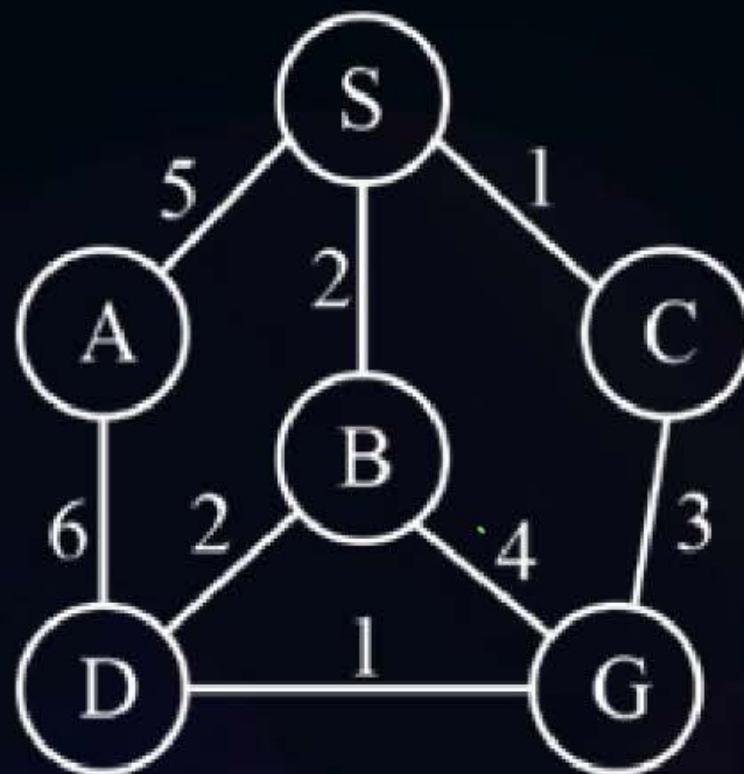
visit again
in open list



Topic : Uninformed Search



BFS
 $S \rightarrow G$



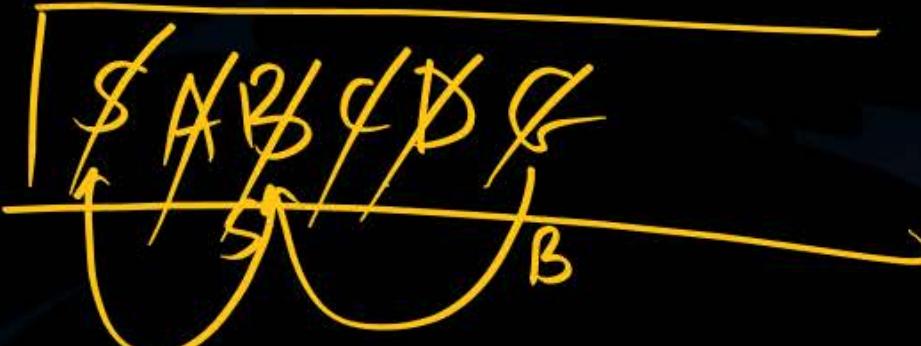
Alphabetical



BFS: len of path
 $S \rightarrow G$?

$S \rightarrow B \rightarrow G$

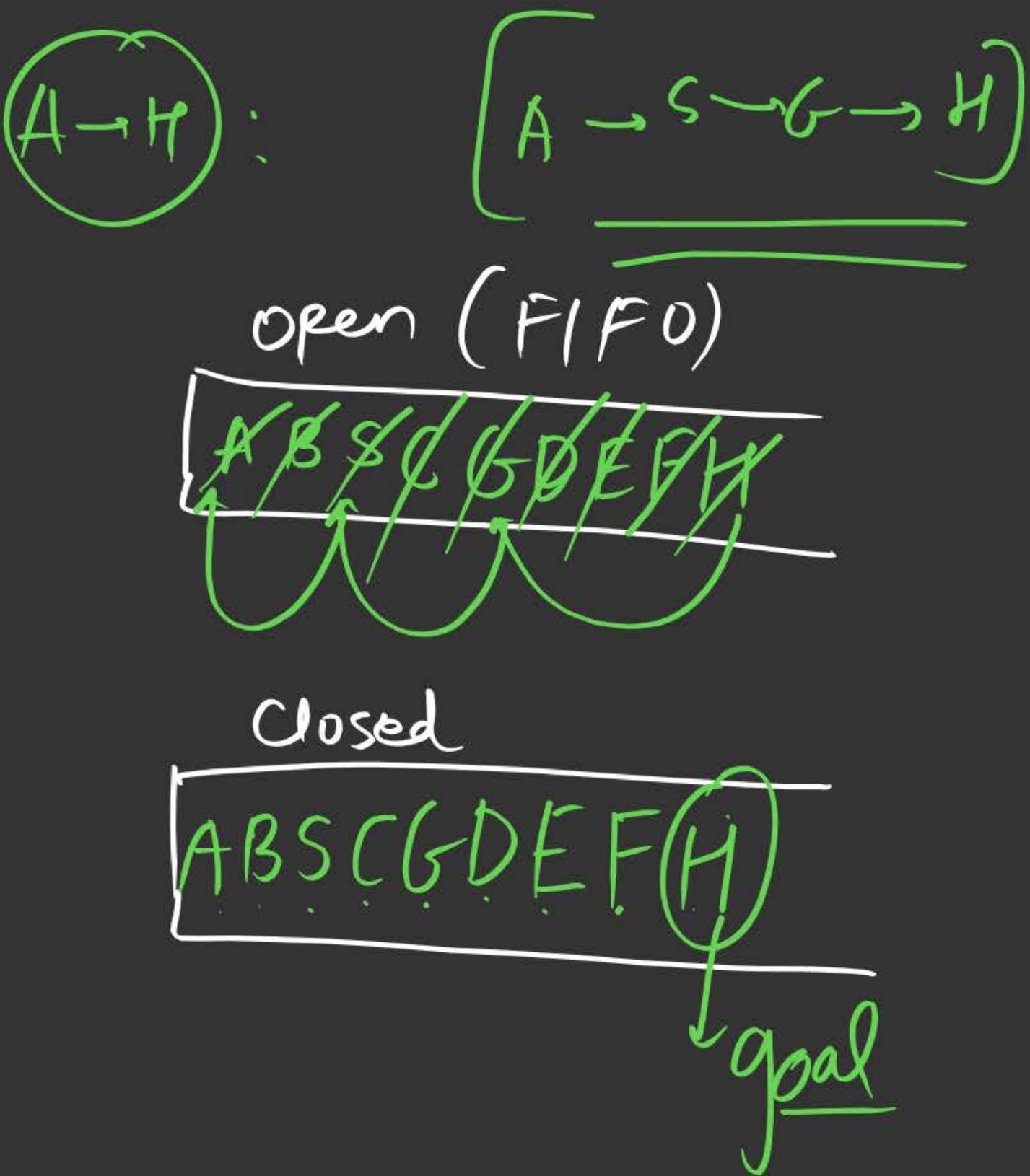
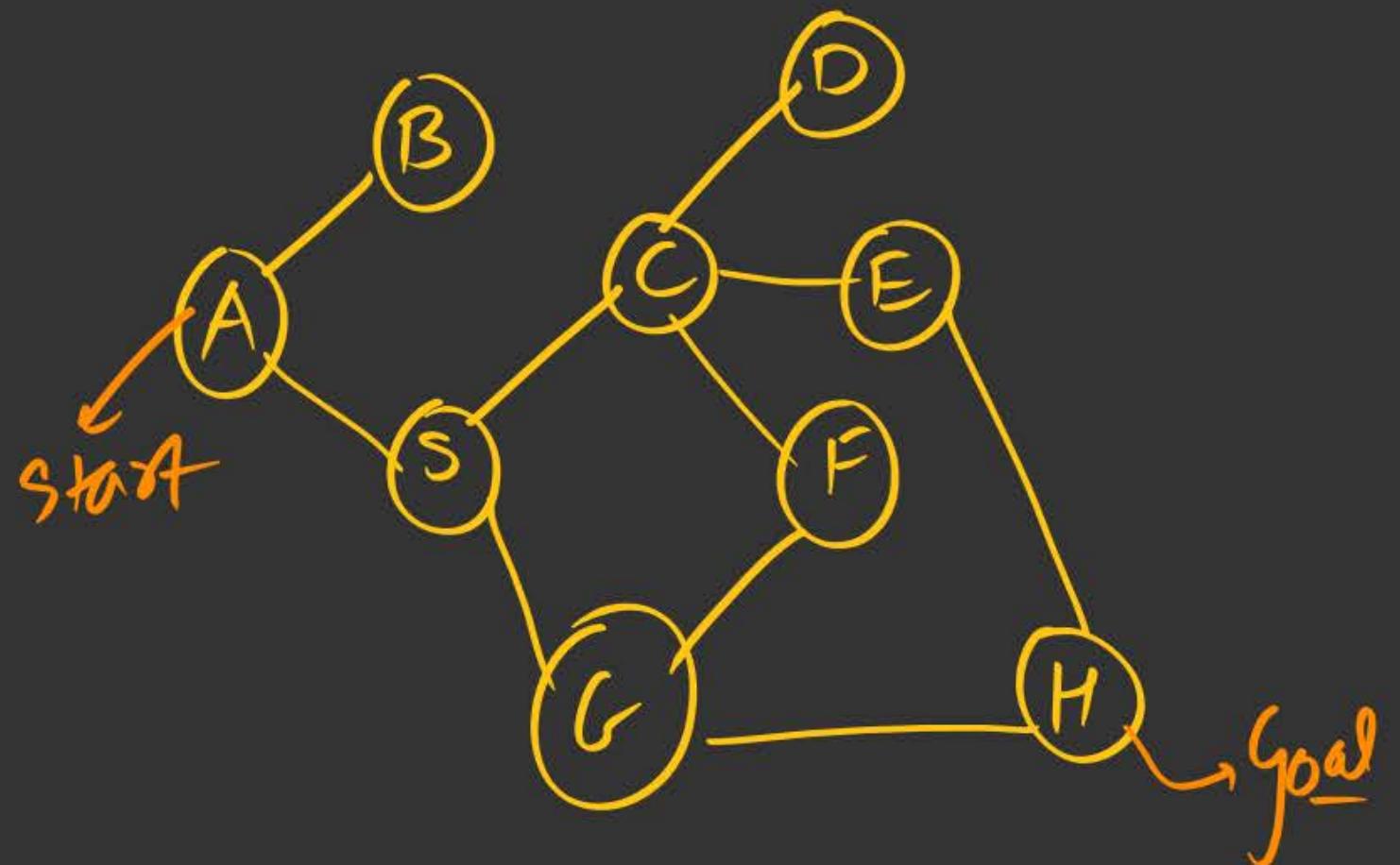
open

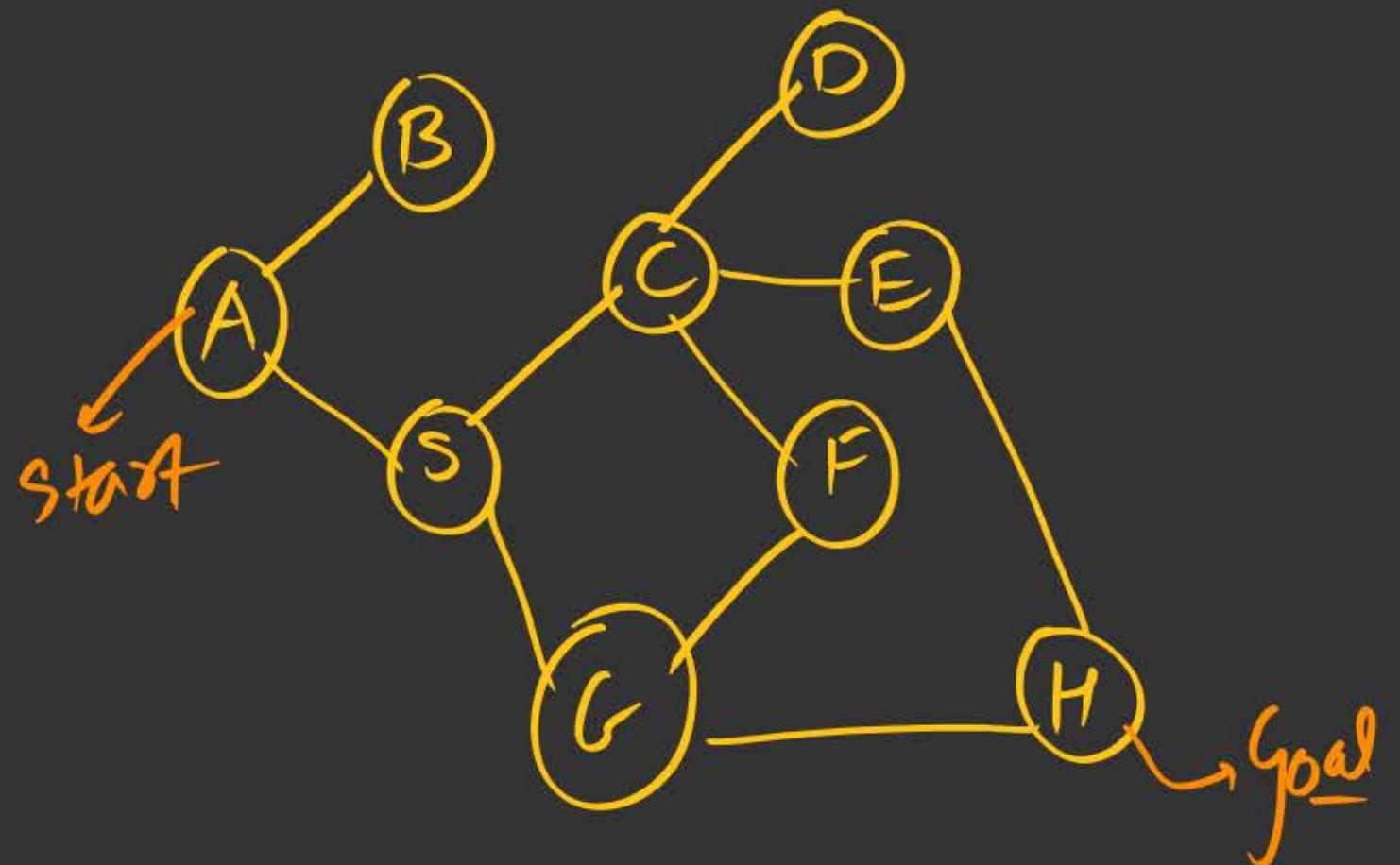


closed



stop





open (LIFO)

~~A S B / G C / F E / D / H G~~

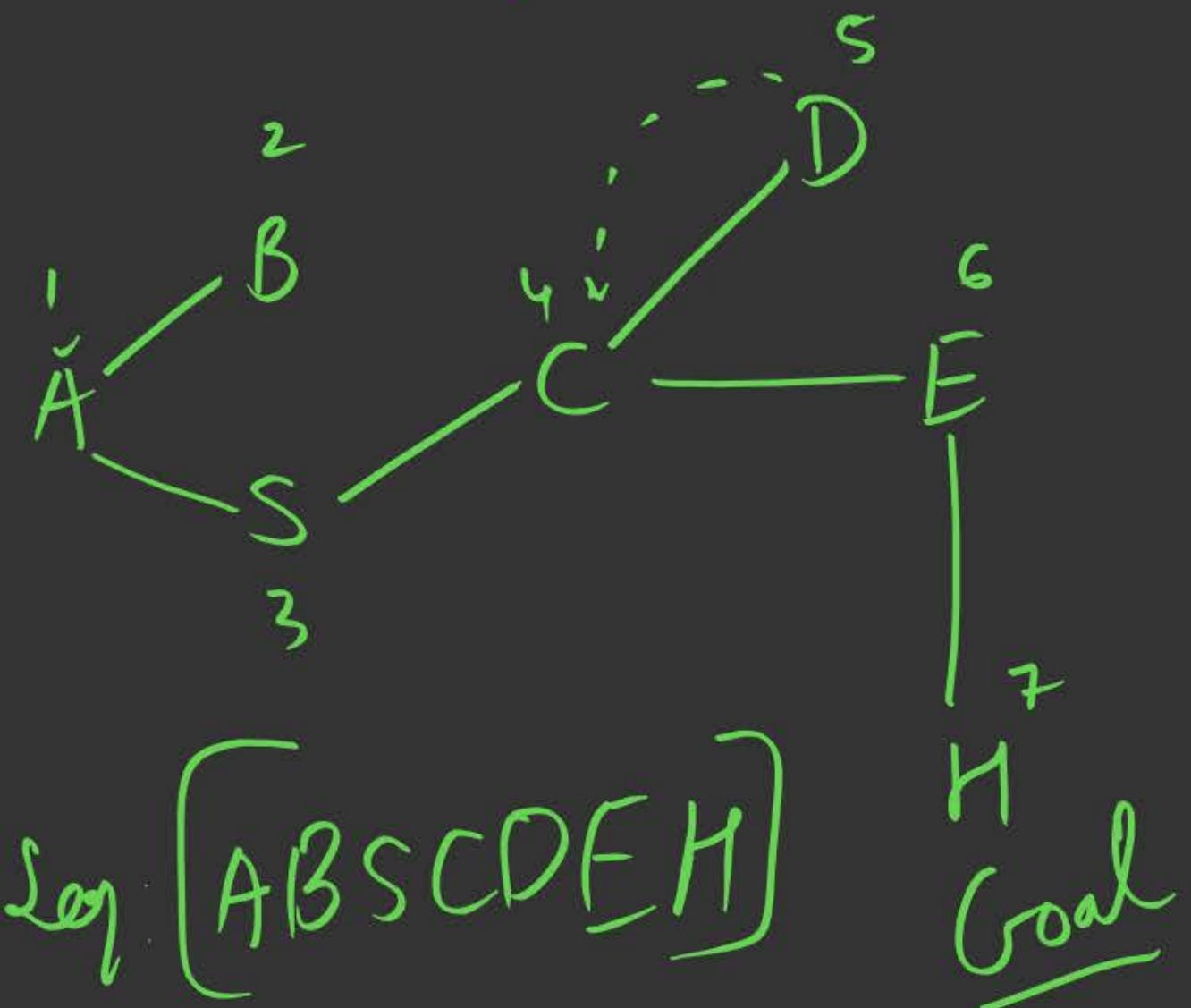
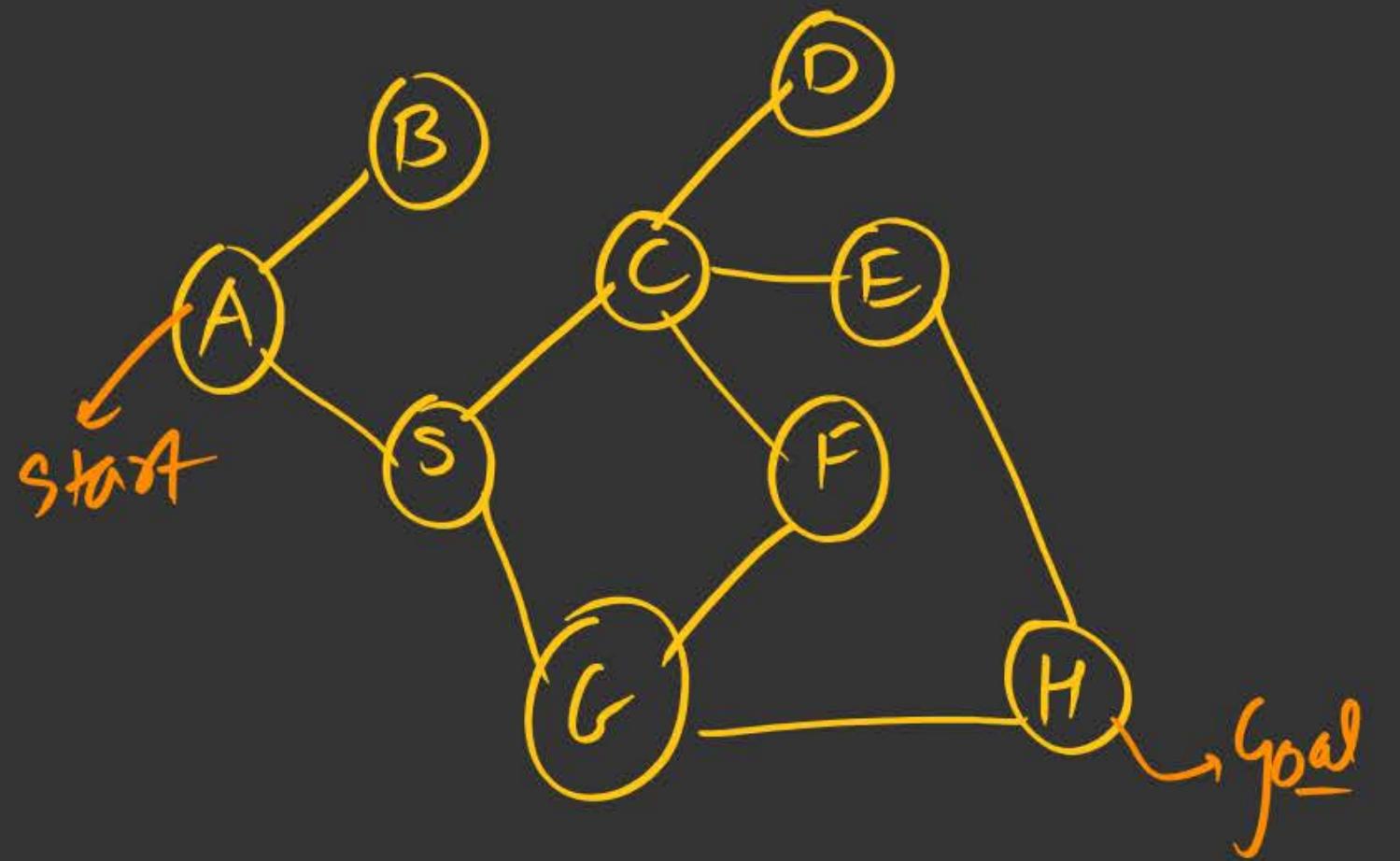
closed

A B S C D E H

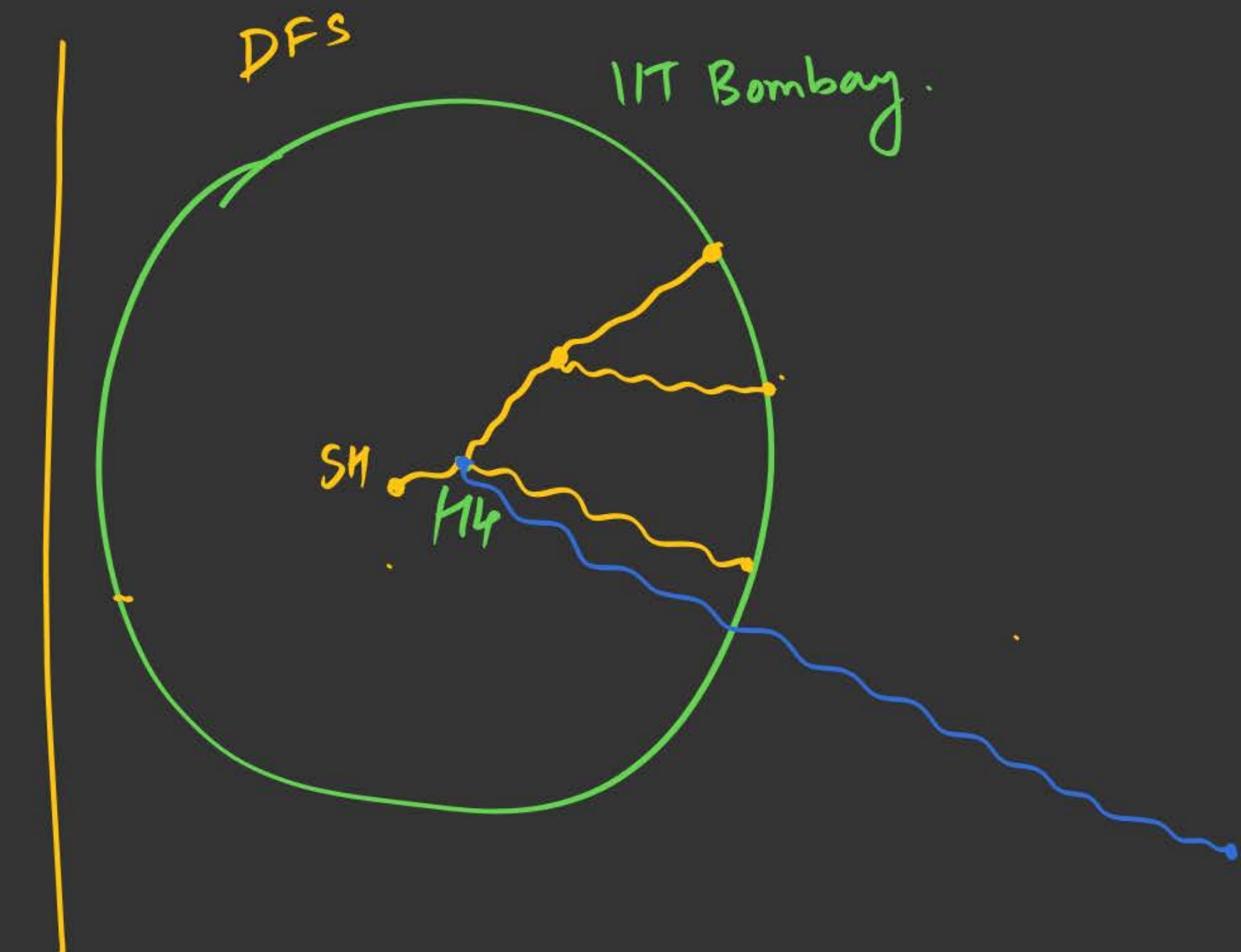
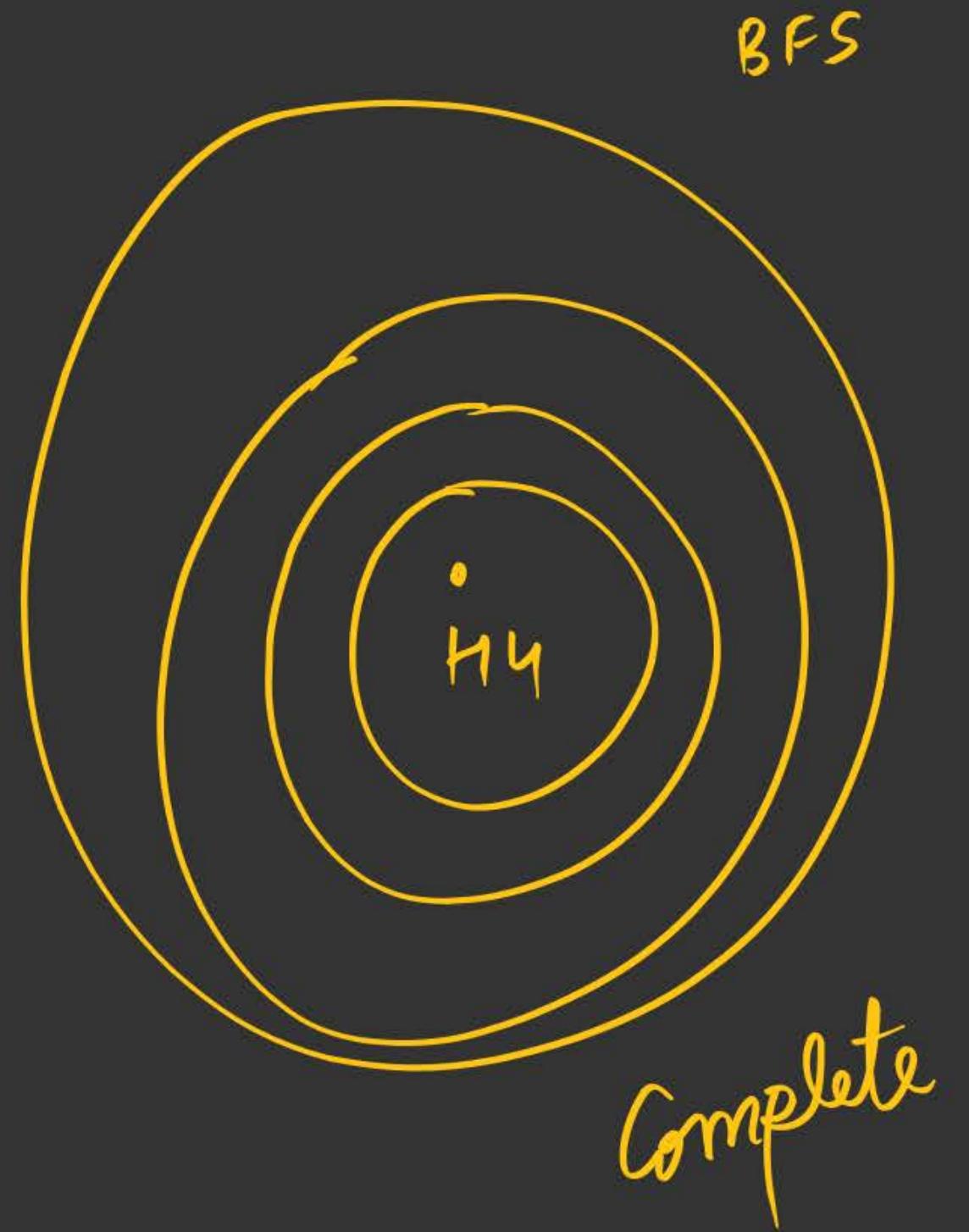
Goal

DFS Spanning Tree

(Alg)



Visited Log [ABSCDEH]
Goal



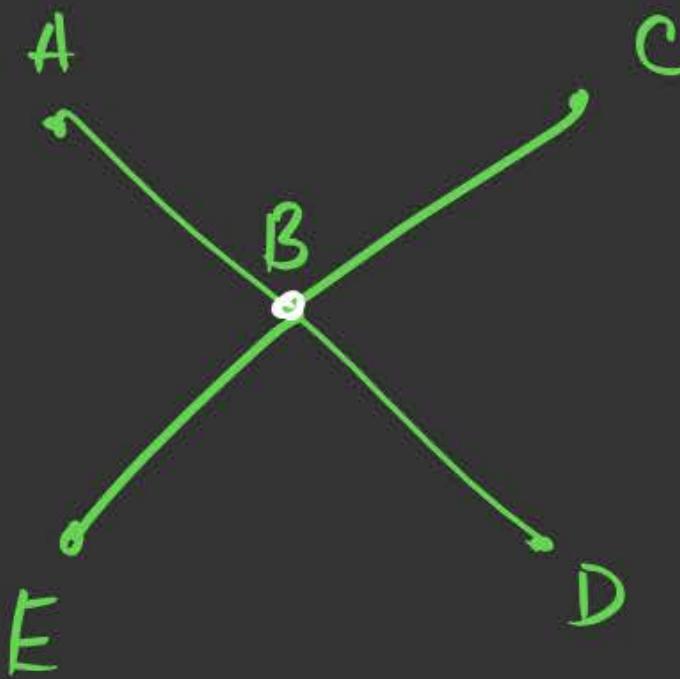
* Properties of DFS

Not
Complete

- ↳ 1) Not optimal search algo
- 2) when Search Space is infinite, then it might not give soln, even if it exists.

(8)

How many ^{diff} possible BFS orderings?



Total orderings

$$= 24 + 24$$

$$= \boxed{48} \checkmark$$

A $\frac{B}{\uparrow} \frac{3 \times 2 \times 1}{\uparrow \uparrow \uparrow}$
Start at A $\xrightarrow{\hspace{1cm}} 6$
" C $\xrightarrow{\hspace{1cm}} 6$
D $\xrightarrow{\hspace{1cm}} 6$
E $\xrightarrow{\hspace{1cm}} 6$

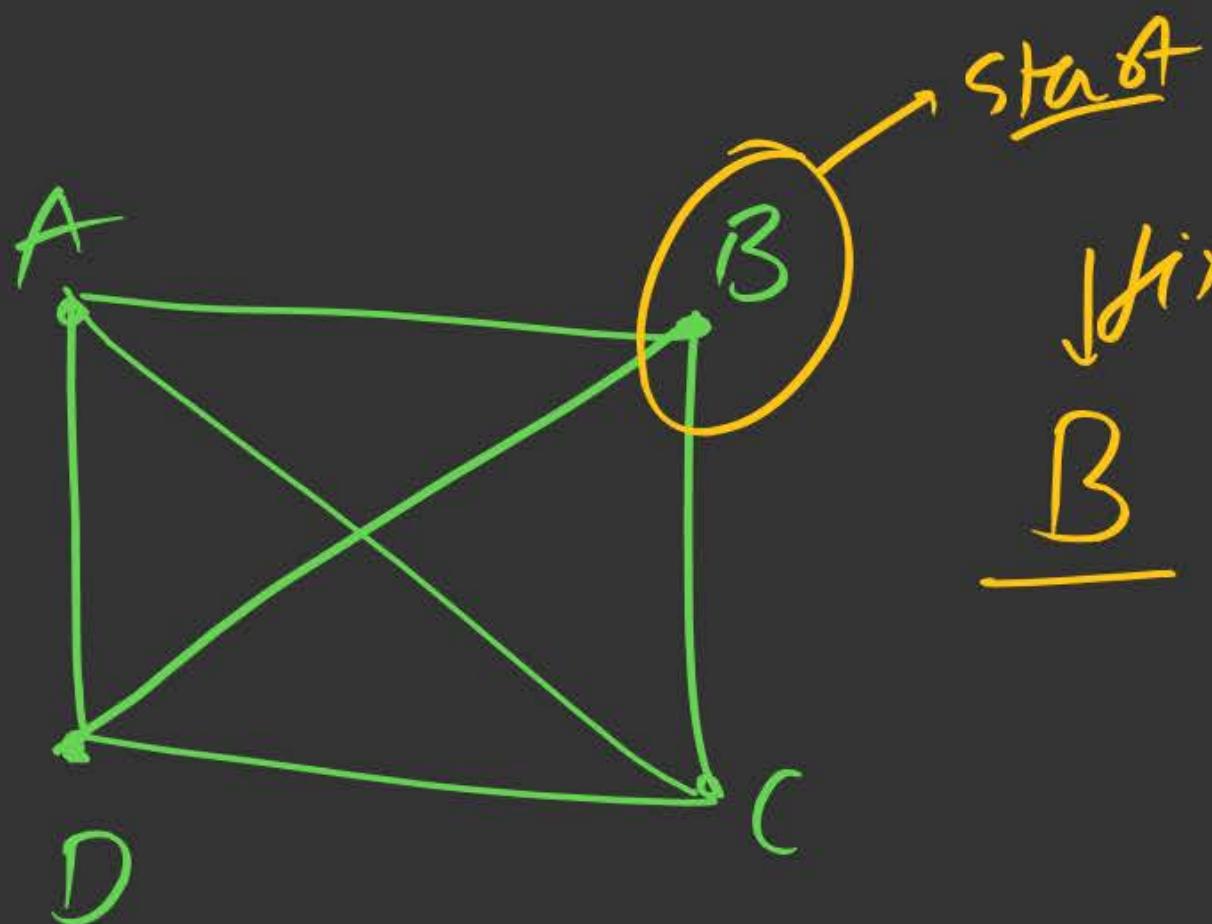
$$6 \times 4 = \underline{\underline{24}}$$

75%

Start at B: $\frac{B}{\uparrow} \frac{4 \times}{\uparrow} \frac{3 \times 2 \times 1}{\uparrow \uparrow \uparrow} \Rightarrow \underline{\underline{24}}$

(Q.2) Given a Complete graph with 4 vertices.

How many off BFS orderings are possible from a given starting node?



fix

$$\frac{B}{\underline{\underline{3 \times 2 \times 1}}} = \underline{\underline{6}}$$

Part 2 :- Starting node not fixed.

→ 24

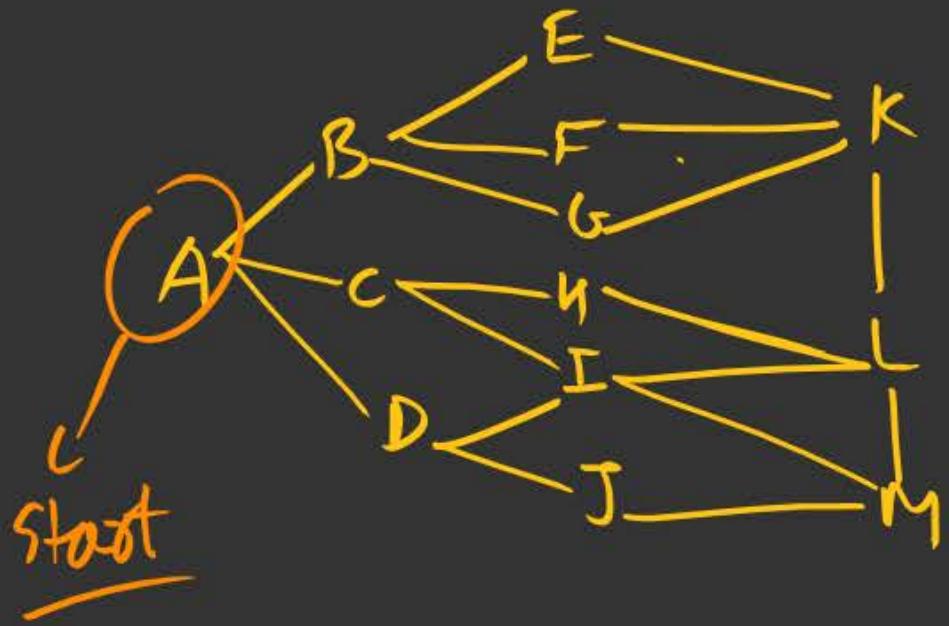
In General :- Complete Graph (K_n)

No. of diff BFS orderings = $n!$

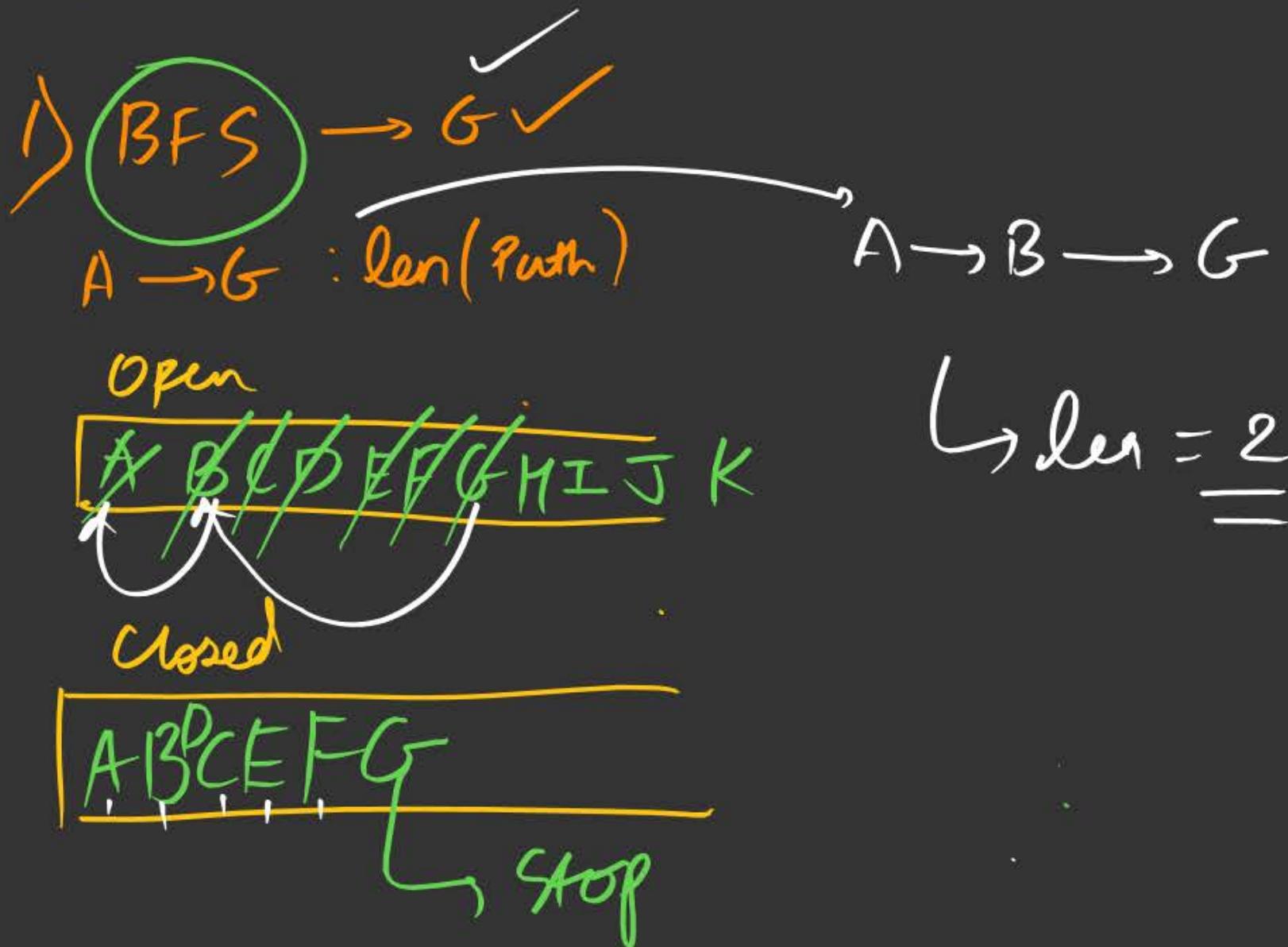
$$n=4 \rightarrow 4! = 24$$

$$n=5 \rightarrow 5! = 120$$

(Q) Alphabetical



Goal: G and I.



②

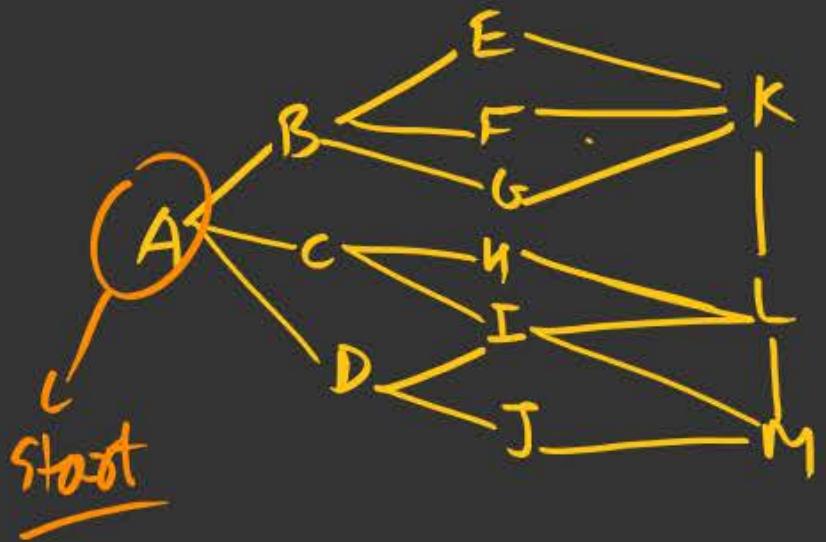
DFS

:

Goal: G & I

(G) ✓

(Q) Alphabetic



DFS: $A \rightarrow G$



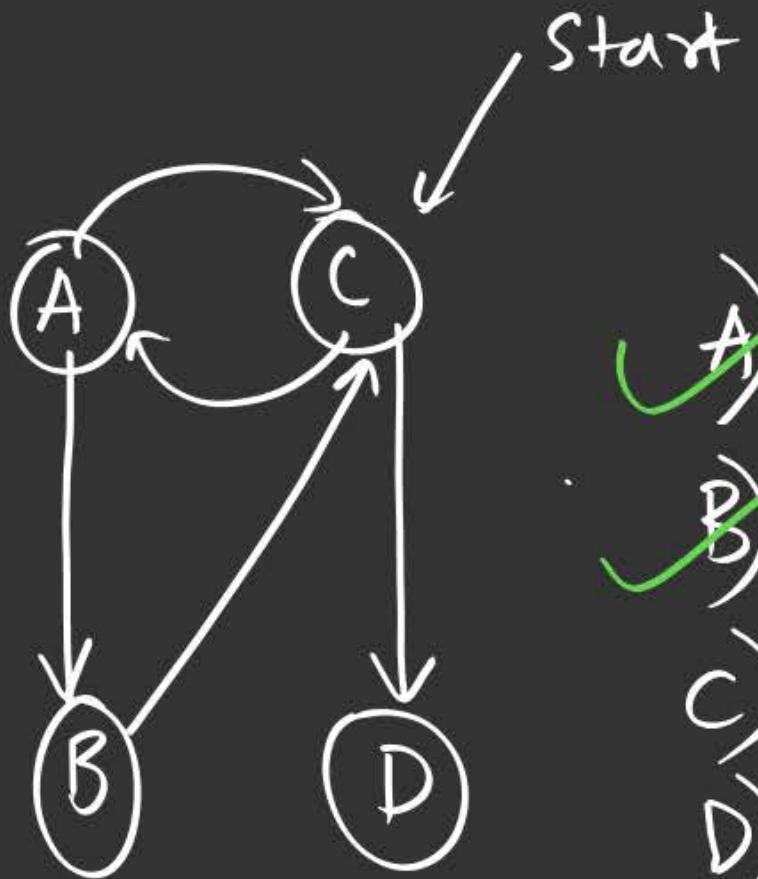
Closed

ABEKF.F.G

goal



(Q)



which is correct
BFS?

A) CADB

B) CDAB

C) CBAD X

D) CDBA X

CADB

A) CADB

B) CDAB

D) CD~~B~~A X

C) CBAD X



THANK - YOU



DS & AI ENGINEERING



Artificial Intelligence

Un-Informed search

Lecture No.- 04

By- Aditya sir



Recap of Previous Lecture



Topic

Topic

Topic

Topic

BFS

DFS



Topics to be Covered



Topic

Topic

Topic

Questions on BFS & DFS

TC & SC of BFS & DFS

DLS



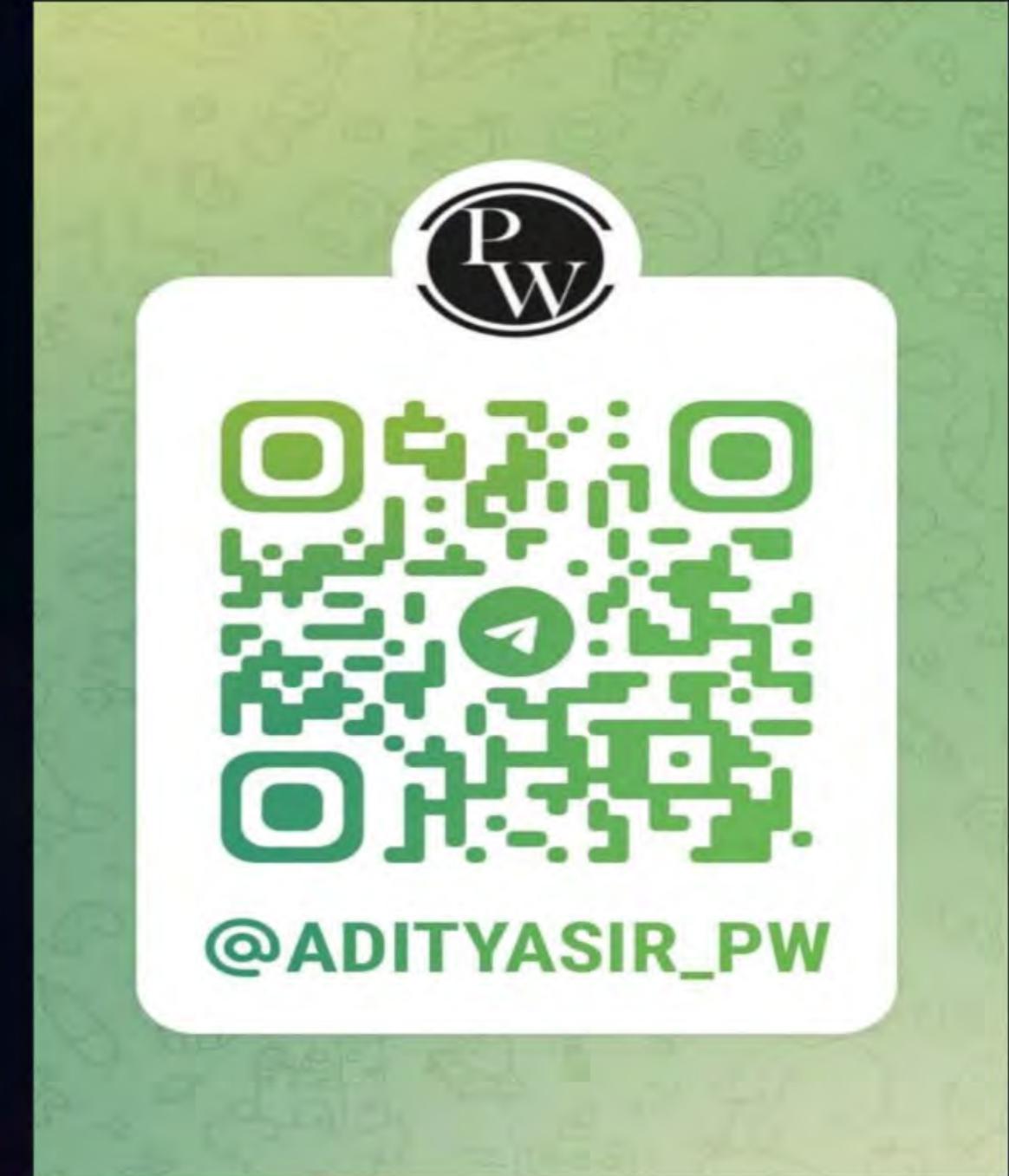
About Aditya Jain sir

1. Appeared for GATE during BTech and secured AIR 60 in GATE in very first attempt - City topper
2. Represented college as the first Google DSC Ambassador.
3. The only student from the batch to secure an internship at Amazon. (9+ CGPA)
4. Had offer from IIT Bombay and IISc Bangalore to join the Masters program
5. Joined IIT Bombay for my 2 year Masters program, specialization in Data Science
6. Published multiple research papers in well known conferences along with the team
7. Received the prestigious excellence in Research award from IIT Bombay for my Masters thesis in ML
8. Completed my Masters with an overall GPA of 9.36/10
9. Joined Dream11 as a Data Scientist
10. Have mentored 15,000+ students & working professionals in field of Data Science and Analytics
11. Have been mentoring & teaching GATE aspirants to secure a great rank in limited time
12. Have got around 27.5K followers on LinkedIn where I share my insights and guide students and professionals.





Telegram



Telegram Link for Aditya Jain sir:
https://t.me/AdityaSir_PW



Topic : Uninformed Search

- Depth-First Search (DFS) is a strategy for traversing or searching tree or graph structures.
- In the context of artificial intelligence (AI), DFS is often used for exploring possible states in search problems, finding solutions in game playing, planning, and more.



Topic : Uninformed Search

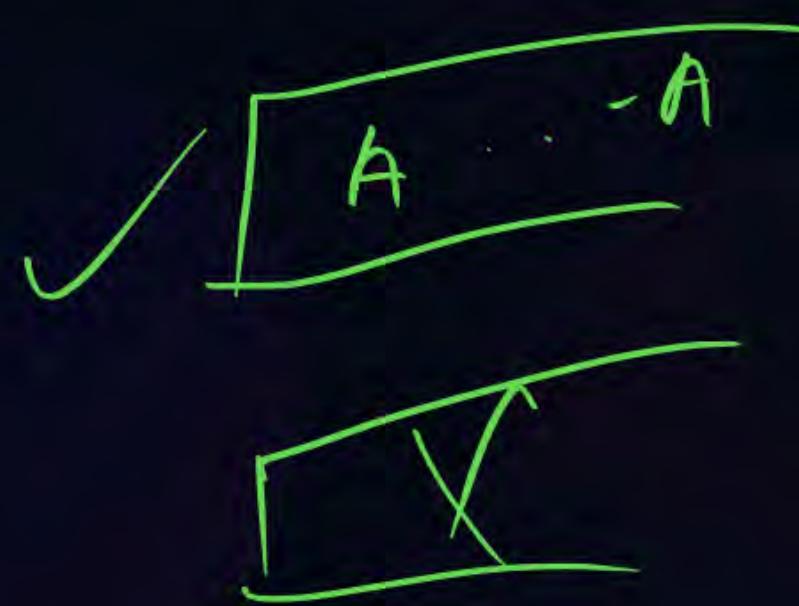
Step-1 \Rightarrow Two list visited, stack

- Start with starting node and insert it into stack.

Step-2 \Rightarrow Now remove the last node in stack, mark it as visited and now insert neighbour node into stack. LIFO

Step-3 \Rightarrow

- Only check that no visited node is to be inserted.
- Repeat step 2,3 till goal node is reached.



DFS
Stack \rightarrow open
visited \rightarrow closed

BFS
Open \rightarrow open
visited \rightarrow closed.



Topic : Uninformed Search





Topic : Uninformed Search

Depth first Search

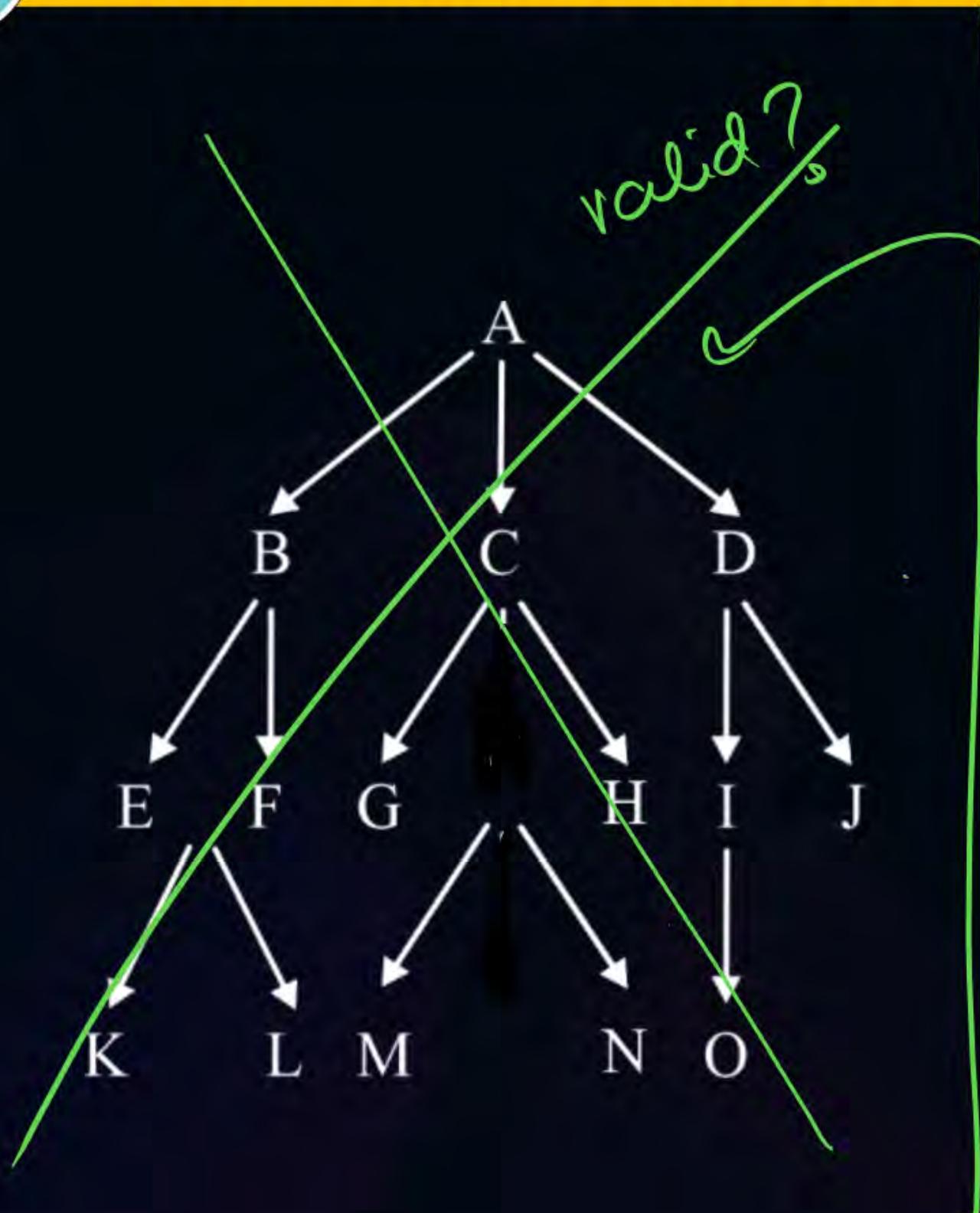
- Key Concepts
- **Traversal Method:** DFS explores as far as possible along **each** branch before backtracking.
- **LIFO Structure:** Uses a stack (Last-In-First-Out) data structure, either implicitly through recursion or explicitly.
- **Completeness:** DFS is not guaranteed to find a solution if one exists, especially in infinite search spaces.)

Not Complete





Topic : Uninformed Search



given

Graph: adj list

A → [B, C, D]

B → [A, E, F, C]

C → [A, B, G, H, D]

D → [A, C, I, J]

E → [B, K, L]

F → [B, G, M]

G → [C, F, N]

H → [C, I, O]

I → [D, H]

J → [D, O]

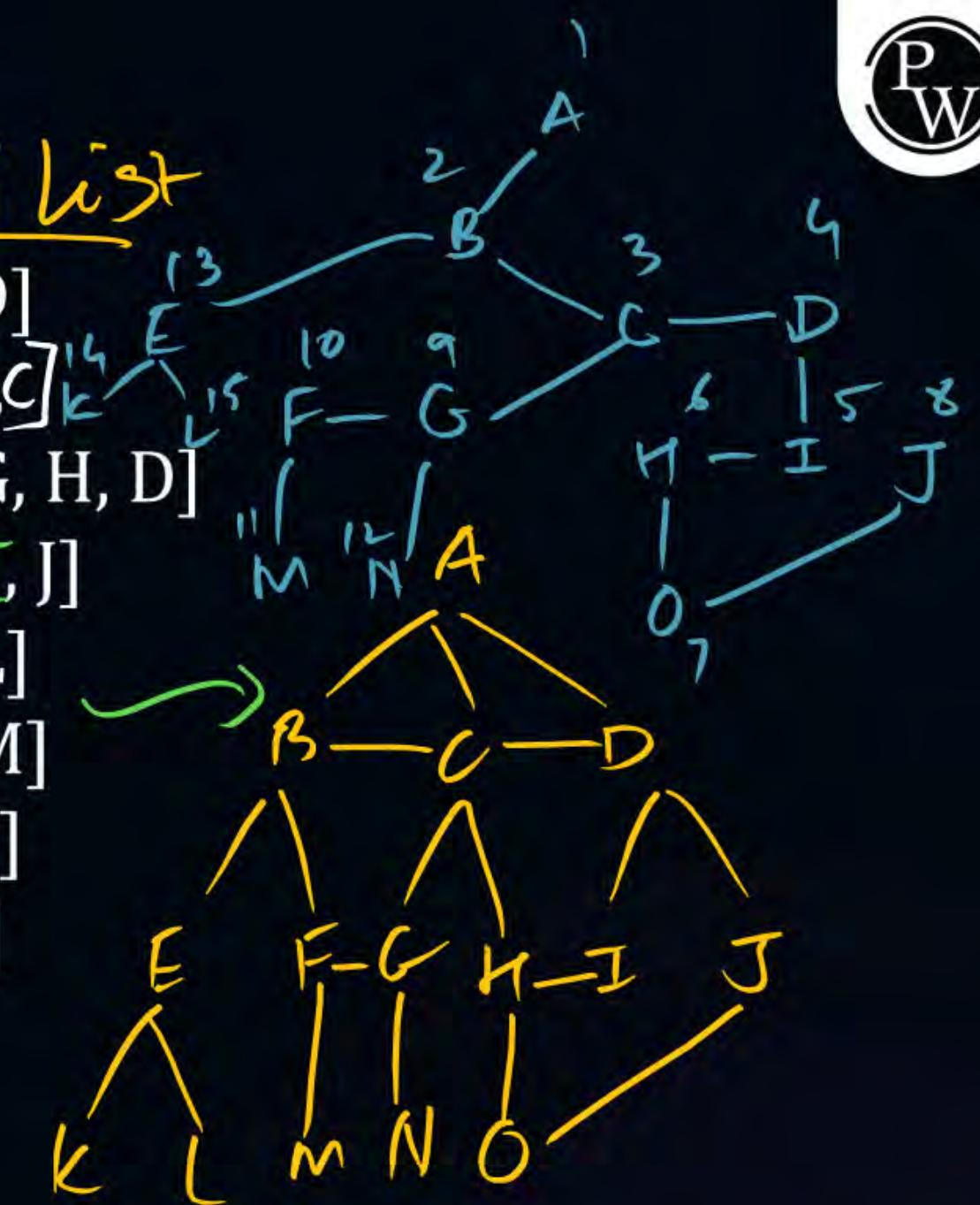
K → [E]

L → [E]

M → [F]

N → [G]

O → [H, J]



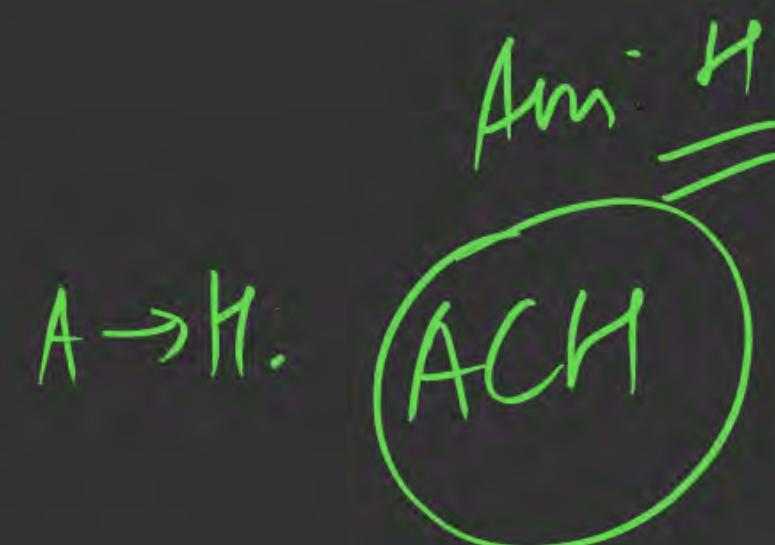
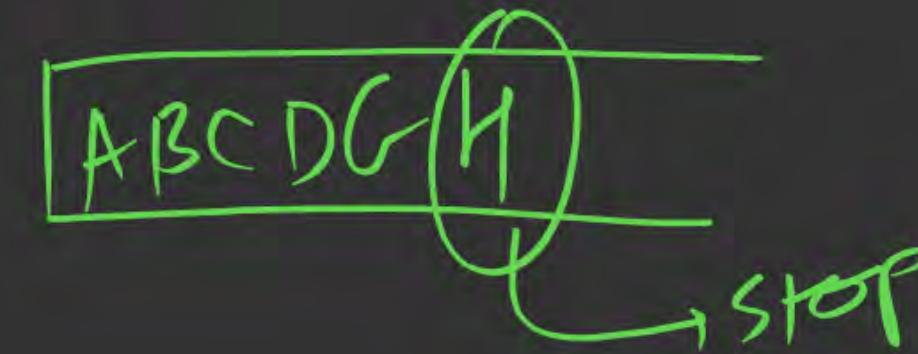
(Q) BFS start at A.
Two Goal states : H & I. which will be reached
First? (alpha order)



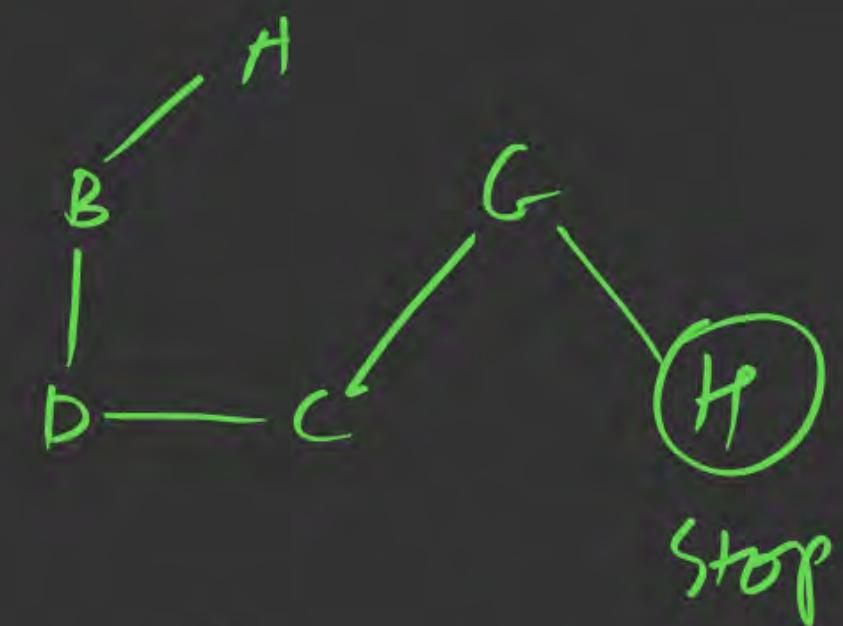
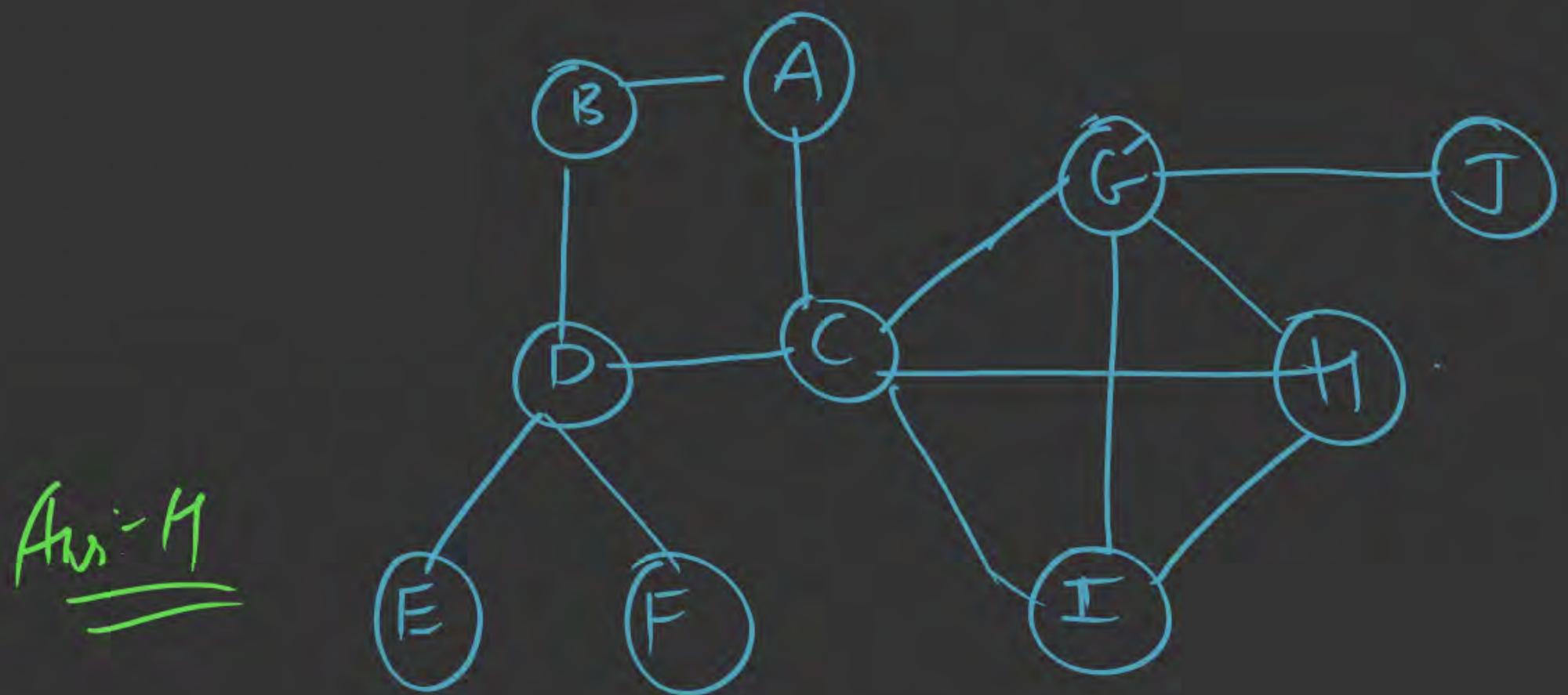
BFS : open

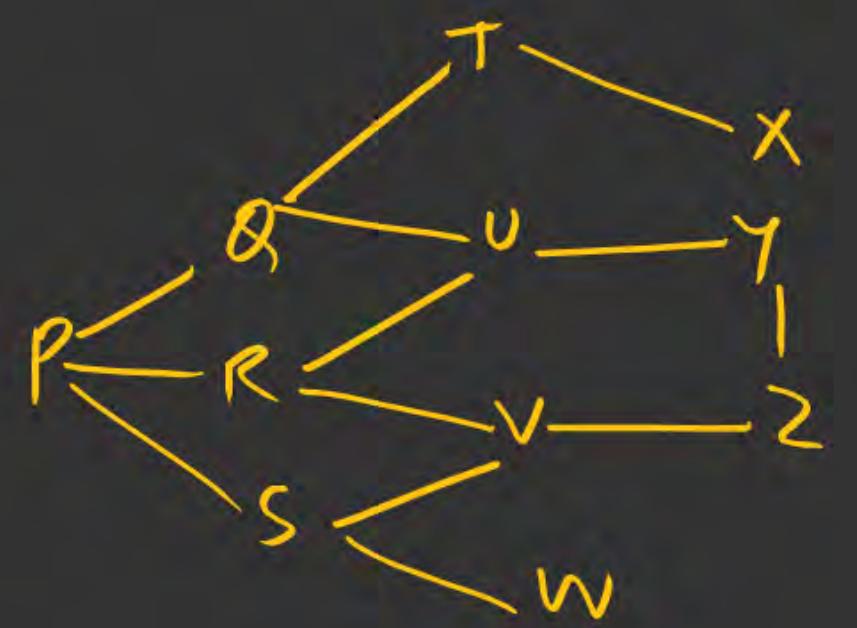


Closed



(Q) DFS start at A.
Two goal states : H & I. which will be reached first?





Ans - X

Start P:

Goal x, z.

which First?

↗ BFS

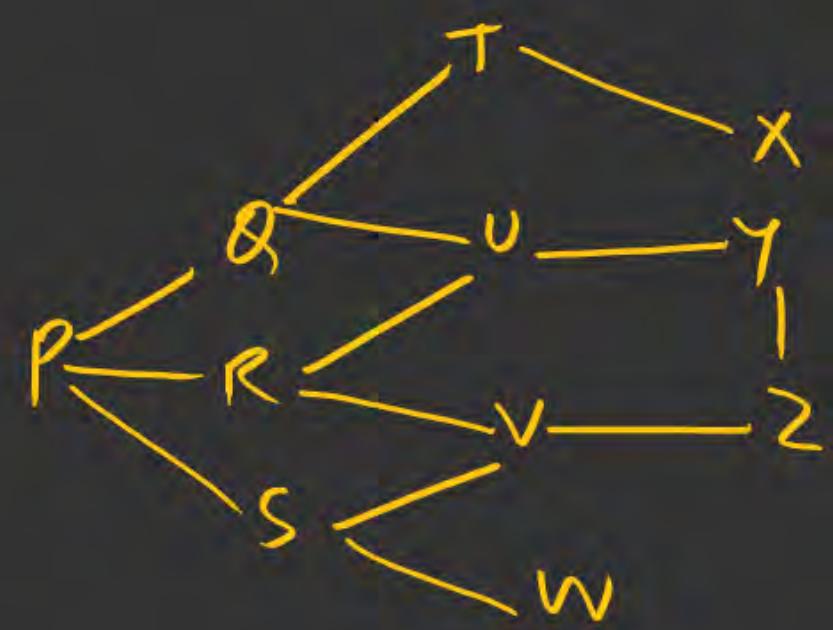
open

~~P Q R S T U V W X Y Z~~

Closed

P Q R S T U V W(X)

goal

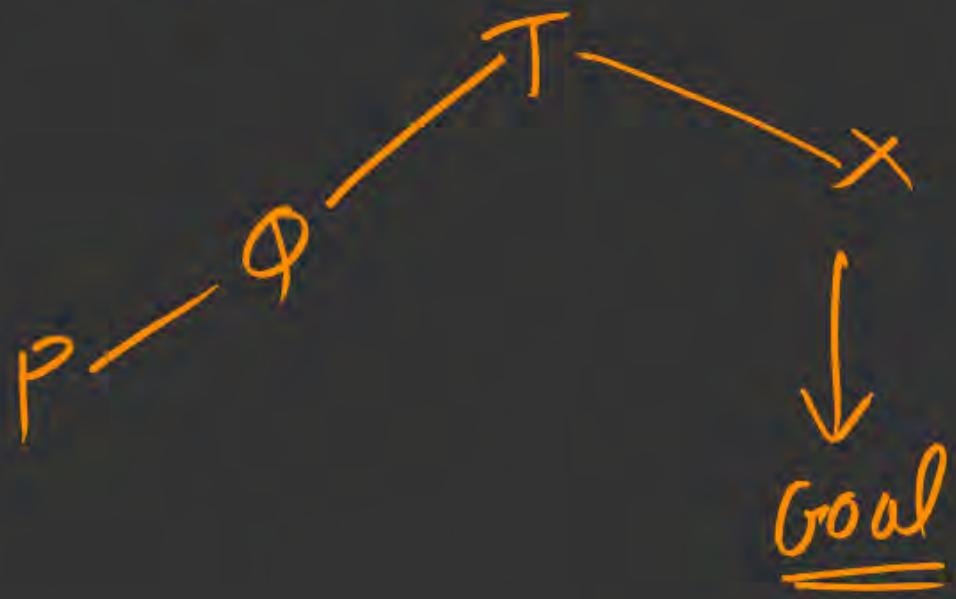


Start P:

Goal x, z.

which First?

2) DFS



1) BFS

→ optimal
(for non-wt graph)

→ complete

2) DFS

→ may or may not
optimal.

→ not complete
(∞ search space)

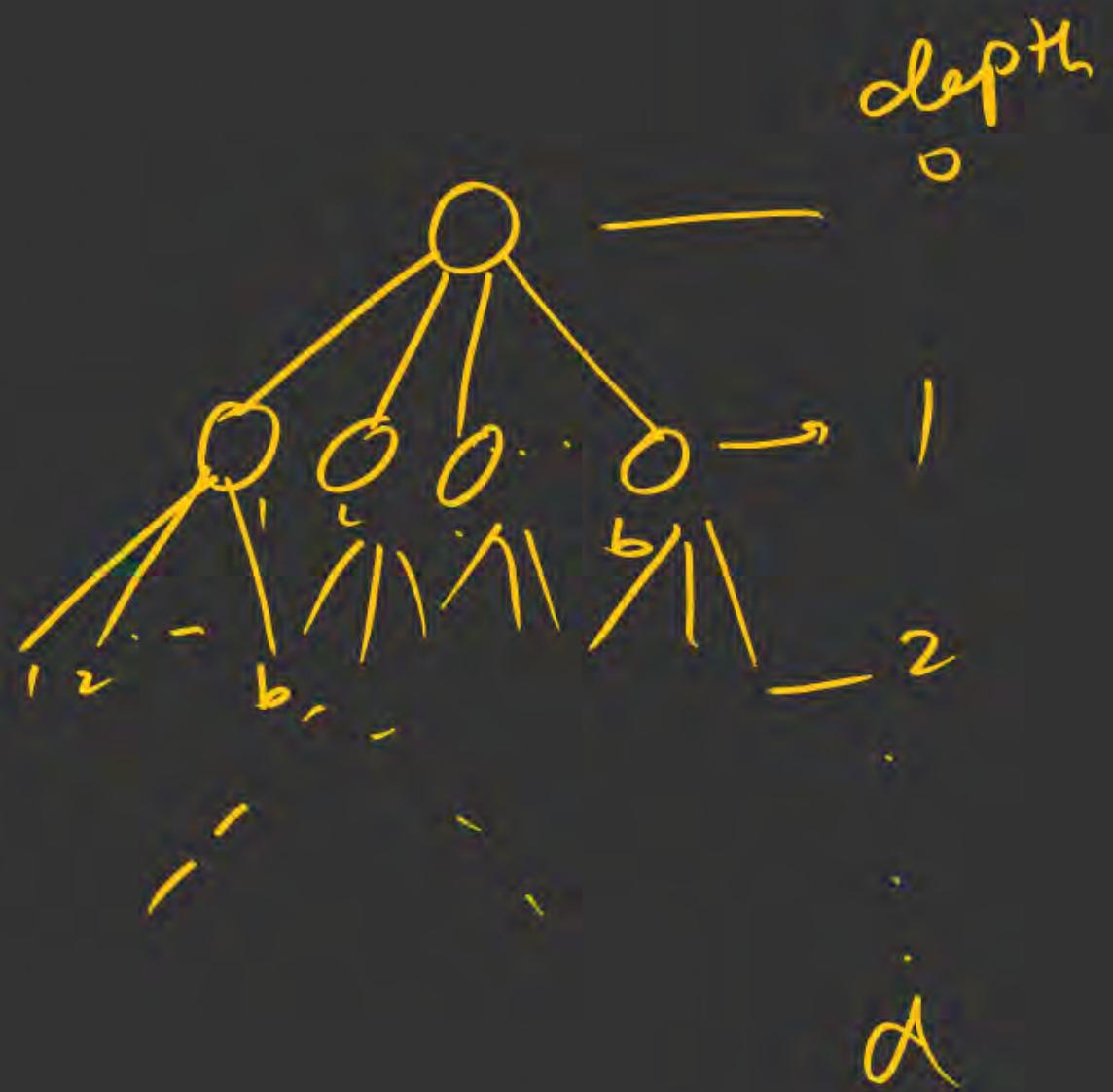
TC: \rightarrow no. of nodes to be visited (Closed list)
before finding goal.

SC: No. of nodes stored in
Open list before finding goal

Tree:

depth $\rightarrow d$

branching factor $\rightarrow b$

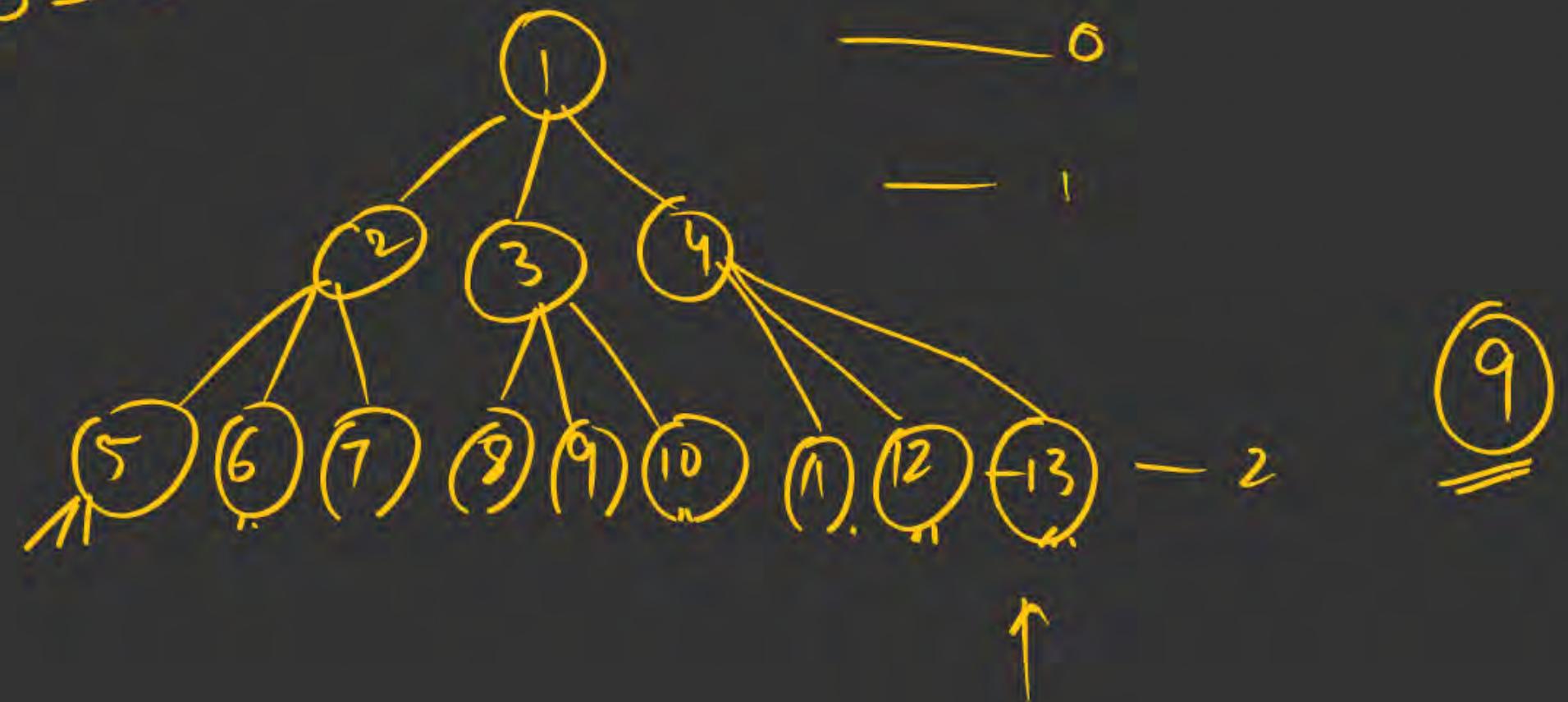


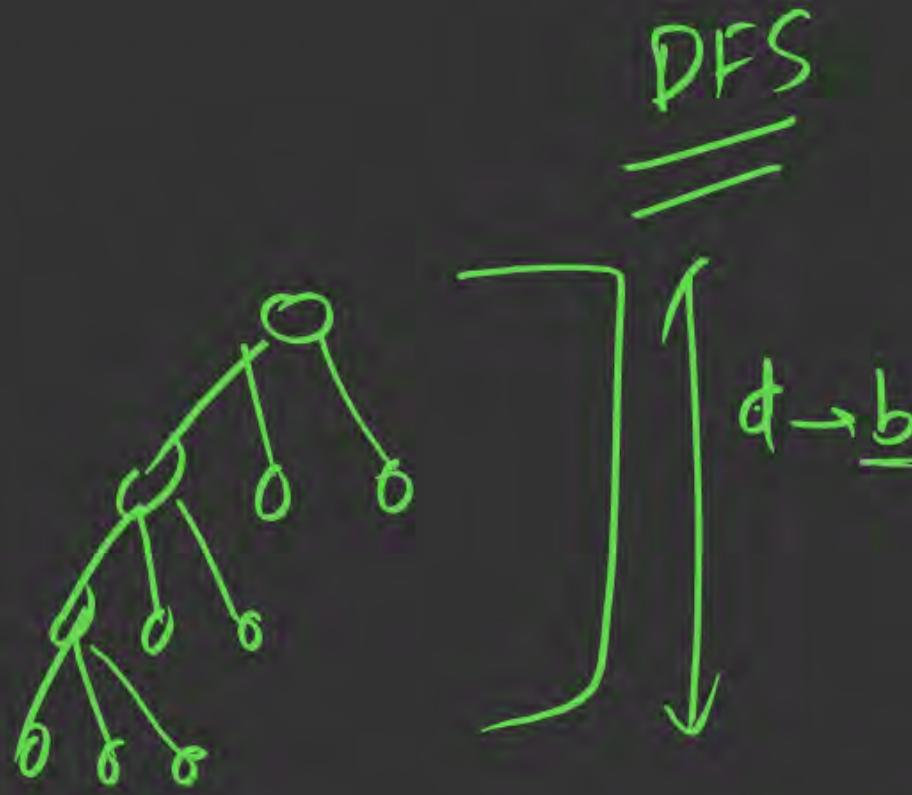
BFS

eg:- $d = 2$
 $b = 3$

$$\frac{SC: O(b^d)}{TC: O(b^d)}$$

Total nodes: $O(b^d)$

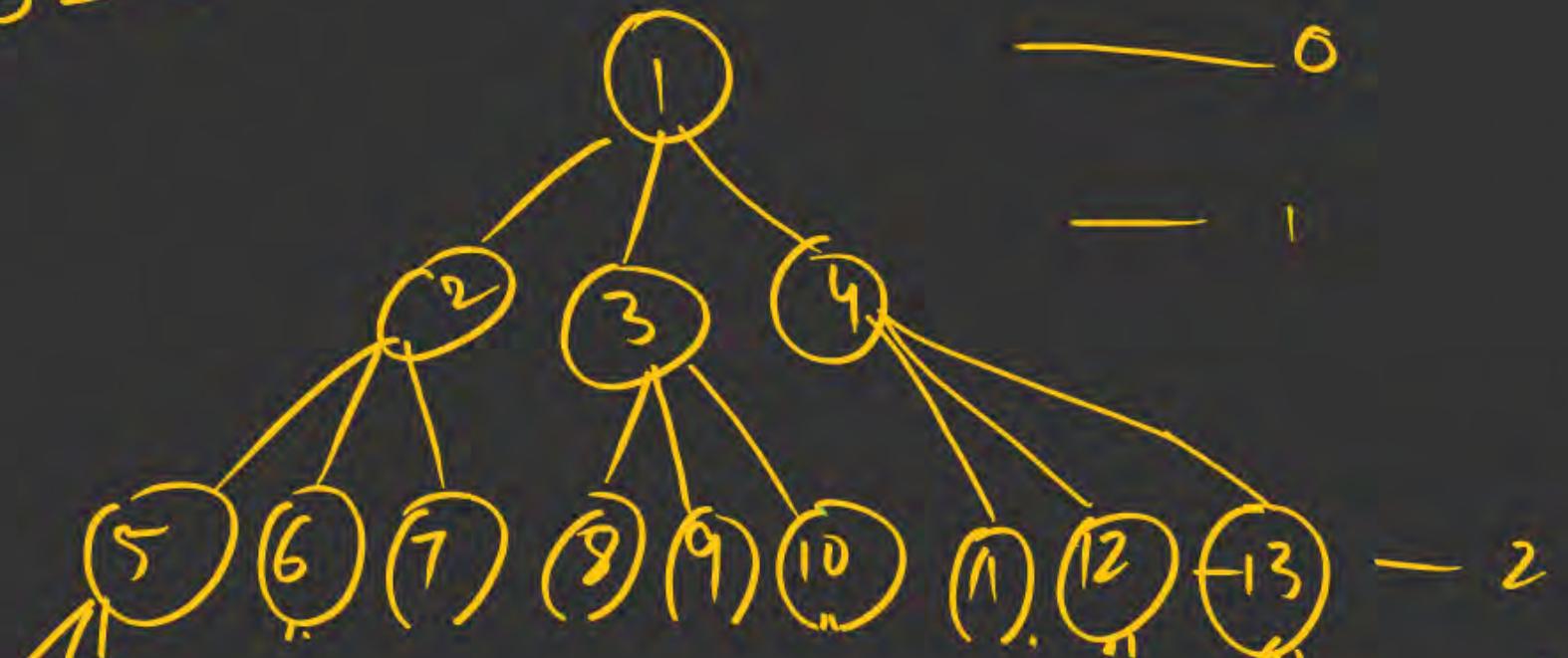




$$d \rightarrow b$$

143 / 765

eg: $d = 2$
 $b = 3$



Total nodes: $O(b^d)$

SC: $O(b * d)$

TC: $O(b^d)$

Summary:

	BFS	DFS
TC	$O(b^d)$	$O(b^d)$
SC	$O(b^d)$	$O(b \times d)$





Topic : Uninformed Search

Depth first Search

Advantage:

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

Disadvantage:

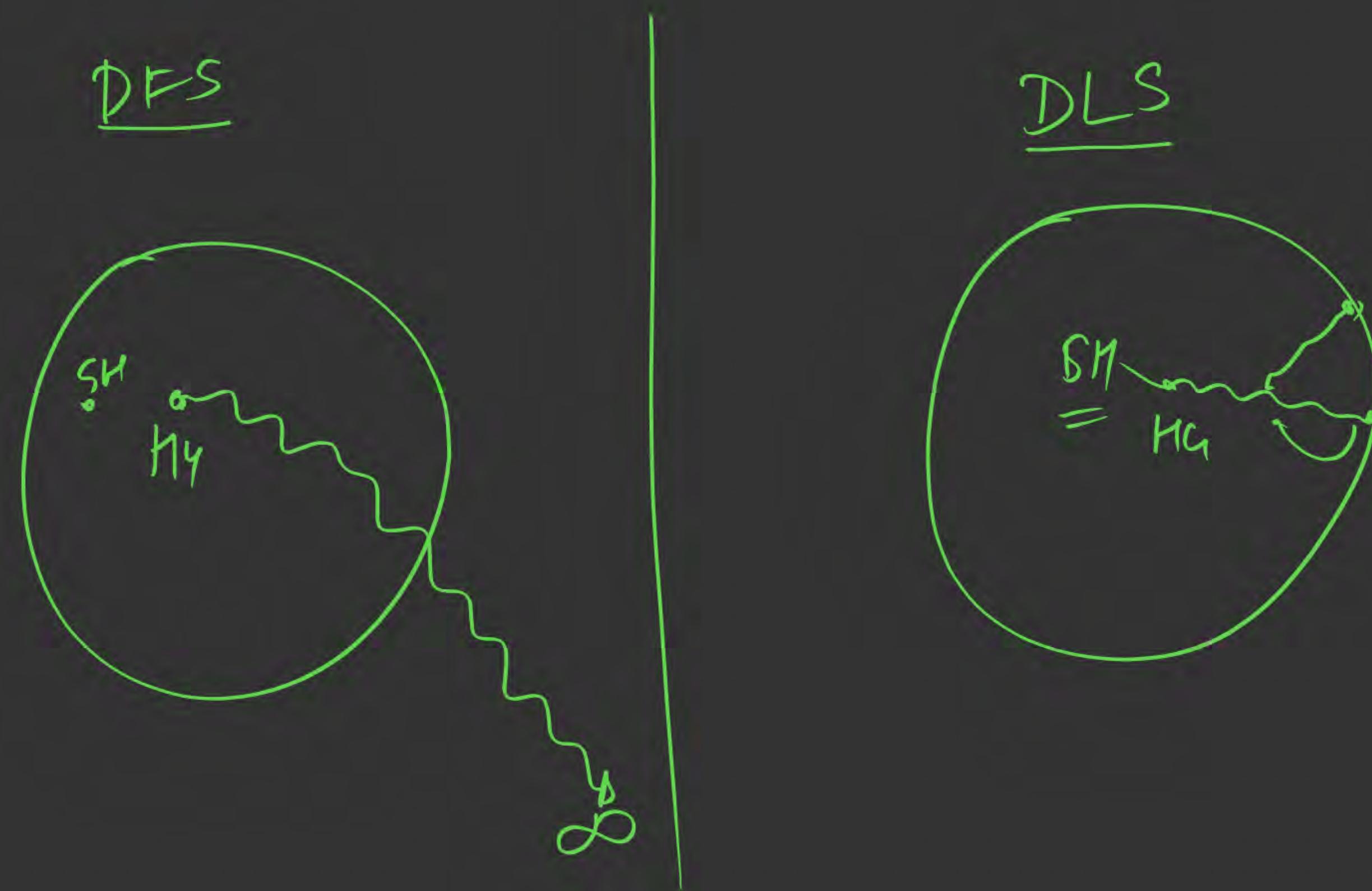
- There is the possibility that many states keep re-occurring, and there is no guarantee of finding the solution.
- DFS algorithm goes for deep down searching and sometime it may go to the infinite loop.



Topic : Uninformed Search

Depth Limited Search (DLS)

- Depth Limited Search is a modified version of DFS that imposes a limit on the depth of the search. This means that the algorithm will only explore nodes up to a certain depth, effectively preventing it from going down excessively deep paths that are unlikely to lead to the goal. By setting a maximum depth limit, DLS aims to improve efficiency and ensure more manageable search times.





Topic : Uninformed Search

Depth Limited Search (DLS)

- How Depth Limited Search Works
 - **Initialization:** Begin at the root node with a specified depth limit.
 - **Exploration:** Traverse the tree or graph, exploring each node's children.
 - **Depth Check:** If the current depth exceeds the set limit, stop exploring that path and backtrack.
 - **Goal Check:** If the goal node is found within the depth limit, the search is successful.
 - **Backtracking:** If the search reaches the depth limit or a leaf node without finding the goal, backtrack and explore other branches.

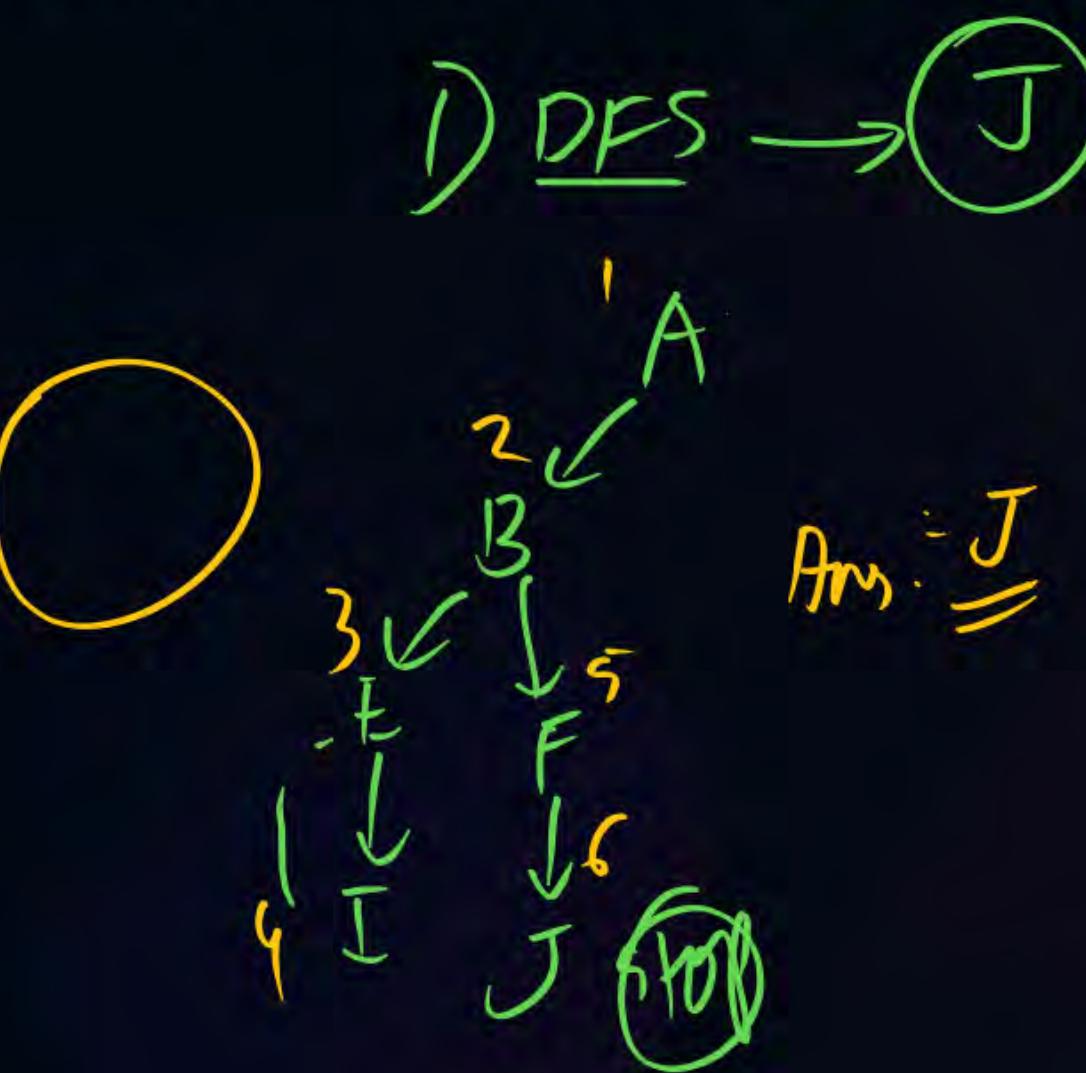


Topic : Uninformed Search

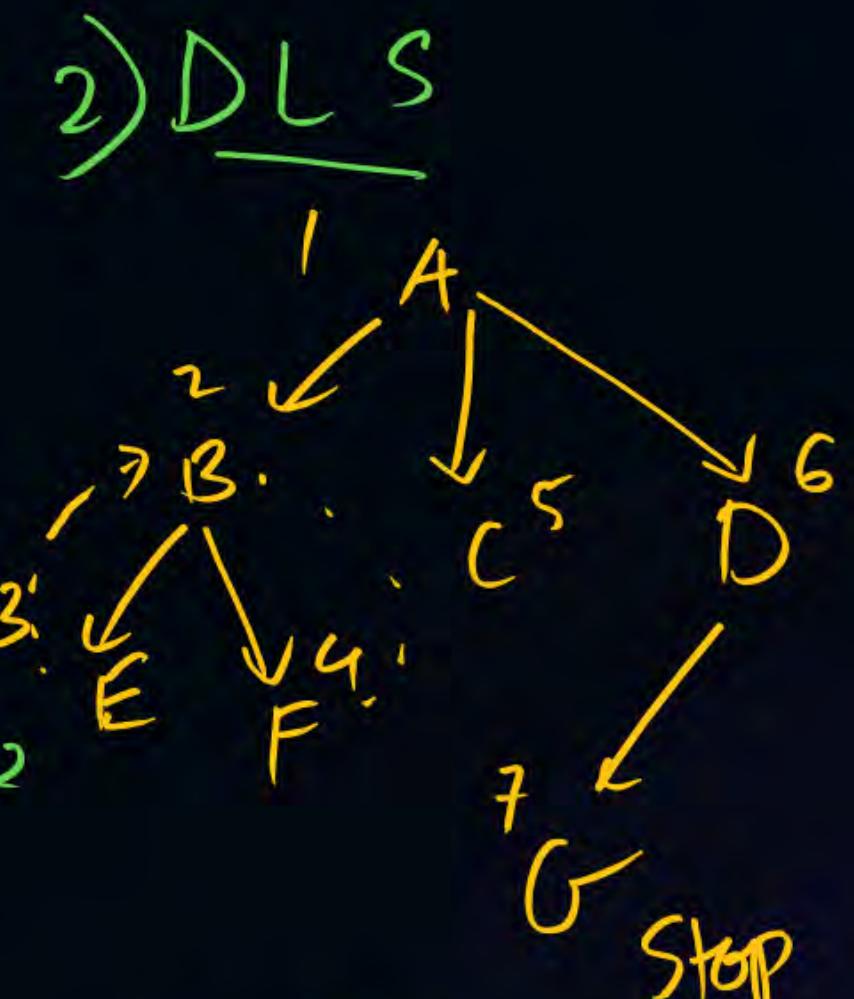
Depth Limited Search (DLS)

Let's define our goal states as nodes G and J.

Apply DFS and DLS with depth limit 2. start at A.



(Root → $d=0$)



Ans: G



DLS



1) Not optimal

2) Not Complete.

(if goal is beyond the depth limit, then it will be missed / skipped).



Topic : Uninformed Search

Depth Limited Search (DLS)

- Performance Measures
- Completeness: The DLS is a complete algorithm in general except the case when the goal node is the shallowest node, and it is beyond the depth limit, i.e. $l < d$, and in this case, we never reach the goal node.
- Optimality: The DLS is a non-optimal algorithm since the depth that is chosen can be greater than d ($l > d$). Thus DLS is not optimal if $l > d$ $O(B^l)$
- Time complexity is expressed as: It is similar to the DFS, i.e. ~~$O(B^L)$~~ , where L is the set depth limit
- Space Complexity is expressed as: It is similar to DFS. $O(BL)$, where L is specified depth limit





THANK - YOU

Tomorrow → 11:30 AM



DS & AI ENGINEERING



Artificial Intelligence

Un-Informed search

Lecture No.- 05

By- Aditya sir



Recap of Previous Lecture



Topic

Topic

Topic

Topic

DFS

BFS

DLS

Topics to be Covered

P
W



Topic

Topic

Topic

Questions

IDDFS

UCS



About Aditya Jain sir

1. Appeared for GATE during BTech and secured AIR 60 in GATE in very first attempt - City topper
2. Represented college as the first Google DSC Ambassador.
3. The only student from the batch to secure an internship at Amazon. (9+ CGPA)
4. Had offer from IIT Bombay and IISc Bangalore to join the Masters program
5. Joined IIT Bombay for my 2 year Masters program, specialization in Data Science
6. Published multiple research papers in well known conferences along with the team
7. Received the prestigious excellence in Research award from IIT Bombay for my Masters thesis in ML
8. Completed my Masters with an overall GPA of 9.36/10
9. Joined Dream11 as a Data Scientist
10. Have mentored 15,000+ students & working professionals in field of Data Science and Analytics
11. Have been mentoring & teaching GATE aspirants to secure a great rank in limited time
12. Have got around 27.5K followers on Linkedin where I share my insights and guide students and professionals.





Telegram



Telegram Link for Aditya Jain sir:
https://t.me/AdityaSir_PW



Topic : Uninformed Search

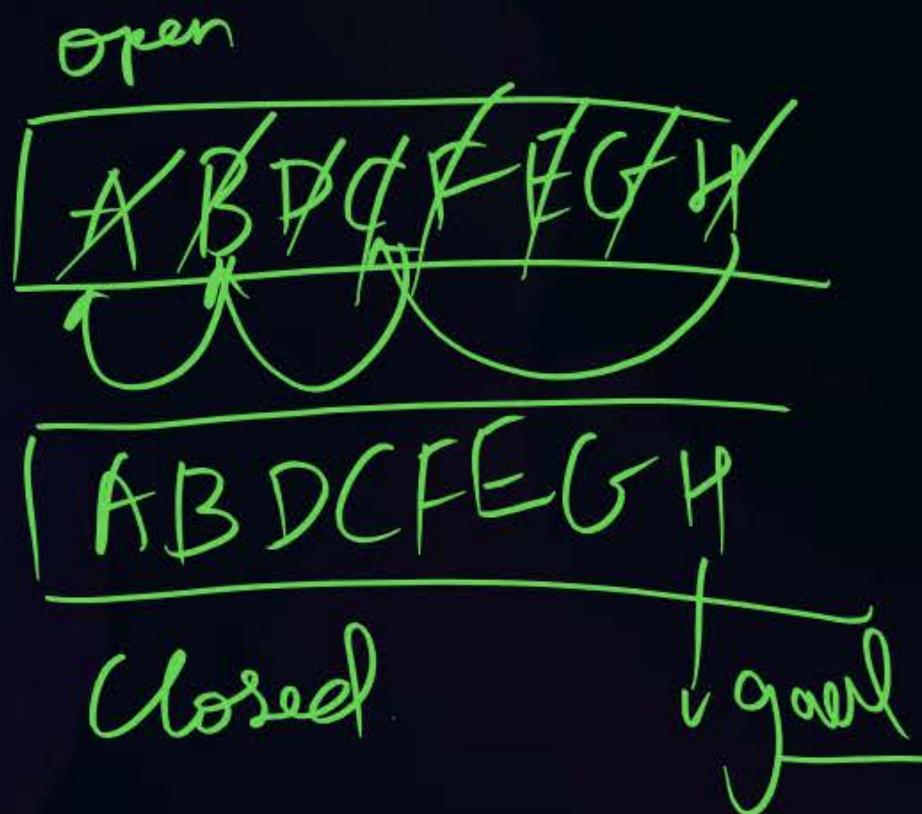


DFS \Rightarrow tree

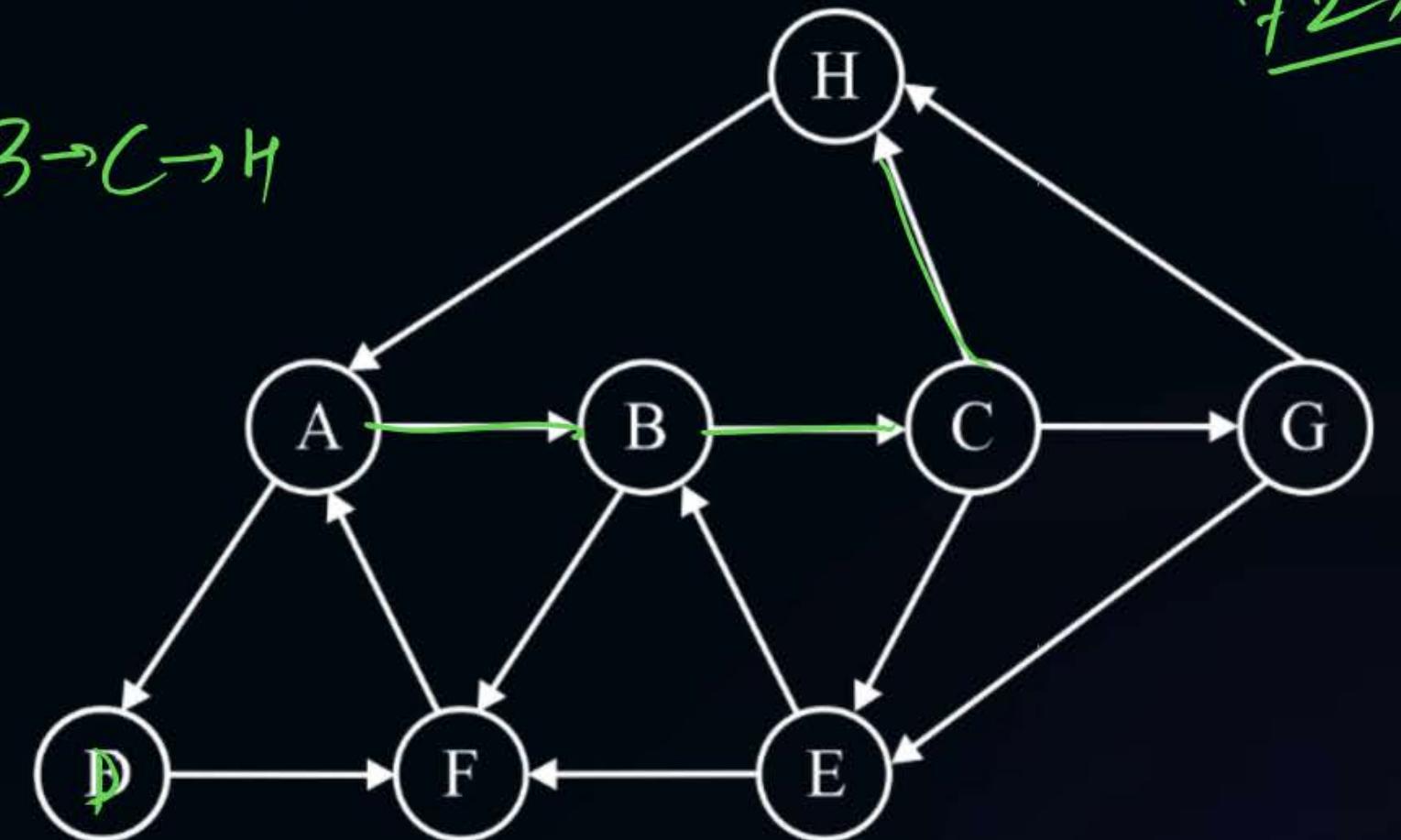
$A \rightarrow H$

\Rightarrow (Alphabetical order)

1) BFS



Path: $A \rightarrow B \rightarrow C \rightarrow H$



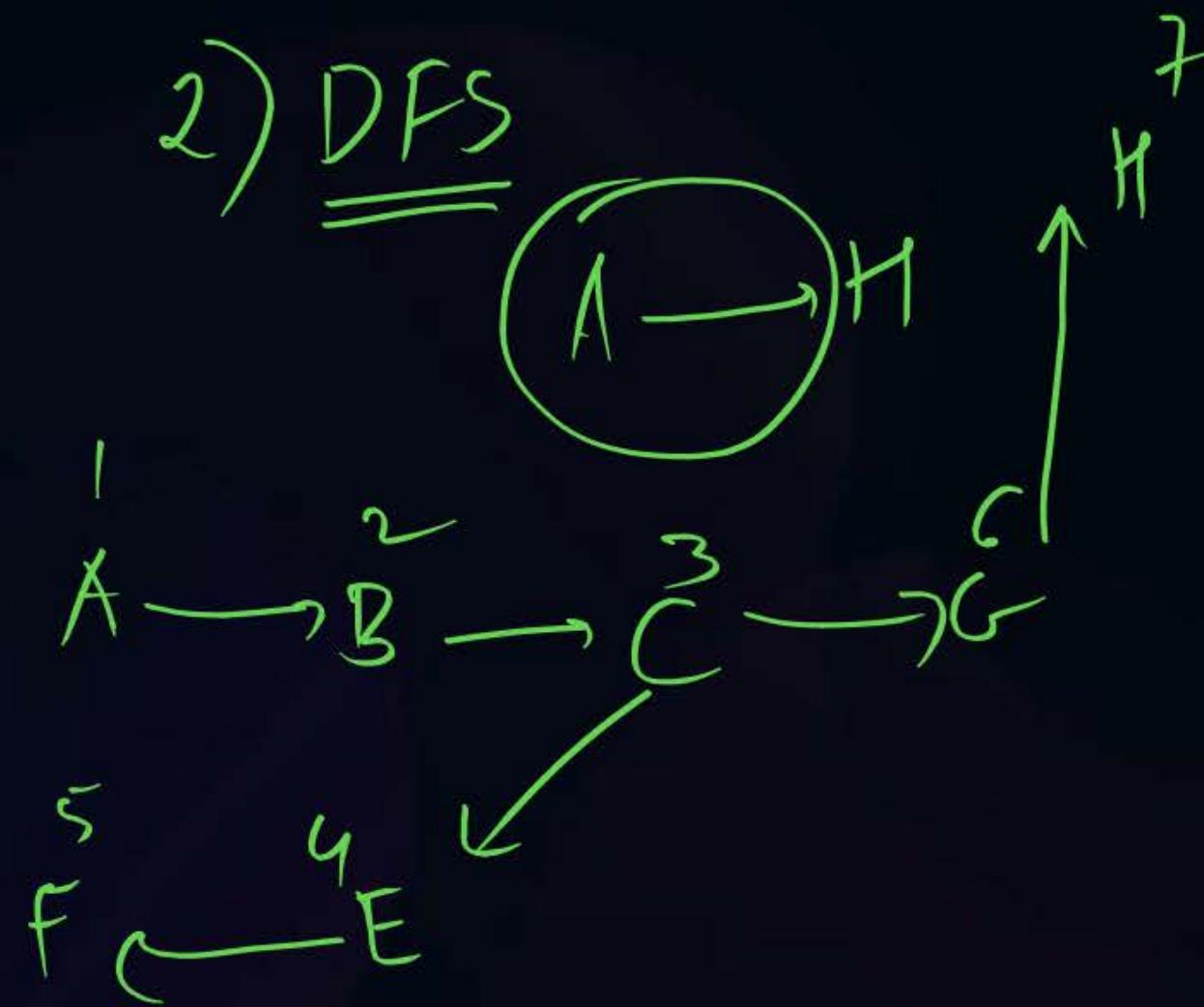


Topic : Uninformed Search

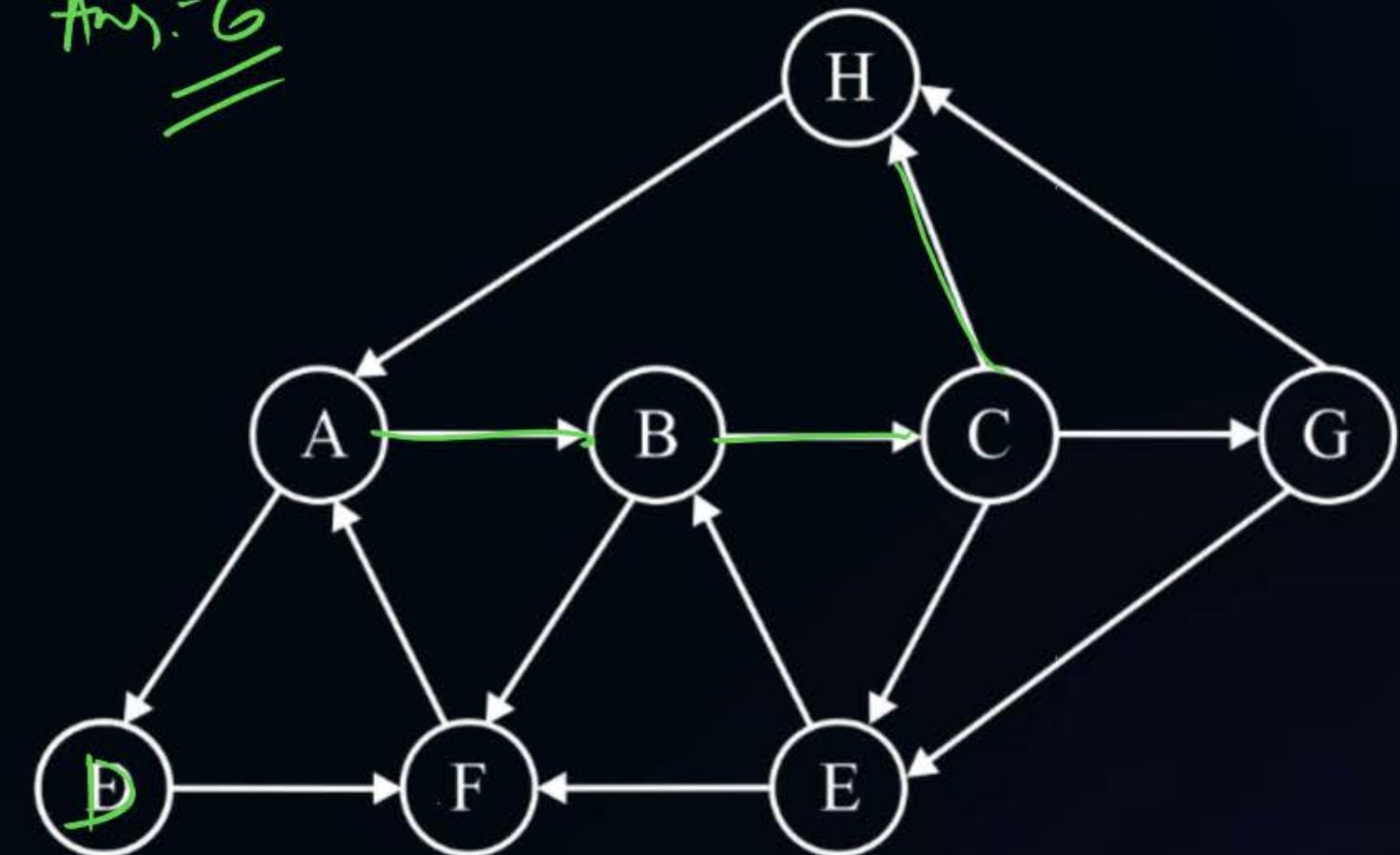
DFS \Rightarrow tree

$A \rightarrow H$

\Rightarrow (Alphabetical order)



Ans:-6

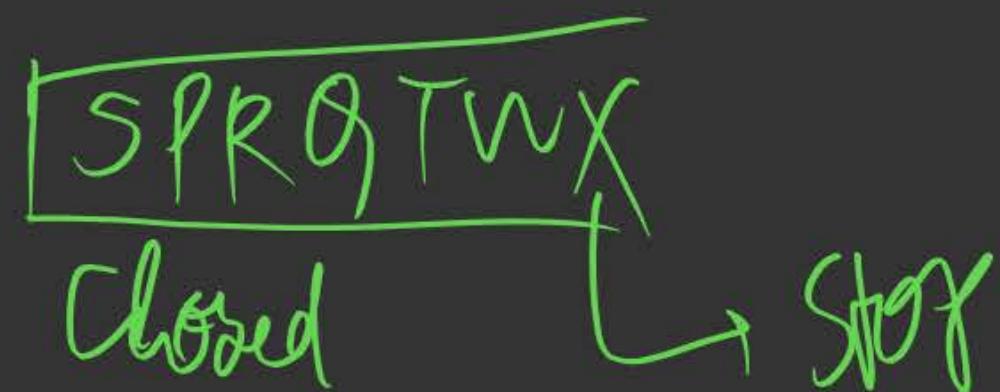
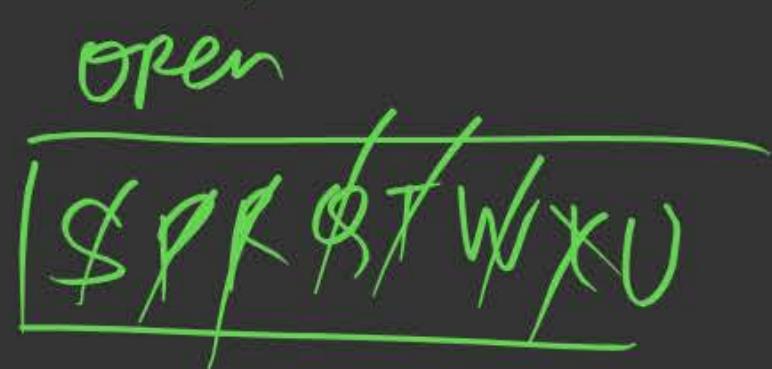
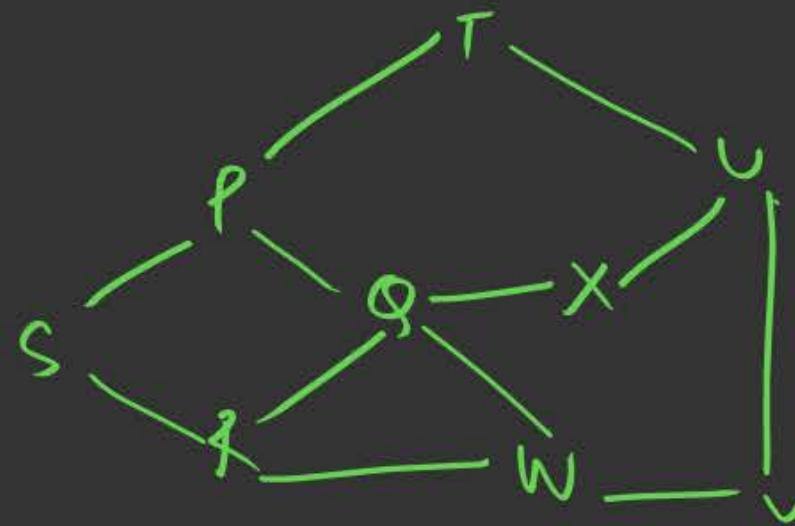


38/

Start: S

Goal(s): U, X

> BFS:

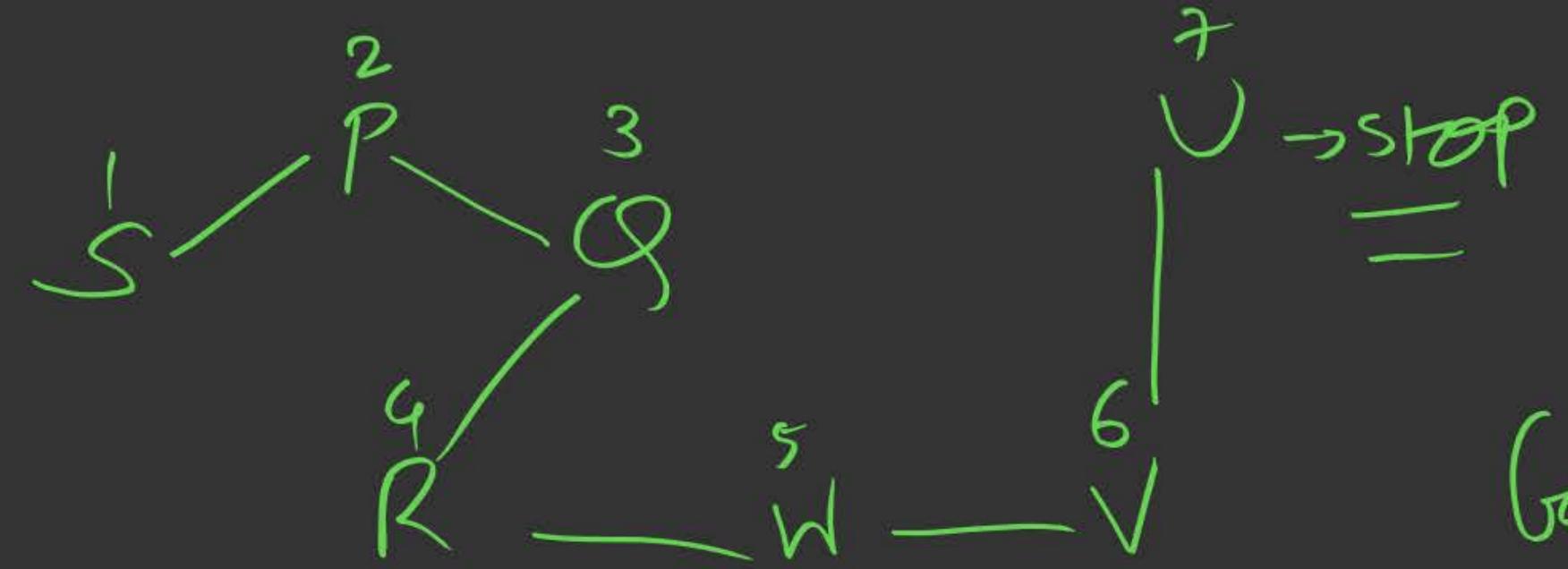
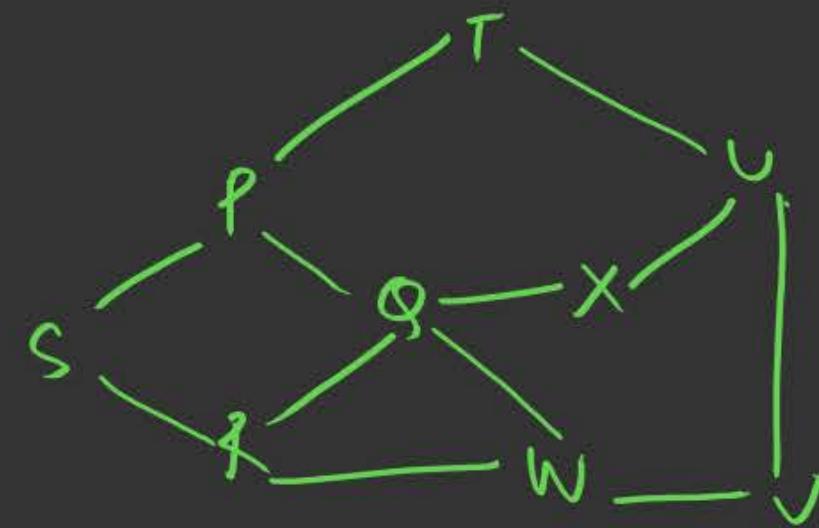


Goal: X

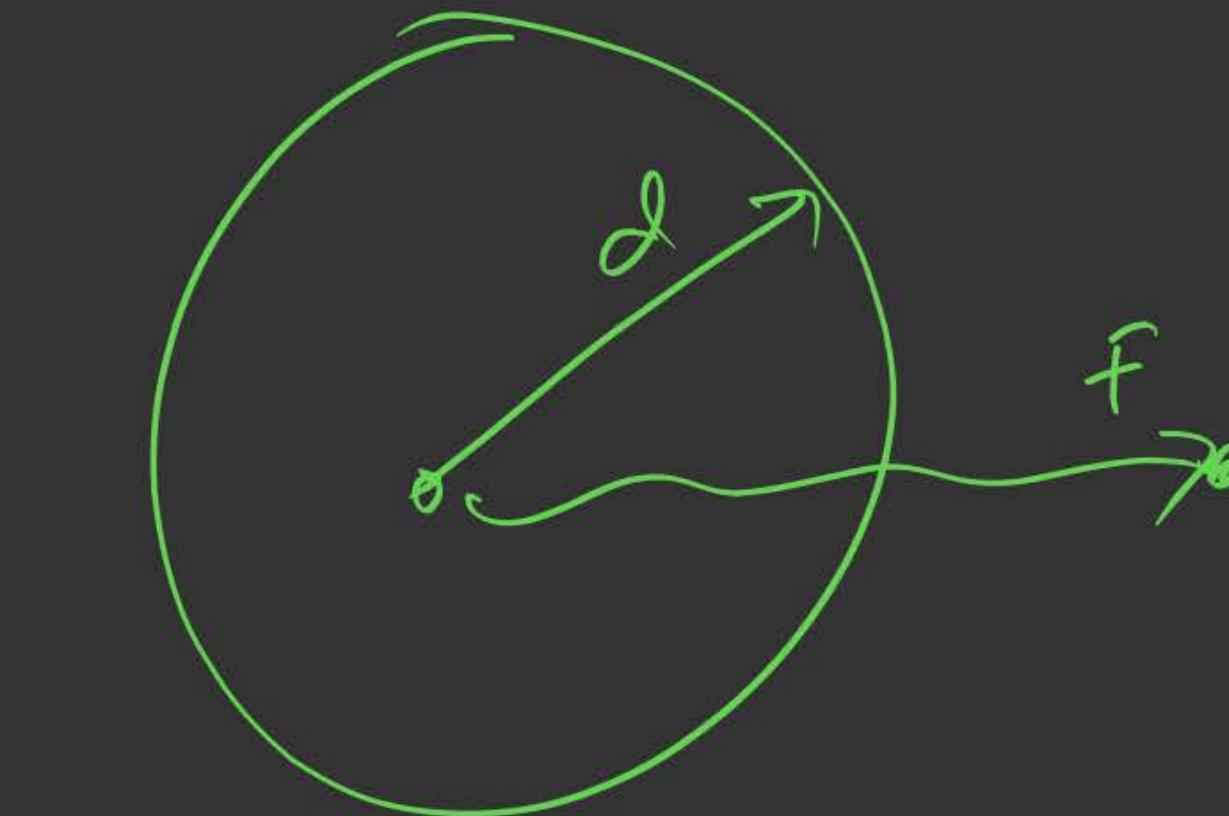
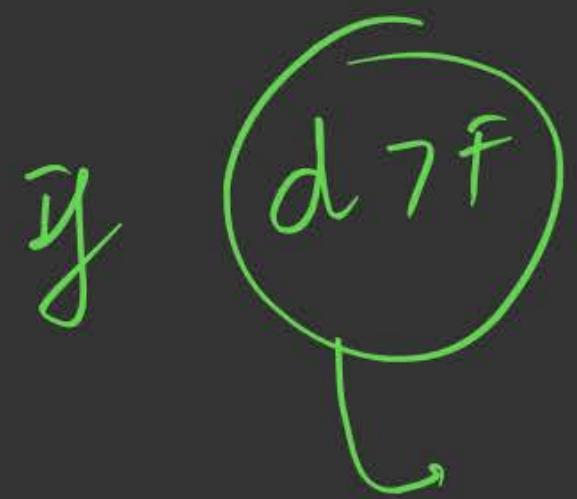
Start: S

Goal(s): U, X

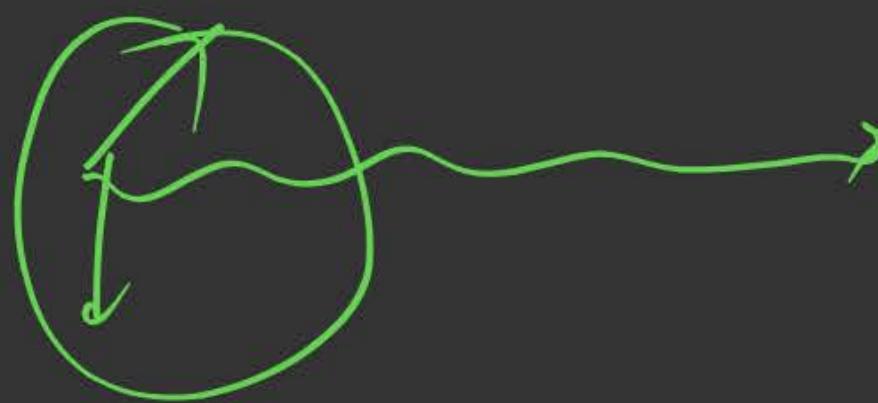
2) DFS



DLS



f > d
not
complete



Search Space



Topic : Uninformed Search



Depth Limited Search (DLS)

- Depth Limited Search is a modified version of DFS that imposes a limit on the depth of the search. This means that the algorithm will only explore nodes up to a certain depth, effectively preventing it from going down excessively deep paths that are unlikely to lead to the goal. By setting a maximum depth limit, DLS aims to improve efficiency and ensure more manageable search times.



Topic : Uninformed Search

Depth Limited Search (DLS)

- How Depth Limited Search Works
 - **Initialization:** Begin at the root node with a specified depth limit.
 - **Exploration:** Traverse the tree or graph, exploring each node's children.
 - **Depth Check:** If the current depth exceeds the set limit, **stop** exploring that path and backtrack.
- **Goal Check:** If the goal node is found within the depth limit, the search is successful.
- **Backtracking:** If the search reaches the depth limit or a leaf node without finding the goal, backtrack and explore other branches.





Topic : Uninformed Search

Depth Limited Search (DLS)

- **Advantages** of Depth Limited Search
 - Depth limited search is better than DFS and requires less time and memory space.
 - DFS assures that the solution will be found if it exists ~~in~~ infinite time.
 - There are applications of DLS in graph theory particularly similar to the DFS.
 - To combat the disadvantages of DFS, we add a **limit** to the depth, and our search strategy performs recursively down the search tree.
- **Disadvantages** of Depth Limited Search
 - The depth limit is compulsory for this algorithm to execute.
 - The goal node may ~~not~~ exist in the depth limit set earlier, which will push the user to iterate further adding execution time.
 - The goal node will not be found if it does not exist in the desired limit.

$P > d$



Topic : Uninformed Search

Depth Limited Search (DLS)

- Performance Measures
- Completeness: The DLS is a complete algorithm in general except the case when the goal node is the shallowest node, and it is beyond the depth limit, i.e. $l < d$, and in this case, we never reach the goal node.
- Optimality: The DLS is a non-optimal algorithm since the depth that is chosen can be greater than d ($l > d$). Thus DLS is not optimal if $l > d$ $\mathcal{O}(B^L)$
- Time complexity is expressed as: It is similar to the DFS, i.e. ~~$O(B^L)$~~ , where L is the set depth limit
- Space Complexity is expressed as: It is similar to DFS. $\mathcal{O}(BL)$, where L is specified depth limit

$L \rightarrow$ depth limit

b^d d
 b^d b
 b^d



Topic : Uninformed Search



Iterative Deepening Search (IDS) or Iterative Deepening Depth First Search (IDDFS)

- There are two common ways to traverse a graph, BFS and DFS. Considering a Tree (or Graph) of huge height and width, both BFS and DFS are not very efficient due to following reasons.
- DFS first traverses nodes going through one adjacent of root, then next adjacent. The problem with this approach is, if there is a node close to root, but not in first few subtrees explored by DFS, then DFS reaches that node very late. Also, DFS may not find shortest path to a node (in terms of number of edges).
- BFS goes level by level, but requires more space.



Topic : Uninformed Search



simp

Iterative Deepening Search (IDS) or Iterative Deepening Depth First Search (IDDFS)

- Iterative Deepening Depth-First Search (IDDFS) combines the depth-first search's space efficiency with the breadth-first search's completeness. It repeatedly performs depth-limited searches with increasing depth limits until the goal is found.

DLS





Topic : Uninformed Search



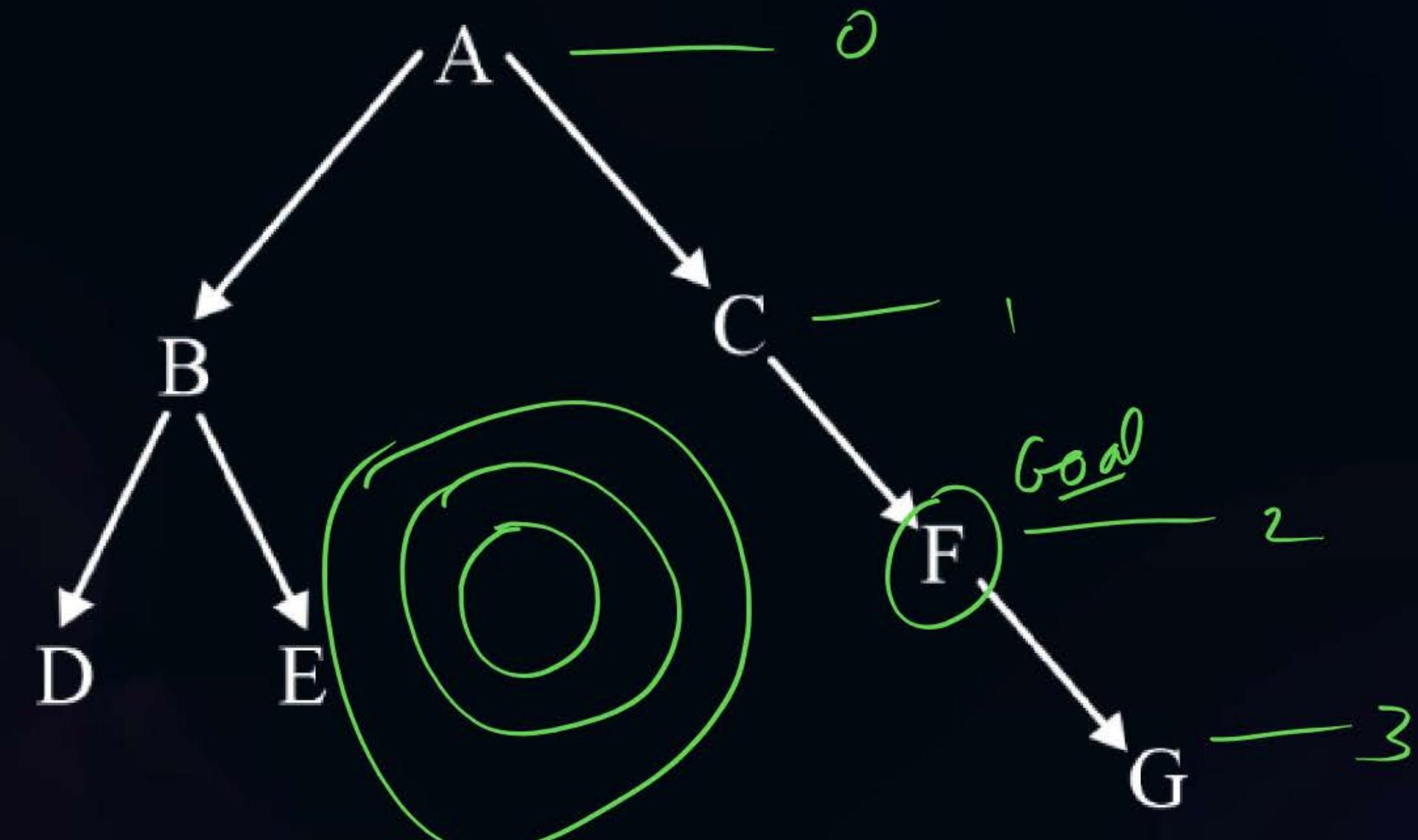
IDFS

$d = \text{depth limit} = 0$

Visited: A

$d = 1$:
A
B
C

$d = 2$:
D
C
B
A
E
F
Goal





Topic : Uninformed Search

Iterative Deepening Search (IDS) or Iterative Deepening Depth First Search (IDDFS)

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 ~~$O(b^d)$~~ .

$$\underline{O(b \times d)}$$





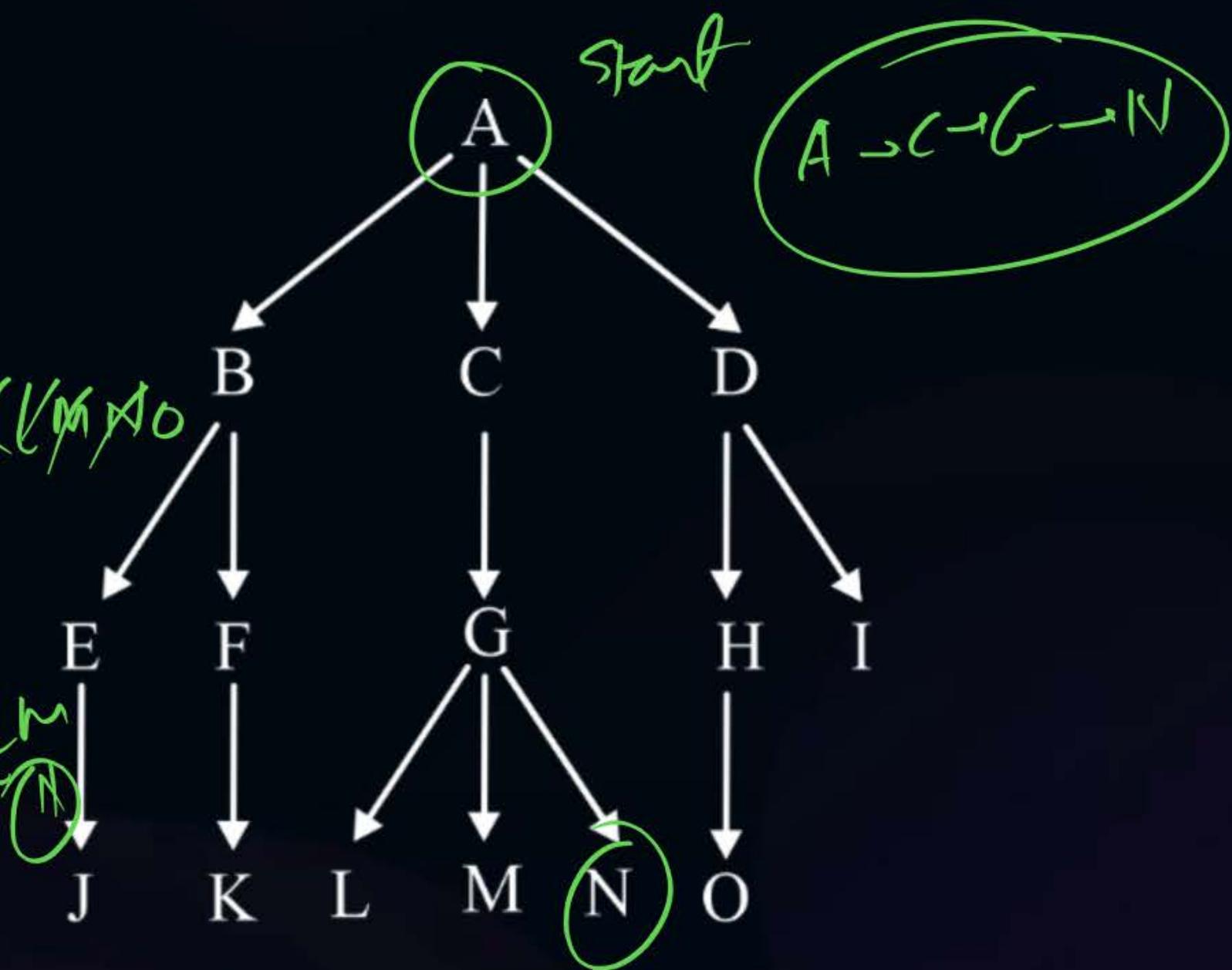
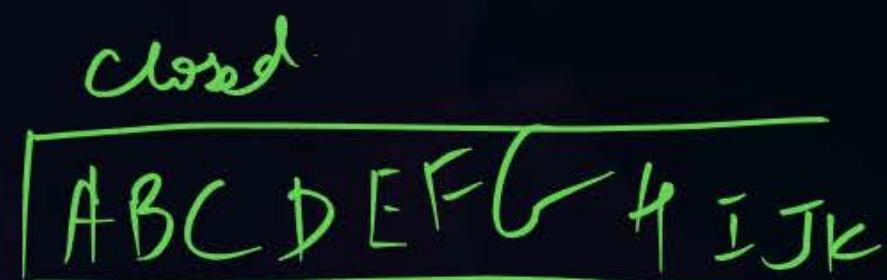
Topic : Uninformed Search



Iterative Deepening Search (IDS) or Iterative Deepening Depth First Search (IDDFS)

We will search for node N.

> BFS





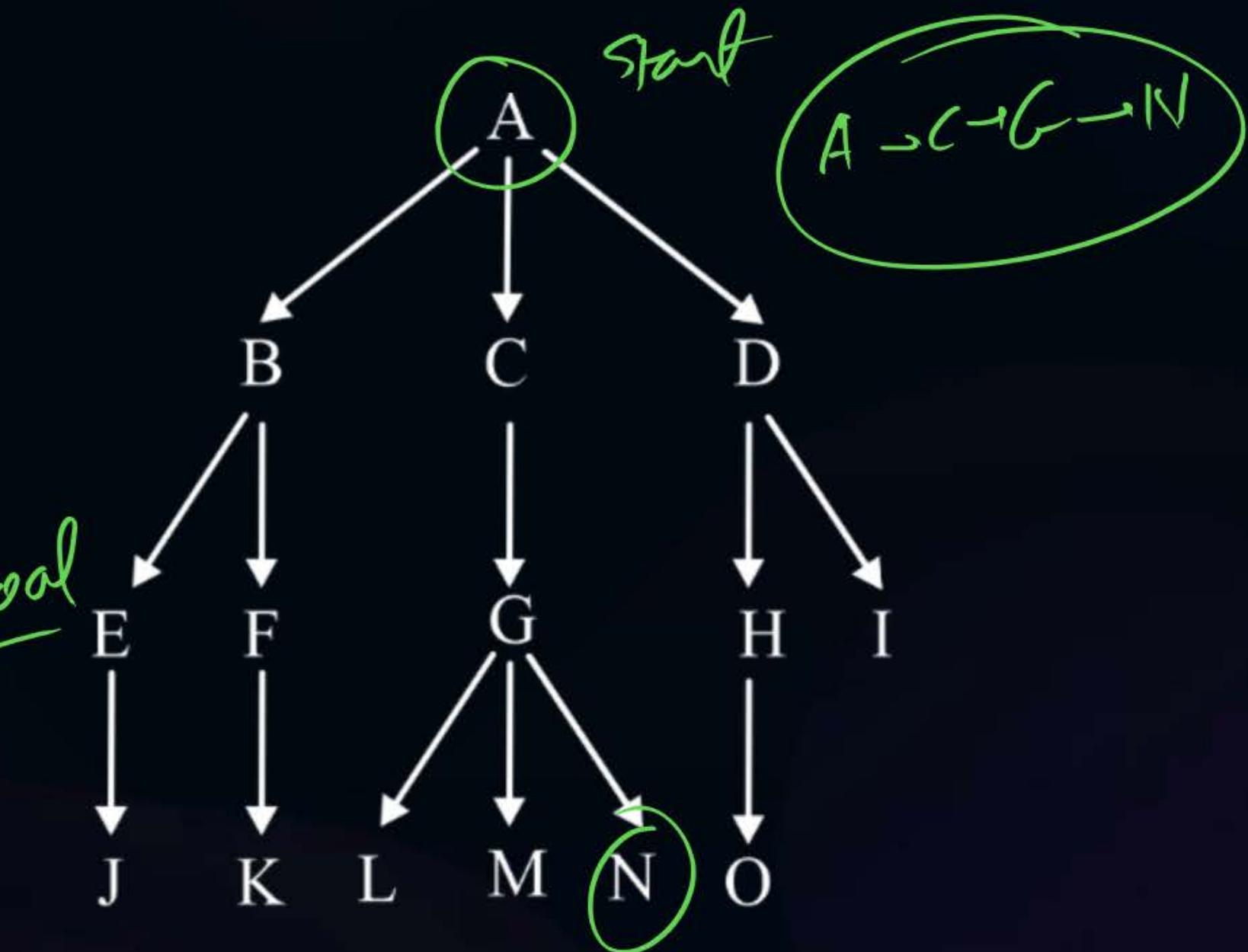
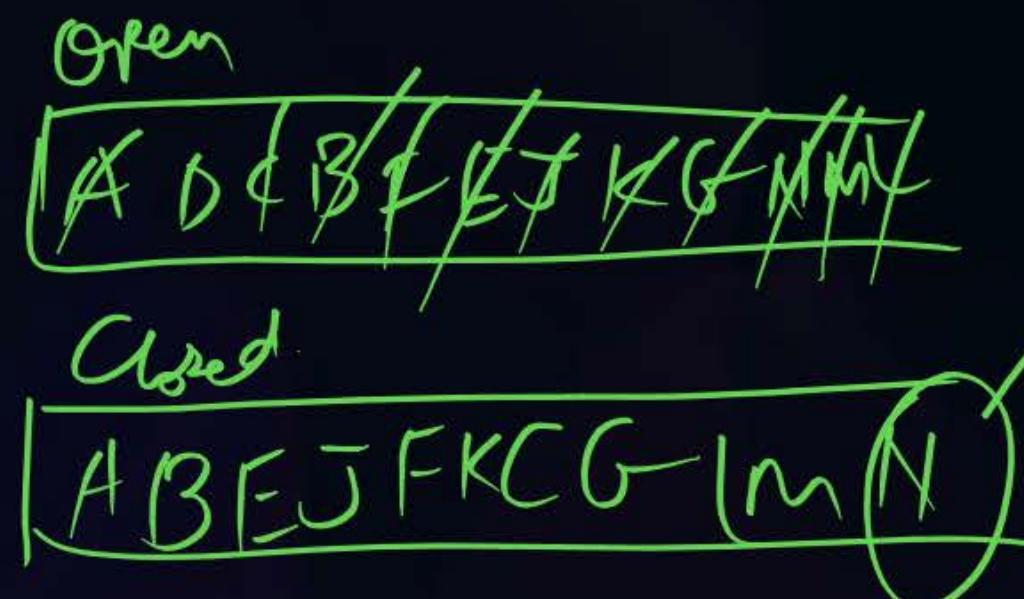
Topic : Uninformed Search



Iterative Deepening Search (IDS) or Iterative Deepening Depth First Search (IDDFS)

We will search for node N.

IDFS





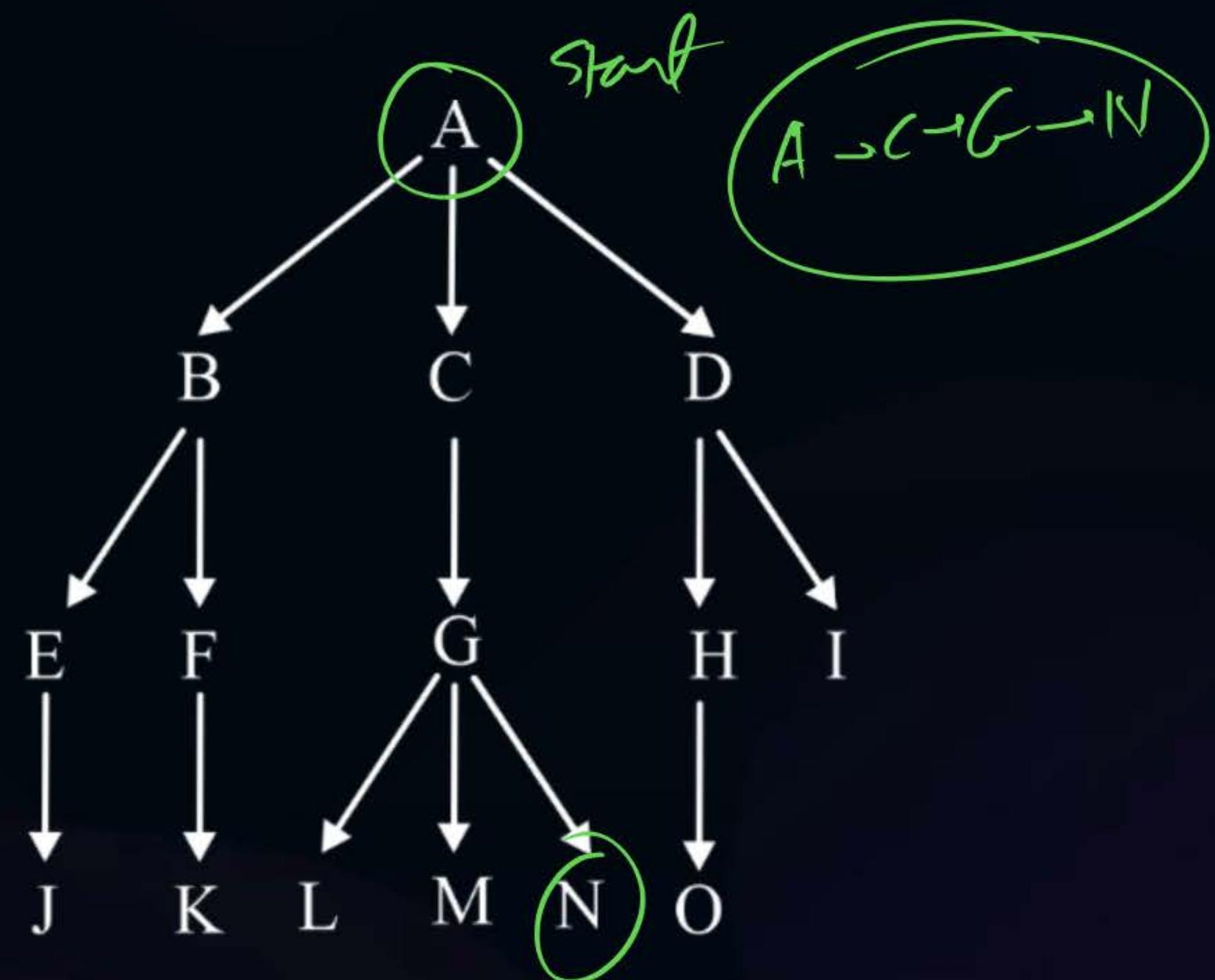
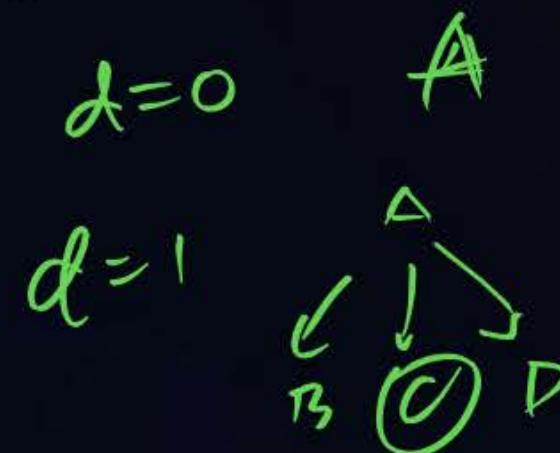
Topic : Uninformed Search

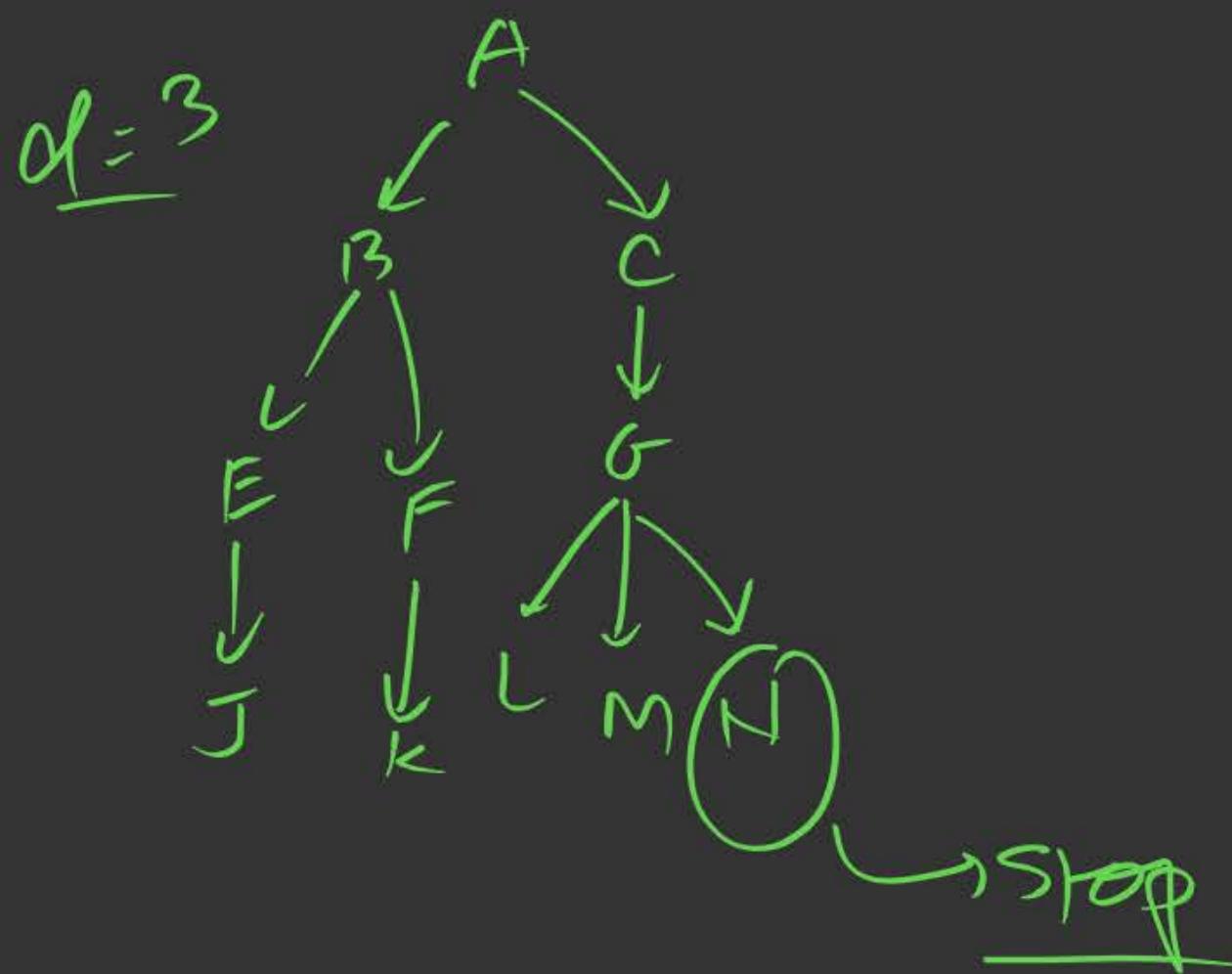
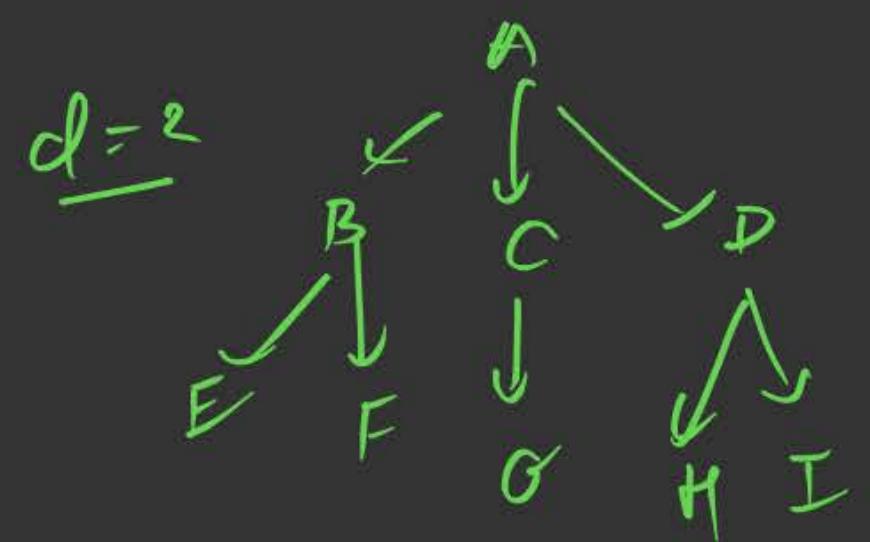


Iterative Deepening Search (IDS) or Iterative Deepening Depth First Search (IDDFS)

We will search for node N.

3) IDDFS





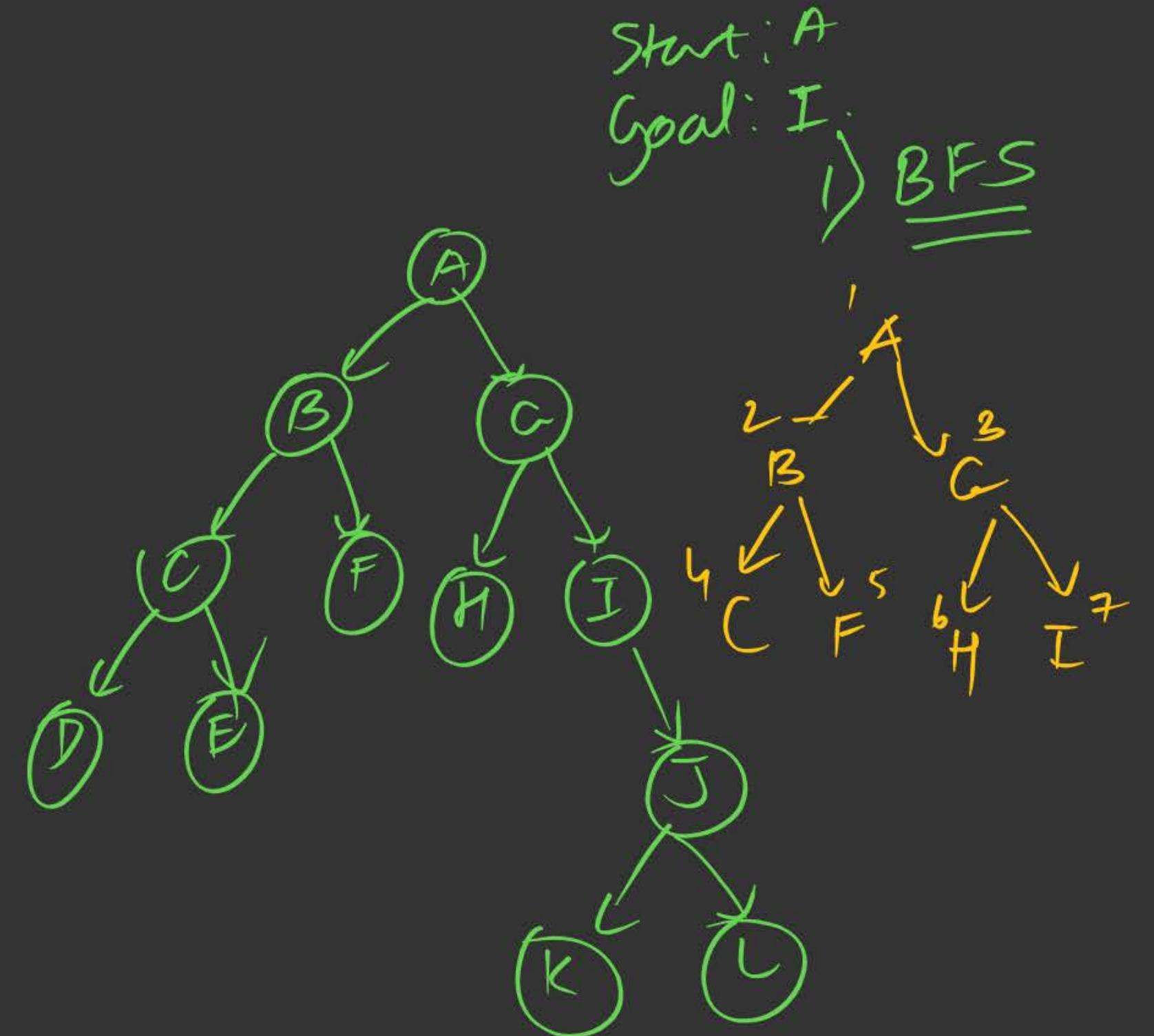


Topic : Uninformed Search

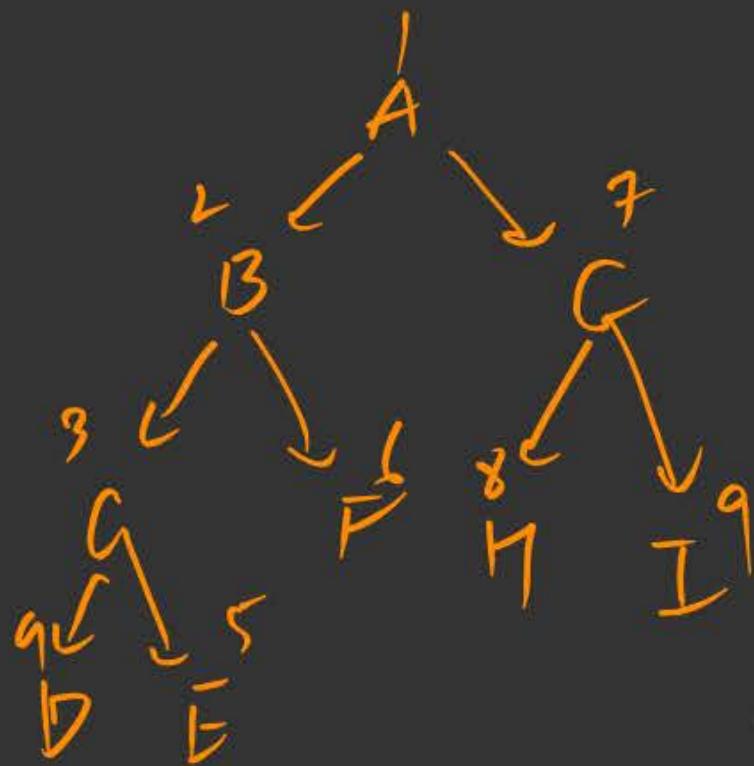
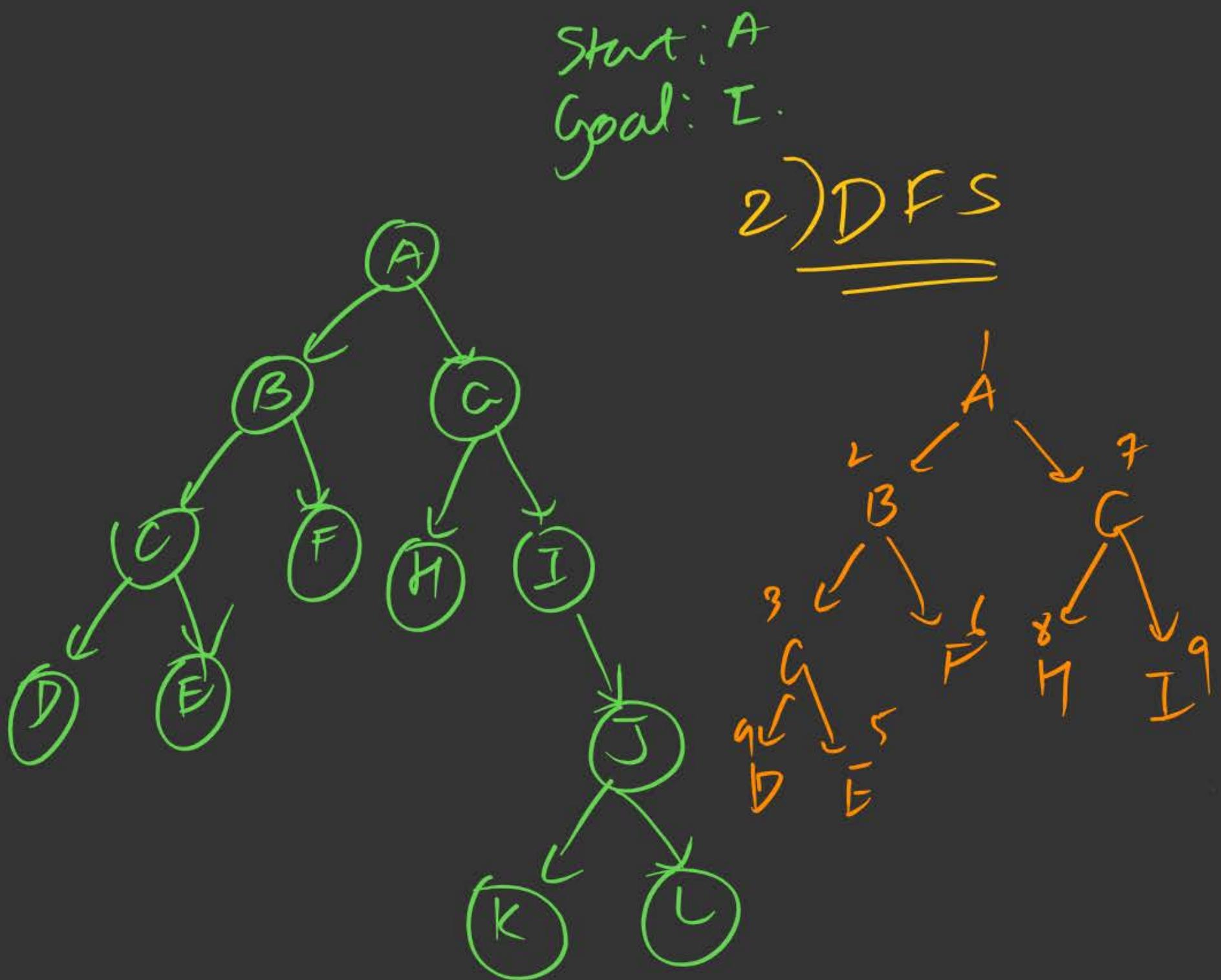


Iterative Deepening Search (IDS) or Iterative Deepening Depth First Search (IDDFS)

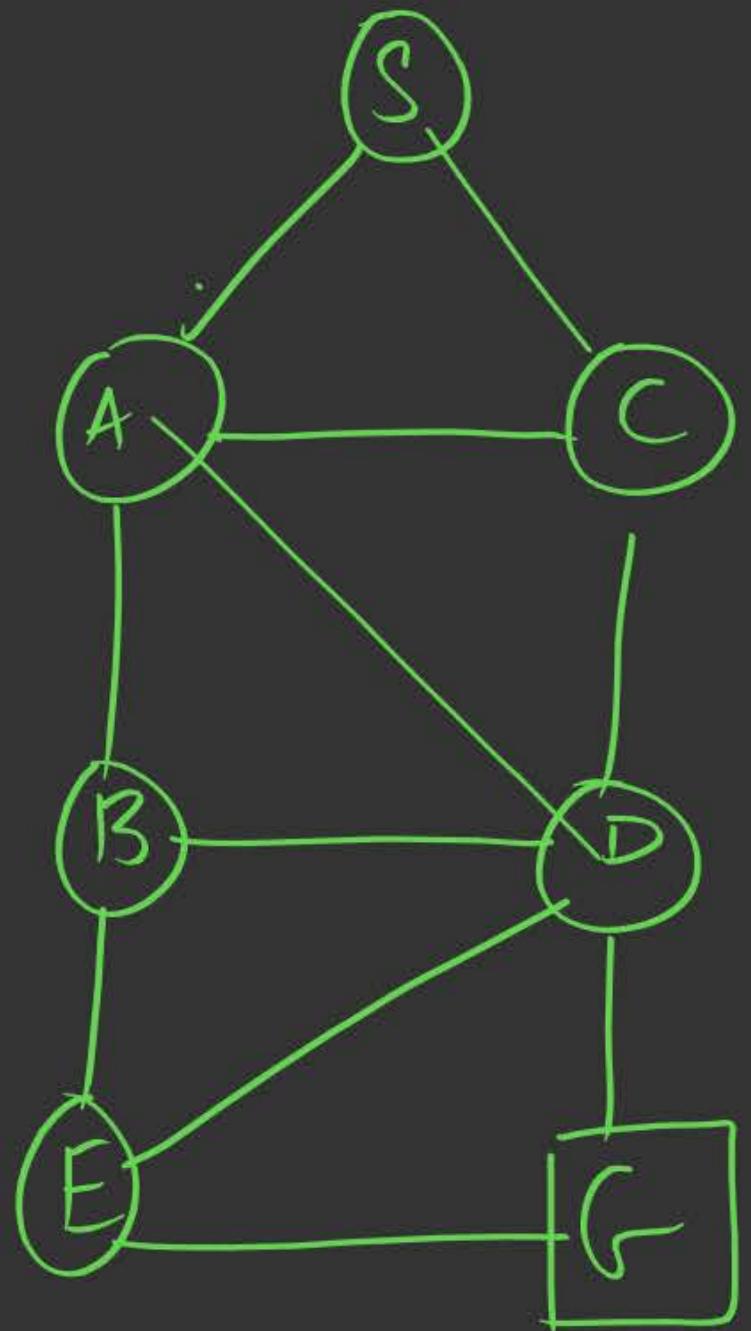
- Space Complexity
- Each iteration of depth-limited DFS uses space proportional to the depth limit.
- IDDFS does not store all nodes at a given depth level simultaneously; it only stores the path from the root to the current node.



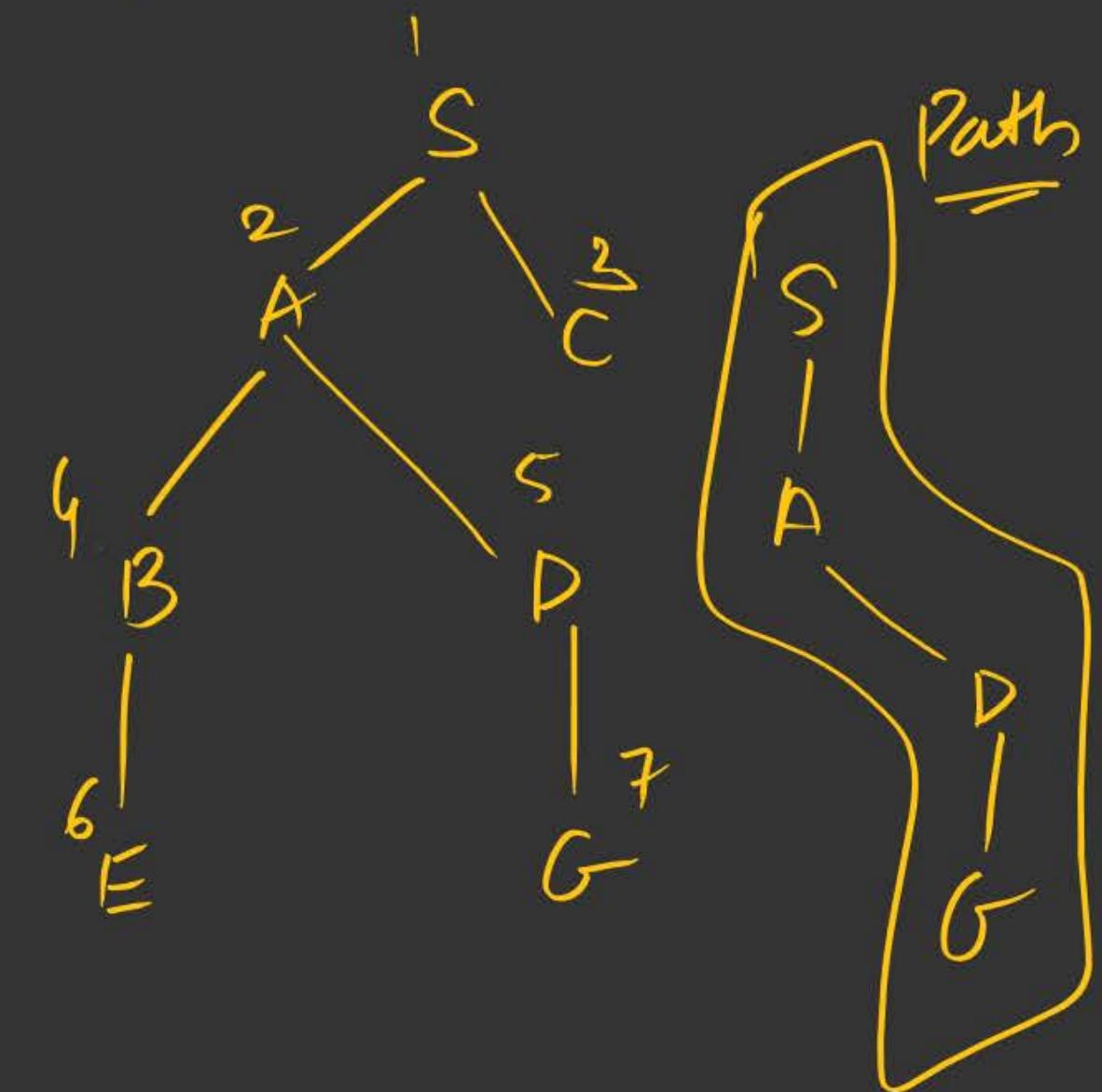
73%



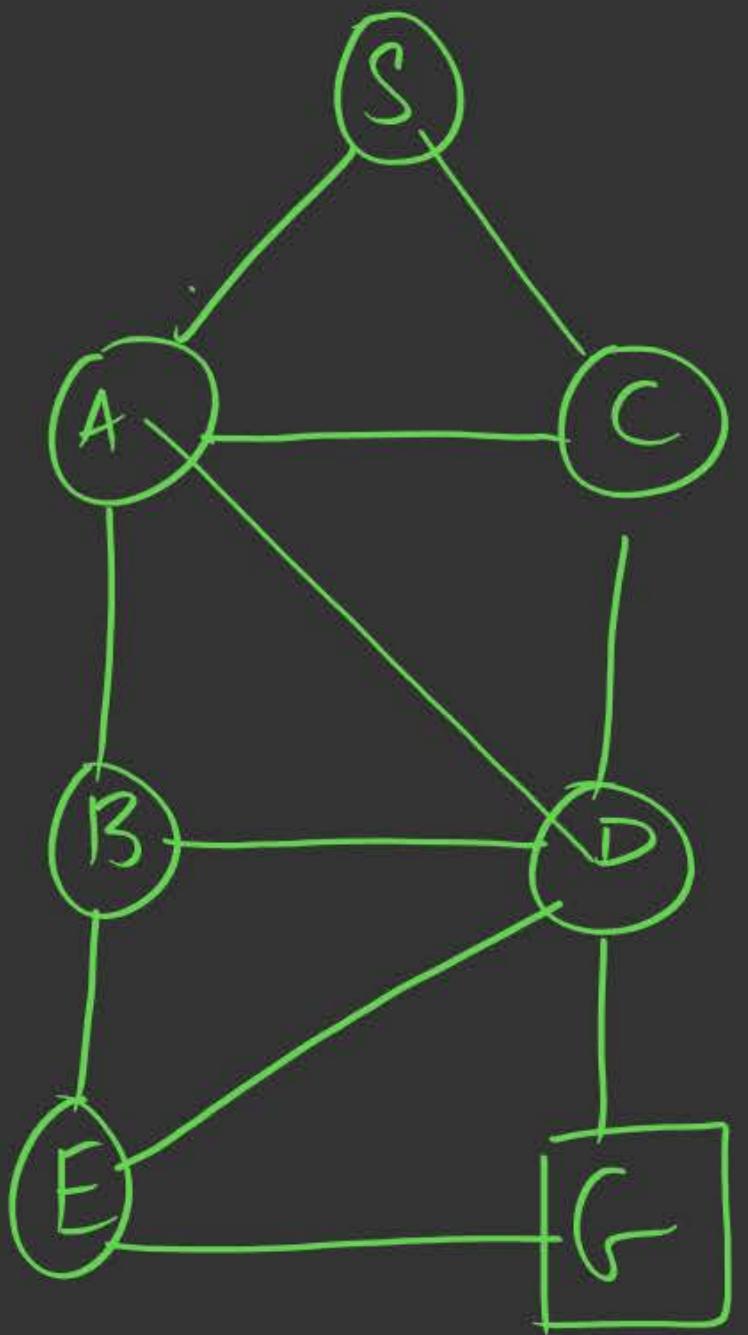
(Q)



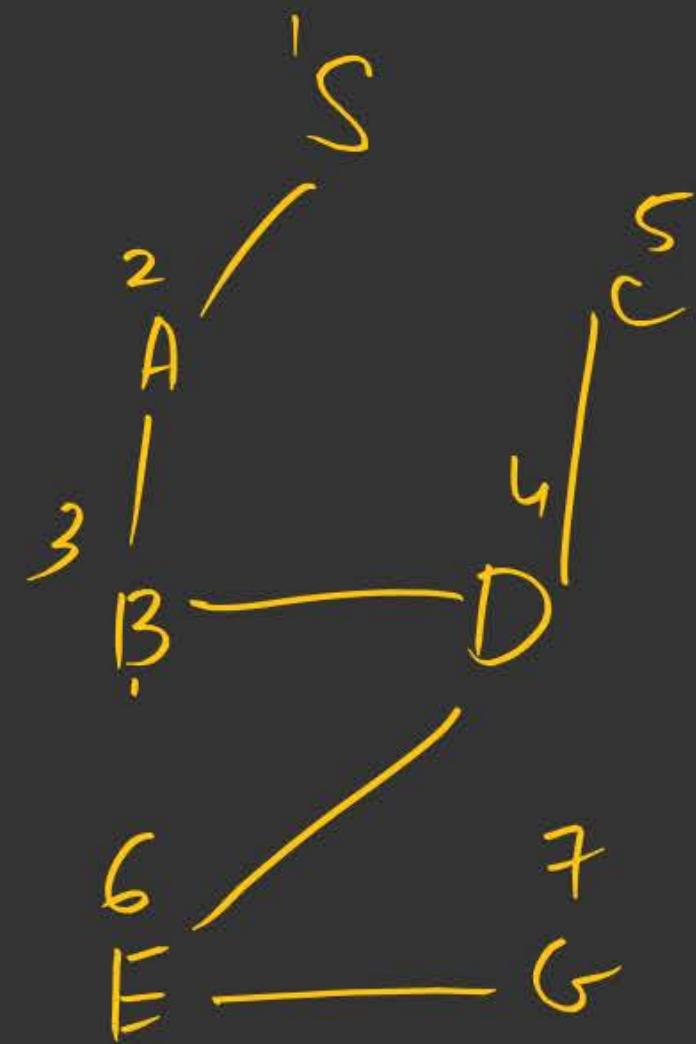
S → G
alphabetical
i) Draw BFS \textcircled{S}



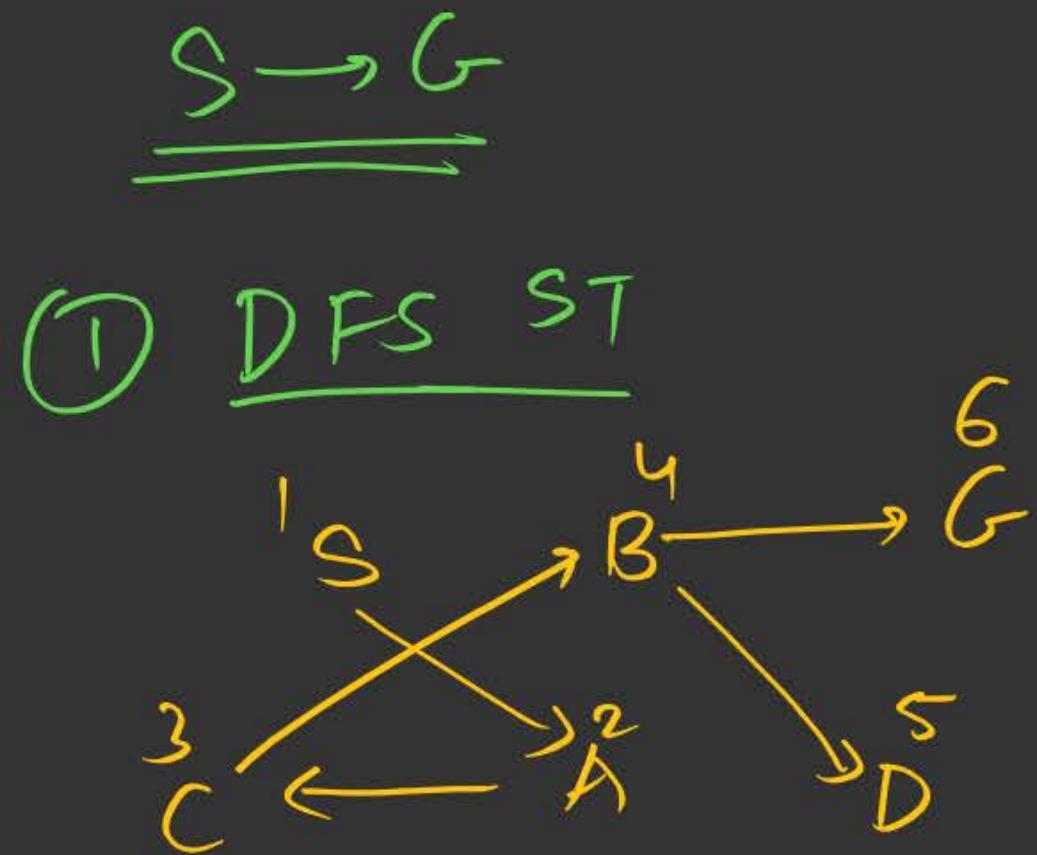
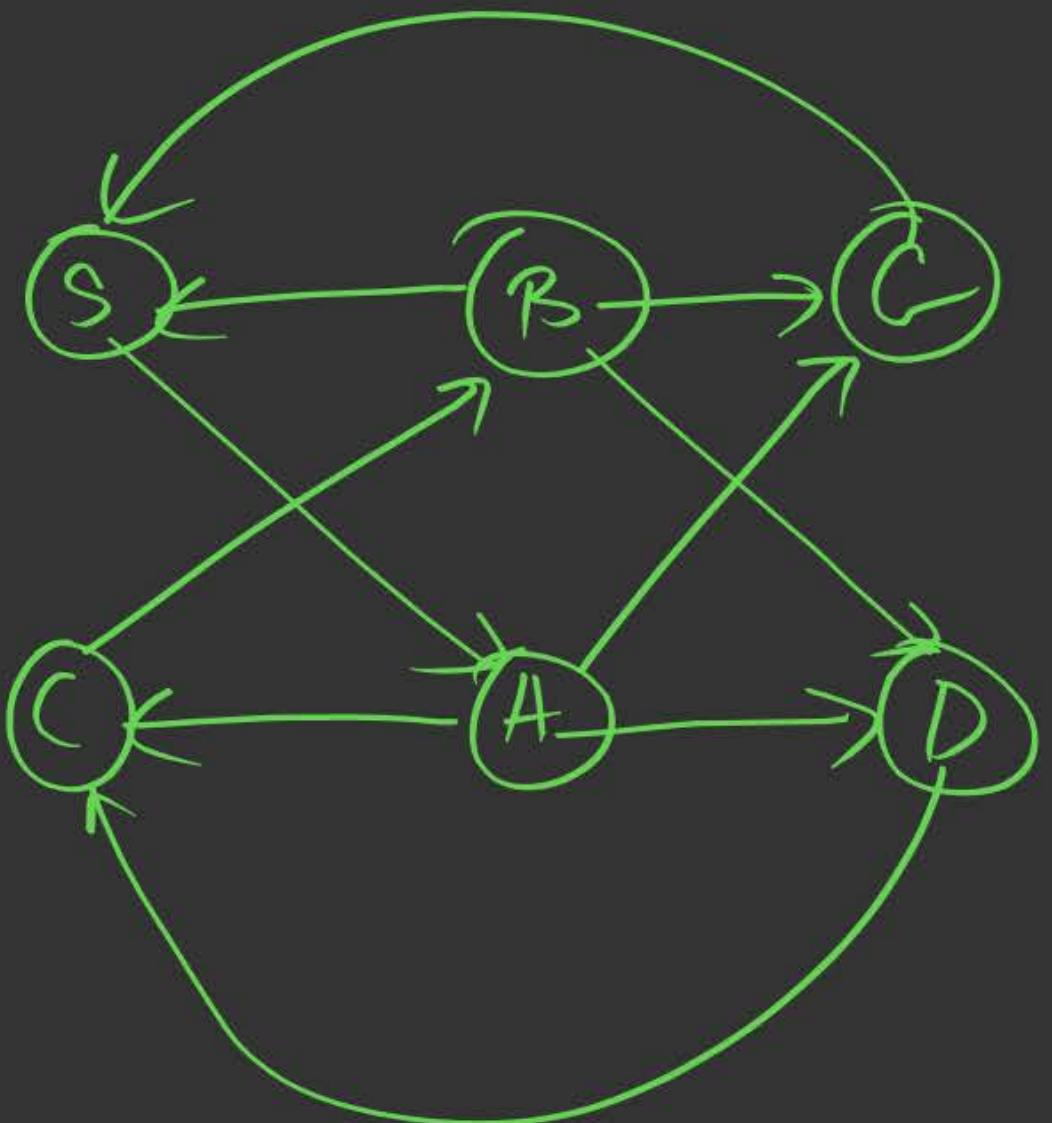
(Q)



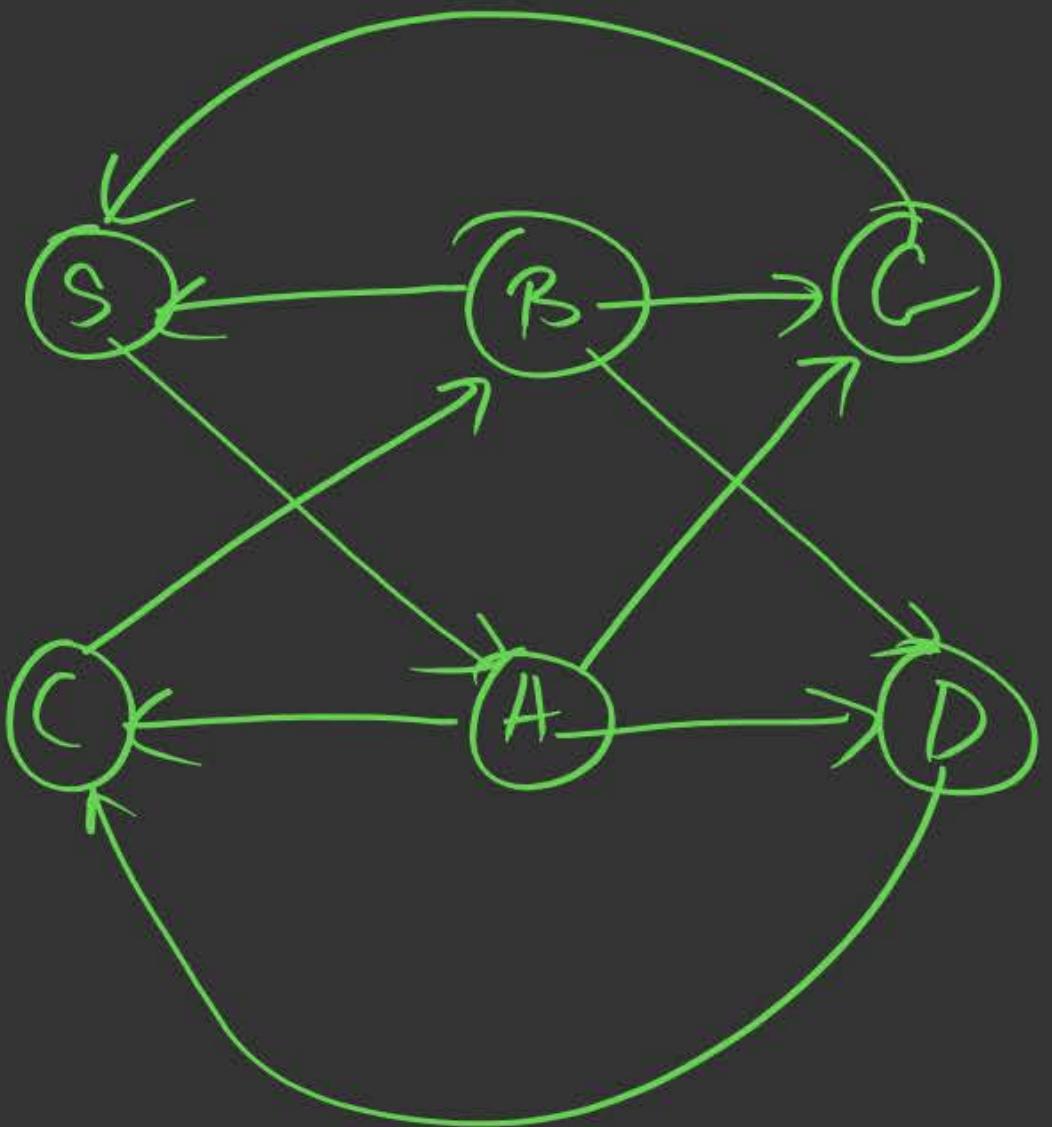
$S \rightarrow C$ alphabetical
i) Draw DFS $\Rightarrow ST$



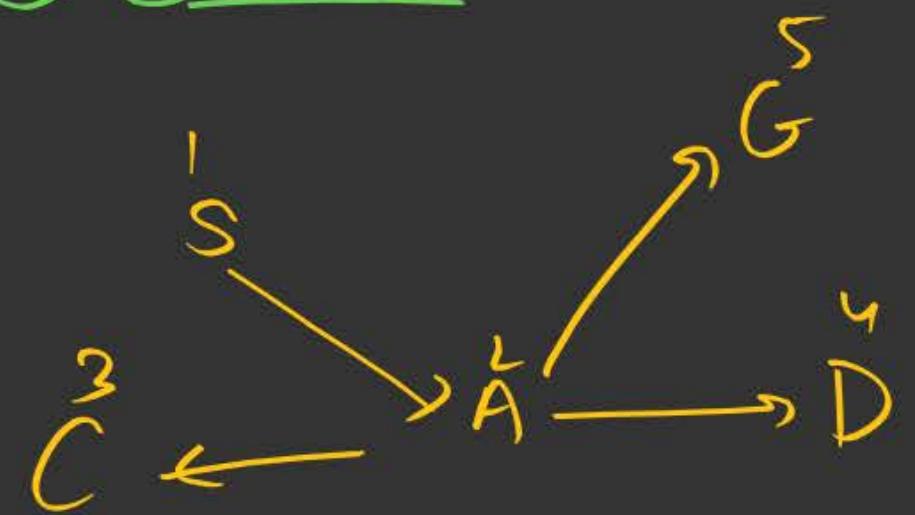
(Q)



(Q)



$\begin{matrix} S \rightarrow G \\ \equiv \\ ② \text{ BFS ST} \end{matrix}$





Topic : Uninformed Search

Uniform Cost Search

Uniform Cost Search (UCS) is a search algorithm used to find the least-cost path from a start node to a goal node in a graph. UCS is a variant of Dijkstra's algorithm and is used in situations where the path costs vary, making it more suitable for weighted graphs.

BFS shortest → less edges

UCS shortest → path cost
~~~~~



# Topic : Uninformed Search

## Uniform Cost Search

### Key Concepts

- Priority Queue UCS uses a priority queue to explore the least-cost path first. Nodes are expanded in order of their cumulative cost from the start node.
- Cost: Each edge in the graph has a cost associated with it. UCS aims to find the path with the minimum cumulative cost.
- Expansion: The algorithm expands the node with the smallest cumulative cost, ensuring that the cheapest path is always chosen.



# Topic : Uninformed Search



SSSP Algo

$S \rightarrow G$

Start = A

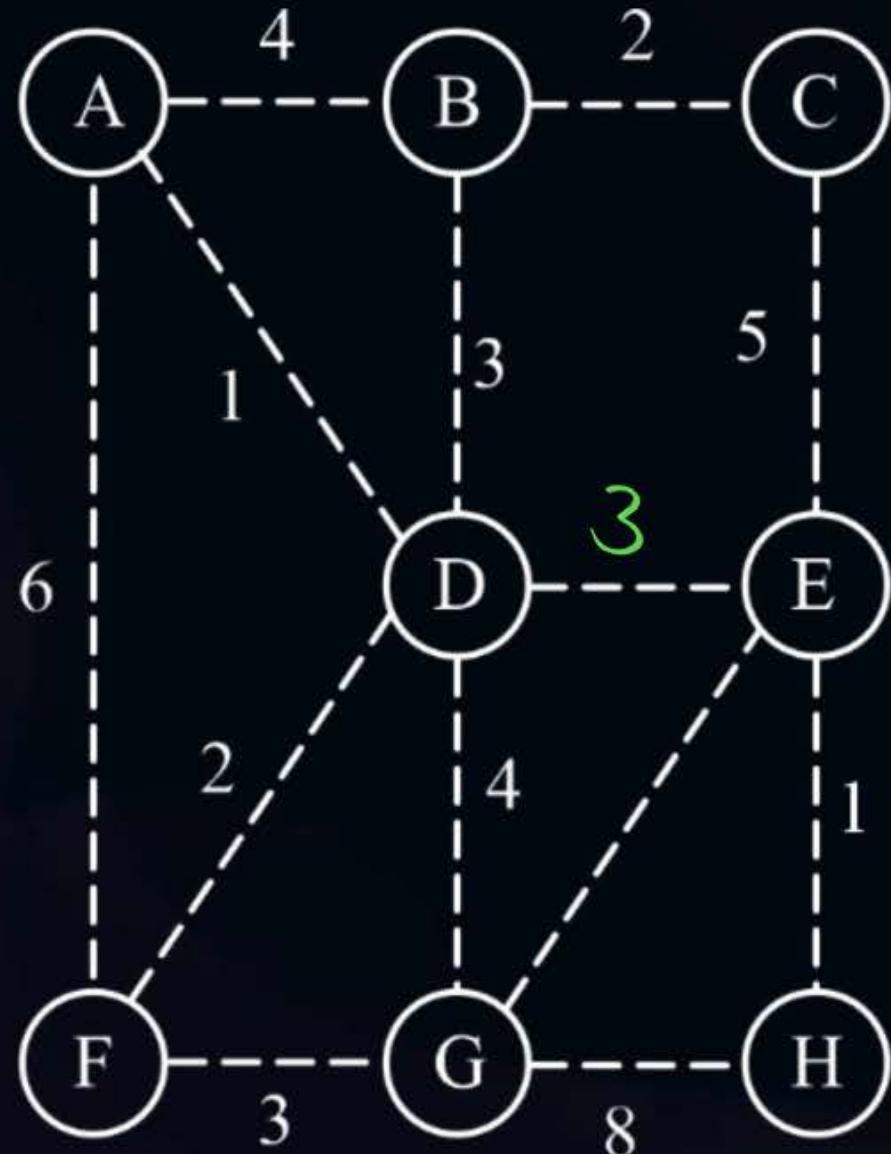
Goal = H

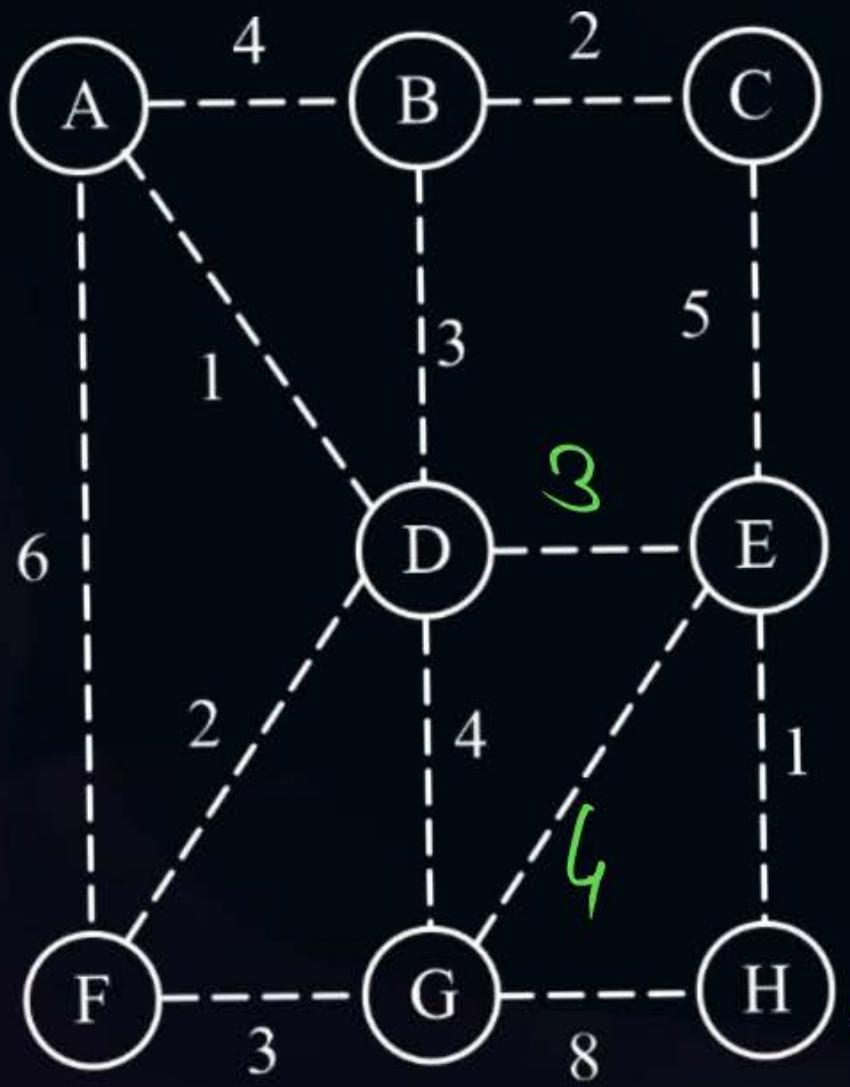
Strategy: UCS.

Ans:-

$A \rightarrow H$ : (5)

path:  $A \rightarrow D \rightarrow E \rightarrow H$





① Dijkstra's SSSP

*Closed/ Visited*

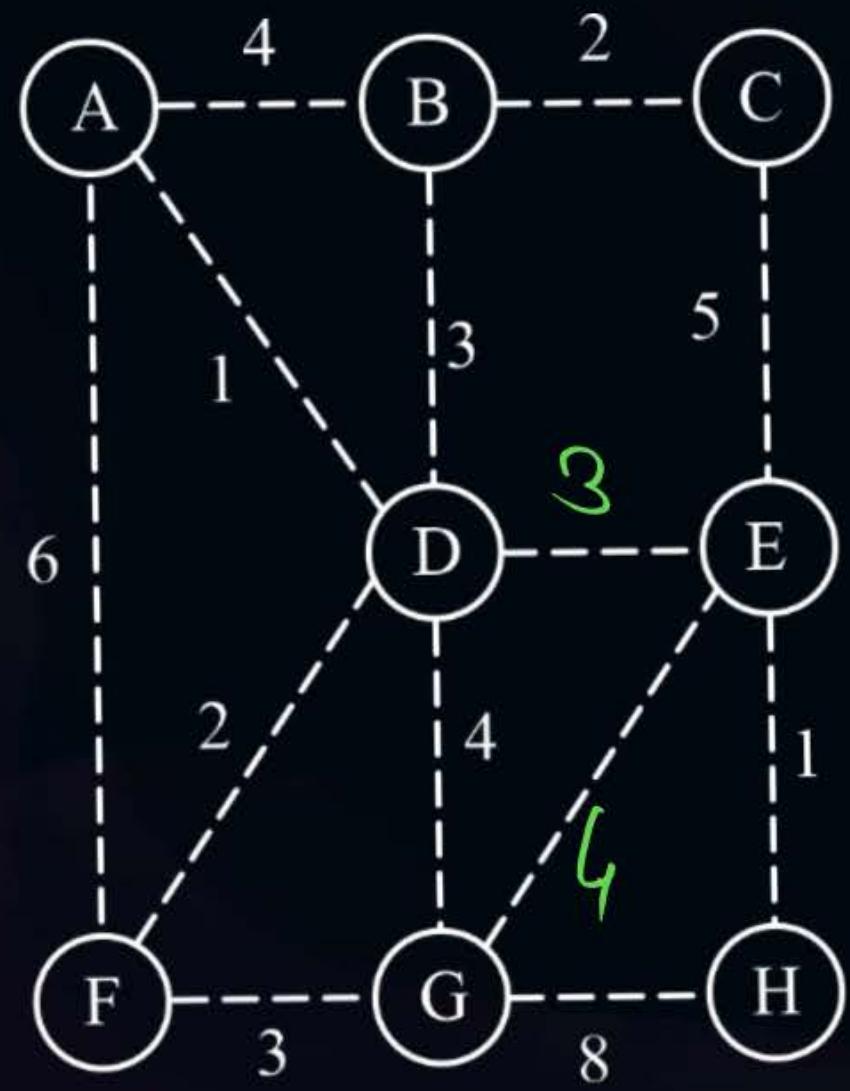
*start*

|                 | A | B | C | D | E | F | G | H |
|-----------------|---|---|---|---|---|---|---|---|
| { }             | ✓ | ✓ |   | ✓ |   |   | ✓ |   |
| (A)             | 0 | 4 | x | 1 | x | 6 | x | x |
| (A,D)           | 0 | 4 | x | 1 | 4 | 3 | 5 | x |
| (A,D,F)         | 0 | 4 | x | 1 | 4 | 3 | 5 | x |
| (A,D,F,B)       | 0 | 4 | 6 | 1 | 4 | 3 | 5 | x |
| (A,D,F,B,E)     | 0 | 4 | 6 | 1 | 4 | 3 | 5 | 5 |
| (A,D,F,B,E,G)   | 0 | 4 | 6 | 1 | 4 | 3 | 5 | 5 |
| (A,D,F,B,E,G,H) | 0 | 4 | 6 | 1 | 4 | 3 | 5 | 5 |

*CAND*

*Goal*

A grid-based diagram showing the state space of Dijkstra's algorithm. The columns represent nodes A through H. The rows represent sets of states. The first row shows the initial state { }. Subsequent rows show the state of each node (0 for unvisited, 4 for distance 4, 6 for distance 6, etc.) and whether it is closed/visited (✓) or not (x). The bottom row shows the goal state (all nodes visited). The grid is highlighted with a green oval.



2) VCS

|     | <u>Open</u> | <u>Closed</u> |   |   |   |   |   |
|-----|-------------|---------------|---|---|---|---|---|
|     |             | A             | D | F | B | E | G |
| x A |             |               |   |   |   |   |   |
| x B | 4           | 4             |   | 9 | 4 | 4 | 4 |
| x D | 1           |               | 1 |   | 1 | 1 | 1 |
| x F | 6           | 3             |   | 3 | 3 | 3 | 3 |
| x E | x           | 5             |   | 5 | 5 | 5 | 5 |
| x G |             |               |   |   |   |   |   |
| x C |             |               |   |   |   |   |   |
| x H |             |               |   |   |   |   |   |

Handwritten annotations in yellow:

- Row 1: A (crossed out)
- Row 2: B (crossed out)
- Row 3: D (crossed out)
- Row 4: F (crossed out)
- Row 5: E (crossed out)
- Row 6: G (crossed out)
- Row 7: C (crossed out)
- Row 8: H (crossed out)
- Column 1: A (crossed out)
- Column 2: D (crossed out)
- Column 3: F (crossed out)
- Column 4: B (crossed out)
- Column 5: E (crossed out)
- Column 6: G (crossed out)
- Cell (B, G): 9 (circled)
- Cell (D, F): 1 (circled)
- Cell (F, B): 3 (circled)
- Cell (E, D): 5 (circled)
- Cell (G, E): 5 (circled)
- Cell (G, H): 5 (circled)
- Cell (H, G): 5 (circled)
- Cell (H, E): 6 (circled)
- Cell (H, F): 6 (circled)
- Cell (H, B): 5 (circled)
- Cell (G, B): 9 (circled)
- Cell (G, E): 9 (circled)
- Cell (G, F): 9 (circled)
- Cell (F, E): 9 (circled)
- Cell (F, G): 9 (circled)
- Cell (E, G): 9 (circled)
- Cell (E, F): 9 (circled)
- Cell (E, B): 5 (circled)
- Cell (E, D): 5 (circled)
- Cell (D, E): 3 (circled)
- Cell (D, B): 1 (circled)
- Cell (D, F): 2 (circled)
- Cell (D, G): 4 (circled)
- Cell (B, D): 3 (circled)
- Cell (B, E): 5 (circled)
- Cell (B, C): 2 (circled)
- Cell (A, B): 4 (circled)
- Cell (A, D): 1 (circled)
- Cell (A, F): 6 (circled)
- Cell (A, G): 3 (circled)
- Cell (A, H): 8 (circled)
- Cell (F, G): 3 (circled)
- Cell (G, H): 1 (circled)
- Cell (G, F): 4 (circled)
- Cell (G, B): 2 (circled)
- Cell (G, D): 4 (circled)
- Cell (G, E): 3 (circled)
- Cell (G, H): 8 (circled)
- Cell (H, G): 1 (circled)
- Cell (H, F): 8 (circled)
- Cell (H, D): 8 (circled)
- Cell (H, E): 8 (circled)

Final result:  $\text{g}_6 = \emptyset$



**THANK - YOU**



# DS & AI ENGINEERING



## Artificial Intelligence

### Un-Informed search

Lecture No.- 06

By- Aditya sir



# Recap of Previous Lecture



Topic

Topic

Topic

Topic

UCS

Questions



# Topics to be Covered



Topic

Topic

Topic

UCS Questions

Beam Search

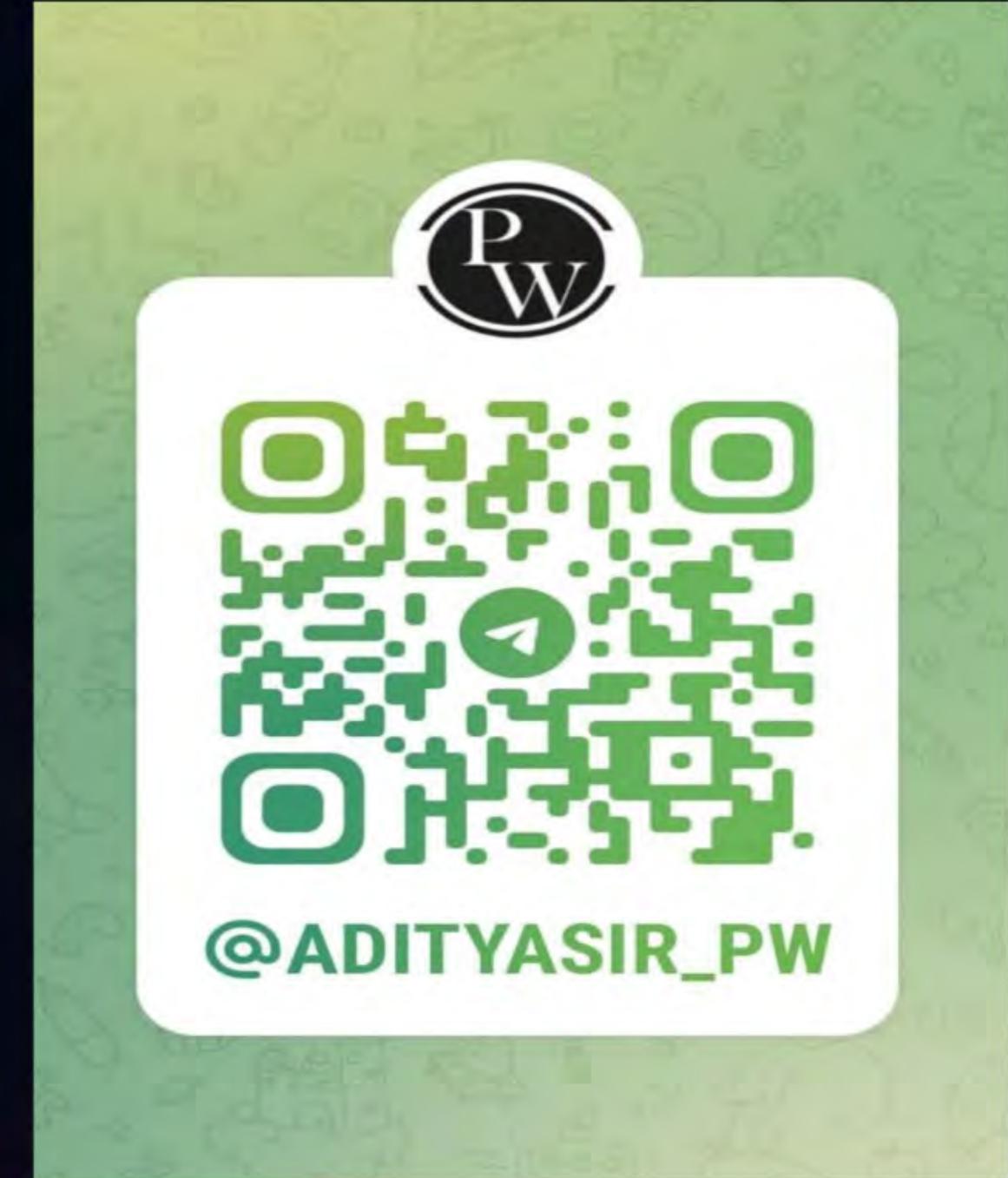
Bidirectional Search



## About Aditya Jain sir

1. Appeared for GATE during BTech and secured AIR 60 in GATE in very first attempt - City topper
2. Represented college as the first Google DSC Ambassador.
3. The only student from the batch to secure an internship at Amazon. (9+ CGPA)
4. Had offer from IIT Bombay and IISc Bangalore to join the Masters program
5. Joined IIT Bombay for my 2 year Masters program, specialization in Data Science
6. Published multiple research papers in well known conferences along with the team
7. Received the prestigious excellence in Research award from IIT Bombay for my Masters thesis in ML
8. Completed my Masters with an overall GPA of 9.36/10
9. Joined Dream11 as a Data Scientist
10. Have mentored 15,000+ students & working professionals in field of Data Science and Analytics
11. Have been mentoring & teaching GATE aspirants to secure a great rank in limited time
12. Have got around 27.5K followers on Linkedin where I share my insights and guide students and professionals.





*Telegram*

**Telegram Link for Aditya Jain sir:**  
**[https://t.me/AdityaSir\\_PW](https://t.me/AdityaSir_PW)**



# Topic : Uninformed Search

## DFS:

- It may stuck into any  $\infty$  length branch and may not be able to reach result.
- 2 Solution

(1) DLS

(2) IDDFS

$l = \infty$   
 $l = 1$   
 $l = 2$





## Topic : Uninformed Search



### Depth Limited Search (DLS):

- Depth Limited Search is a modified version of DFS that imposes a limit on the depth of the search. This means that the algorithm will only explore nodes up to a certain depth, effectively preventing it from going down excessively deep paths that are unlikely to lead to the goal. By setting a maximum depth limit, DLS aims to improve efficiency and ensure more manageable search times.
- It may give no result
- If the goal state is below the mentioned depth.



# Topic : Uninformed Search

## Depth Limited Search (DLS):

- will not stuck into branch with  $\infty$  nodes.
- d : depth of tree
- b : branching factor

## Advantage:

- Less search time if the goal state is with in given depth.
- Time and space complexity
- $TC \Rightarrow O(b^L)$
- L : depth limit given for DLS
- $SC \Rightarrow O(b \times L)$



$$\underline{d < L}$$

$$d < L$$



## Topic : Uninformed Search

### Depth Limited Search (DLS):

#### Disadvantage:

- Will not give result if the goal node is below specified depth.

$$\underline{d > l}$$



## Topic : Uninformed Search

### Iterative Deepening Search (IDS) or Iterative Deepening Depth First Search (IDDFS):

- There are two common ways to traverse a graph, BFS and DFS. Considering a Tree (or Graph) of huge height and width, both BFS (optimal But space) and DFS space less, not optimal sol are not very efficient due to following reasons.
- DFS first traverses nodes going through one adjacent of root, then next adjacent. The problem with this approach is, if there is a node close to root, but not in first few subtrees explored by DFS, then DFS reaches that node very late. Also, DFS may not find shortest path to a node (in terms of number of edges).
- BFS goes level by level, but requires more space.



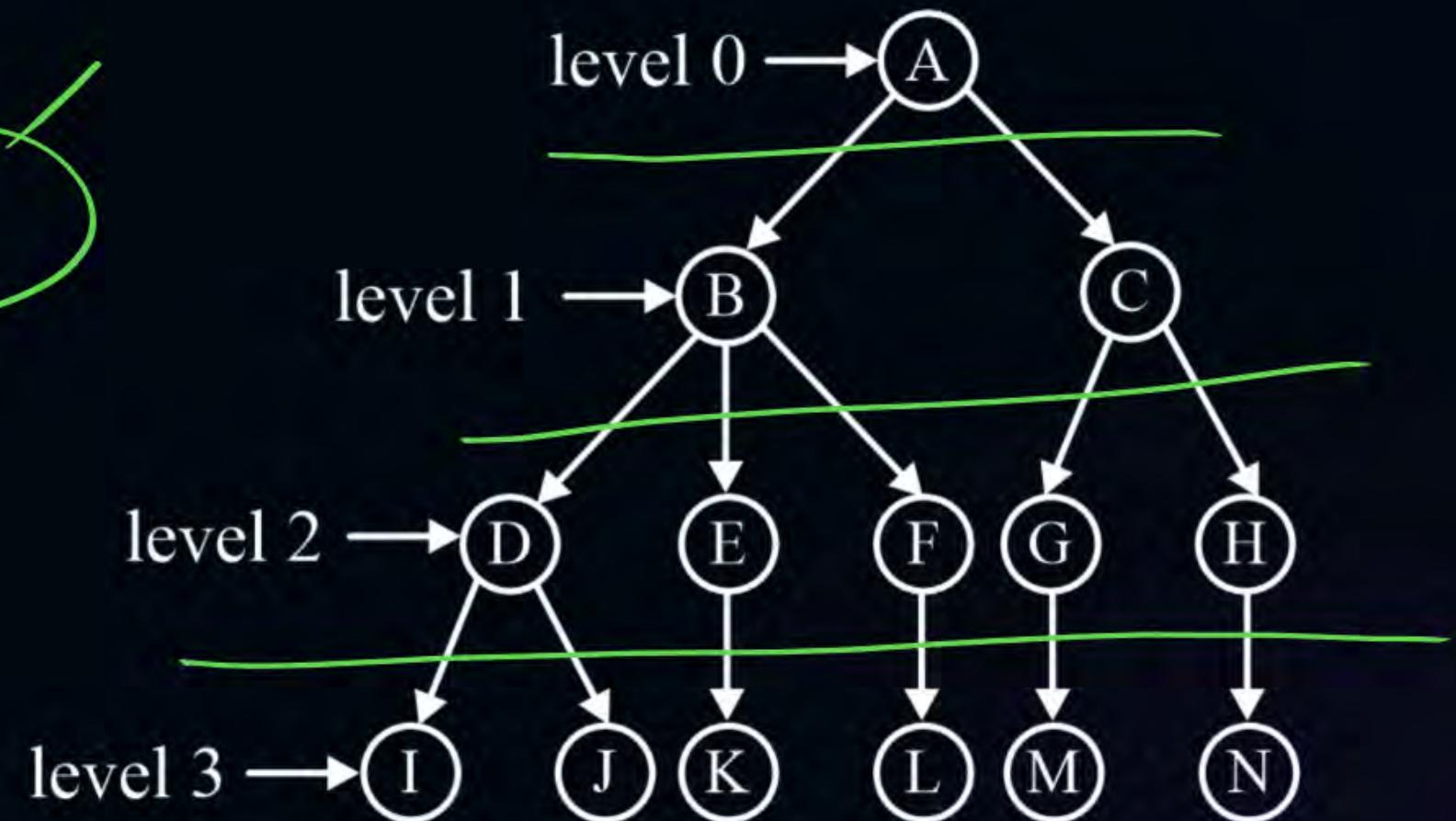
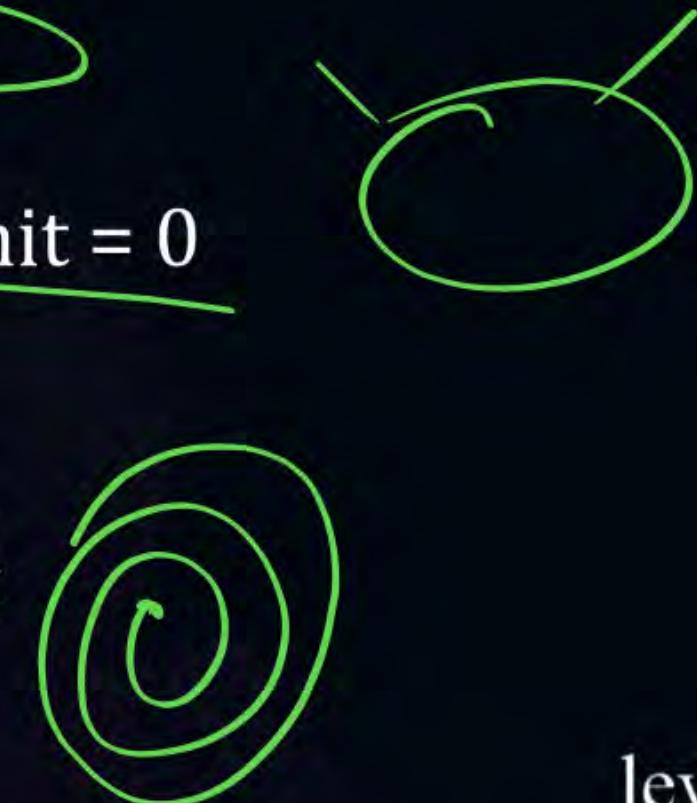
# Topic : Uninformed Search

## Iterative Deepening Search (IDS) or Iterative Deepening Depth First Search (IDDFS):

### IDDFS:

Better than DLS , DFS, BFS

- No depth is pride fine
- we start with depth limit = 0
- we start with level 0.
- goal node reached no.
- inc the depth limit by 1
- apply DFS.





# Topic : Uninformed Search

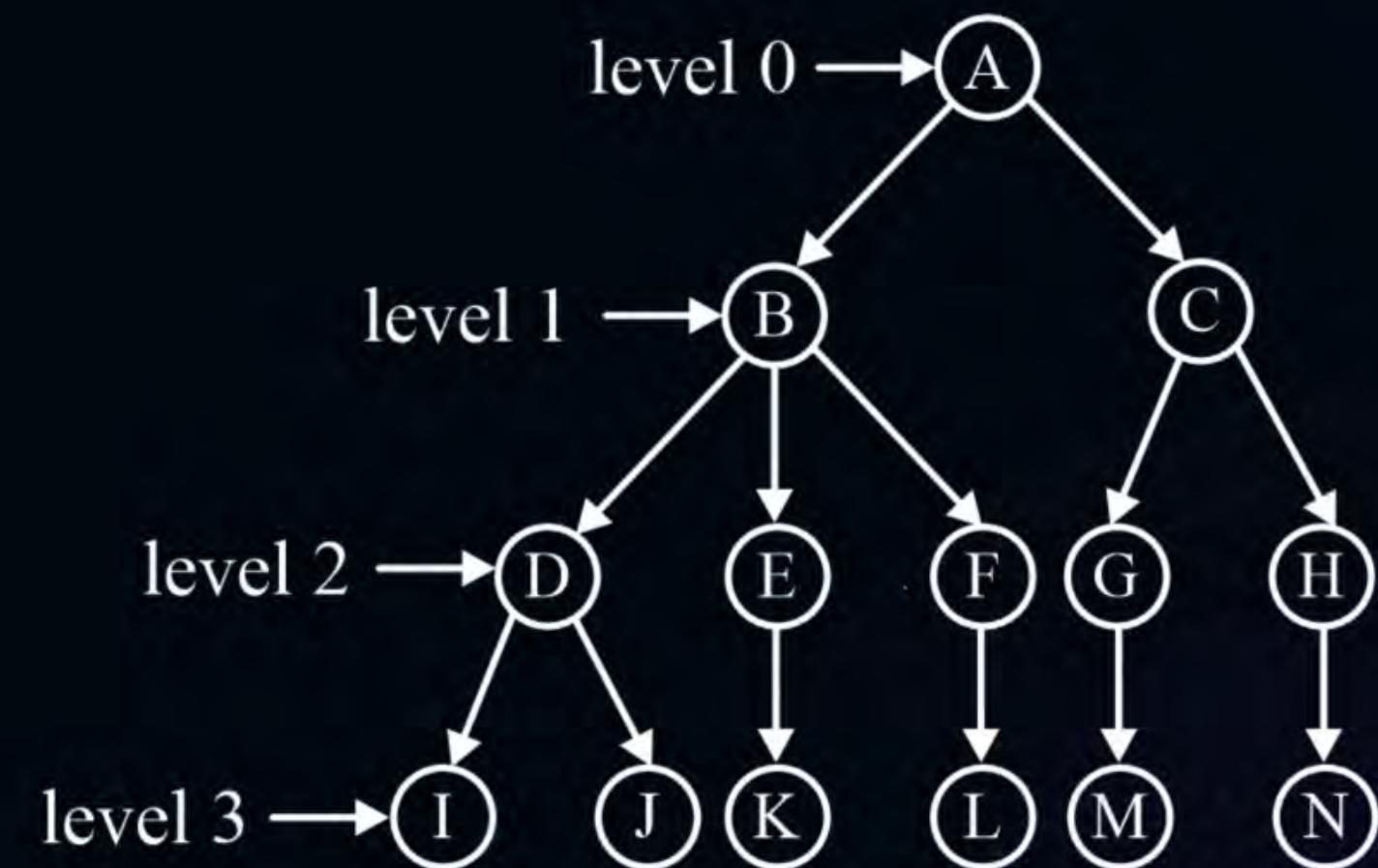


**Iterative Deepening Search (IDS) or Iterative Deepening Depth First Search (IDDFS):**

**Goal node (alpha)**

I, H

- BFS
- ~~IDS DFS~~
- IDDFS





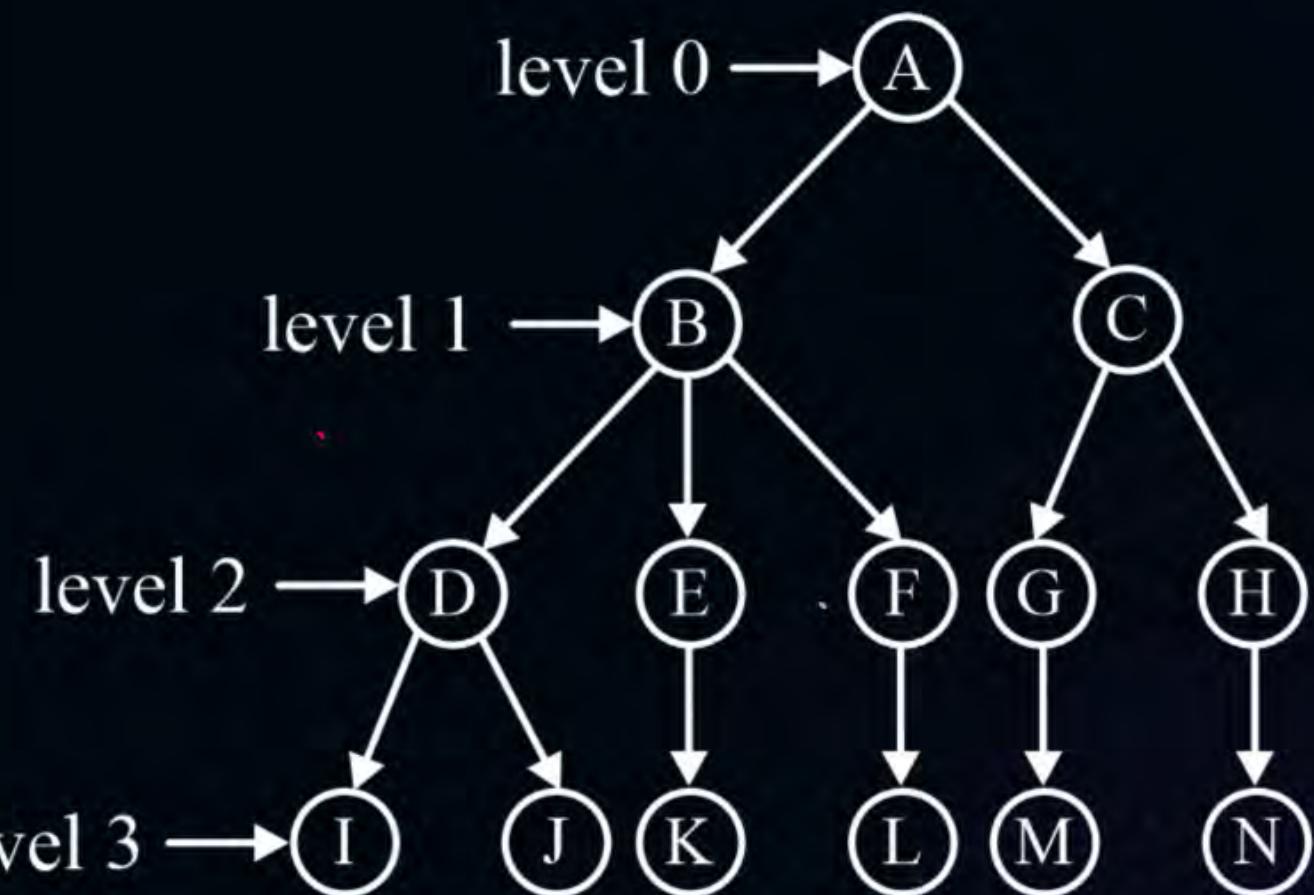
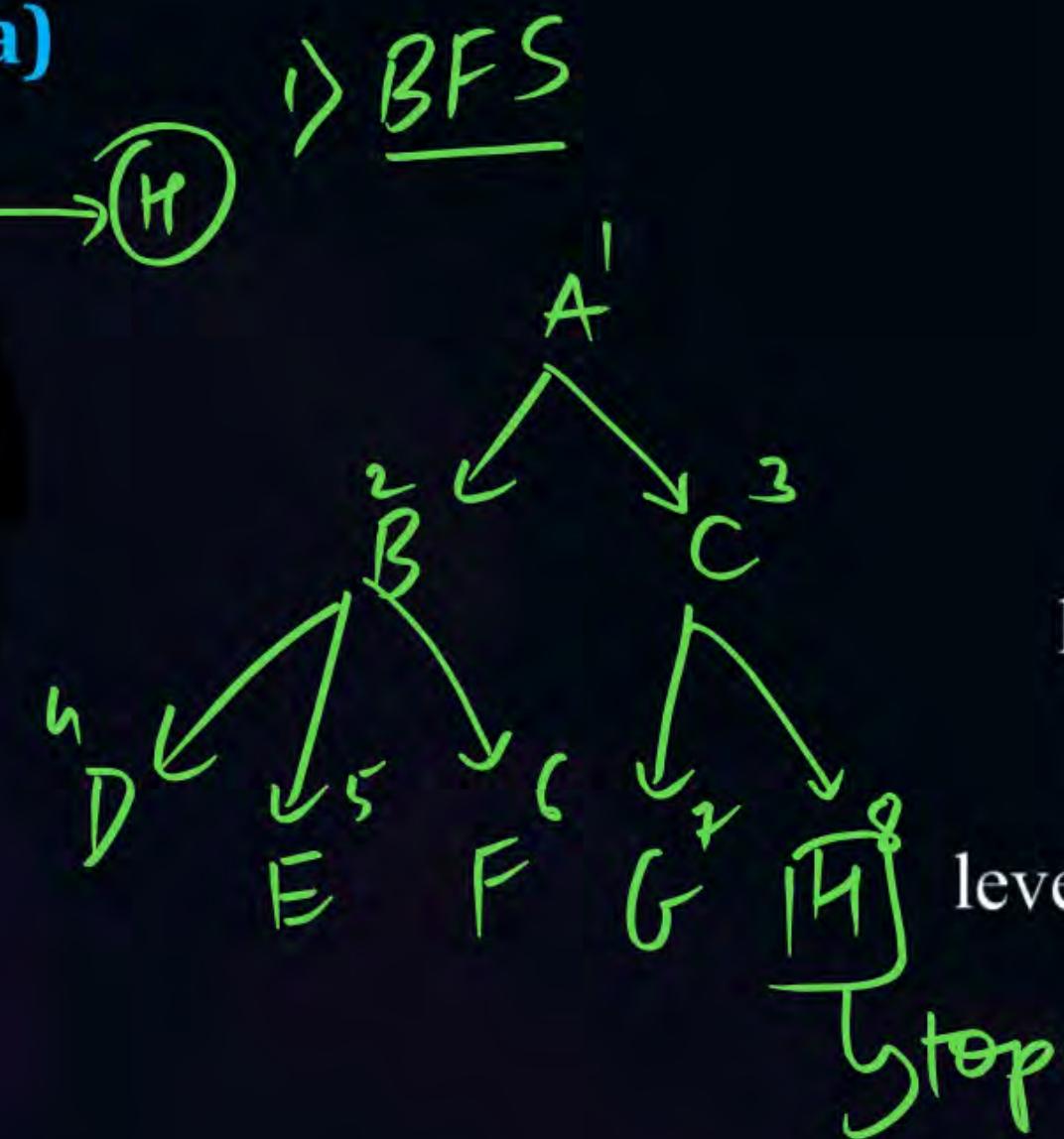
# Topic : Uninformed Search



**Iterative Deepening Search (IDS) or Iterative Deepening Depth First Search (IDDFS):**

**Goal node (alpha)**

- ~~I, H~~
- BFS
- ~~IDS DFS~~
- IDDFS





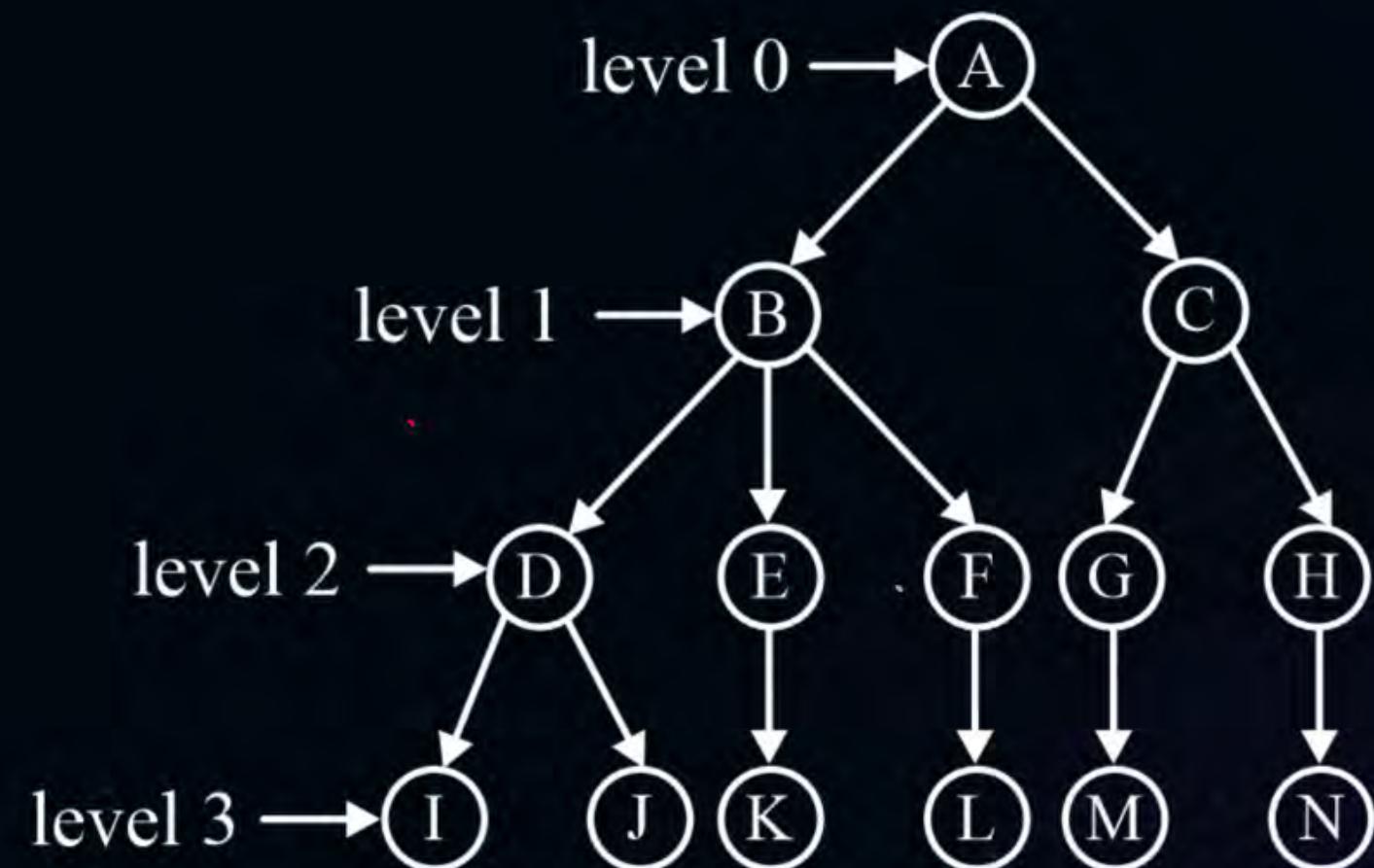
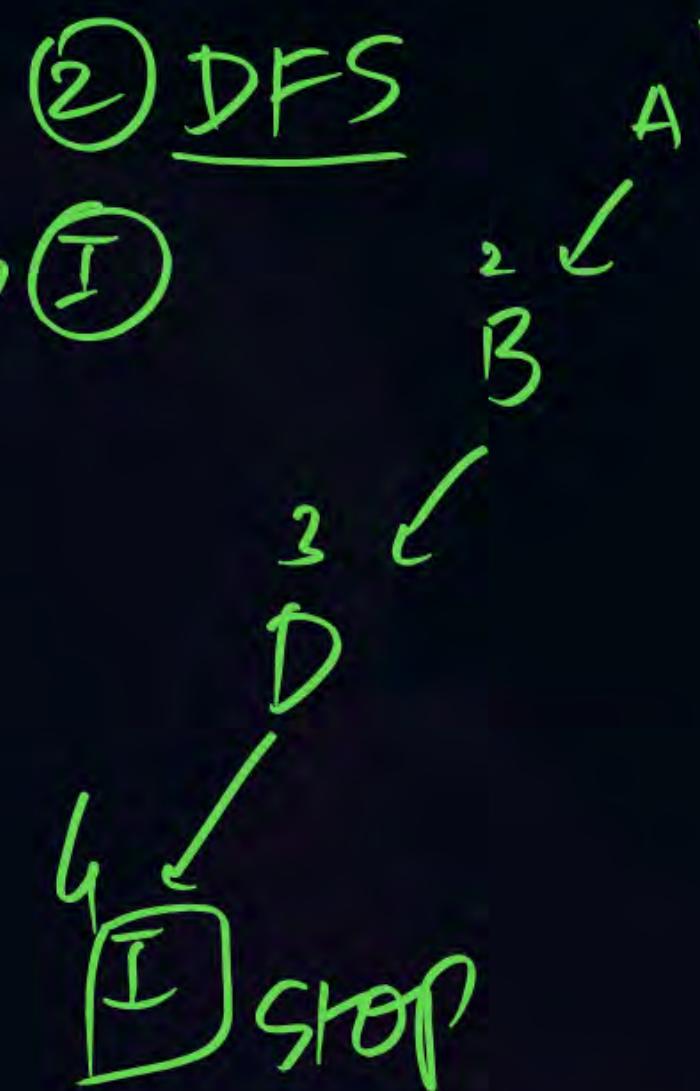
# Topic : Uninformed Search



**Iterative Deepening Search (IDS) or Iterative Deepening Depth First Search (IDDFS):**

**Goal node (alpha)**

- ~~I, H~~
- BFS
- ~~IDS~~ DFS → I
- IDDFS





# Topic : Uninformed Search



Iterative Deepening Search (IDS) or Iterative Deepening Depth First Search (IDDFS):

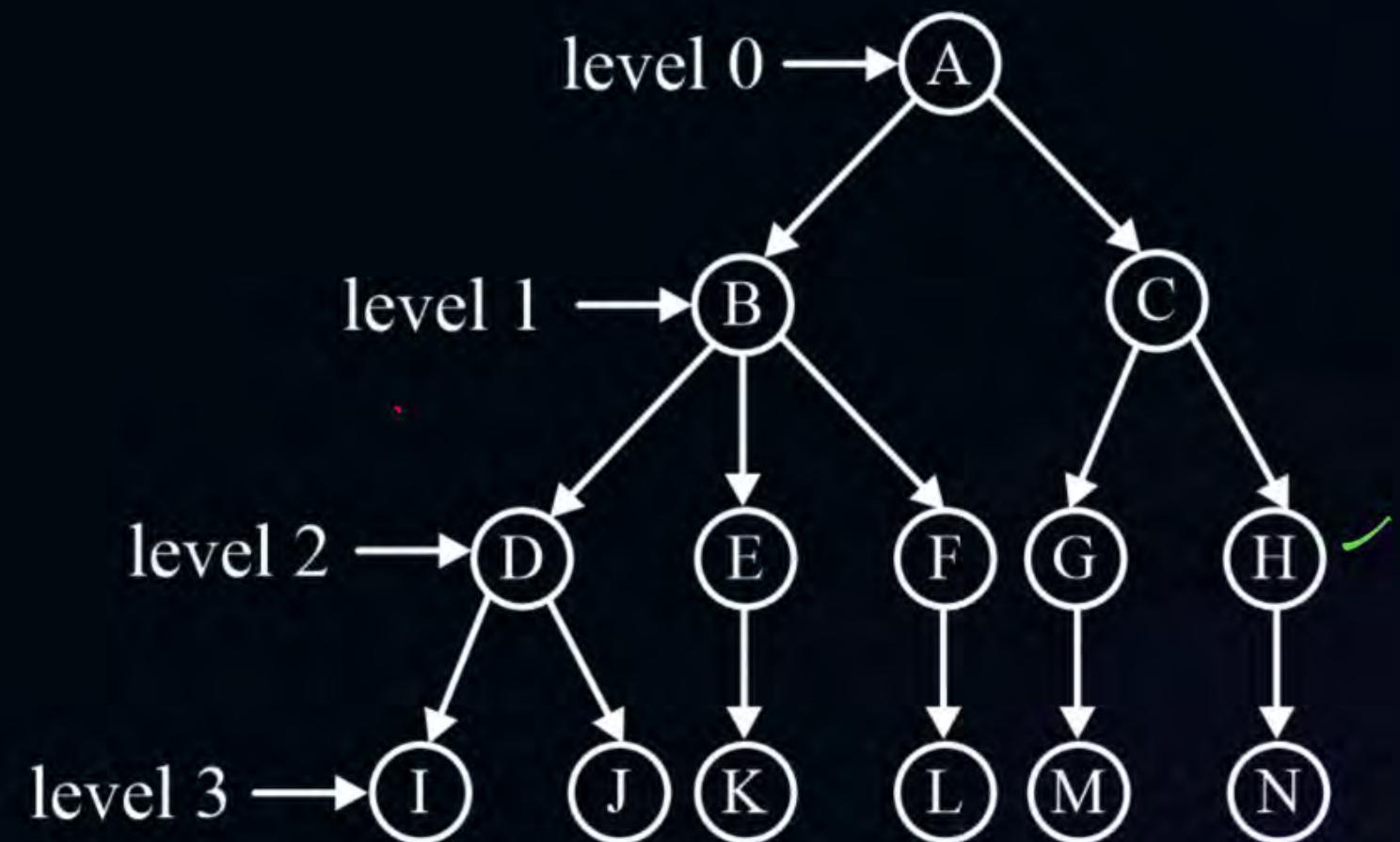
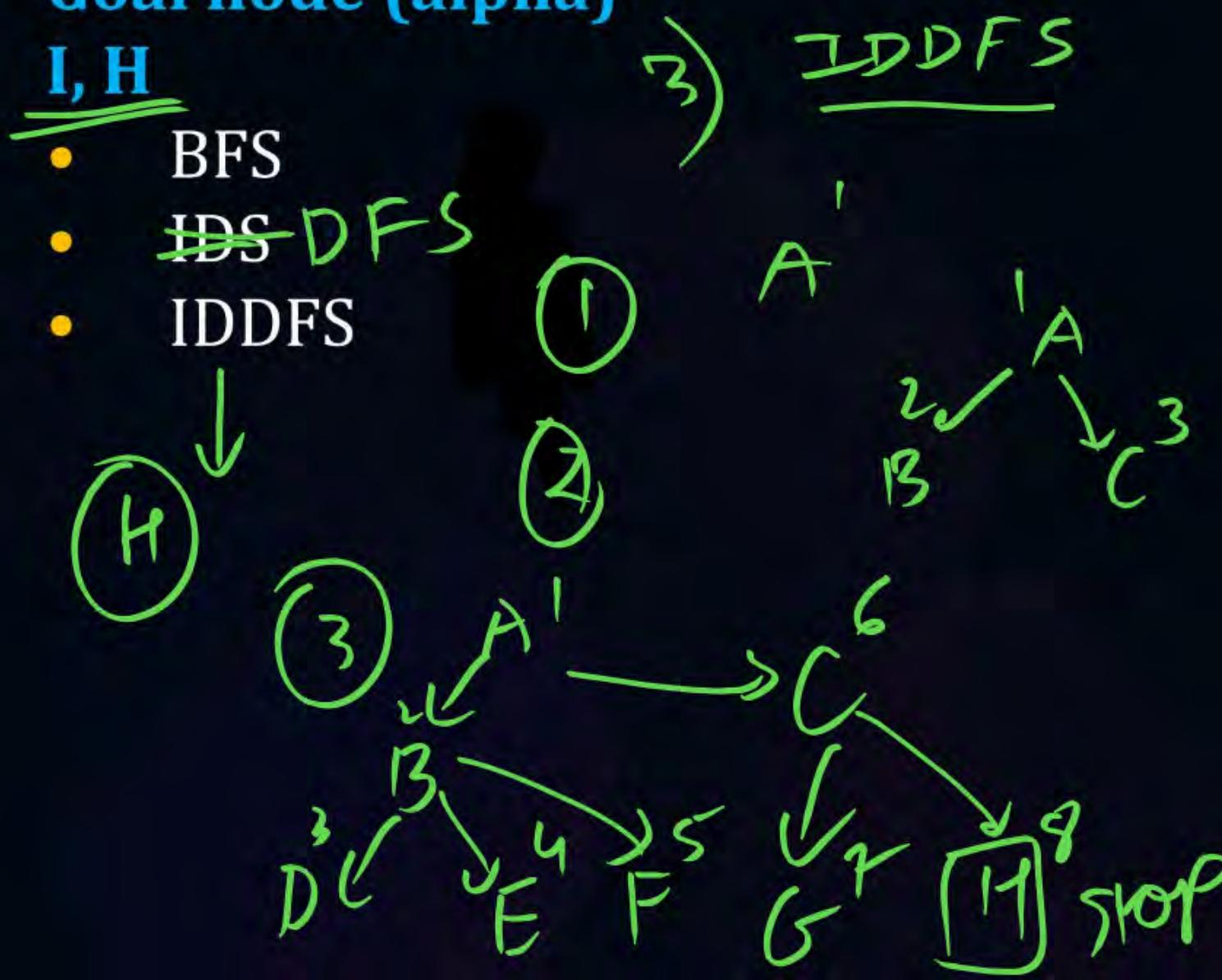
Goal node (alpha)

I, H

BFS

~~IDS~~ DFS

IDDFS





## Topic : Uninformed Search

**Iterative Deepening Search (IDS) or Iterative Deepening Depth First Search (IDDFS):**

1. Completeness
  2. Optimal result
  3. Less space complexity
- BFS      DFS

IDDFS:

- DFS → less space complexity
- Result BFS → advantage optimum result.



## Topic : Uninformed Search

### Iterative Deepening Search (IDS) or Iterative Deepening Depth First Search (IDDFS):

- Iterative Deepening Depth-First Search (IDDFS) combines the depth-first search's space efficiency with the breadth-first search's completeness. It repeatedly performs depth-limited searches with increasing depth limits until the goal is found.





## Topic : Uninformed Search

### Uniform Cost search:

Uniform Cost Search (UCS) is a search algorithm used to find the least-cost path from a start node to a goal node in a graph. UCS is a variant of Dijkstra's algorithm and is used in situations where the path costs vary, making it more suitable for weighted graphs.



# Topic : Uninformed Search

## Uniform Cost search:

### Key Concepts:

- Priority Queue UCS uses a priority queue to explore the least-cost path first. Nodes are expanded in order of their cumulative cost from the start node.
- Cost: Each edge in the graph has a cost associated with it. UCS aims to find the path with the minimum cumulative cost.
- Expansion: The algorithm expands the node with the smallest cumulative cost, ensuring that the cheapest path is always chosen.

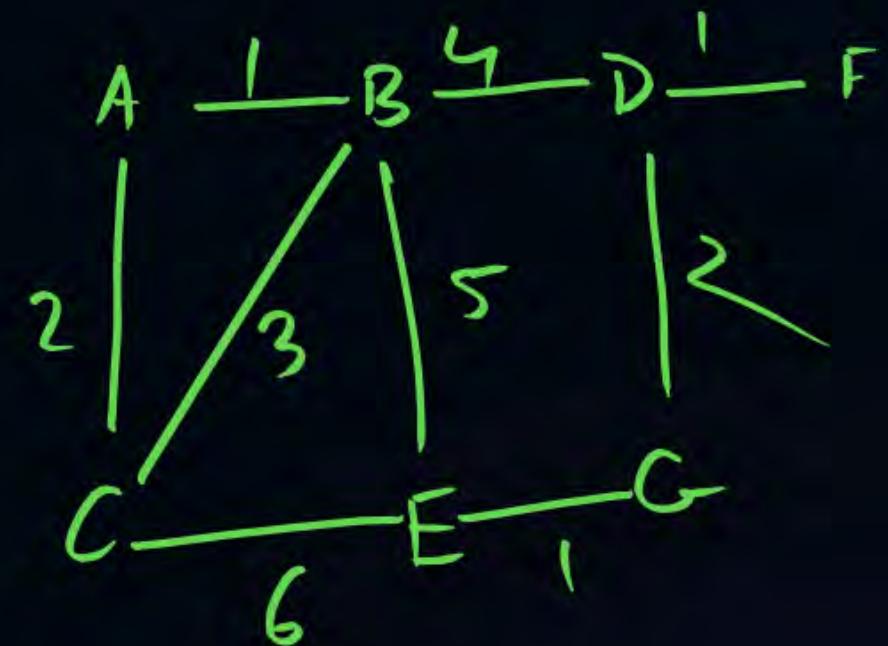
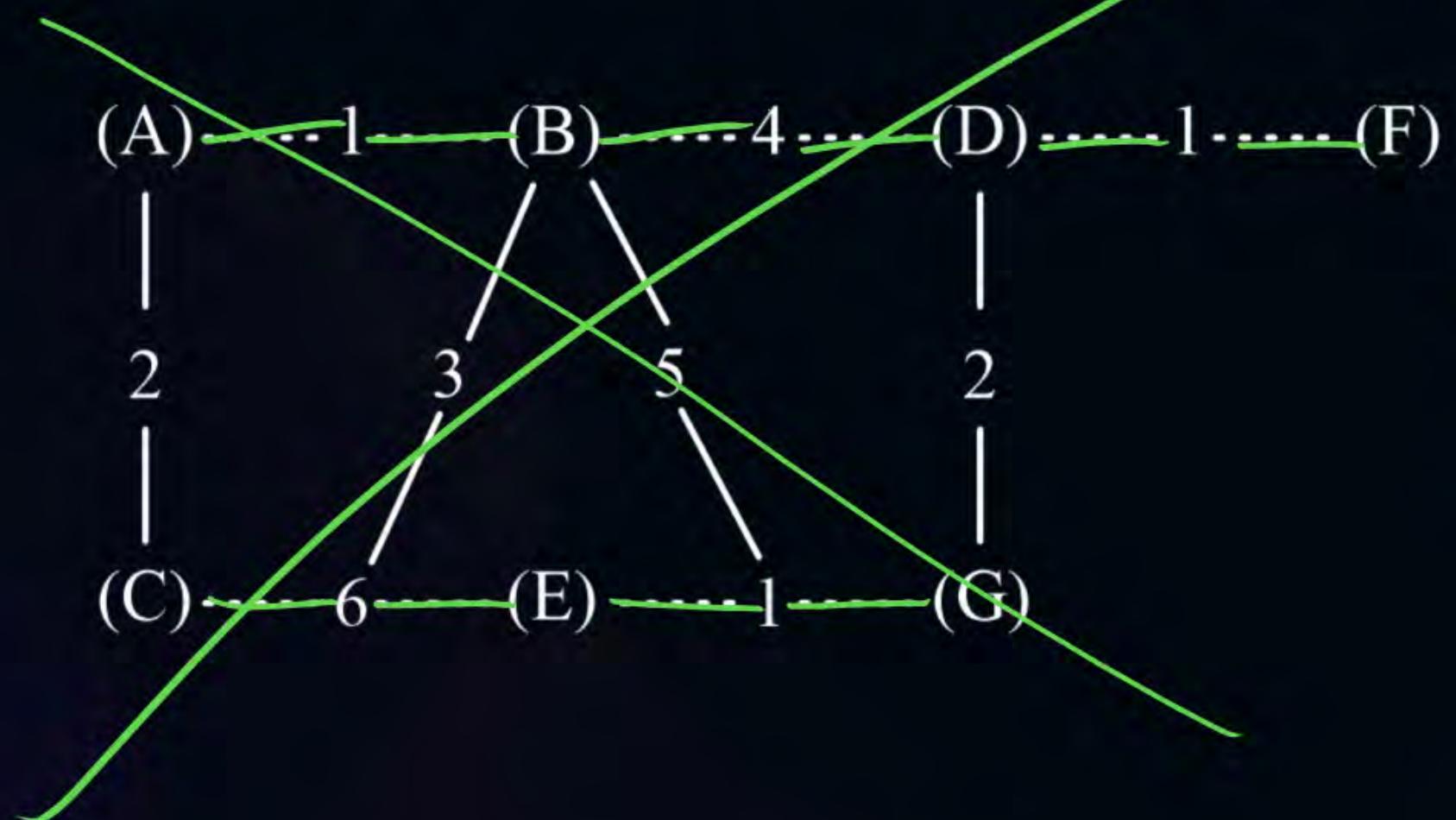




# Topic : Uninformed Search

## Uniform Cost search:

Here we use a Priority Queue.

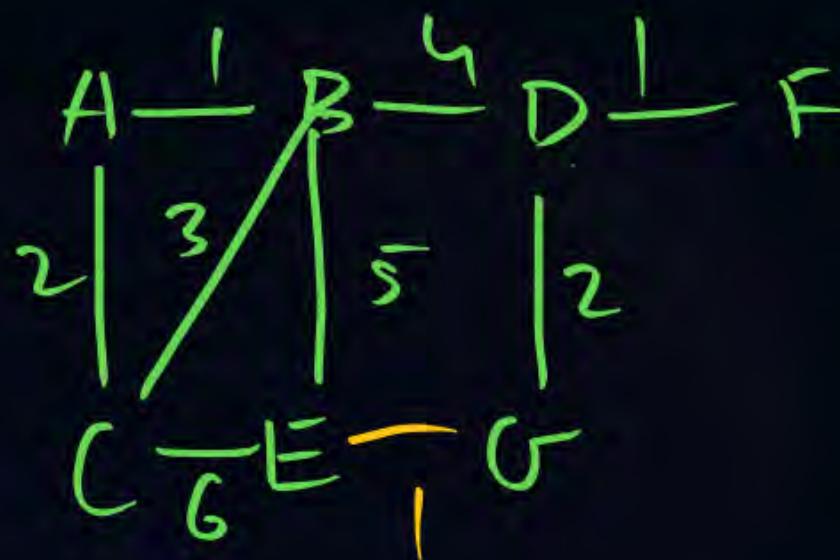




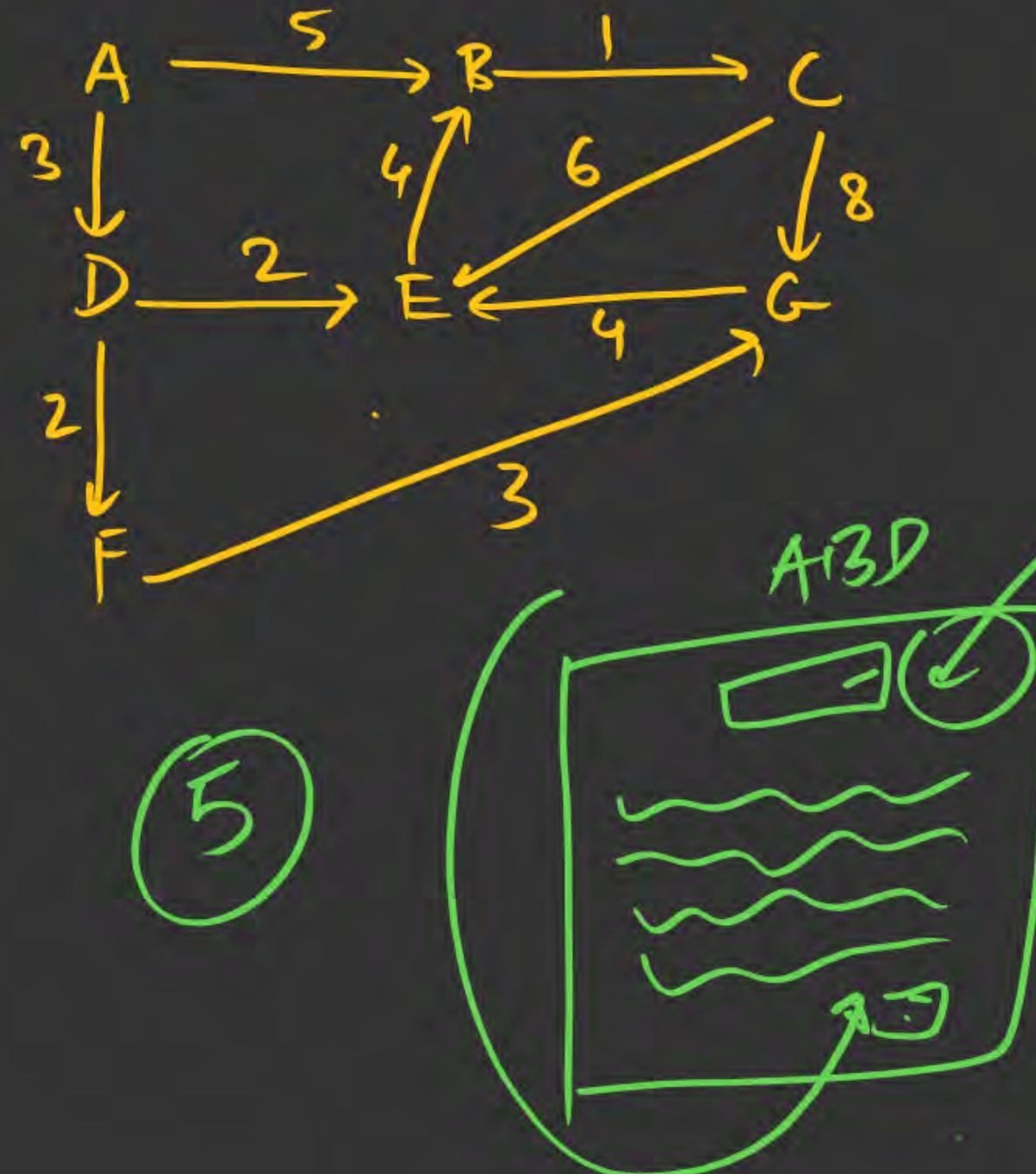
# Topic : Uninformed Search

## Uniform Cost search:

Your task is to find the least cost path from city A to city G using Uniform Cost Search (UCS).



|   | Open  | Closed      |
|---|-------|-------------|
| A | A     |             |
| B | 1     |             |
| C | 2     | 2           |
| D | x 5   | 5           |
| E | x 6   | 6           |
| F | x x x | 6 7 7 7 7   |
| G |       | 6 6 6 6 6 ✓ |



VCS

A  $\rightarrow$  G

|   | Open    | Closed  |  |
|---|---------|---------|--|
| A |         | A       |  |
| B | 5       | 5       |  |
| D | 3       | 3       |  |
| E |         | 3       |  |
| F | X       | 5       |  |
| C | X       | 5       |  |
| G | X X X X | 6 6 6 6 |  |

Cost: 8

$D \rightarrow F \rightarrow G$

ADF(G)

$8 \rightarrow \underline{\text{goal}}$

$$TC \& SC \text{ of } \underline{UCS} \\ = O(b^{(C^*/\epsilon)})$$

$C^*$  = approx optimal path cost of the Soln.

$\epsilon$  = smallest path weight in entire graph

UCS



If all paths are of equal cost ( $\epsilon_e$ )

$$C^* = (\text{depth of the graph} * \epsilon_e) \rightarrow \underline{\text{Worst Case}}$$
$$= d \times \epsilon_e$$

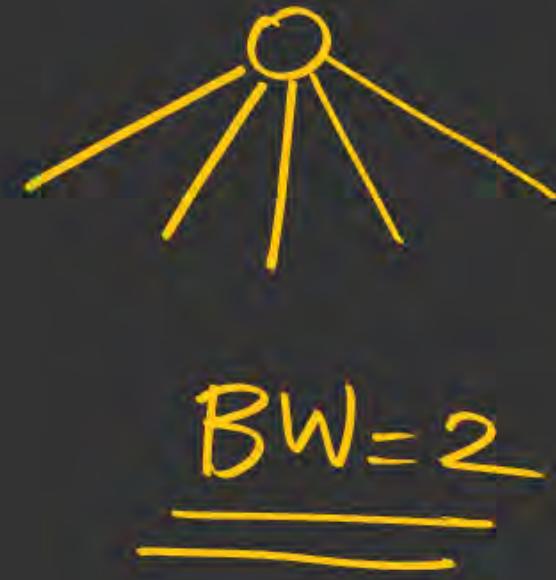
WC TC & SC of UCS =  $O(b^{d * \epsilon_e / \epsilon_e}) = \underline{\underline{O(b^d)}}$

BFS

## Beam Search:

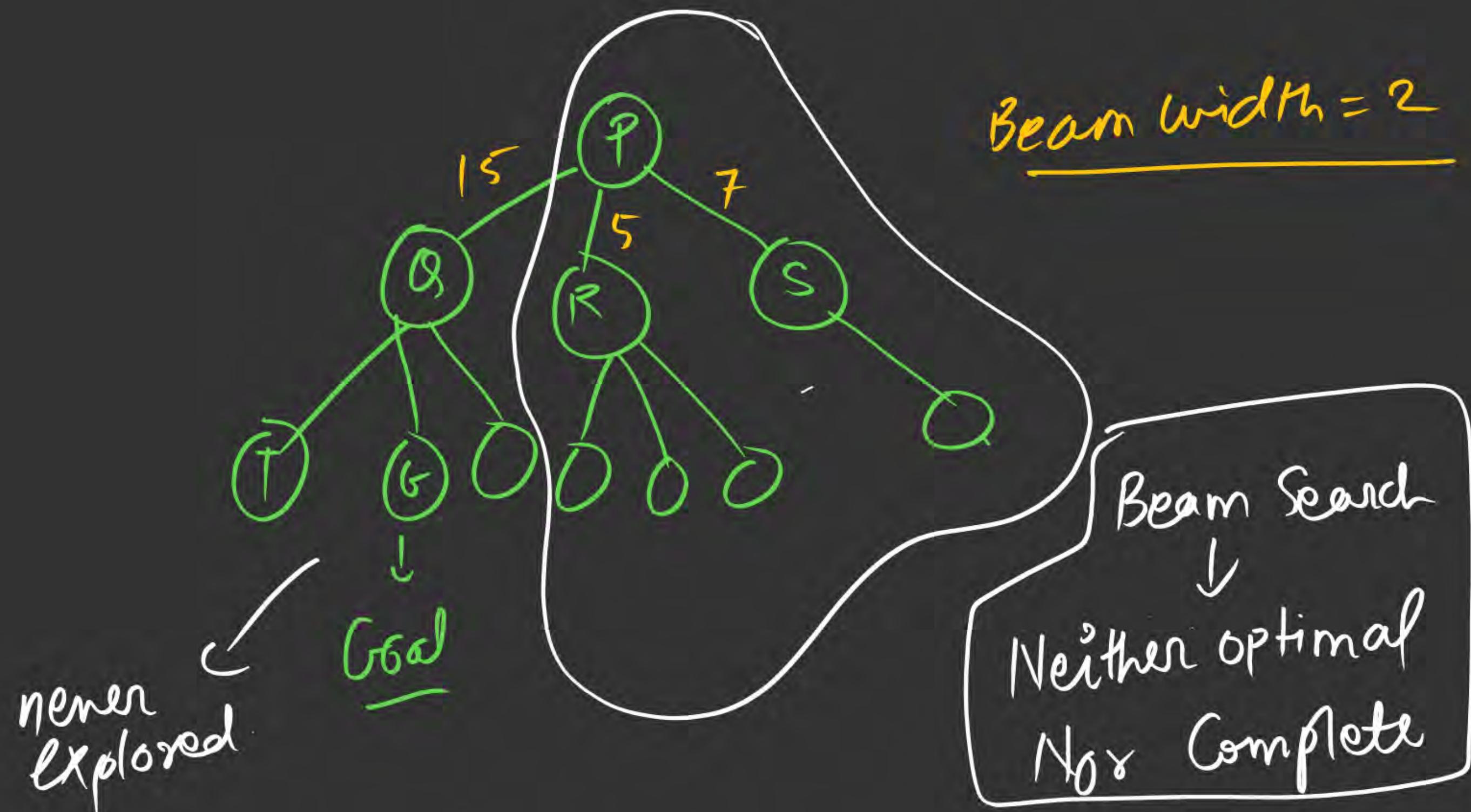


UCS → all childs of visited node  
↓  
Select the one with min cost.  
Large SC due to many childs (larger b)

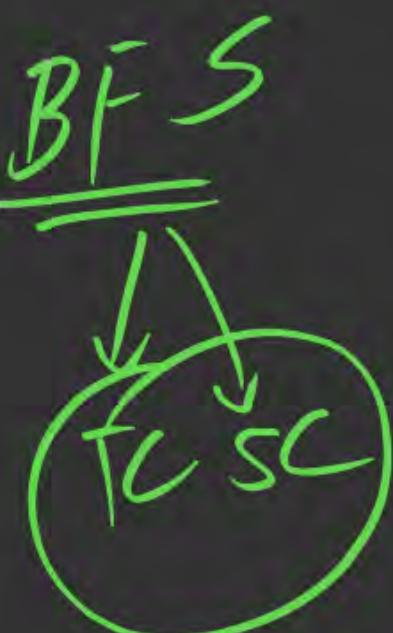


Beam Search → [Beam width]

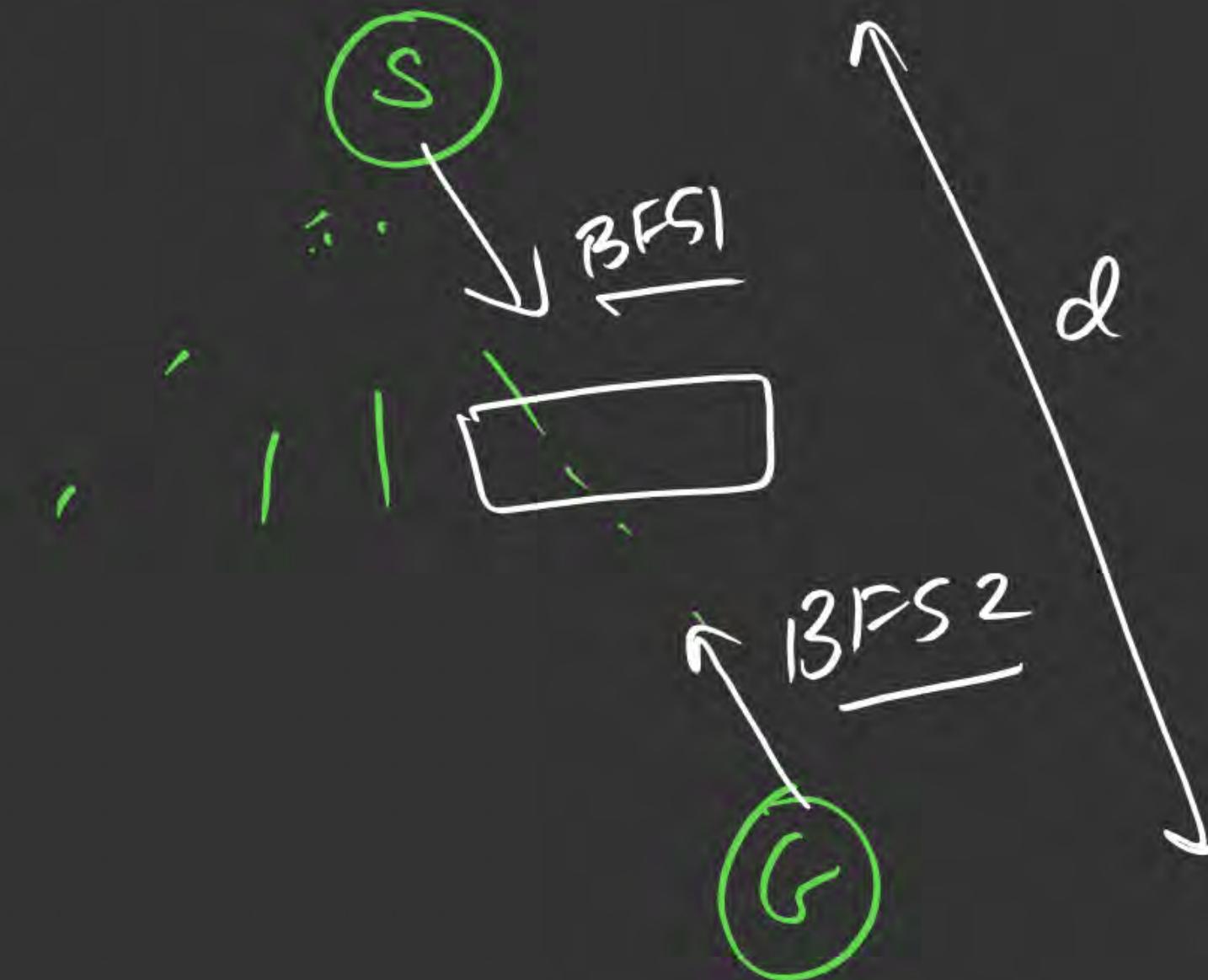
limit child and prefer the ones with min cost.



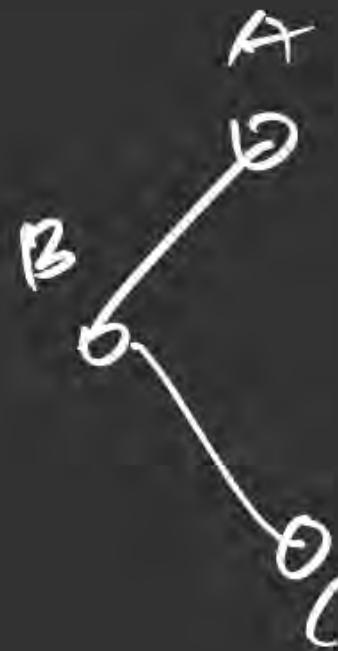
## \* Bidirectional Search :

BFS   $\rightarrow \underline{\underline{O(b^d)}}$  due to large  
number of visited nodes

Bidirectional  
Search



$$\frac{2^8 \rightarrow 2^4}{}$$



TC & SC of Bidirectional Search:

Graph depth = d

Both BFS from S & G will meet at d/2

$$TC \& SC = \underbrace{O(b^{d/2} + b^{d/2})}_{= O(b^{d/2})}$$

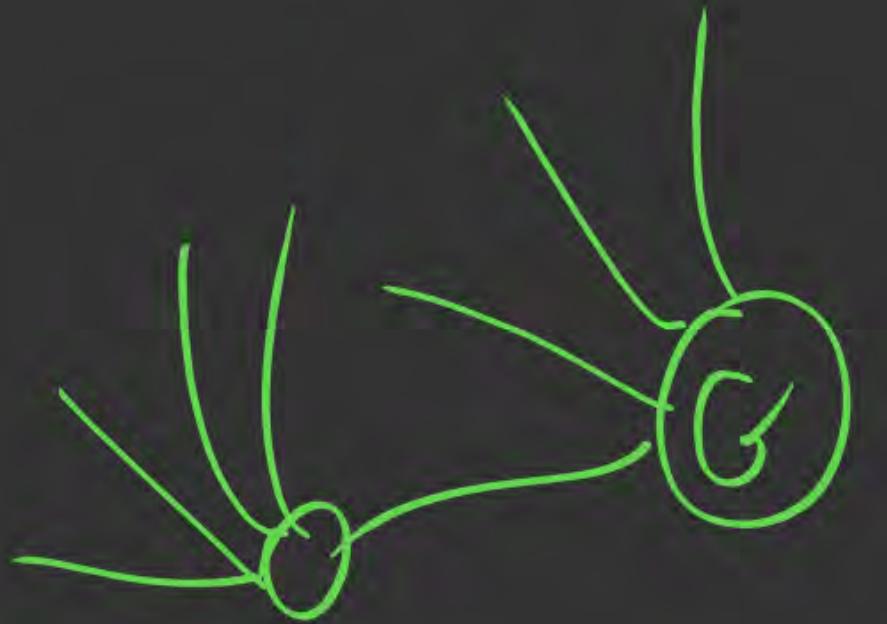
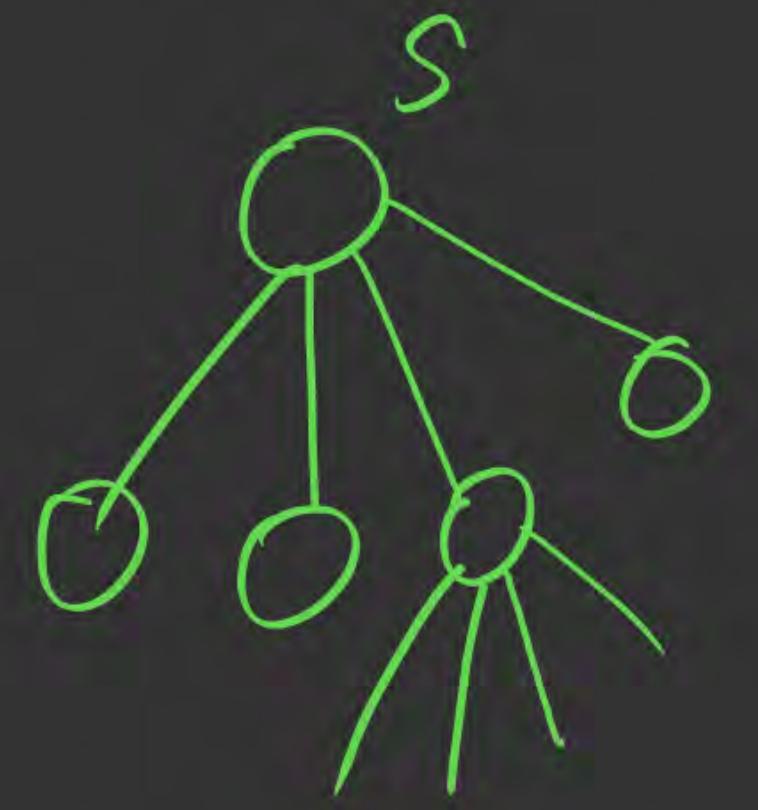
## Bidirectional Search:

Adv:

- ① TC & SC reduced  $\Rightarrow \underline{O(b^{d/2})}$

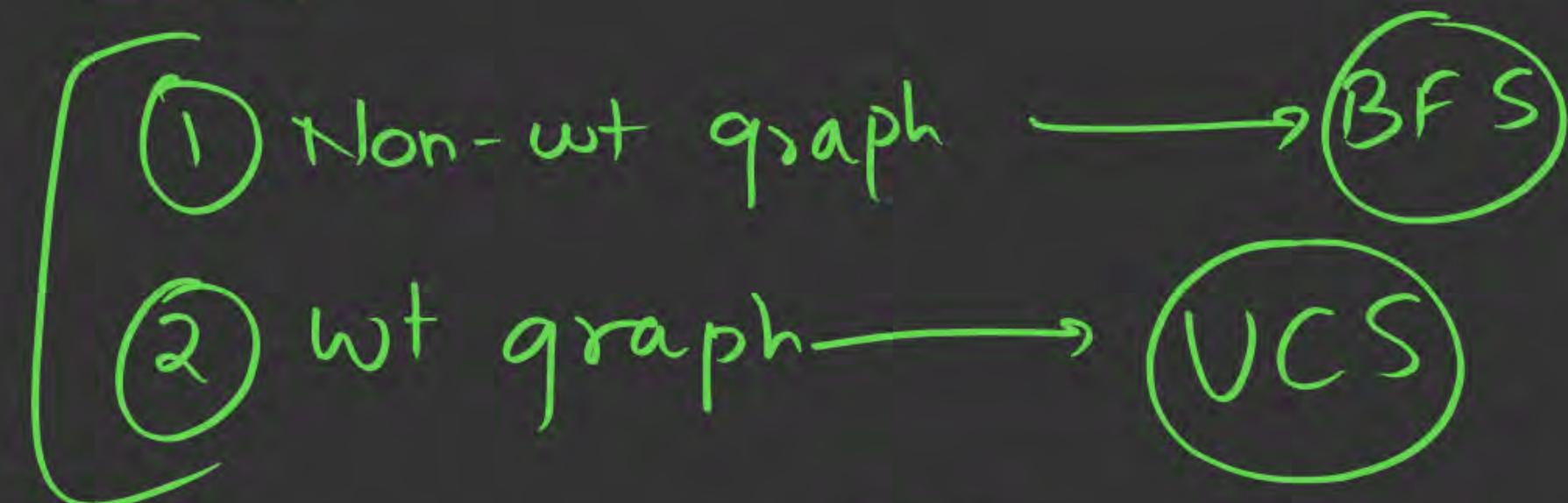
Disadvantages:

- ① Presence of multiple goals
- ② Should be able to have operation to move to parent nodes



Uninformed

Types :





**THANK - YOU**



# DS & AI ENGINEERING



## Artificial Intelligence

### Informed Search

Lecture No.- 01

By- Aditya sir



# Recap of Previous Lecture



Topic

Topic

Topic

Topic

Uninformed Search

# Topics to be Covered

P  
W



Topic

Informed Search

→ GBFS

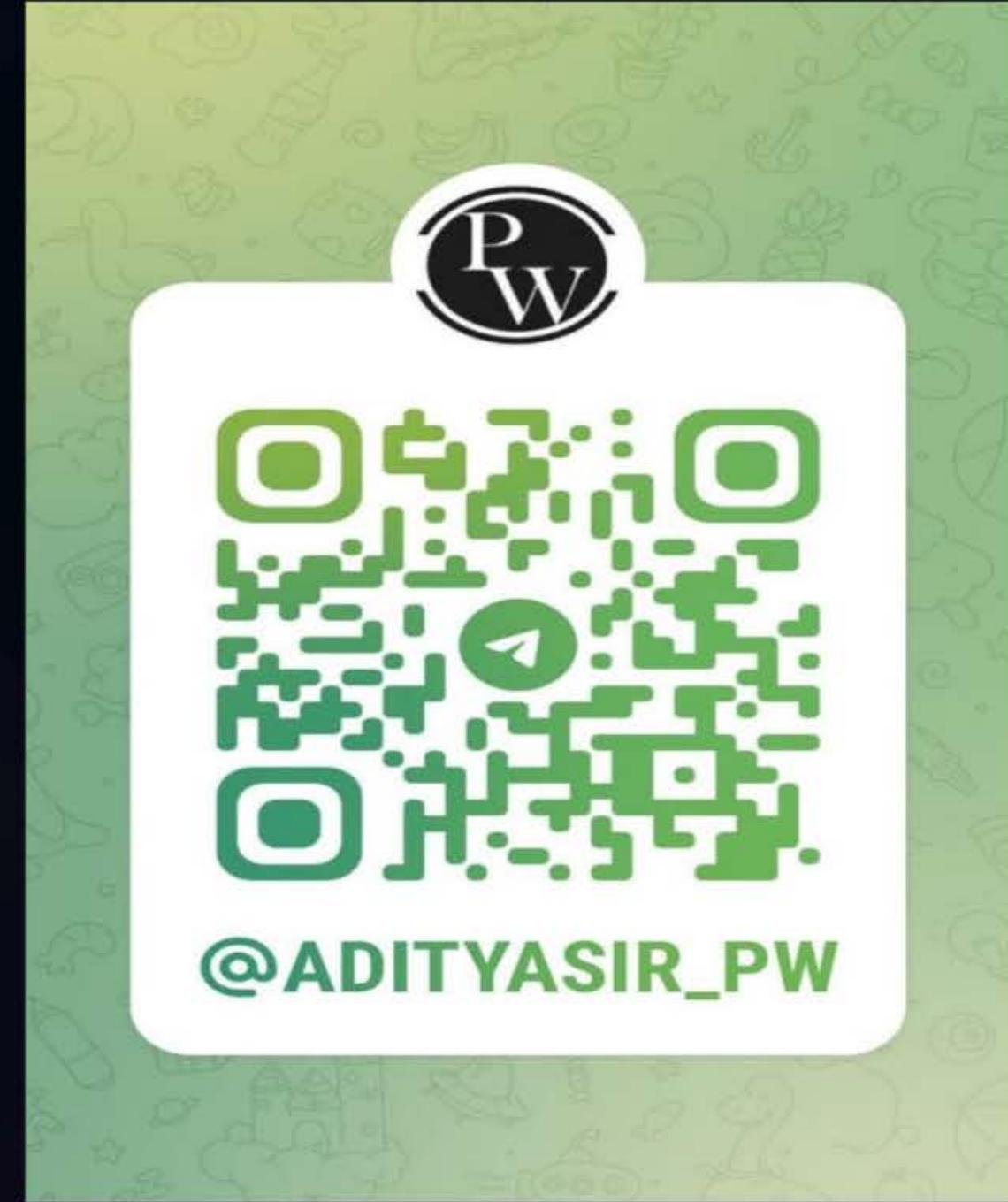
→ A\* Search



## About Aditya Jain sir

1. Appeared for GATE during BTech and secured AIR 60 in GATE in very first attempt - City topper
2. Represented college as the first Google DSC Ambassador.
3. The only student from the batch to secure an internship at Amazon. (9+ CGPA)
4. Had offer from IIT Bombay and IISc Bangalore to join the Masters program
5. Joined IIT Bombay for my 2 year Masters program, specialization in Data Science
6. Published multiple research papers in well known conferences along with the team
7. Received the prestigious excellence in Research award from IIT Bombay for my Masters thesis in ML
8. Completed my Masters with an overall GPA of 9.36/10
9. Joined Dream11 as a Data Scientist
10. Have mentored 15,000+ students & working professionals in field of Data Science and Analytics
11. Have been mentoring & teaching GATE aspirants to secure a great rank in limited time
12. Have got around 27.5K followers on Linkedin where I share my insights and guide students and professionals.



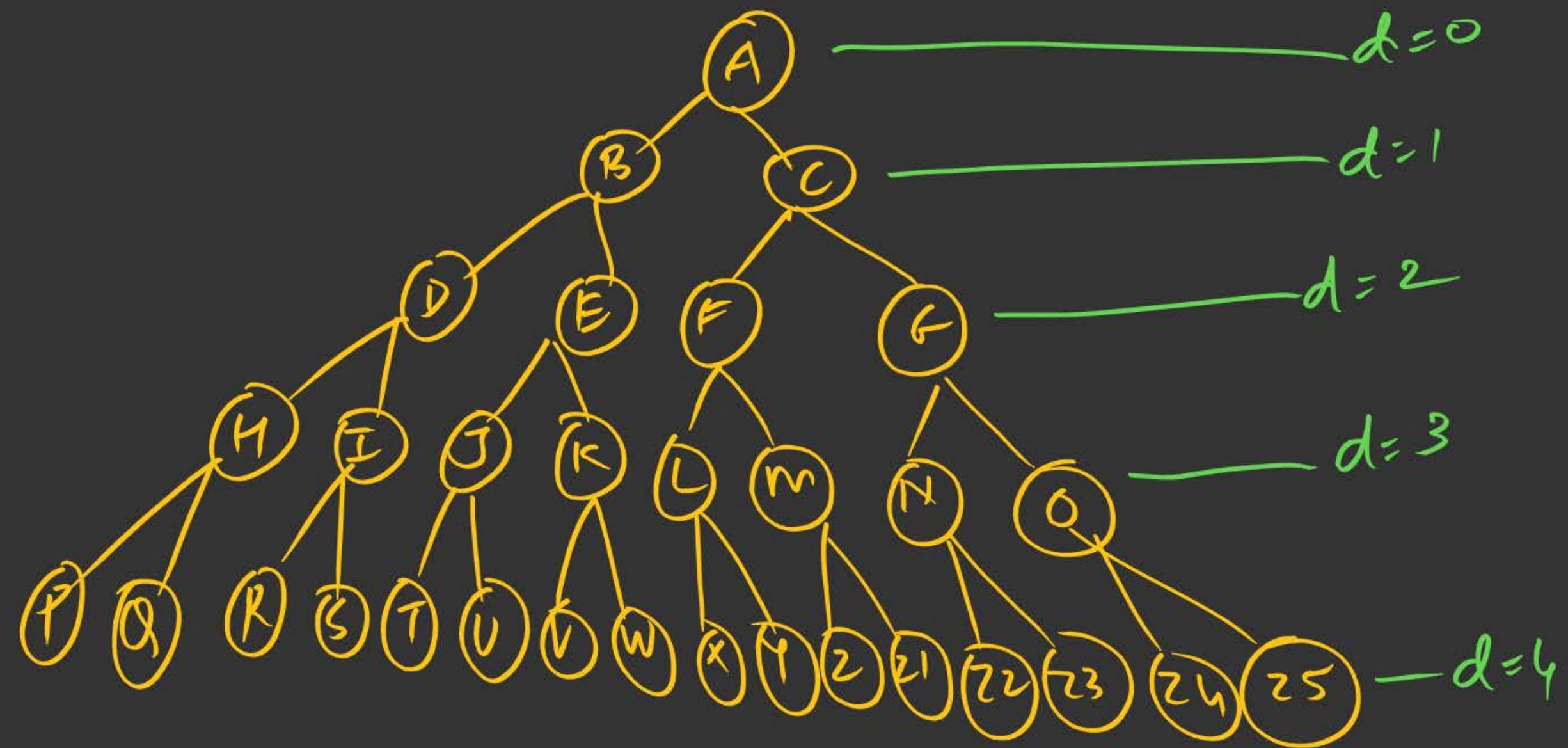


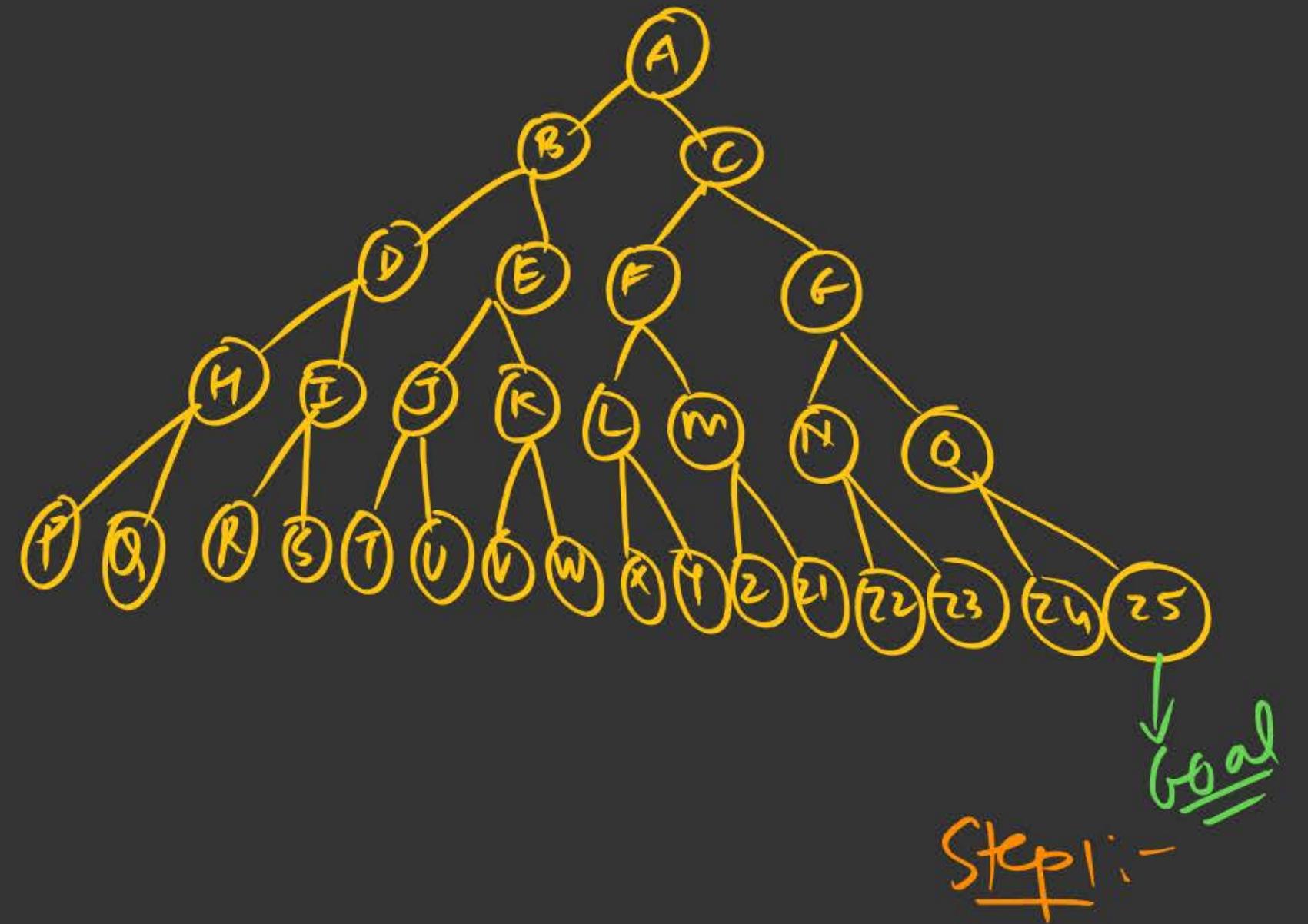
Telegram

Telegram Link for Aditya Jain sir:

[https://t.me/AdityaSir\\_PW](https://t.me/AdityaSir_PW)

Bidirectional Search Eg:



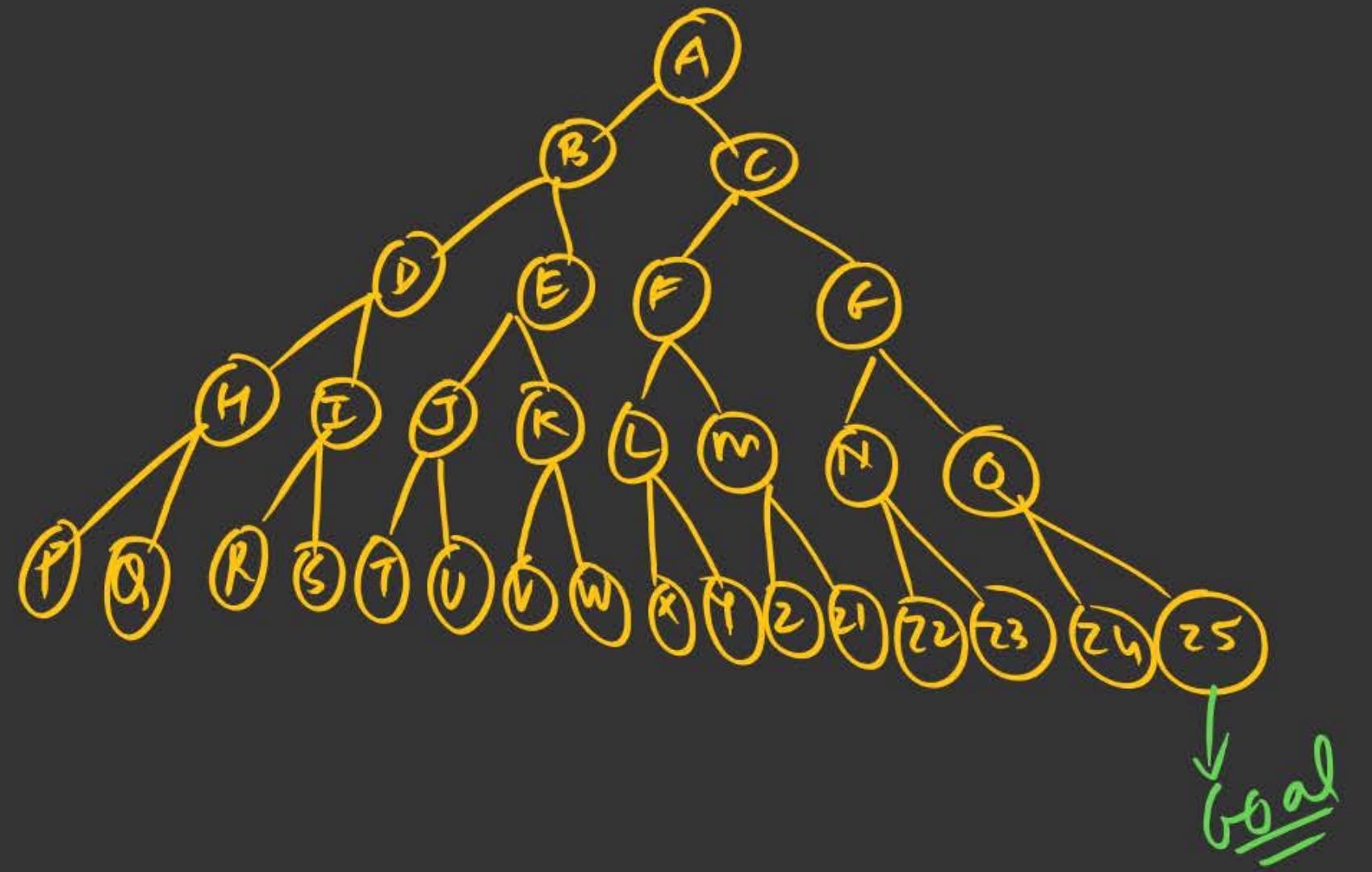


O<sub>1</sub>  
~~A B C~~

O<sub>1</sub>  
A  
C<sub>1</sub>

O<sub>2</sub>  
~~Z S O~~

O<sub>2</sub>  
Z<sub>5</sub>  
C<sub>2</sub>



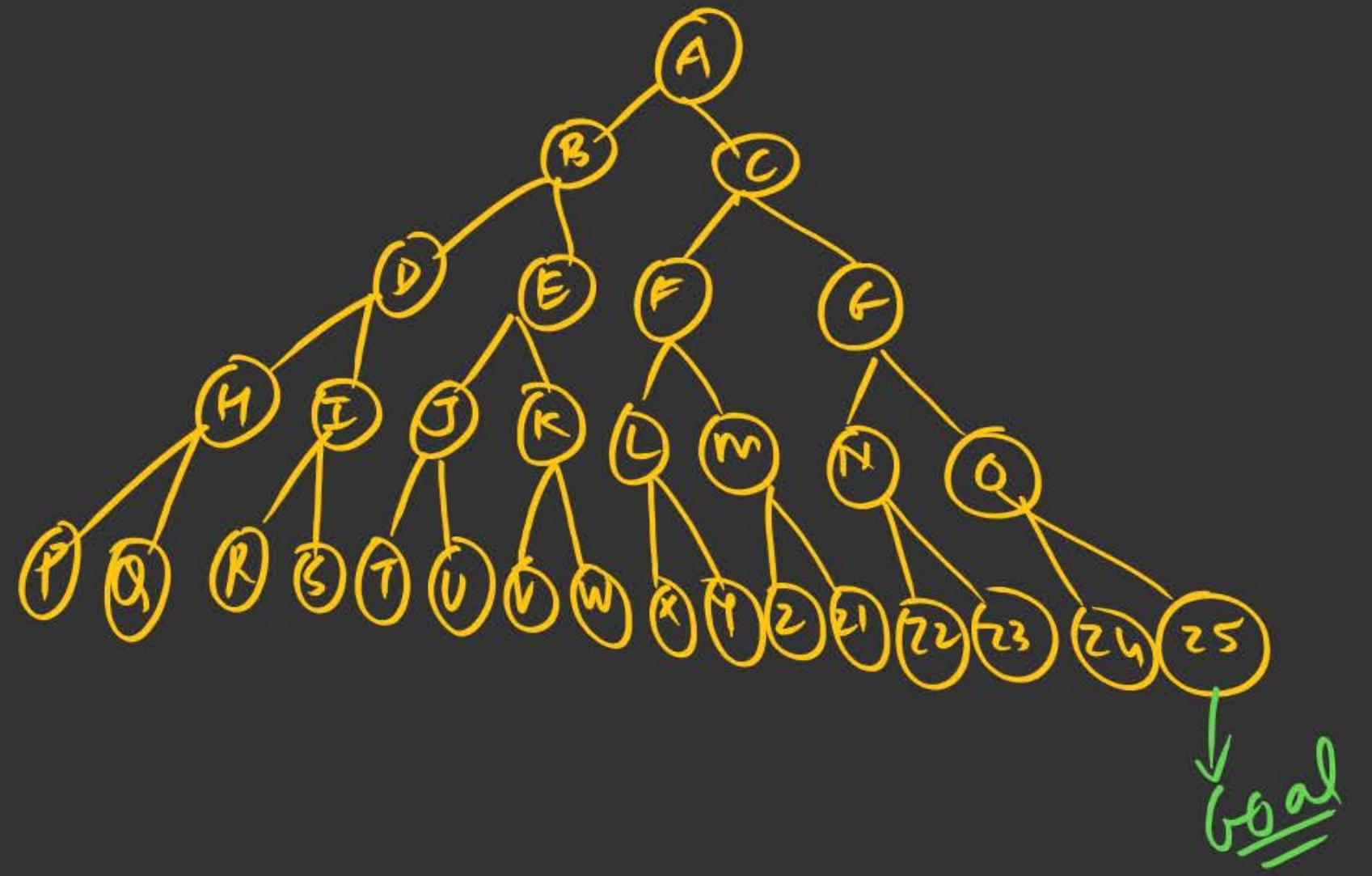
Step 2

O<sub>1</sub>  
~~A B C D E~~

O<sub>2</sub>  
~~Z<sub>5</sub> Z<sub>6</sub> G Z<sub>4</sub>~~

O<sub>1</sub>  
Z<sub>5</sub> Z<sub>6</sub>

O<sub>2</sub>  
Z<sub>5</sub> Z<sub>6</sub>



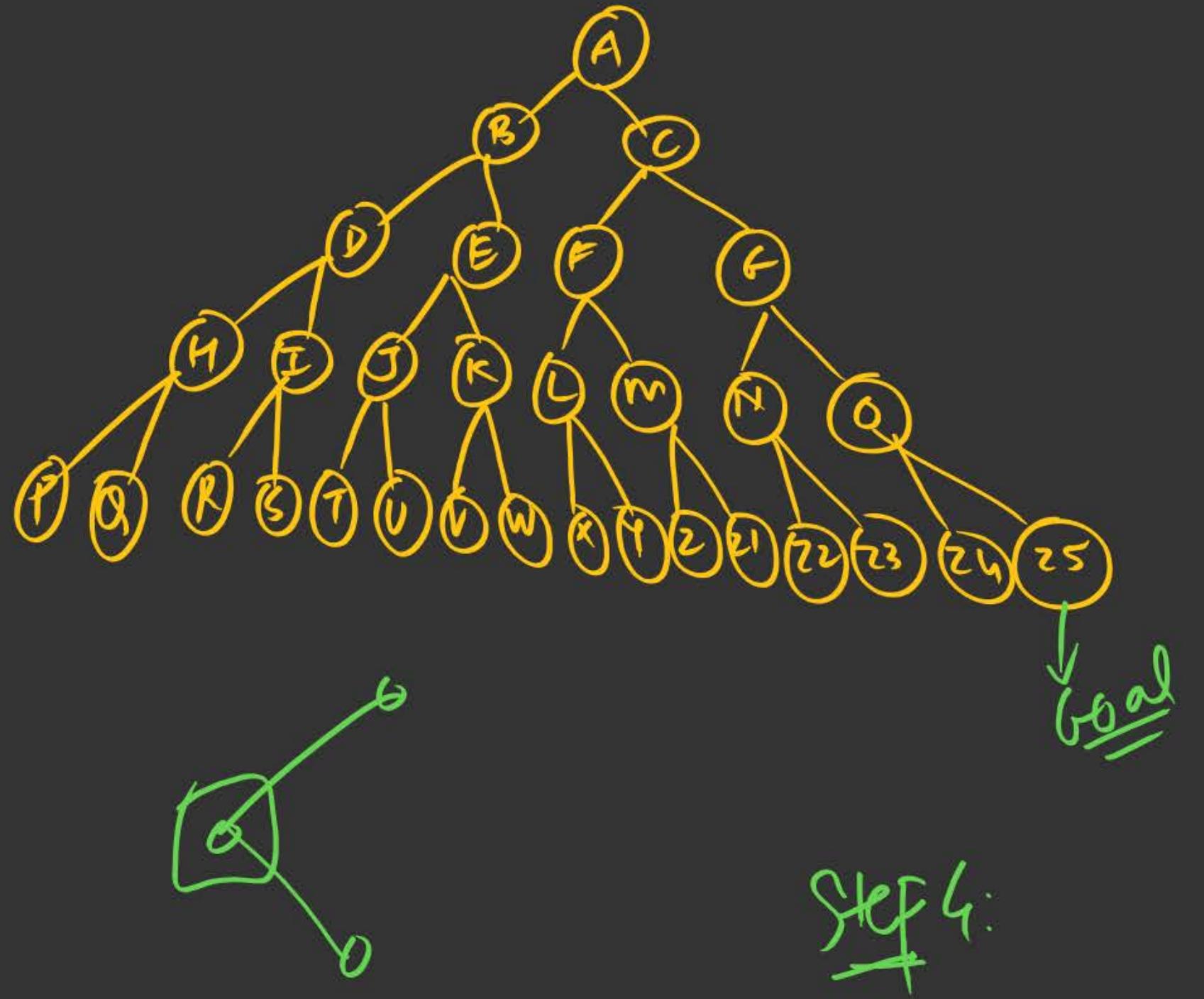
Step 3 :

O<sup>1</sup>  
~~A B C D E F G~~

C<sub>1</sub>  
 ABC

O<sup>2</sup>  
~~Z5 O D Z4 C N~~

C<sub>2</sub>  
 Z5 O G

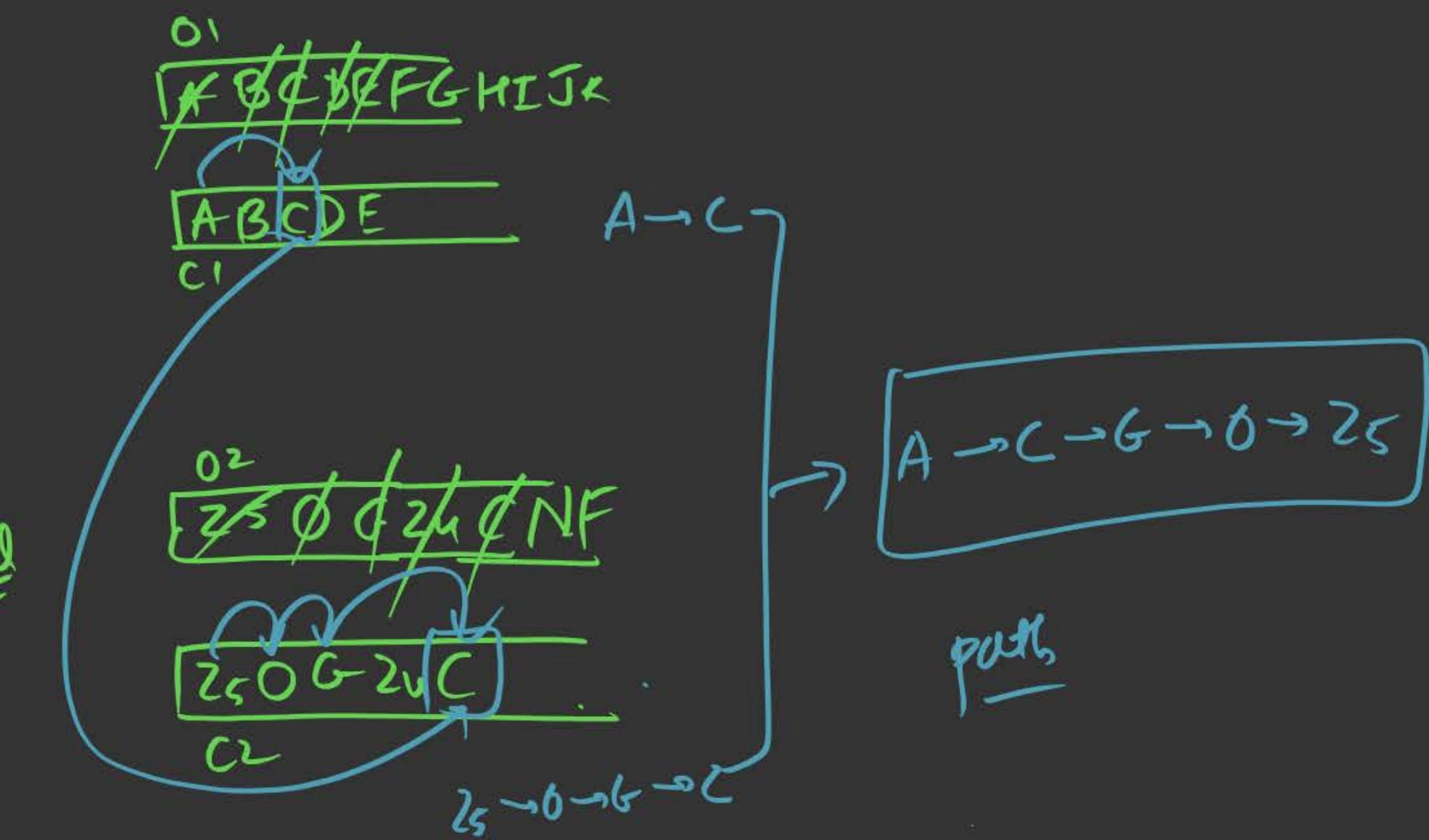
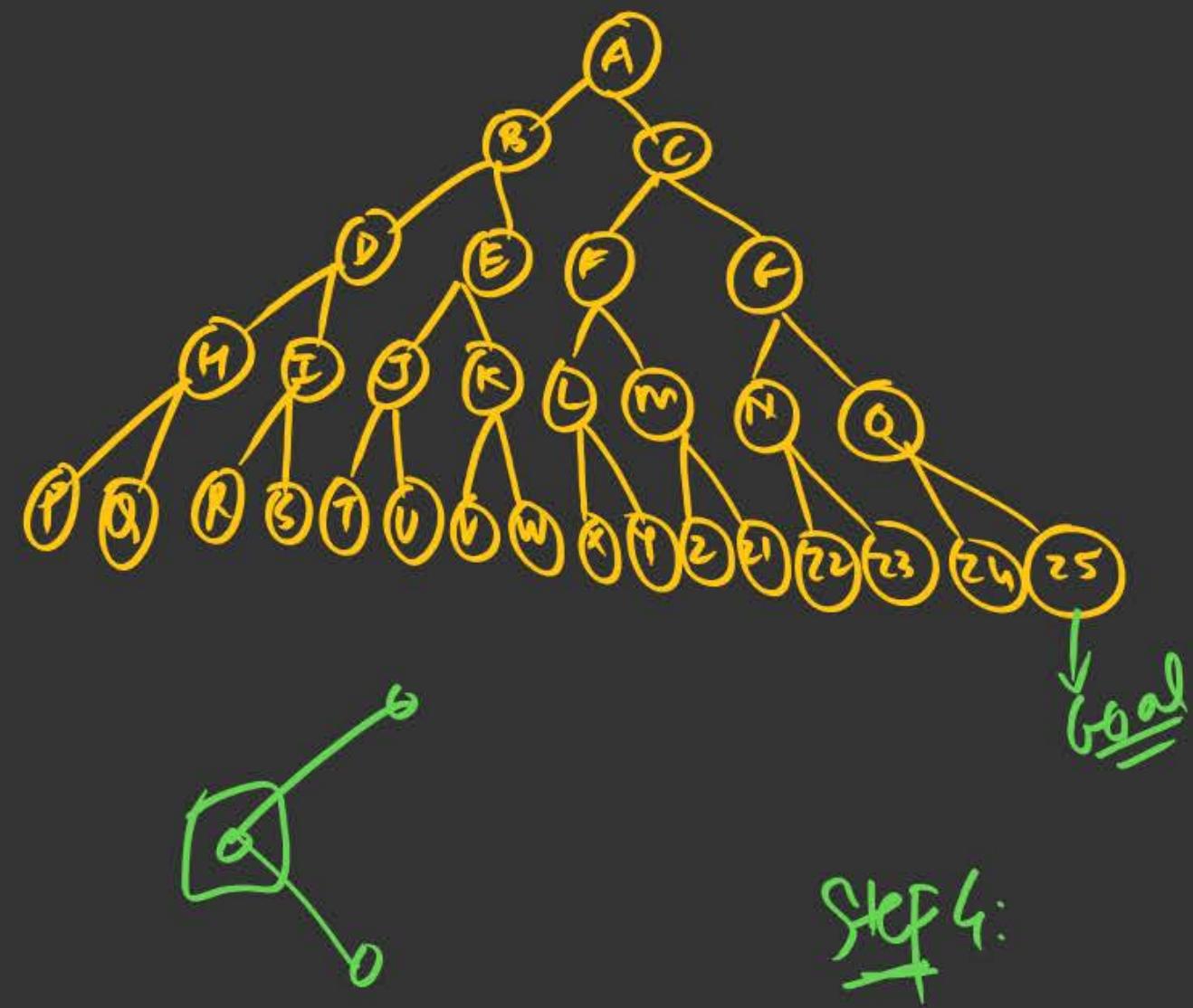


O<sub>1</sub>  
~~A B C D E F G H I~~

C<sub>1</sub>  
 ABCD

O<sub>2</sub>  
~~Z Y X W V U T S R Q P~~ C N

C<sub>2</sub>  
 Z Y X W V U T S R Q P



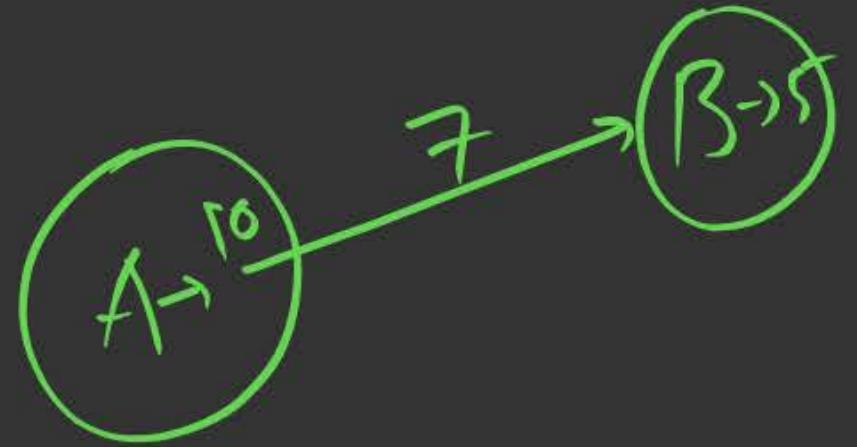
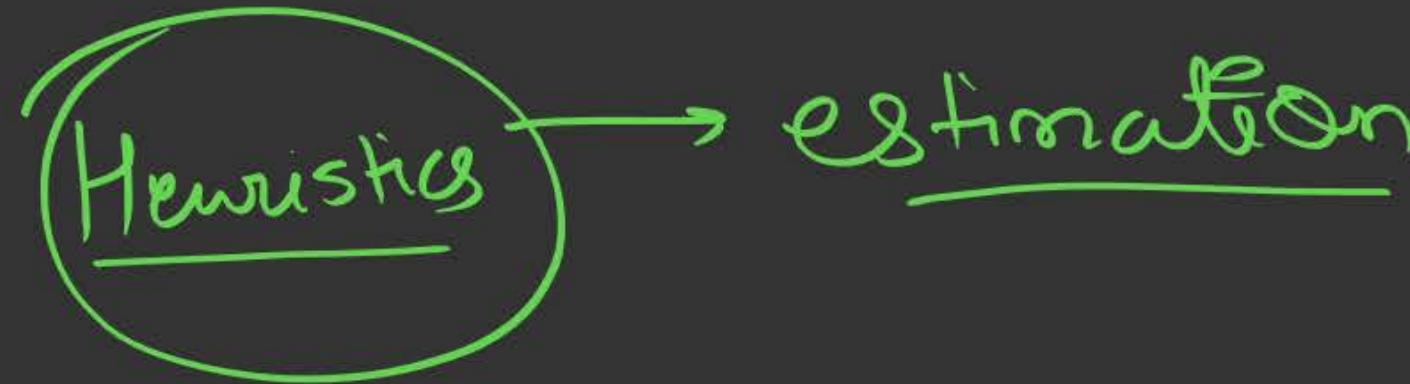
| Algo                    | Complete | Optimal | TC                    | SC                  |  |
|-------------------------|----------|---------|-----------------------|---------------------|--|
| 1) BFS                  | Yes      | Yes     | $O(b^d)$              | $O(b^d)$            |  |
| 2) DFS                  | No       | No      | $O(b^d)$              | $O(b \times d)$     |  |
| 3) UCS                  | Yes      | Yes     | $O(b^{C^*/\epsilon})$ | $O(b^{C^*/\alpha})$ |  |
| 4) Bidirectional Search | Yes      | Yes     | $O(b^{d/2})$          | $O(b^{d/2})$        |  |

Informed Search

Informed Searches:



more efficient  
than  
uninformed  
search



Heuristic function: gives an estimate of the  
smallest/cheapest cost of  
going from a node to the goal node.

① GBFS

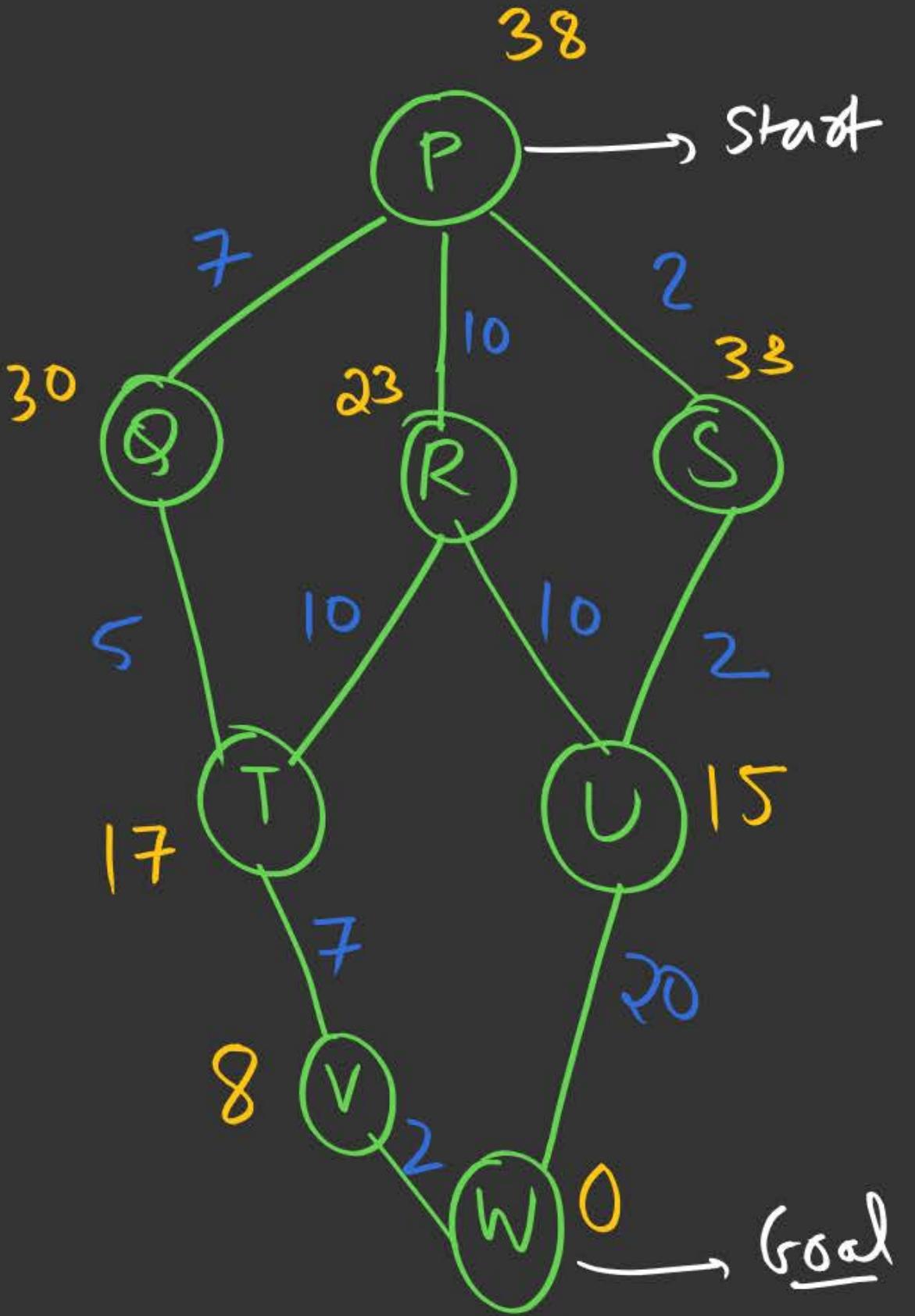


Greedy Best First Search.

## GBFS:

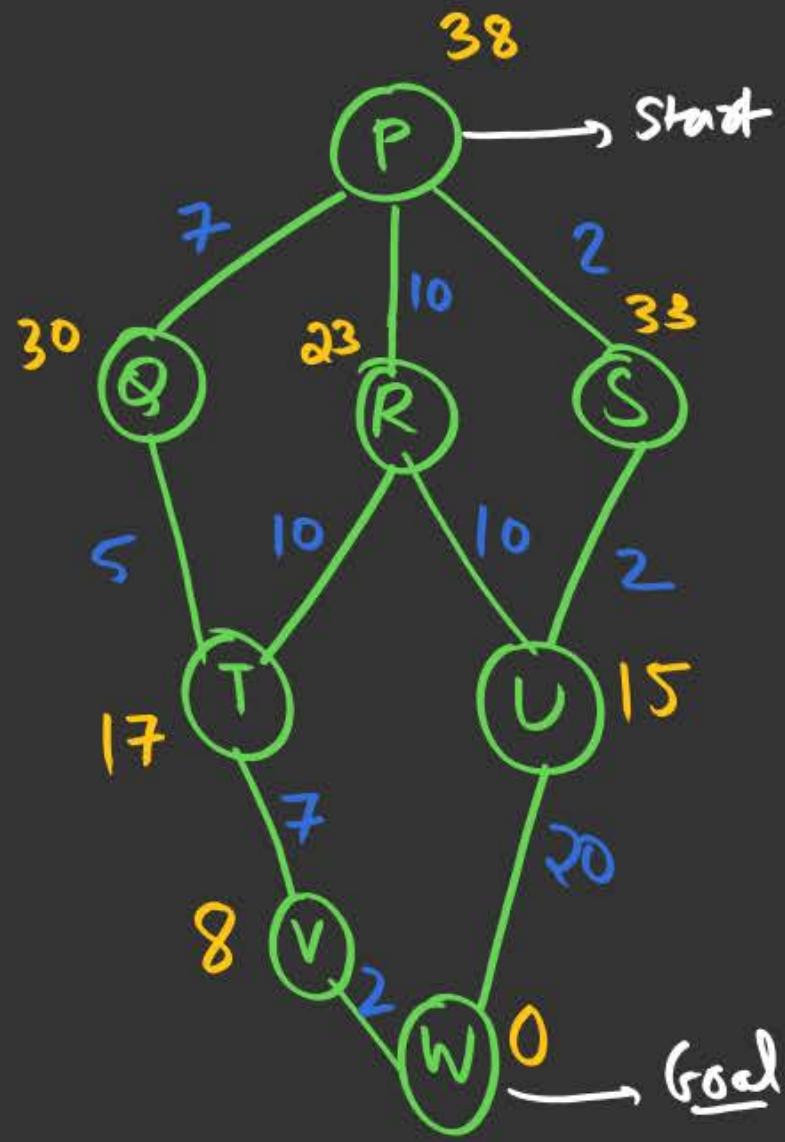
- move → on basis of  $h(n)$ .
- If multiple child nodes, move to the child with  $\min h(n)$ .

Eg:-



① UCS

Eg:-

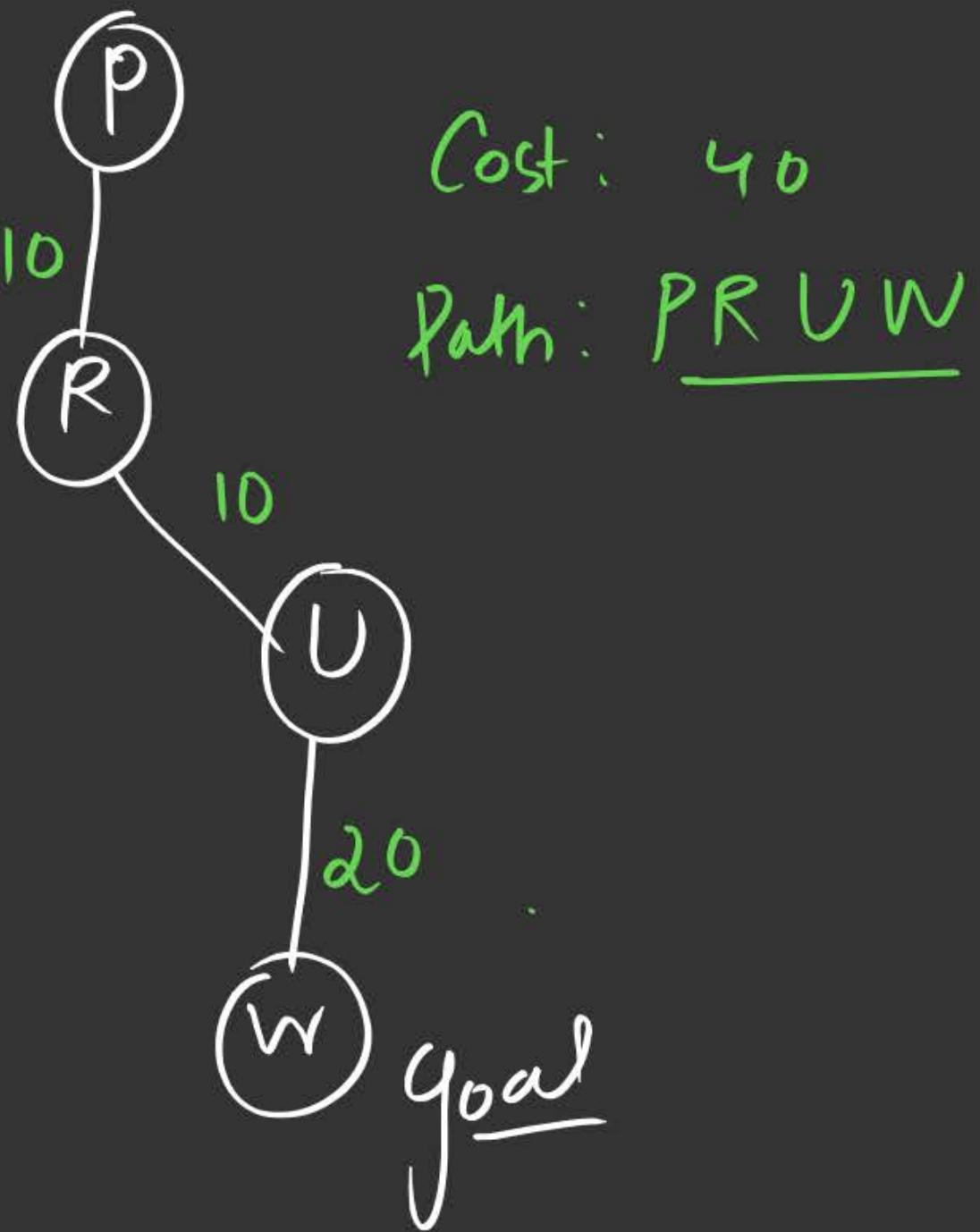
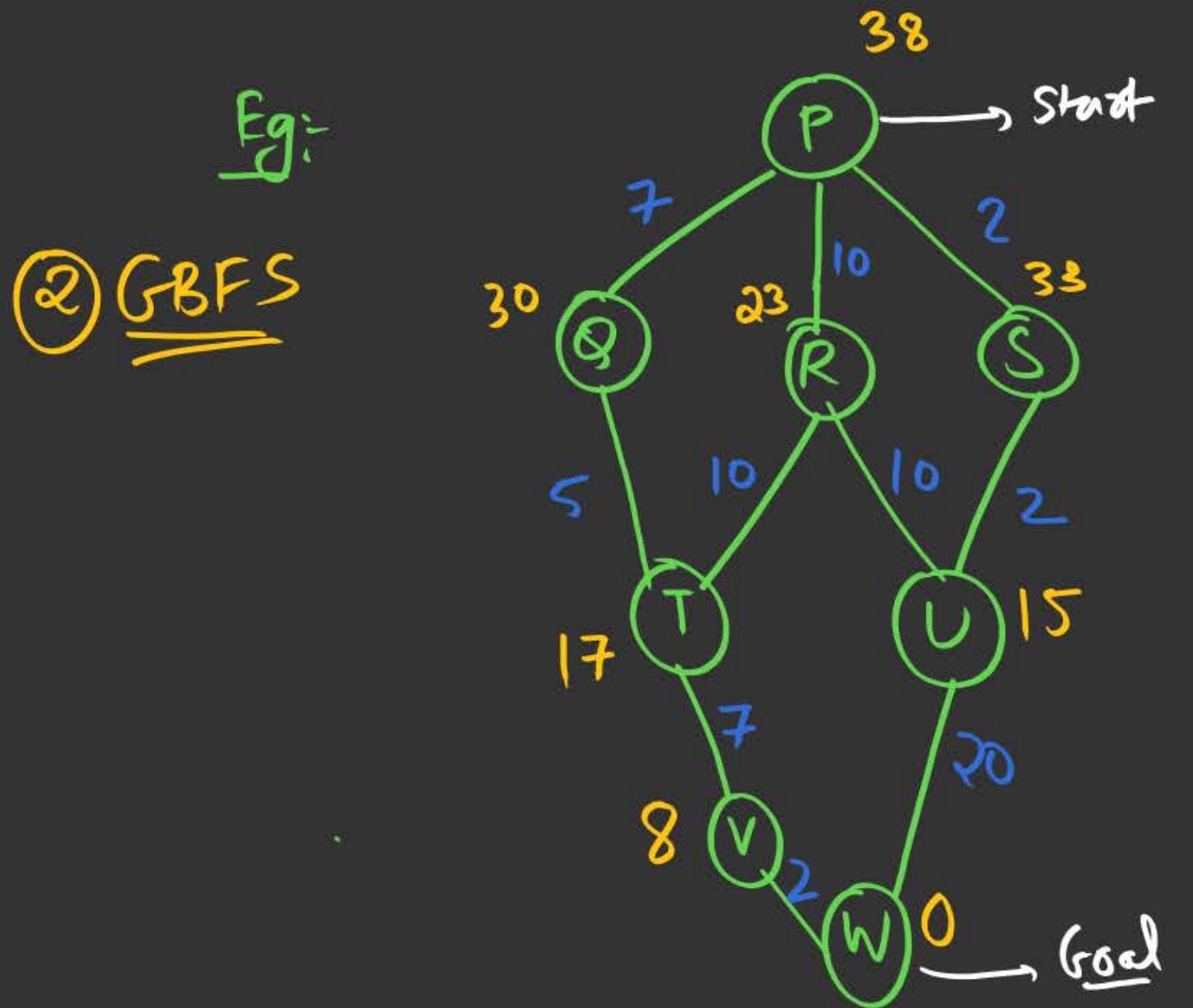


① UCS

|    | Open | Closed |    |    |    |    |    |
|----|------|--------|----|----|----|----|----|
| P* | P    | S      | U  | Q  | R  | T  | V  |
| Q* | 7    | 7      | 7  | 7  | 7  | 7  | 7  |
| R* | 10   | 10     | 10 | 10 | 10 | 10 | 10 |
| S* | 2    | 2      | 2  | 2  | 2  | 2  | 2  |
| U* | x    | 4      | 4  | 4  | 4  | 4  | 4  |
| W  | x    | x      | 24 | 24 | 24 | 24 | 24 |
| T* | x    | x      | x  | 12 | 12 | 12 | 12 |
| V  | x    | x      | x  | x  | x  | 19 | 19 |

$$\text{UCS: Cost} = 21$$

$(P \rightarrow Q \rightarrow T \rightarrow V \rightarrow W)$



Imp:

Disadvantages: If the Heuristics are incorrect/Bad.

W.C TC & SC.

$$\int O(b^d)$$

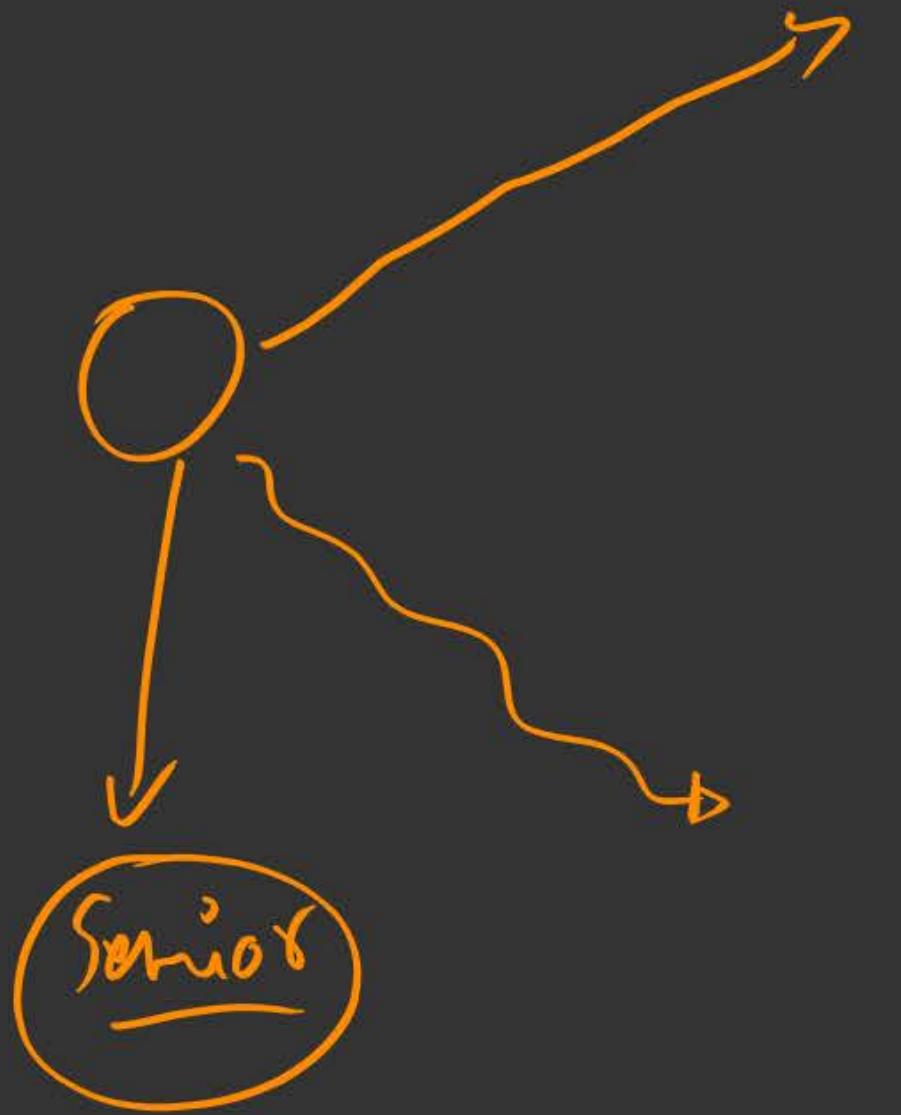
visit all nodes

- ① Not Optimal  $\rightarrow$  <sup>as</sup> not considering path costs
- ② Not Complete
  - Cycle
  - Dead-end

Advantages:

If Heuristics are correct:

- ① OPTIMAL ✓
- ② Complete ✓



②

## A\* Search :

↳ Similar to UCS.

UCS: Cost of a node =  $\overline{\text{sum of cost of all the path from start to that node.}}$

↓  
 $g(n)$

↓  
 $g(n)$

\* GBFS:  $\rightarrow h(n)$

cost of node = sum of all the Heuristics  
 $(h(n)) \leftarrow$

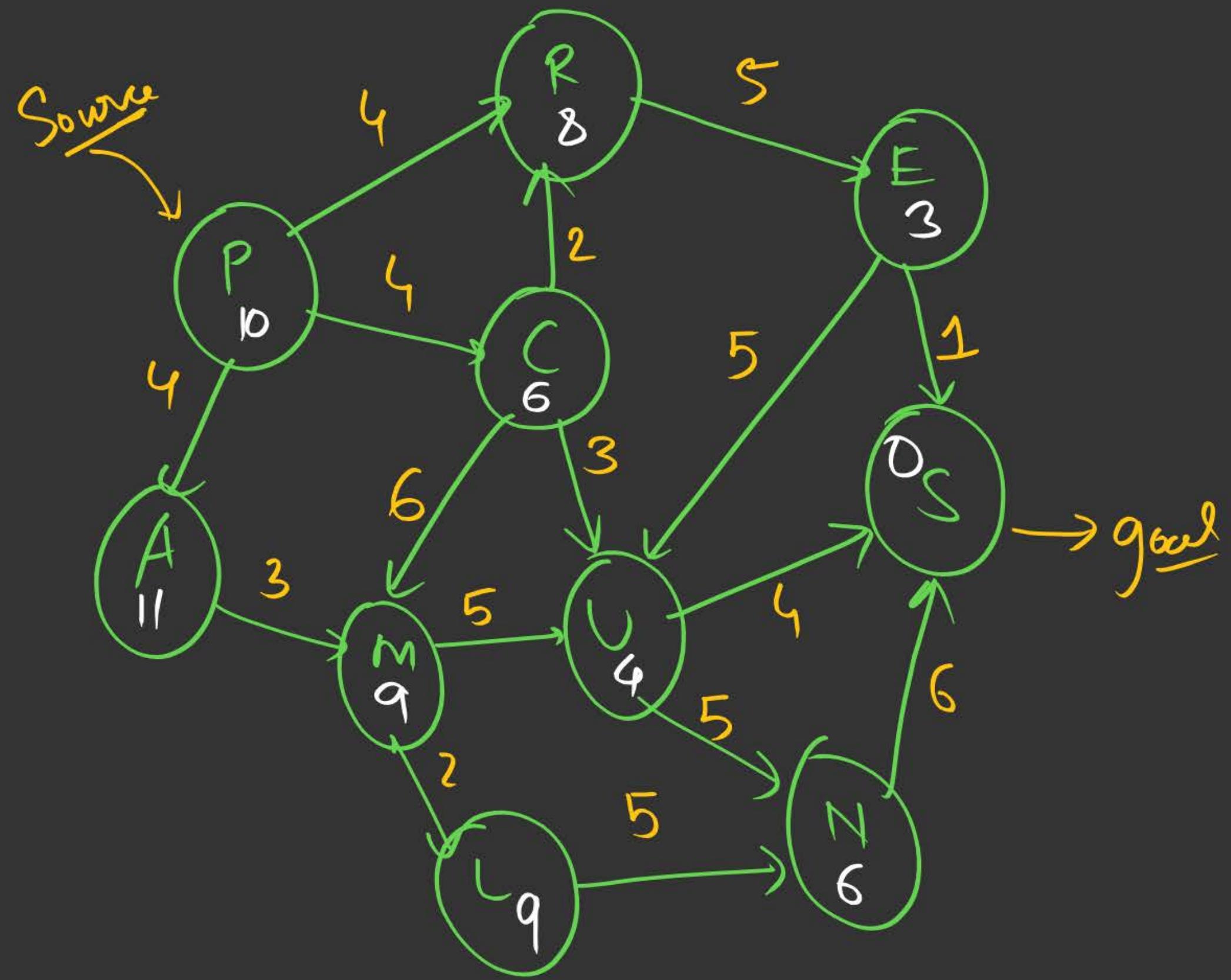
## A\* Search :

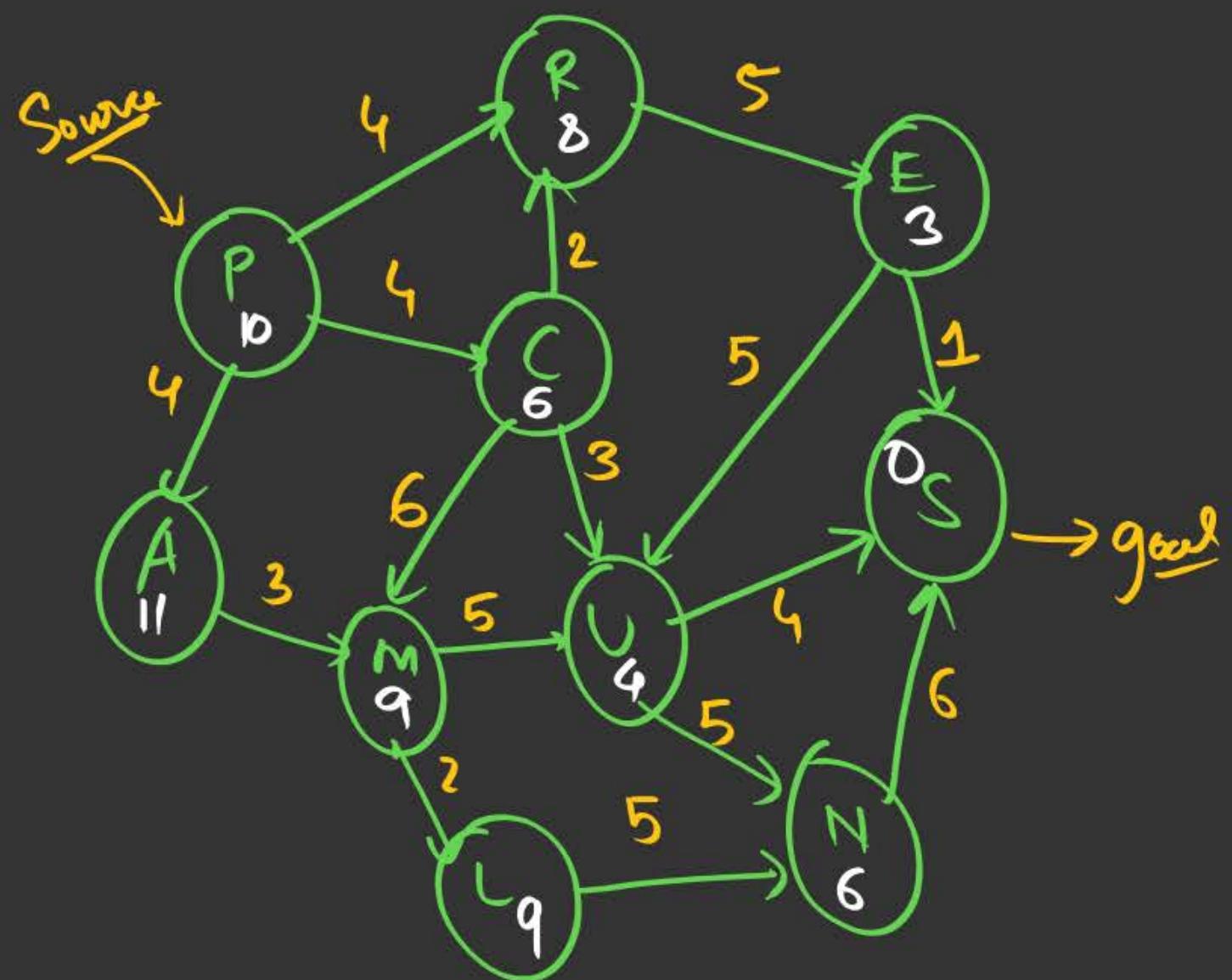
Same as UCS, but Cost of a node ( $f(n)$ )

$$f(n) = g(n) + h(n)$$

path cost till that node.

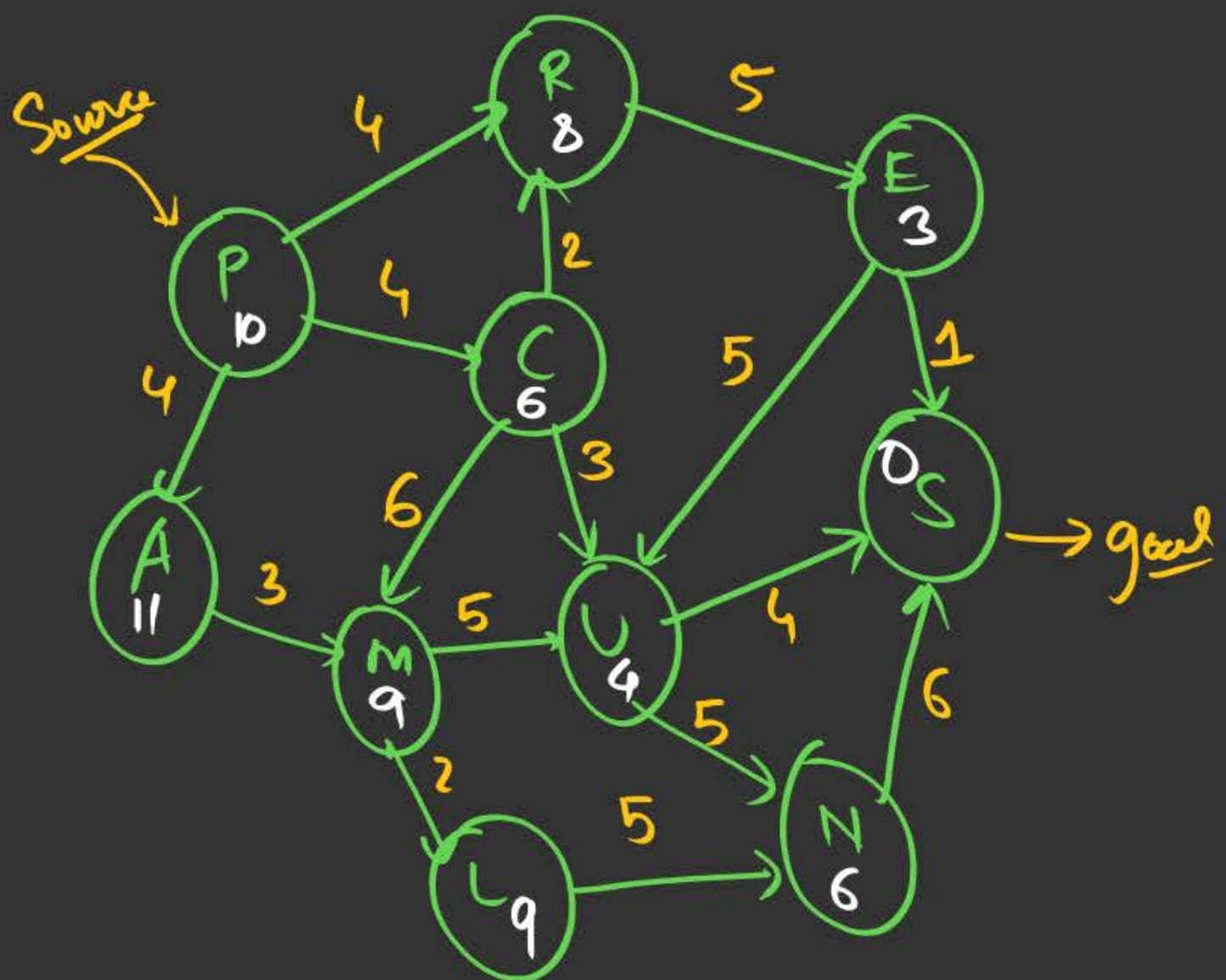
Heuristic value  $g$  that  
node





① UCS

| open | closed |   |   |   |    |    |    |        |
|------|--------|---|---|---|----|----|----|--------|
| P X  | P      | A | C | R | M  | U  | E  | L      |
| A X  | 4      | 4 | 4 | 4 | 4  | 4  | 4  | 4      |
| C X  | 4      | 4 | 4 | 4 | 4  | 4  | 4  | 4      |
| R X  | 4      | 4 | 4 | 4 | 4  | 4  | 4  | 4      |
| M X  | X      | 7 | 7 | 7 | 7  | 7  | 7  | 7      |
| U X  | X      | X | 7 | 7 | 7  | 7  | 7  | 7      |
| E X  | X      | X | X | 9 | 9  | 9  | 9  | 9      |
| L X  | X      | X | X | X | 9  | 9  | 9  | 9      |
| N S  | X      | X | X | X | 12 | 12 | 12 |        |
|      | X      | X | X | X | 11 | 10 | 10 | → goal |



## ② A\* Search

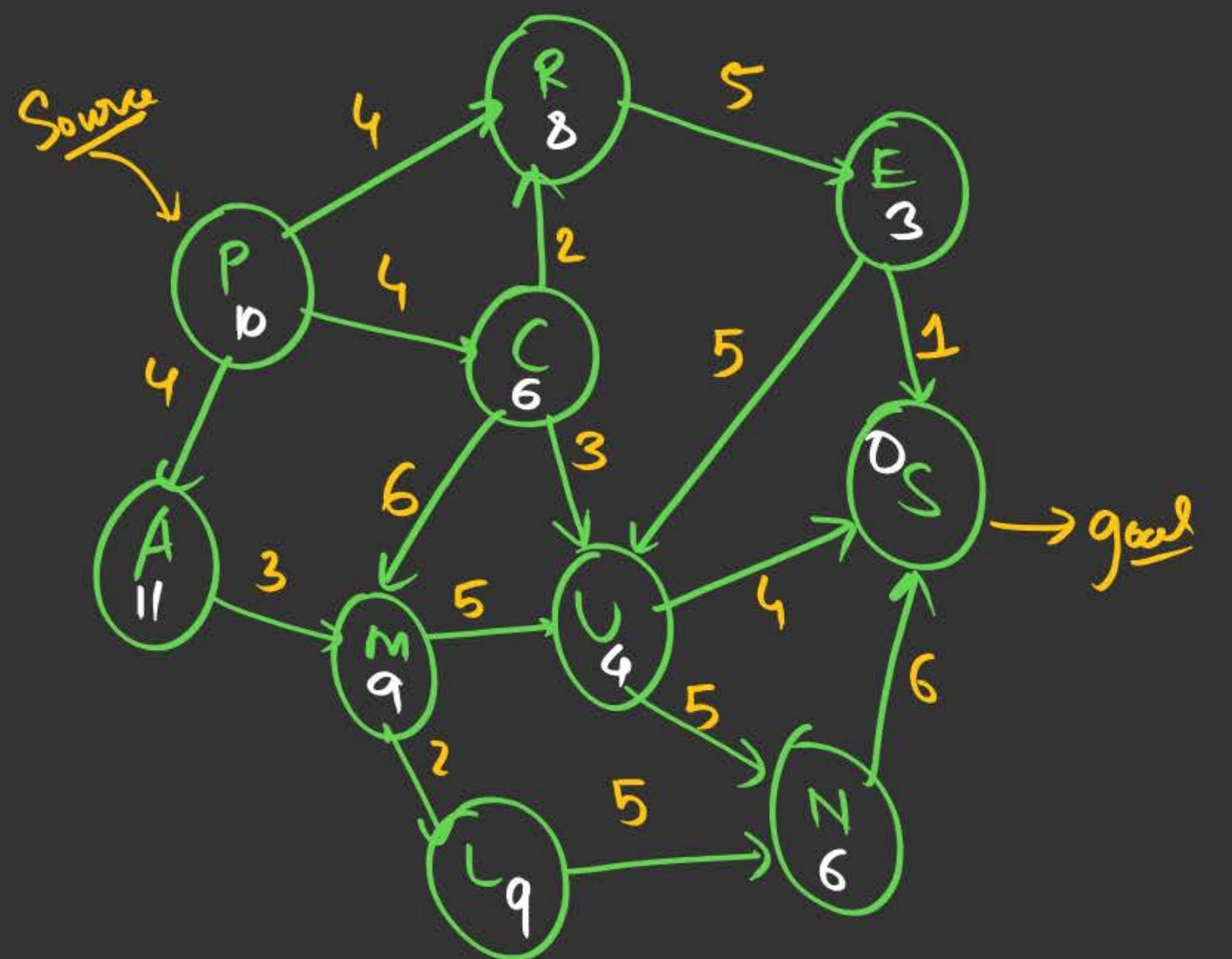
Open | Closed.

|   | P | C | U | S  |
|---|---|---|---|----|
| P | X |   |   |    |
| A |   |   |   |    |
| C | X |   |   |    |
| R |   |   |   |    |
| M | X |   |   |    |
| U | X |   |   |    |
| S | X | X |   |    |
| N | X | X |   | 18 |

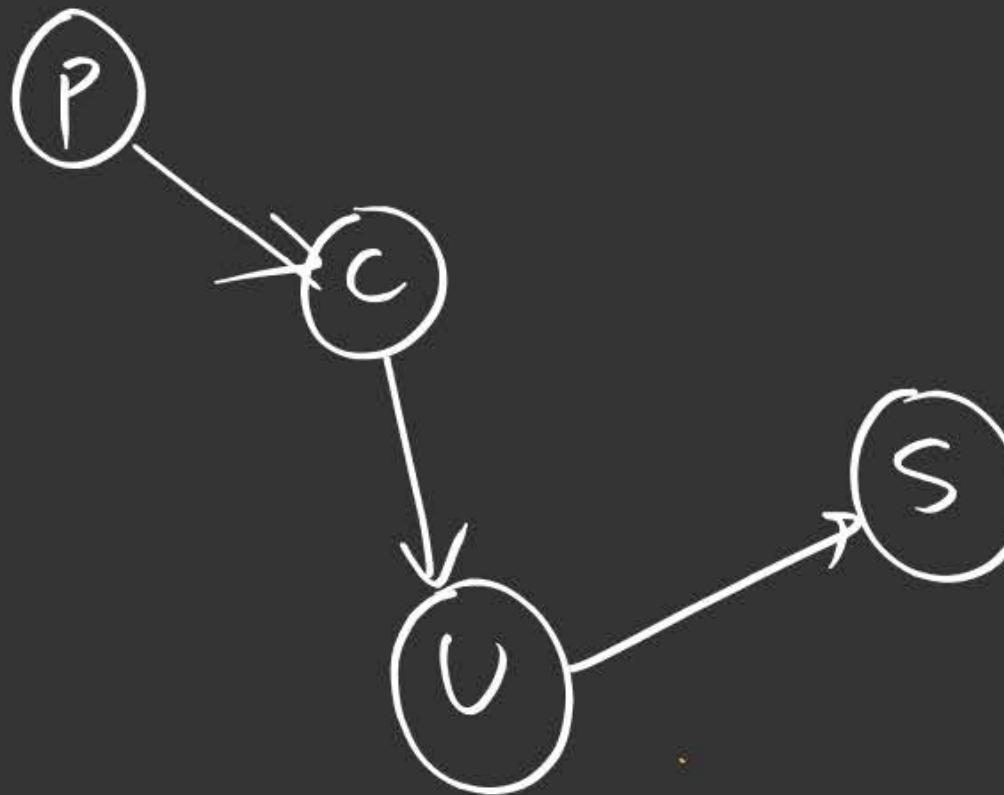
Path: **PCUS**

Cost = 11

Reached goal



③ GBFS





**THANK - YOU**



# DS & AI ENGINEERING



## Artificial Intelligence

### Informed search

Lecture No.- 02

By- Aditya sir



# Recap of Previous Lecture



Topic

Topic

Topic

Topic

Intro to Informed Search

GBFS

A\*

# Topics to be Covered



Topic

Topic

Topic

Properties of Heuristics.

Types of A\*



## About Aditya Jain sir

1. Appeared for GATE during BTech and secured AIR 60 in GATE in very first attempt - City topper
2. Represented college as the first Google DSC Ambassador.
3. The only student from the batch to secure an internship at Amazon. (9+ CGPA)
4. Had offer from IIT Bombay and IISc Bangalore to join the Masters program
5. Joined IIT Bombay for my 2 year Masters program, specialization in Data Science
6. Published multiple research papers in well known conferences along with the team
7. Received the prestigious excellence in Research award from IIT Bombay for my Masters thesis in ML
8. Completed my Masters with an overall GPA of 9.36/10
9. Joined Dream11 as a Data Scientist
10. Have mentored 15,000+ students & working professionals in field of Data Science and Analytics
11. Have been mentoring & teaching GATE aspirants to secure a great rank in limited time
12. Have got around 27.5K followers on Linkedin where I share my insights and guide students and professionals.





**Telegram Link for Aditya Jain sir:**  
**[https://t.me/AdityaSir\\_PW](https://t.me/AdityaSir_PW)**



# Topic : Analysis of Informed Search



## Heuristics

- ↳ ① Admissible
- ② Consistent



# Topic : Informed Search

## Type of Heuristic values

In the context of search algorithms, especially in artificial intelligence and pathfinding (like A\* algorithm), a heuristic function helps estimate the cost of reaching the goal from a given state. There are two important properties that a heuristic function can have: admissibility and consistency (also known monotonicity).

\* Informed search will give optimal result if  $h(n)$  are consistent & admissible.

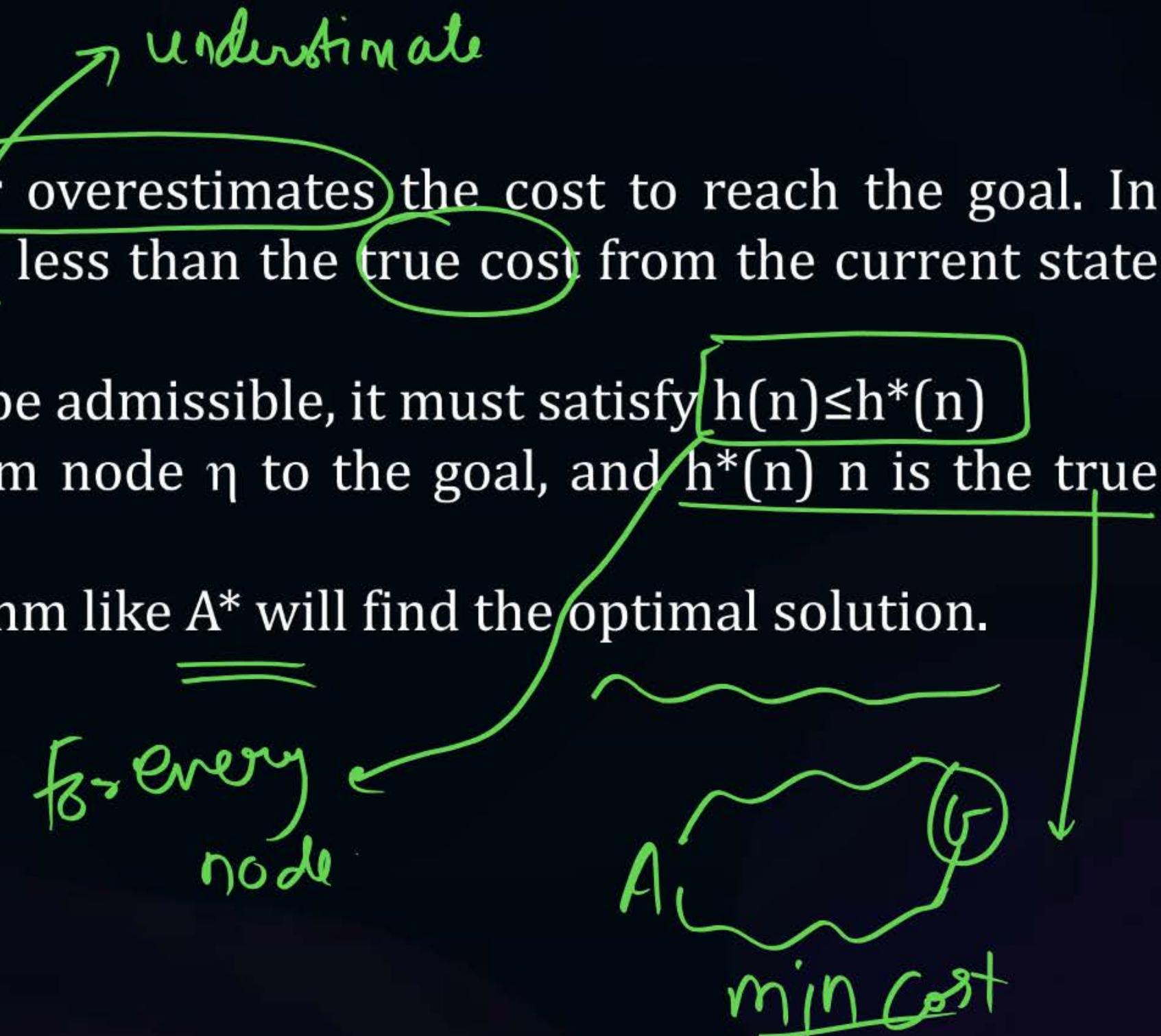


# Topic : Informed Search

## Type of Heuristic values

### Admissible Heuristic

- A heuristic is admissible if it never overestimates the cost to reach the goal. In other words, it is always equal to or less than the true cost from the current state to the goal.
- Formally, for a heuristic  $h(n)$  to be admissible, it must satisfy  $h(n) \leq h^*(n)$
- where  $h(n)$  is the heuristic cost from node  $\eta$  to the goal, and  $h^*(n)$  is the true cost from  $n$  to the goal.
- Admissibility ensures that an algorithm like  $A^*$  will find the optimal solution.





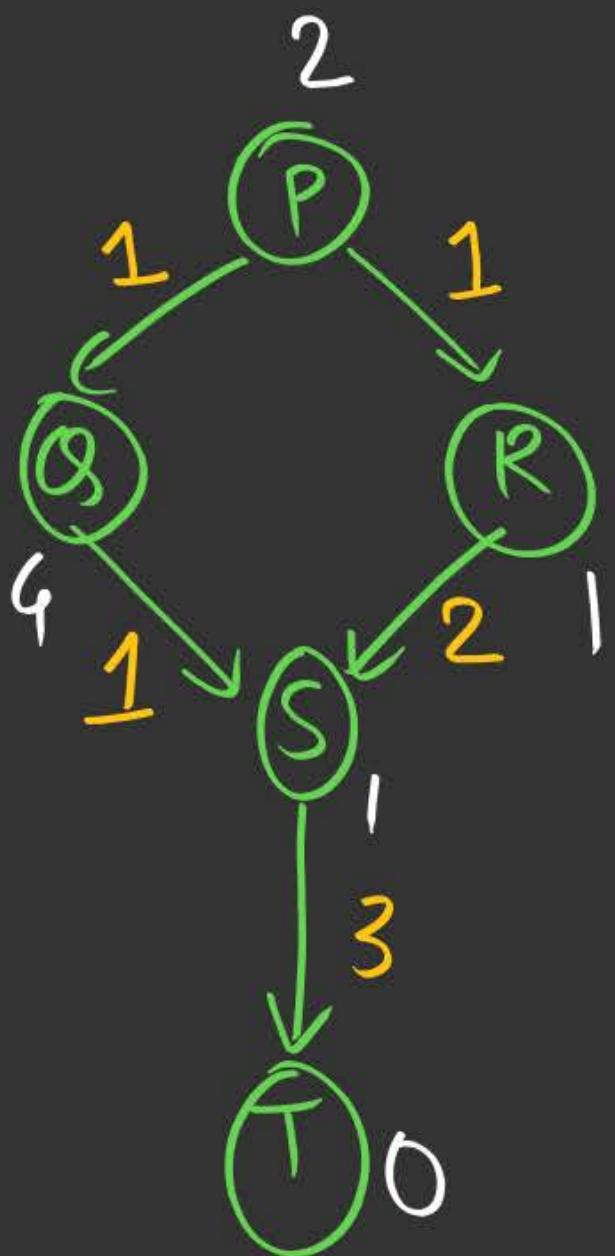
# Topic : Informed Search

## Type of Heuristic values

### Consistent (Monotonic) Heuristic

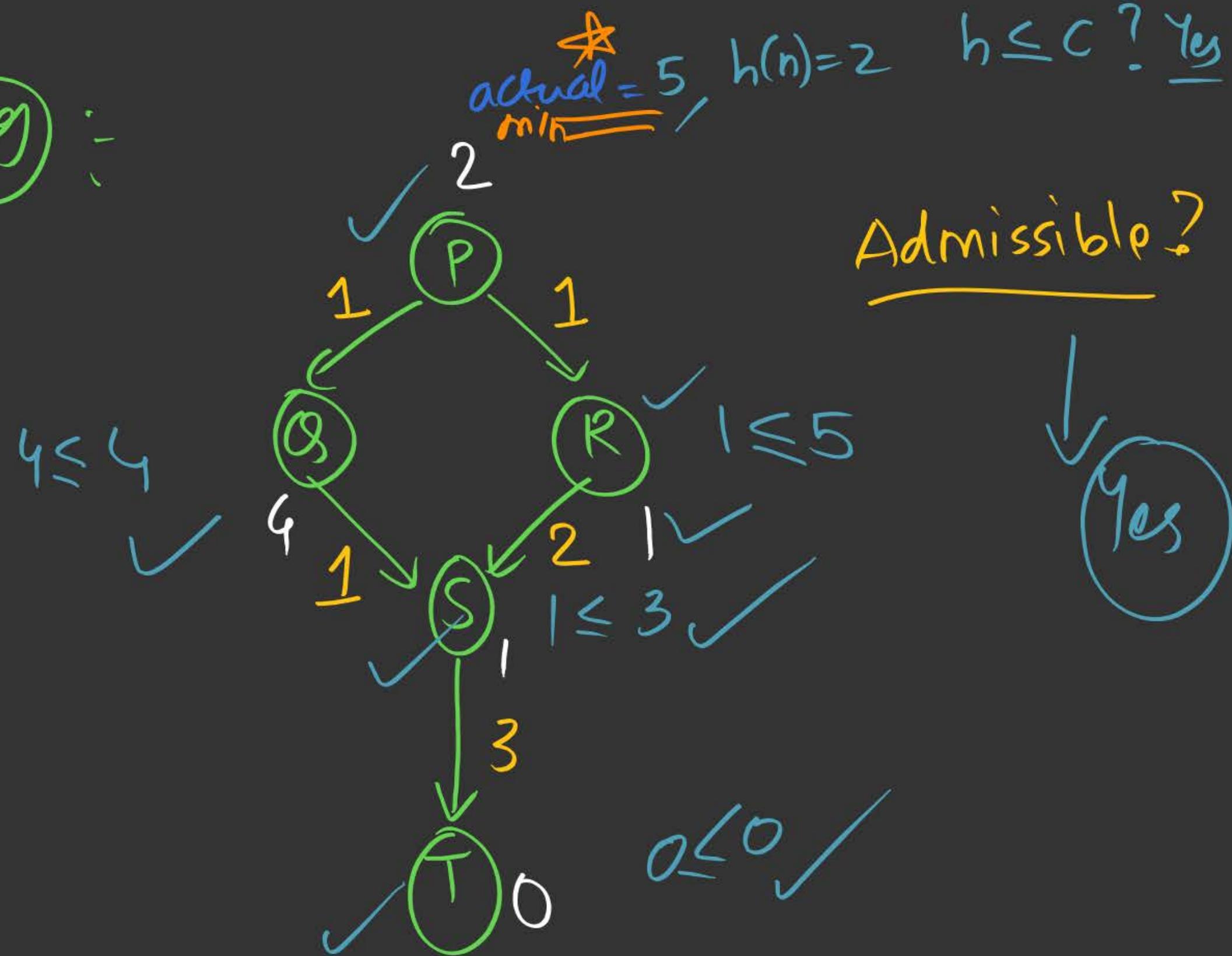
- A heuristic is consistent if the estimated cost is always less than or equal to the estimated cost from any neighbouring node plus the step cost of reaching that neighbour.
- Formally, for a heuristic  $h(n)$  to be consistent, it must satisfy the following condition for every node  $n$  and every successor  $n'$  of  $n$ :  $(h(n) \leq c(n,n') + h(n'))$  where  $c(n,n')$  is the actual cost from node  $n$  to its successor  $n'$ , and  $h(n')$  is the heuristic cost from  $n'$  to the goal.

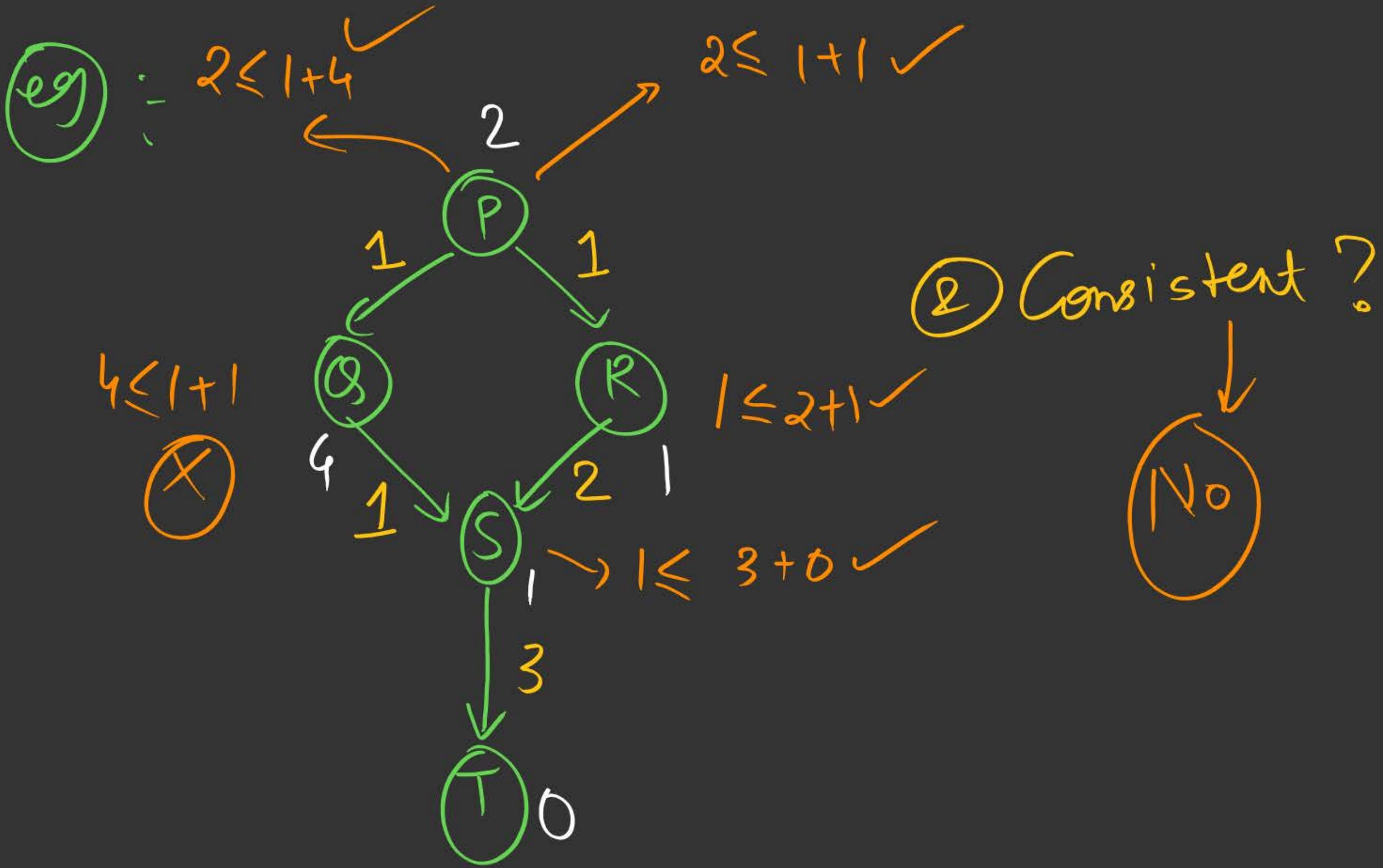
(eg) :



- Is
- 1) Admissible?
  - 2) Consistent?

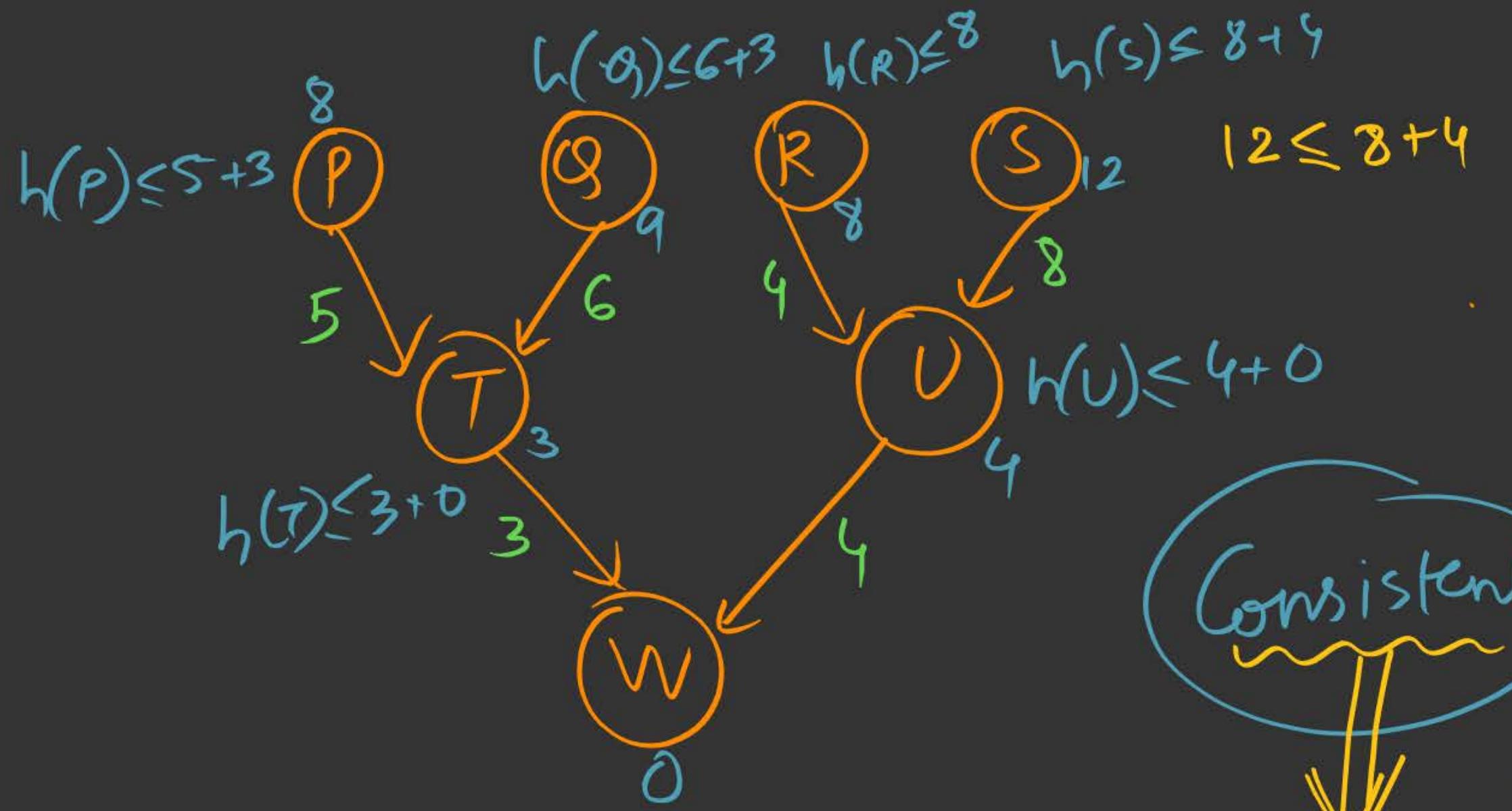
(eg) :







If Heuristic Consistent  $\rightarrow$  then automatically  
admissible



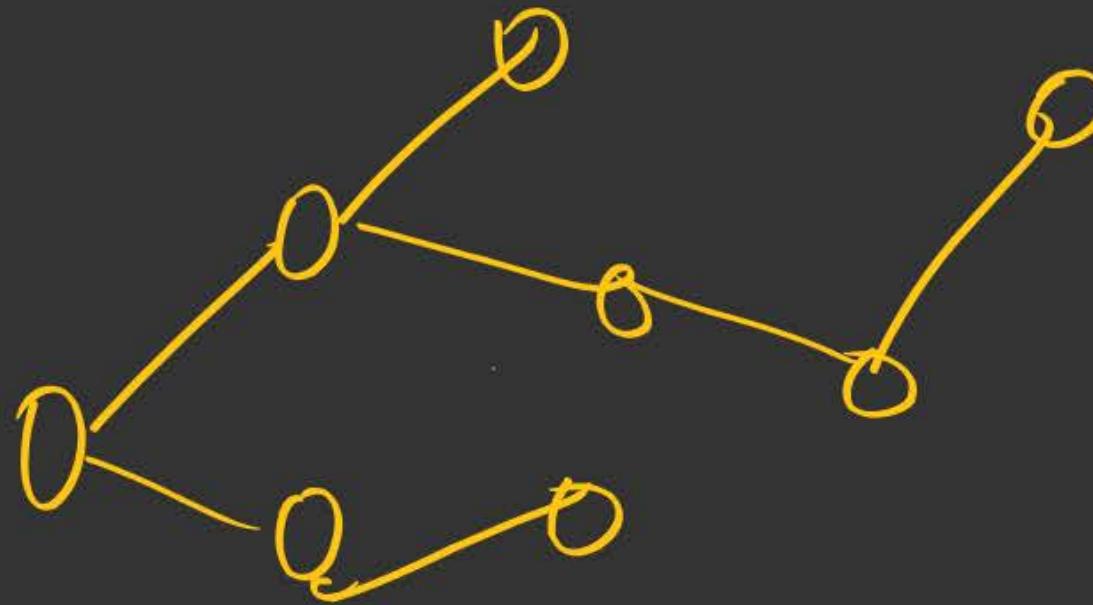
Consistent

Admissible

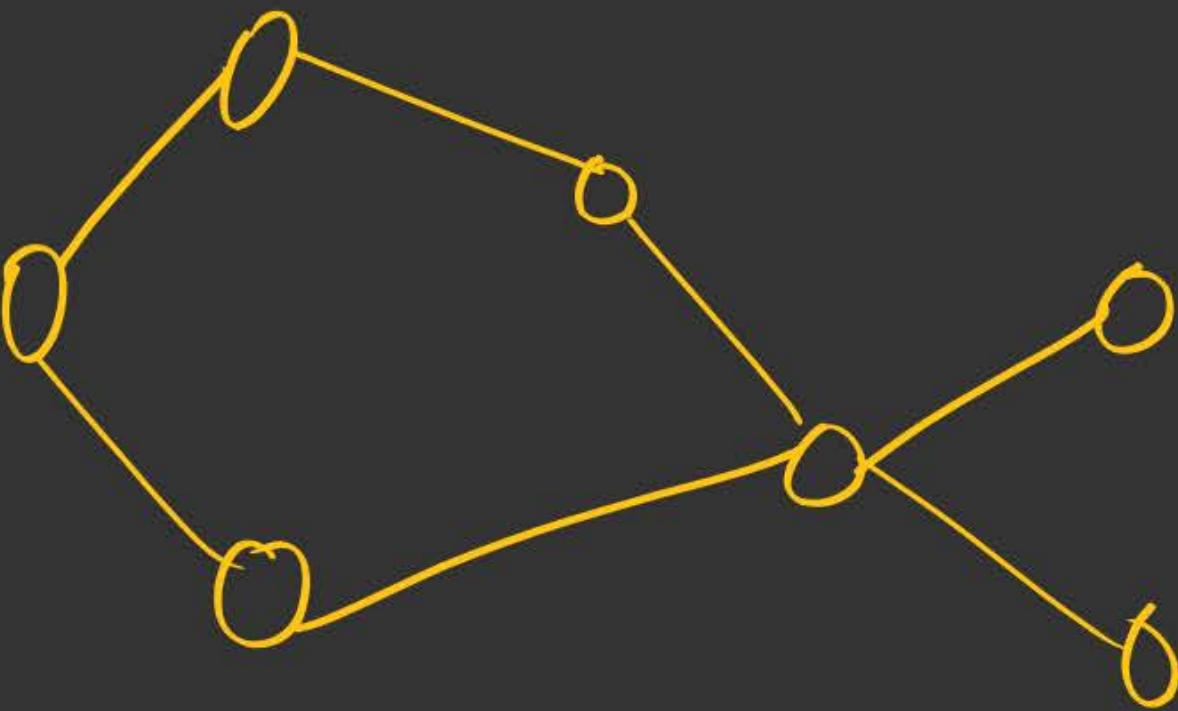
But not  
vise versa

Graph vs Tree Search: (DSA pov)

Tree  $\rightarrow$  Acyclic



Graph → Can have Cycles

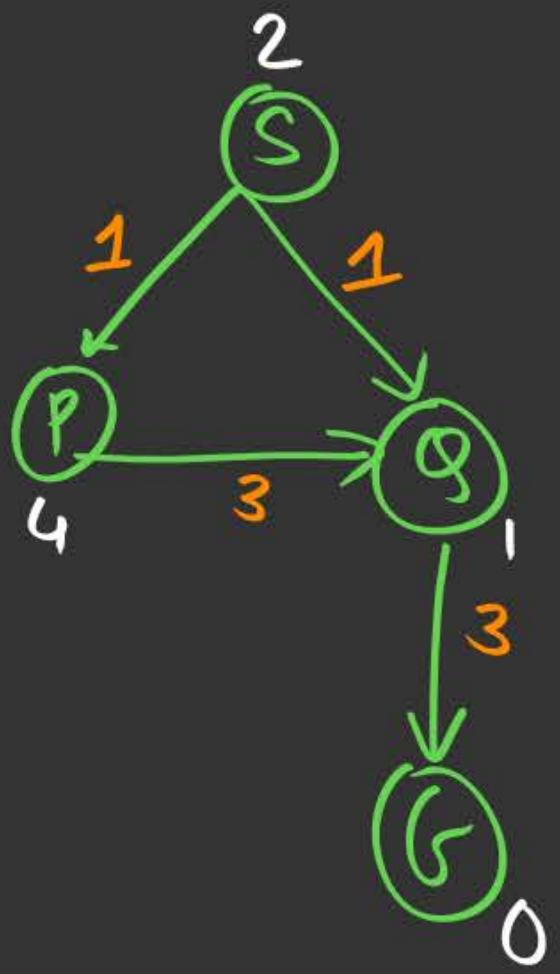


# Tree & Graph Search (AI POV)

① Graph Search: a node present in a closed list  
cannot be visited again.

② Tree Search: " " closed list  
can be visited again.  
(basically, can be further minimised and  
then again get selected)

eg:



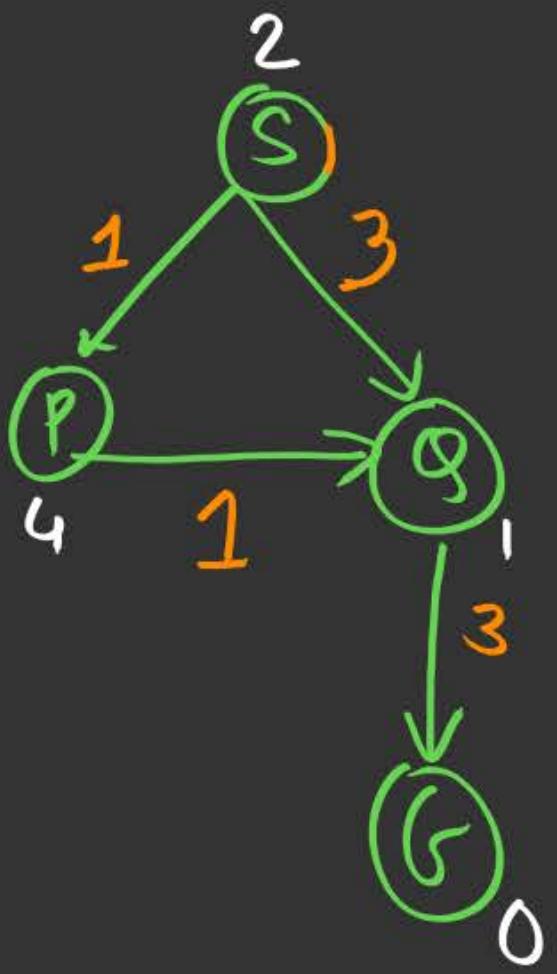
A\* Graph Search:

|    | Open | Closed     |  |
|----|------|------------|--|
| S* | S    | Q          |  |
| P  | 5    | 5          |  |
| Q  | 2    | 2          |  |
| G  | X    | (4) → goal |  |

S → Q → G

Cost: 4

(2)

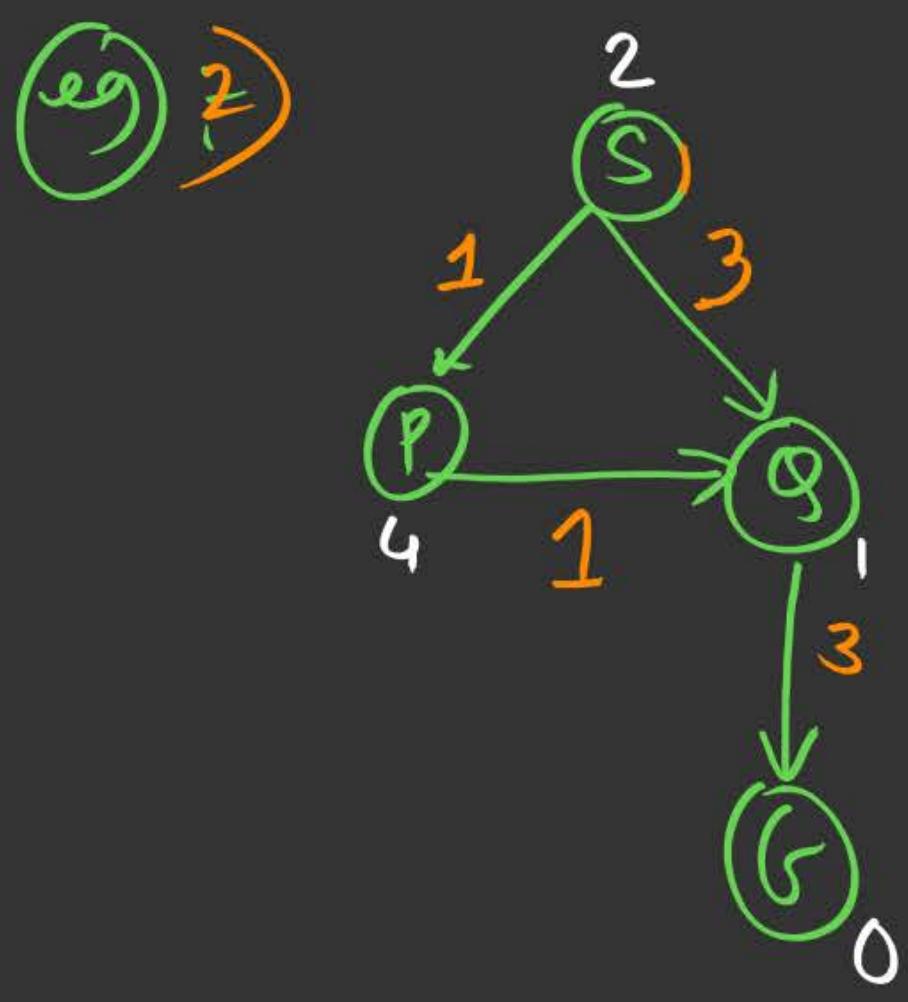


## ① A\* Graph Search

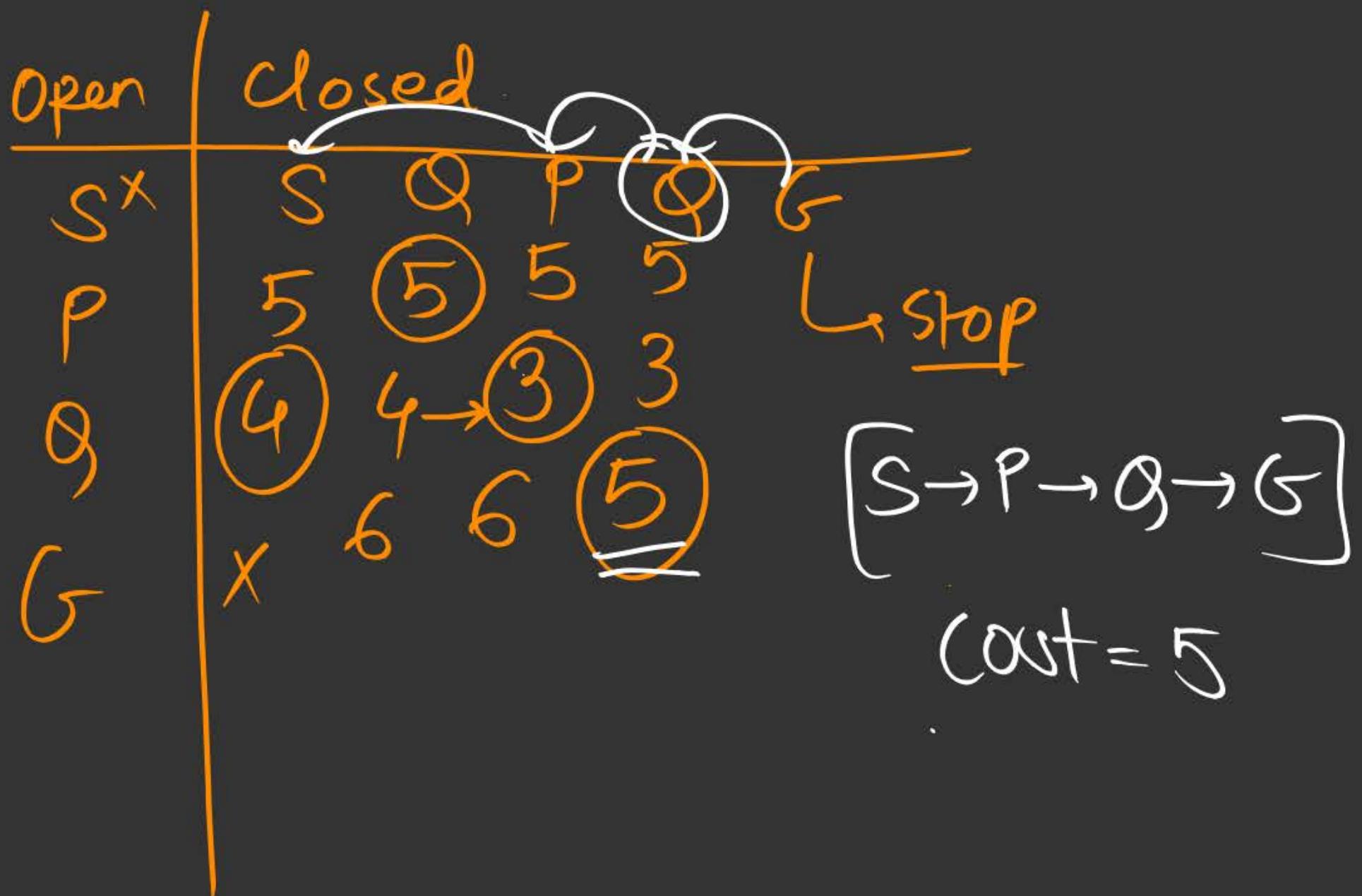
|    | Open | Closed |     |
|----|------|--------|-----|
| S* | S    | ∅      | P G |
| P  | 5    | 5      | 5   |
| Q  | 4    | 4      |     |
| G  | X 6  | 6      |     |

$S \rightarrow Q \rightarrow G$

Cost = 6



① A\* Tree Search



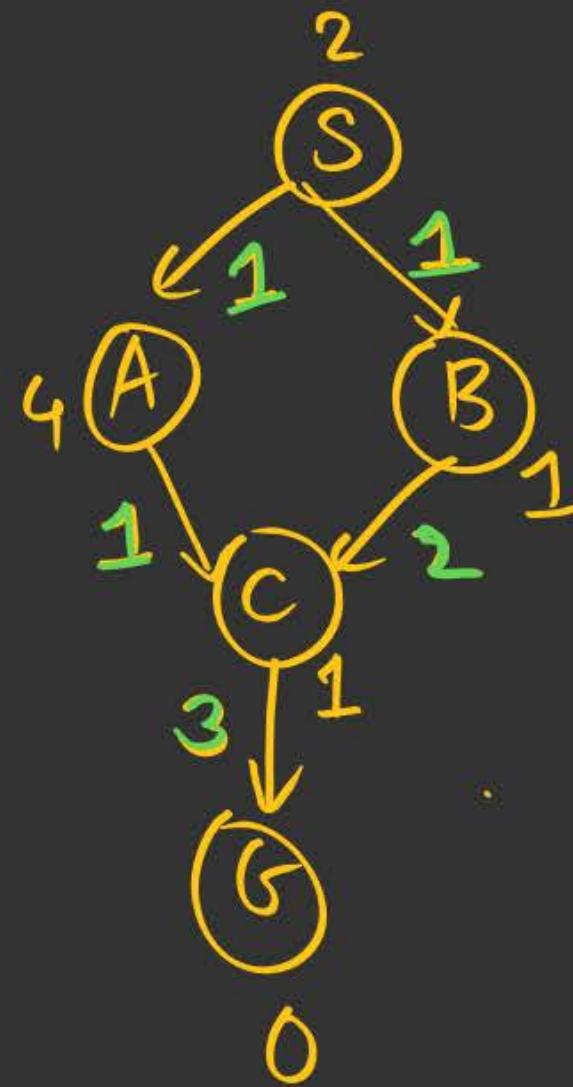
① WC TC & SC <sup>of</sup>  
both can be same.

② Tree Search are better to provide  
Optimal Solution

③ Tree Search can be less efficient as  
they may visit a node multiple times.

| Property                         | A* Graph Search                                         | A* Tree Search                     |
|----------------------------------|---------------------------------------------------------|------------------------------------|
| 1) Closed list used?             | Yes                                                     | No                                 |
| 2) Re-explores node?             | No (unless better path found)                           | Yes                                |
| 3) Optimal with admissible $h$ ? | No always<br>(needs consistent $h$ )<br>or re-expansion | Always                             |
| 4) Space usage                   | High (visited nodes stored)                             | High (all paths stored)            |
| 5) Time efficiency               | More efficient<br>(if good heuristic)                   | Less efficient due to re-expansion |

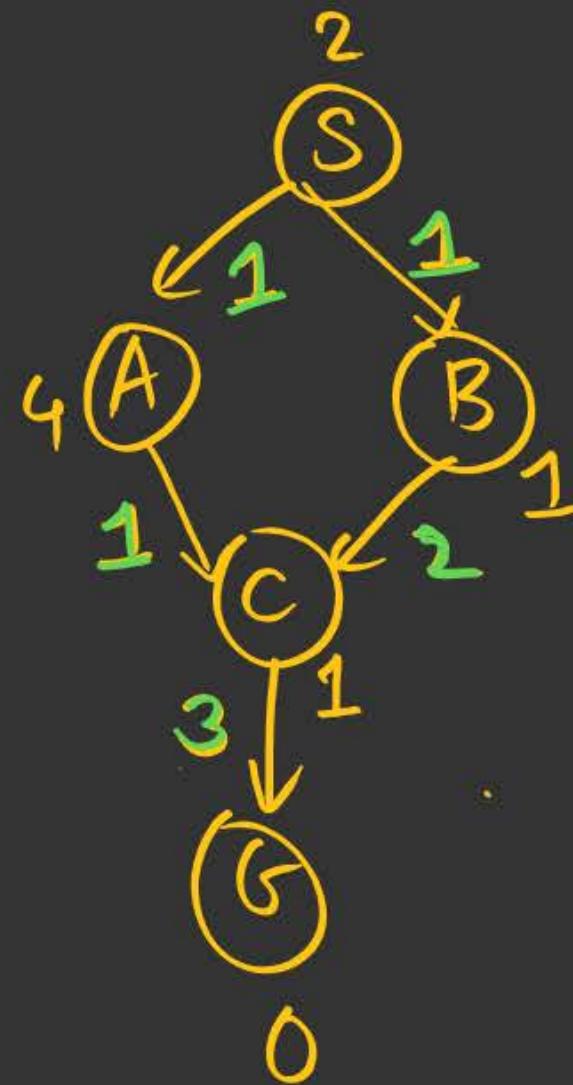
① Admissible but not Consistent.



$A^*$   
1) graph Search

|    | open | closed |   |   |   |                                                                   |
|----|------|--------|---|---|---|-------------------------------------------------------------------|
|    | S*   | S      | B | C | A | G                                                                 |
| A  | 5    | 5      | 5 | 5 | 5 |                                                                   |
| B* | 2    | 2      | 2 | 2 | 2 | STOP                                                              |
| C  | x    | 4      | 4 | 4 |   | <span style="border: 1px solid black; padding: 2px;">S BCG</span> |
| G  | xx   | 6      | 6 | 6 |   | <u>Cost = 6</u>                                                   |

① Admissible but not Consistent.



② A\* Tree Search

|   | Open | Closed |   |   |   |   |
|---|------|--------|---|---|---|---|
| S | S    | B      | C | A | C | G |
| A | 5    | 5      | 5 | 5 | 5 |   |
| B | 2    | 2      | 2 | 2 | 2 |   |
| C | 4    | 4      | 3 | 3 | 3 |   |
| G | 6    | 6      | 5 | 5 | 5 |   |

stop

SACG

cost = 5

Diagram illustrating the A\* Tree Search process. The search space is divided into Open and Closed sets. The Open set contains nodes S, B, C, A, C, and G. The Closed set contains nodes S, B, C, A, C, and G. The cost of reaching each node is shown in the Open set. The path from S to G is highlighted in yellow. The final solution is SACG with a total cost of 5.

# Summary for A\* Search Optimality

| Admissible | Consistent | A* Graph Search | A* Tree Search |
|------------|------------|-----------------|----------------|
| X          | X          | X               | X              |
| ✓          | X          | X               | ✓              |
| ✓          | ✓          | ✓               | ✓              |



**THANK - YOU**



# DS & AI ENGINEERING



## Artificial Intelligence

### Informed search

Lecture No.- 03

By- Aditya sir



# Recap of Previous Lecture



Topic

Topic

Topic

Topic

Properties of Heuristics

↳ Admissible  
↳ Consistent

A\* Search

↳ Tree Search  
↳ Graph Search

# Topics to be Covered



Topic

Topic

Topic

A\* Search Continued . . .





## About Aditya Jain sir

1. Appeared for GATE during BTech and secured AIR 60 in GATE in very first attempt - City topper
2. Represented college as the first Google DSC Ambassador.
3. The only student from the batch to secure an internship at Amazon. (9+ CGPA)
4. Had offer from IIT Bombay and IISc Bangalore to join the Masters program
5. Joined IIT Bombay for my 2 year Masters program, specialization in Data Science
6. Published multiple research papers in well known conferences along with the team
7. Received the prestigious excellence in Research award from IIT Bombay for my Masters thesis in ML
8. Completed my Masters with an overall GPA of 9.36/10
9. Joined Dream11 as a Data Scientist
10. Have mentored 15,000+ students & working professionals in field of Data Science and Analytics
11. Have been mentoring & teaching GATE aspirants to secure a great rank in limited time
12. Have got around 27.5K followers on Linkedin where I share my insights and guide students and professionals.





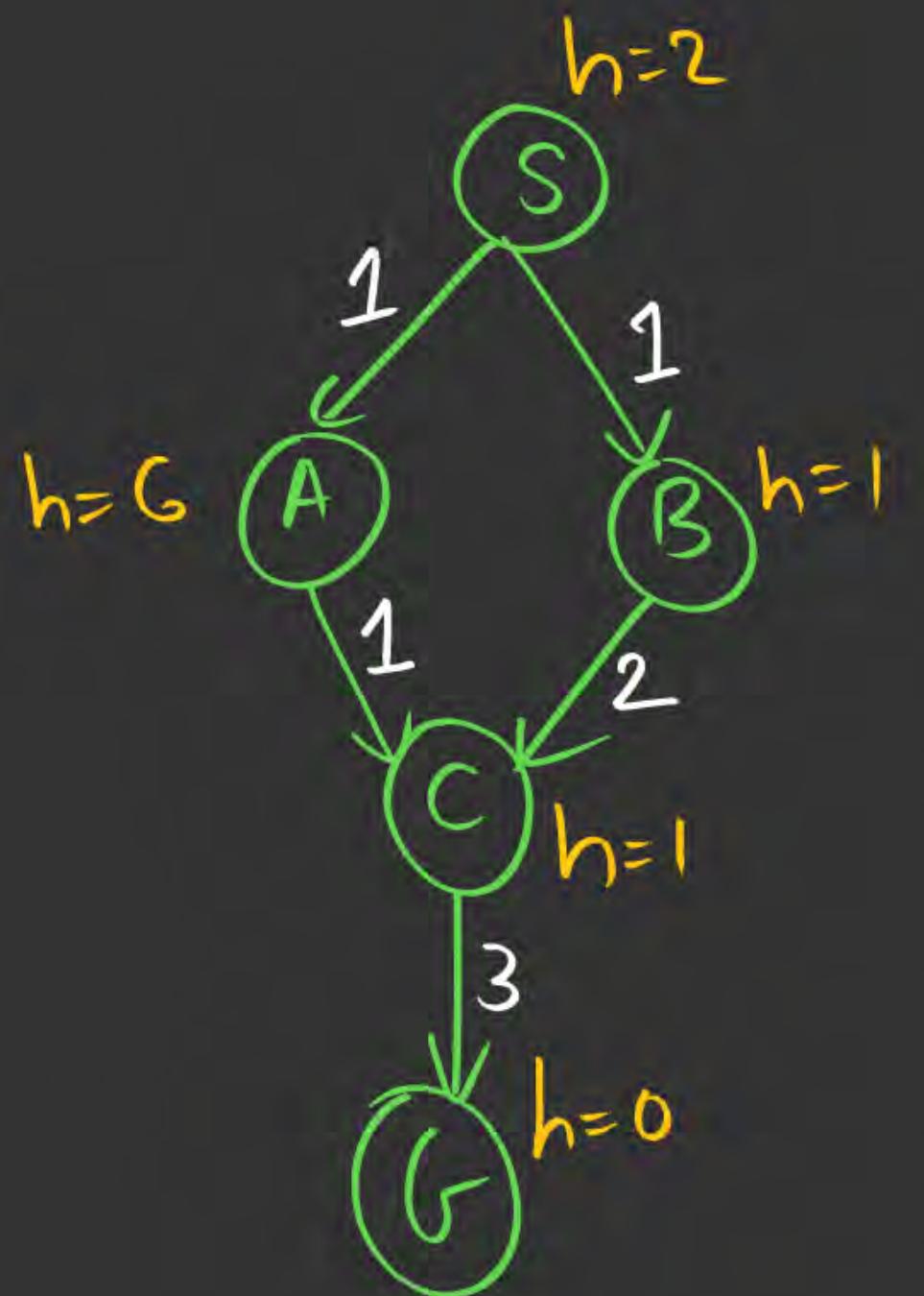
## Revision : Graph and Tree Search

- **Graph Search** → We make a closed list and never visit closed node.
- **Tree Search** → We do not make closed list.

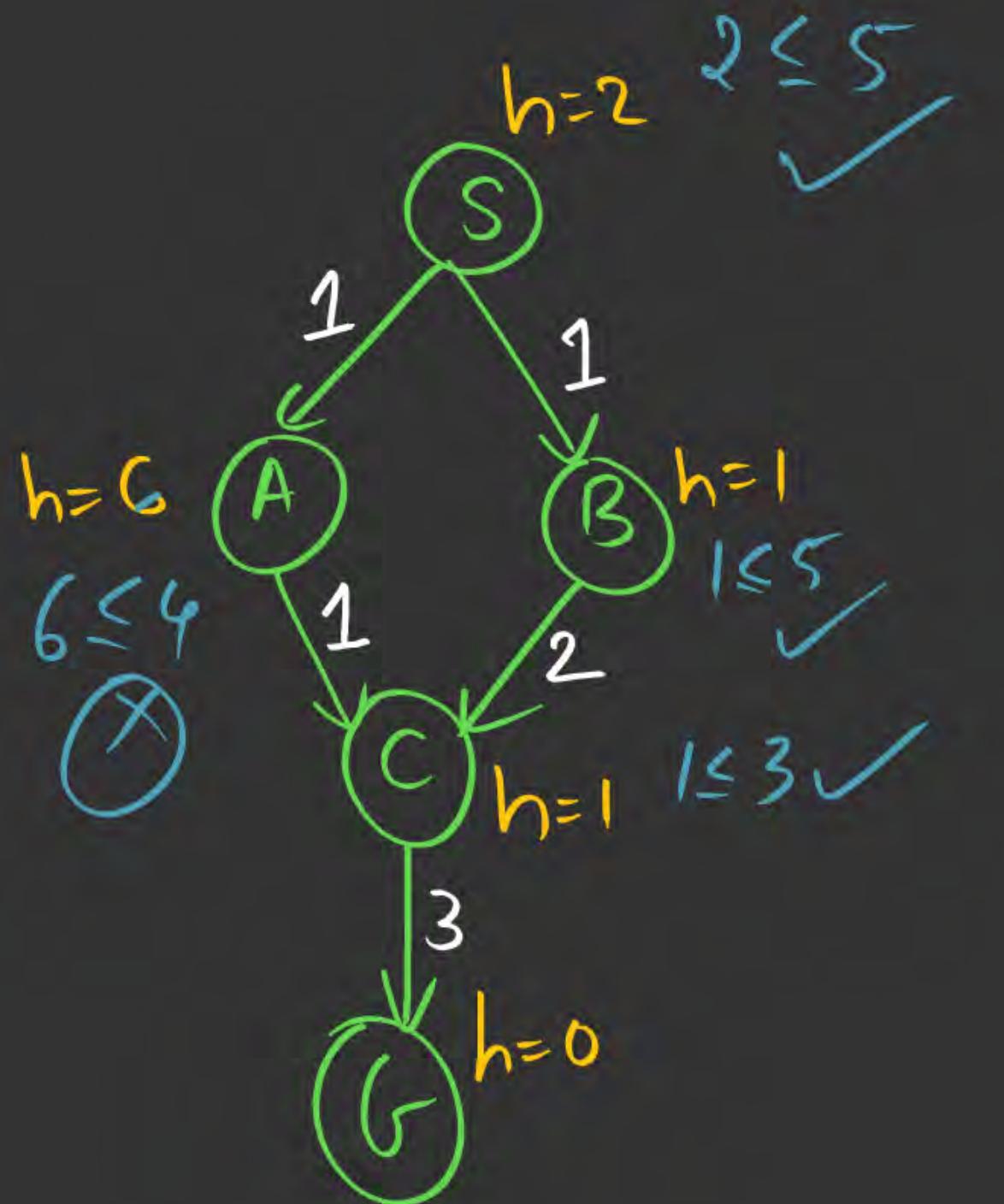


## Revision : Graph and Tree Search

- **Graph Search** → Consistent  $h^0$  optimal solution.
- **Tree Search** → Admissible  $h^0$  optimal solution.  
  

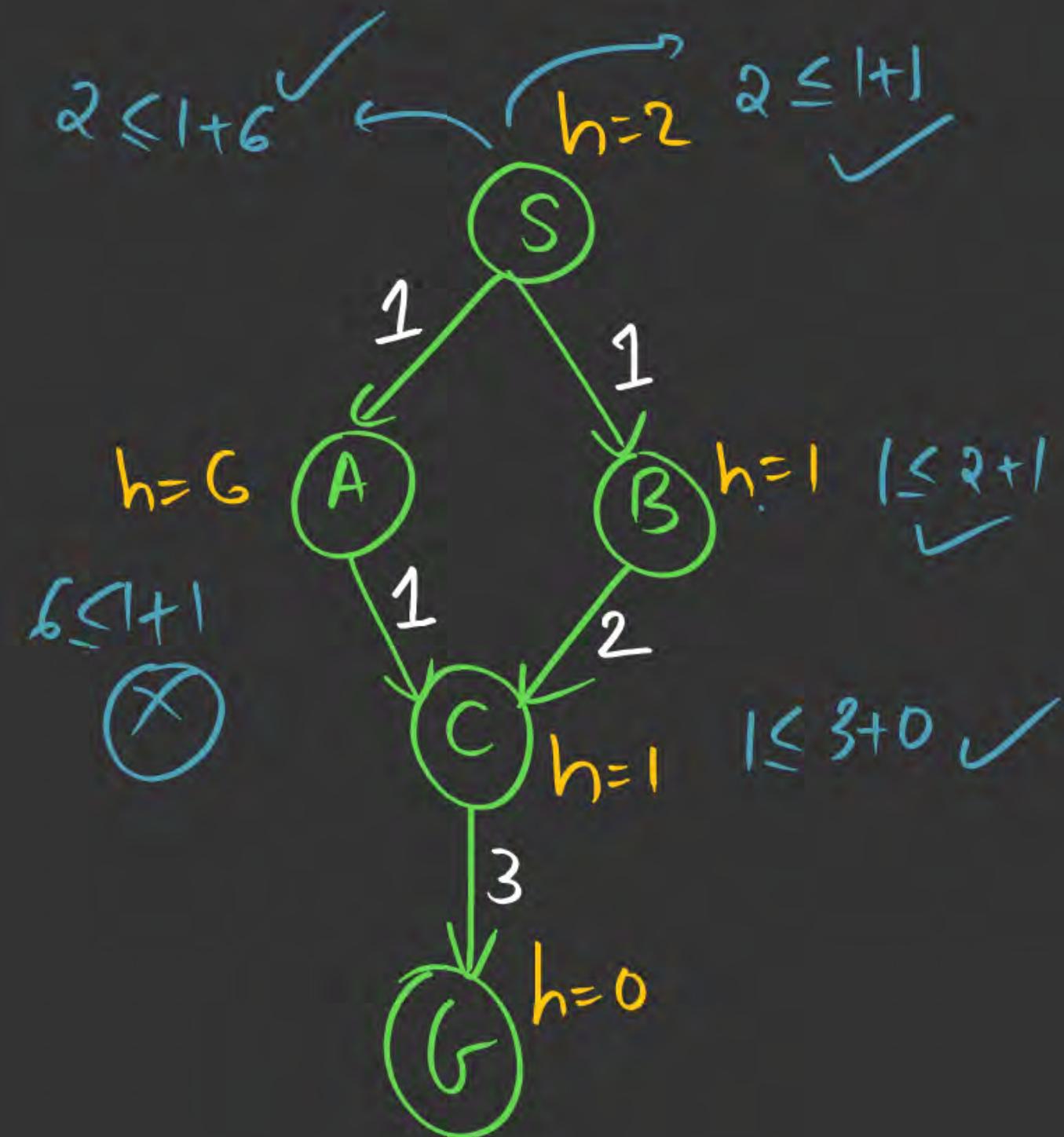



- 1) Admissible?
- 2) Consistent?



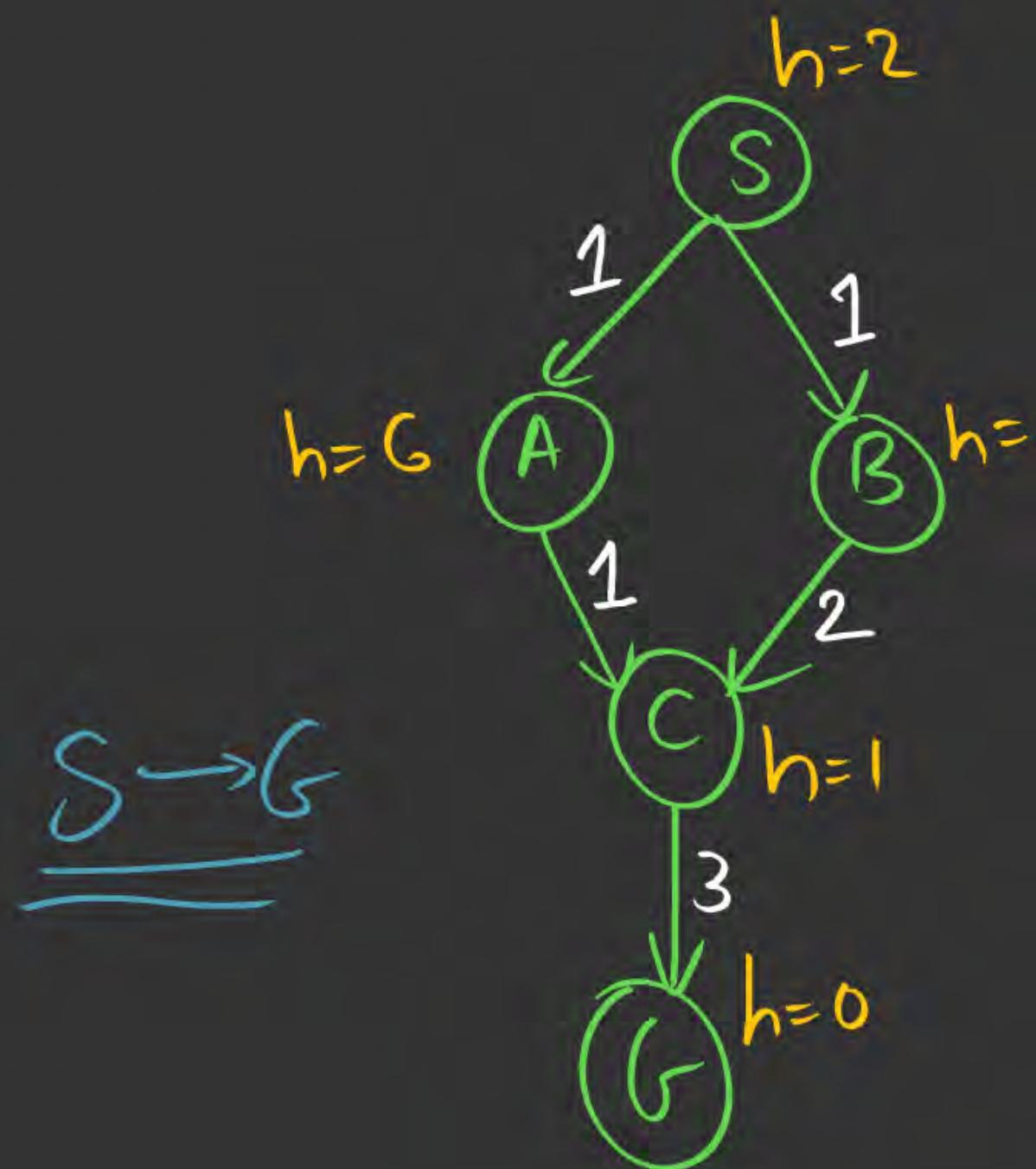
i) Admissible?  $\rightarrow \text{No}$

$$h(n) \leq h^*(n)$$



2) Consistent? → No

$$h(n) \leq c(n, n') + h(n')$$



## A\* Graph Search

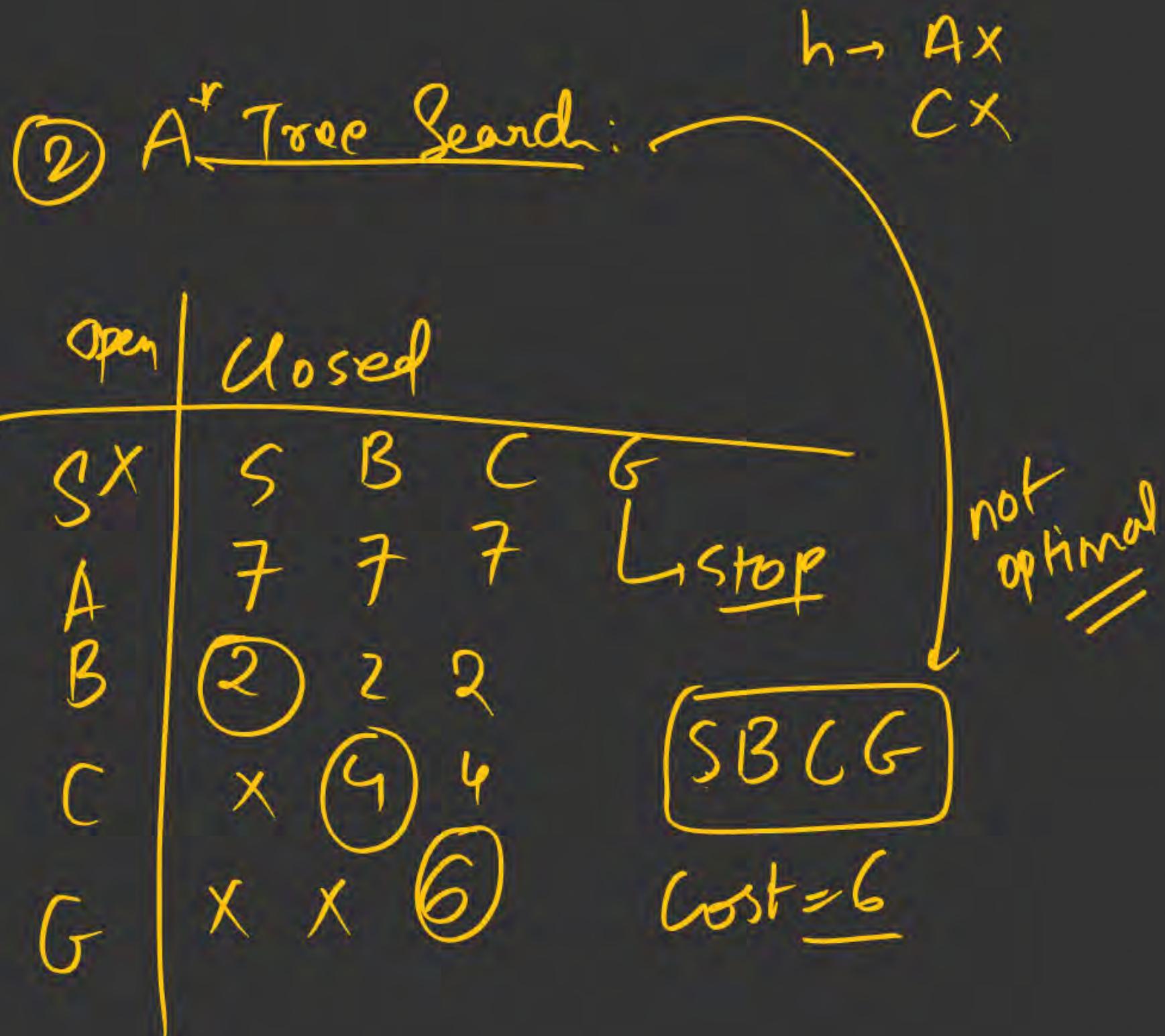
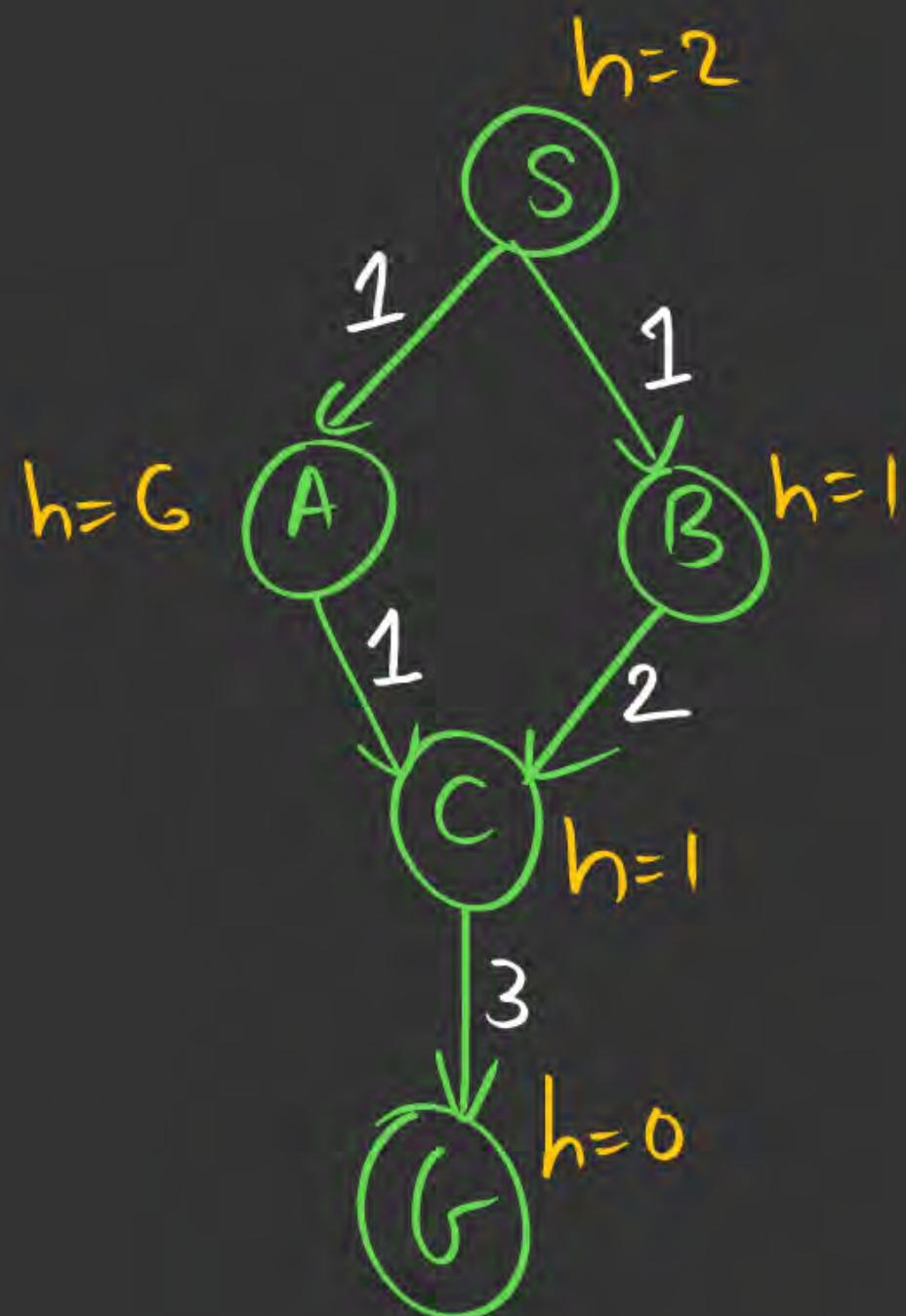
|    | Open | Closed |
|----|------|--------|
| S* | S    | B C G  |
| A  | 7    | 7 7    |
| B  | 2    | 2 2    |
| C  | 4    |        |
| G  | 6    |        |

Stop  
 SBCG  
 Cost = 6

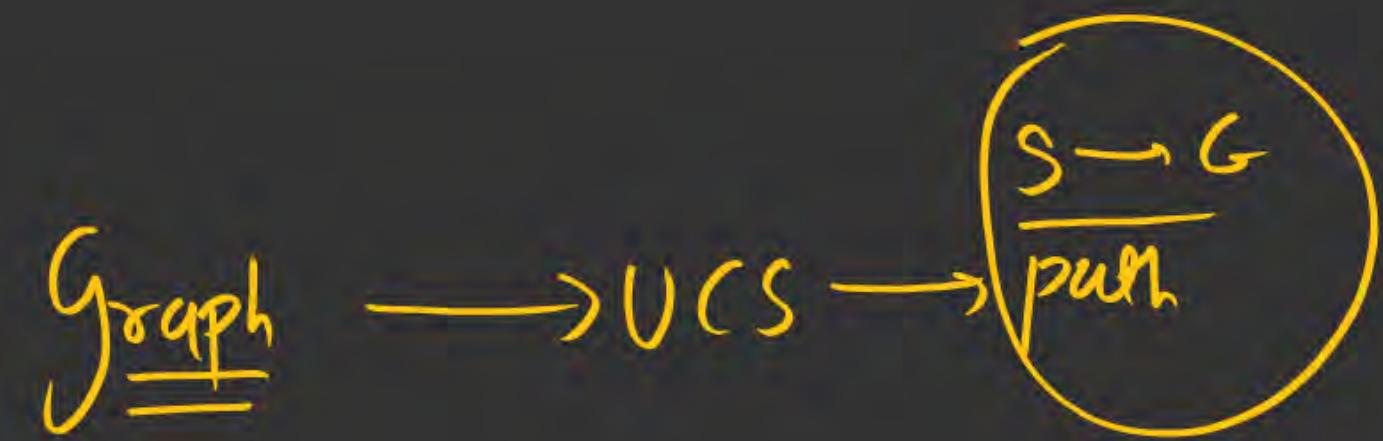
Bad Heuristics

$f(A^*)$   
 $f(C^*)$

not optimal



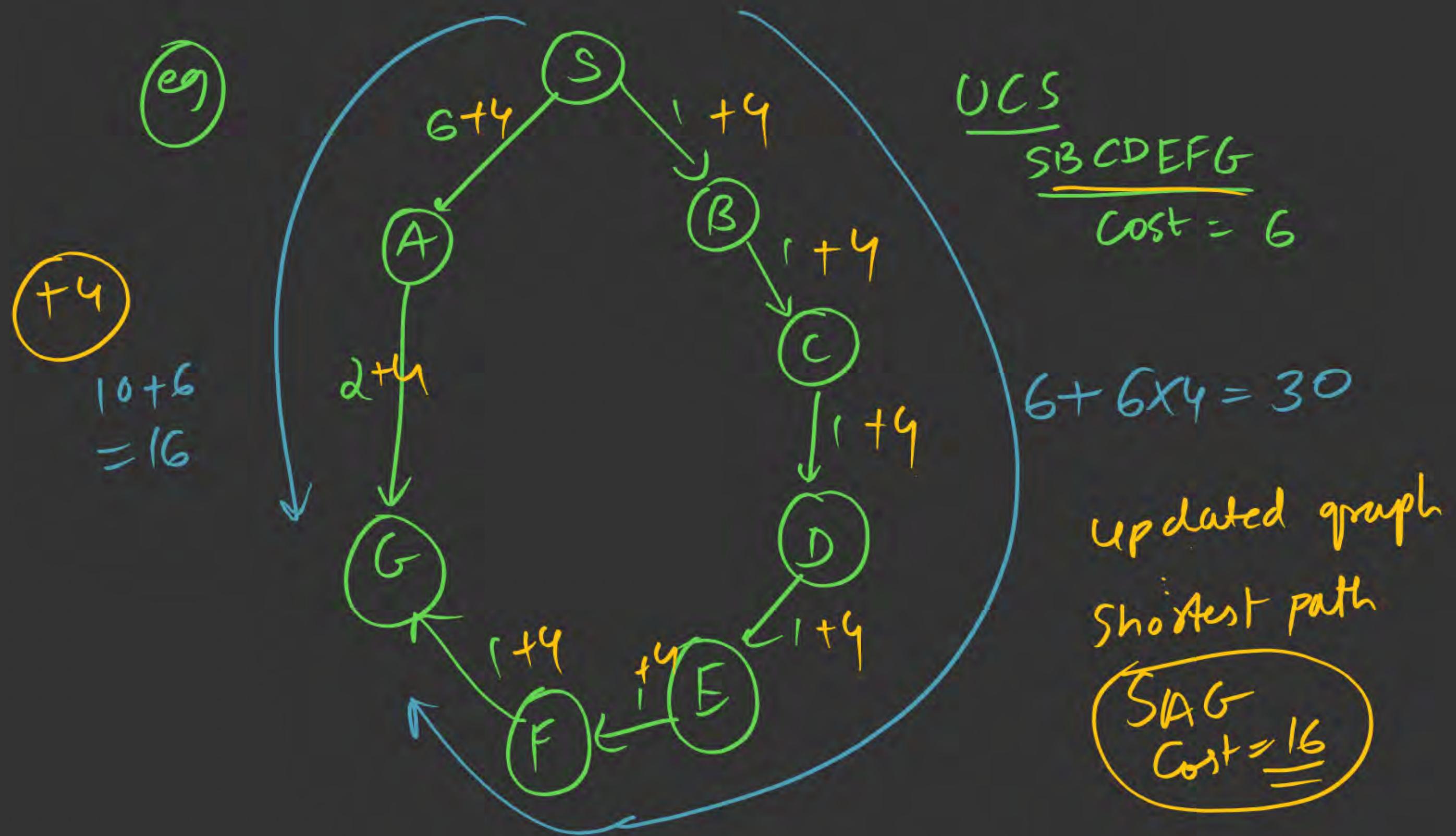
(Q.1)



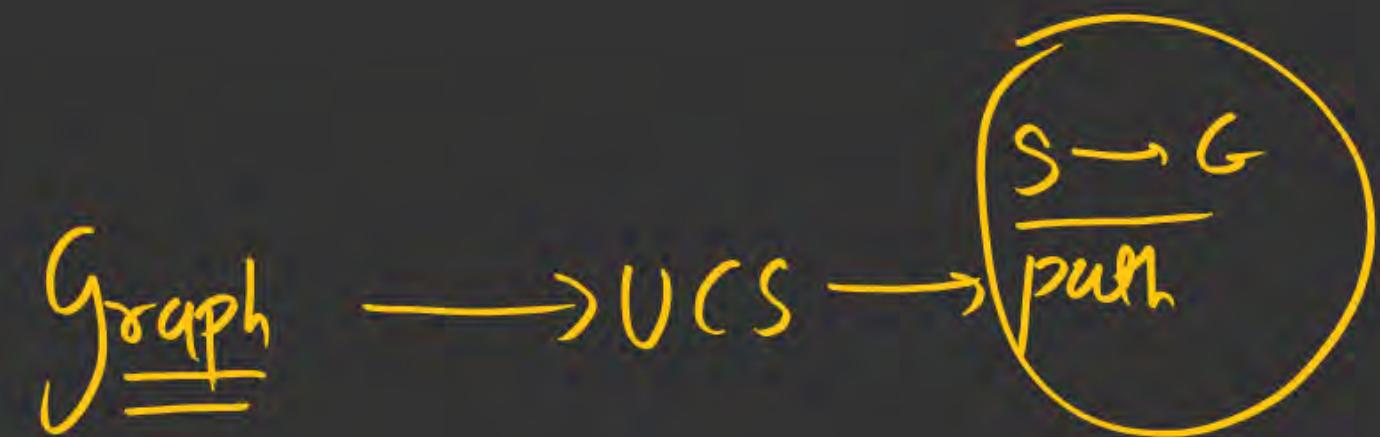
↓ + C

Graph' → UCS → Can path change?  
Yes it may change

The diagram shows a flow from "Graph'" to "UCS" to a question "Can path change?". A blue circle contains the word "Yes" and an arrow points from it to the text "it may change".

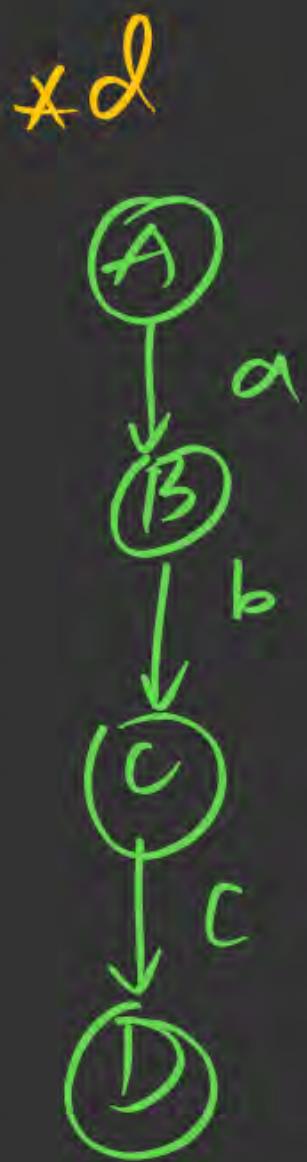


(q.2)

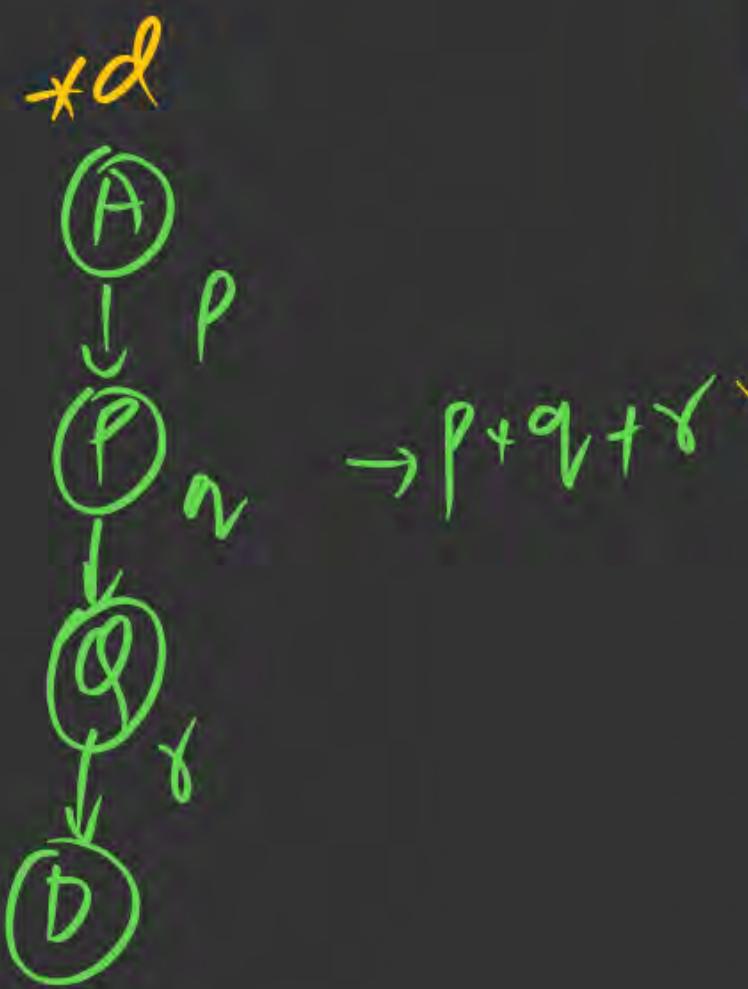


$\downarrow *C$  (every edge multiplied by some const  $C$ )

Graph'  $\rightarrow$  UCS  $\rightarrow$  Can path change?  
No, will not change



$$\rightarrow \underline{\underline{a+b+c}}$$



$$\rightarrow p+q+r$$

$$(a+b+c) < (p+q+r)$$

$$(a+b+c)*d < (p+q+r)*d$$

Path will  
not change



## Revision : Time and Space Complexity

- $A^* \Rightarrow O(b^d)$  → We might store all nodes (Visit all nodes)



# Revision : Heuristic – consistent and admissible

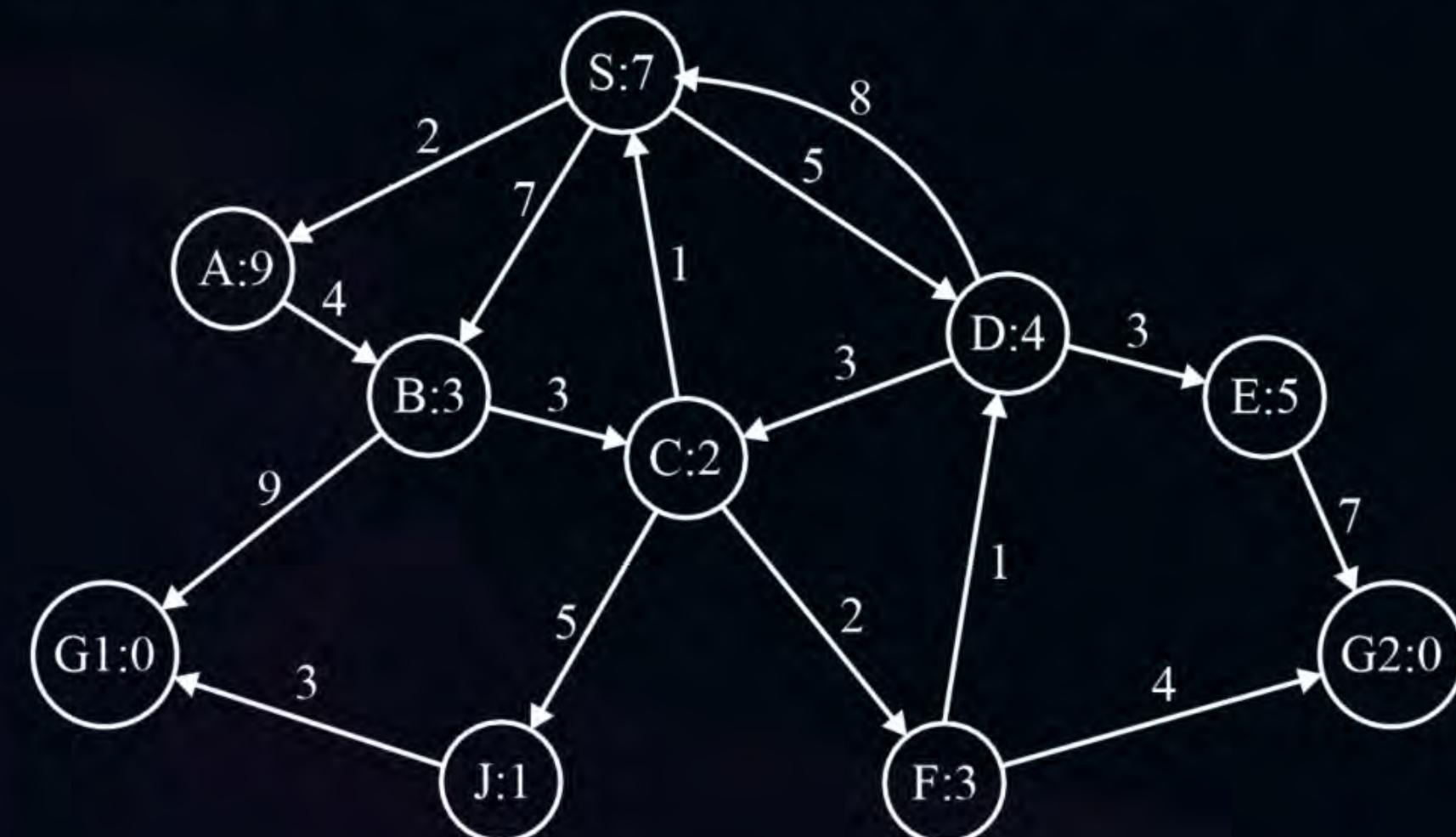
Summary:

|                               | A* tree | A* graph |
|-------------------------------|---------|----------|
| If C heuristic                | ✓       | ✓        |
| If A heuristic                | ✓       | ✗        |
| If neither C nor A heuristics | ✗       | ✗        |



## Topic : Artificial Intelligence

#Q. Consider the search space below, where S is the start node and G1 and G2 satisfy the goal test. Arcs are labeled with the cost of traversing them and the heuristic cost to a goal is reported inside nodes (so lower scores are better).





## Topic : Artificial Intelligence

For A\* search, indicate which goal state is reached at what cost and list, in order, all the states popped off of the OPEN list. You use a search graph to show your work.

Note: When all else is equal, nodes should be removed from OPEN in alphabetical order.

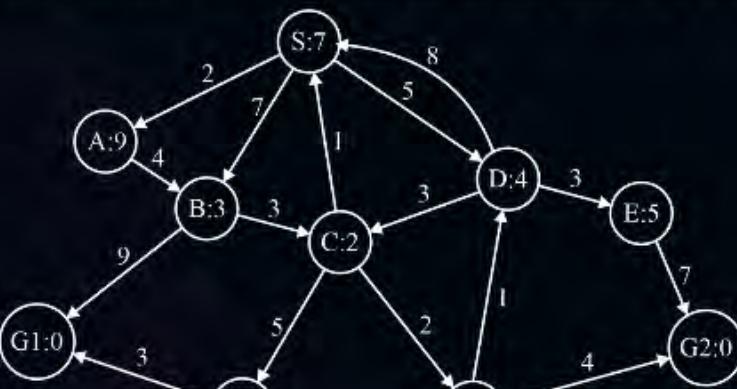


## Topic : Artificial Intelligence



#Q.

Consider the search space below, where S is the start node and G1 and G2 satisfy the goal test. Arcs are labeled with the cost of traversing them and the heuristic cost to a goal is reported inside nodes (so lower scores are better).

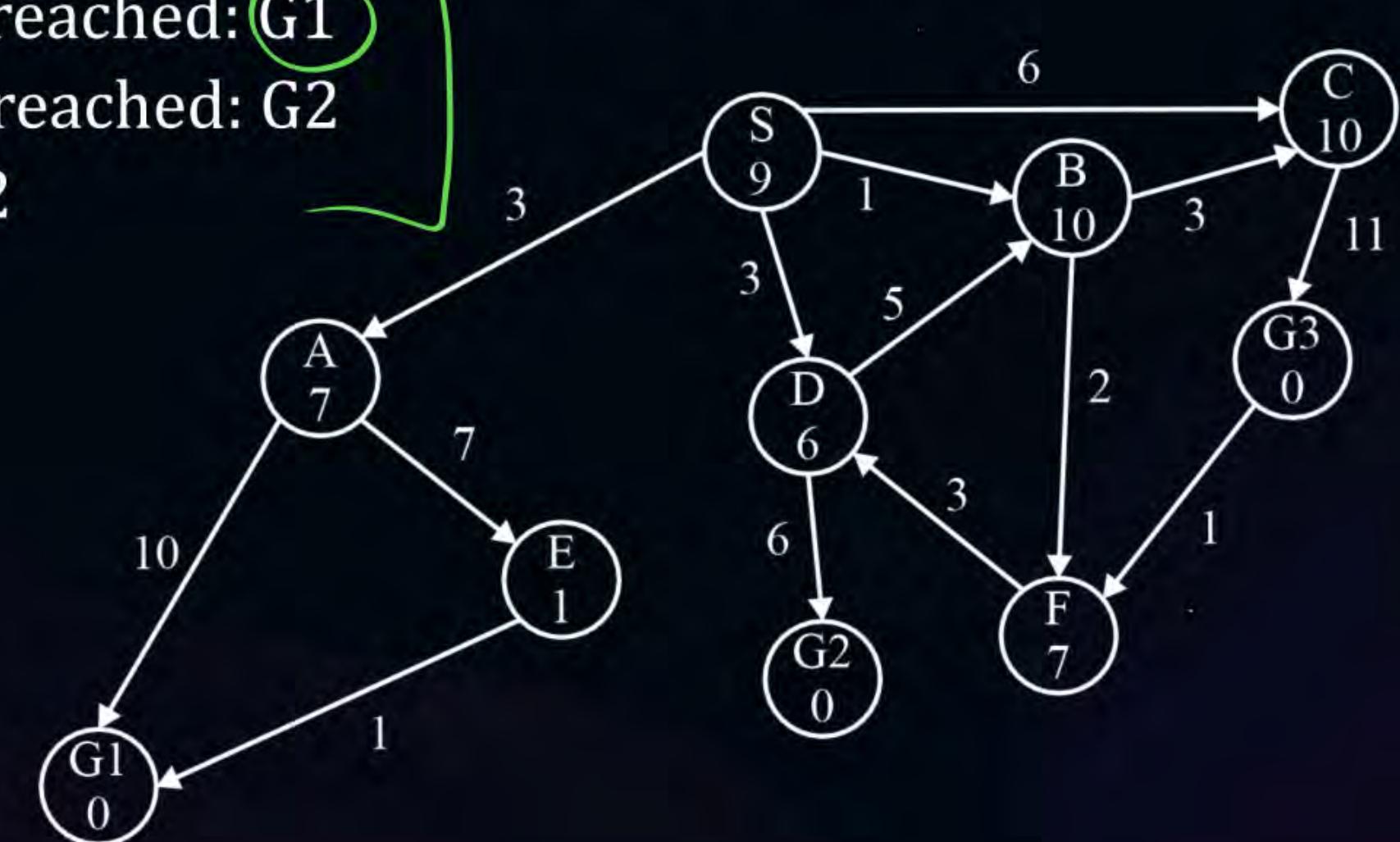
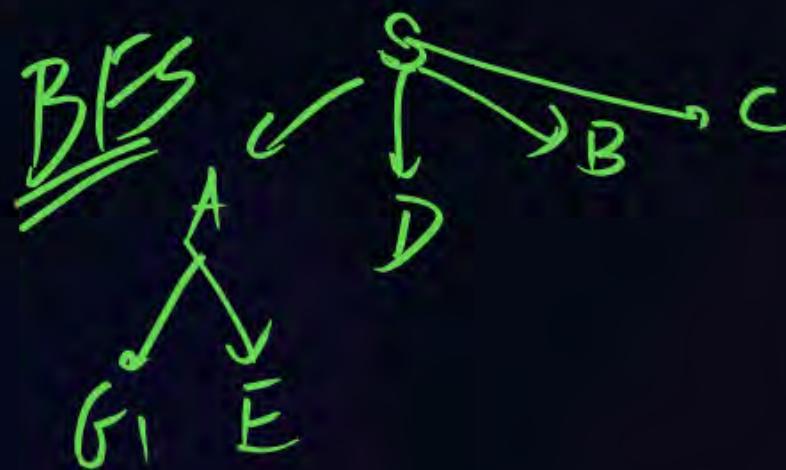




## Topic : Artificial Intelligence

#Q. Consider the search graph below, where S is the start node and G1, G2, and G3 are goal states. Arcs are labeled with the cost of traversing them and the heuristic cost to a goal is shown inside the nodes. For each of the three search strategies below, indicate which of the goal states is reached:

- Ans* {
- (a) Breadth-first search. Goal reached: G1
  - (b) Uniform cost search. Goal reached: G2
  - (c) A\* search. Goal reached: G2





# Topic : Artificial Intelligence

## Advantages

- **Optimality:** A\* is guaranteed to find the least-cost path to the goal if the heuristic used is admissible (never overestimates the true cost) and consistent (satisfies the triangle inequality).
- **Efficiency:** A\* is generally more efficient than uninformed search algorithms like Dijkstra's algorithm because it uses heuristics to guide the search. This often results in exploring fewer nodes, reducing the overall computation.
- **Flexibility:** A\* can be adapted to various problems by changing the heuristic function. This flexibility makes it applicable to a wide range of scenarios, from simple grid navigation to complex graph-based pathfinding.  $f = h + g$
- **Combines Advantages of Greedy and Uniform-Cost Search:** A\* combines the strengths of greedy best-first search (h which is fast but not always optimal) and uniform-cost search (which is optimal but can be slow) by balancing exploration and exploitation.

$f = h + g$



# Topic : Artificial Intelligence

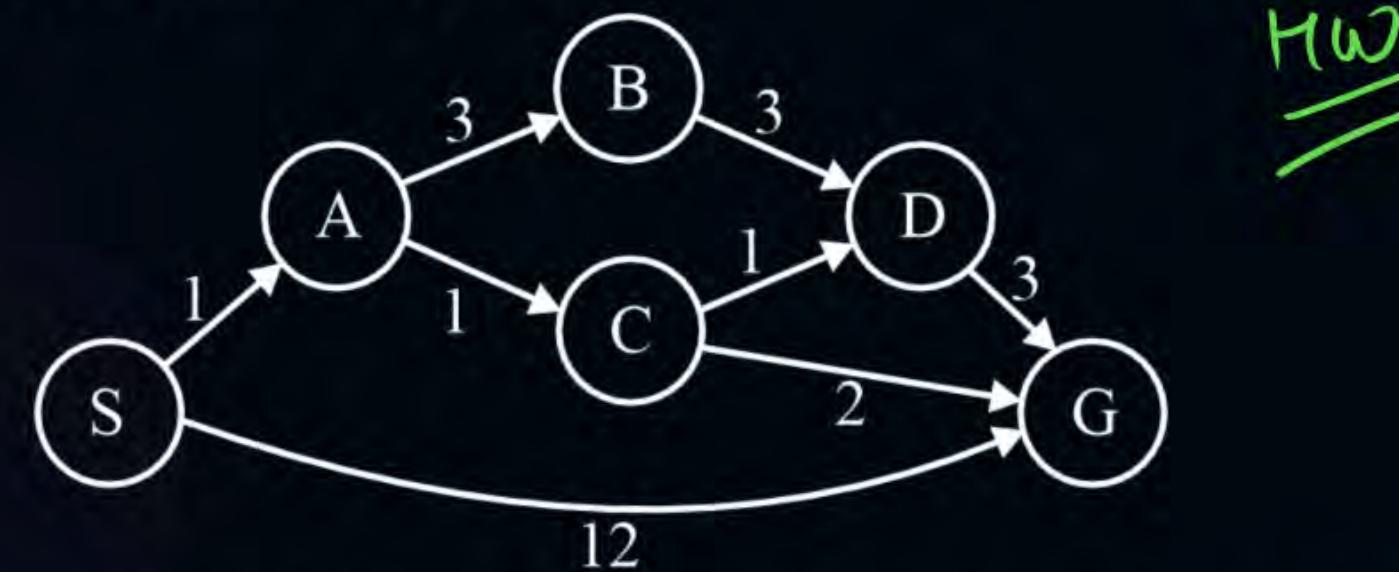
## Disadvantages

- **High Memory Usage:** A\* requires storing all generated nodes in memory, which can lead to high memory consumption, especially in large search spaces. This makes it less suitable for problems with very large state spaces.
- **Computationally Intensive:** In the worst case, A\* can be computationally expensive, especially if the branching factor is high or the heuristic is not well-optimized. The algorithm's time complexity is exponential in the worst case.
- **Heuristic Dependency:** The performance of A\* heavily depends on the quality of the heuristic. If the heuristic is poorly chosen, the algorithm can degrade to a less efficient search, potentially exploring many unnecessary nodes.
- **Not Always Practical for Real-Time Applications:** Due to its potential high time and space complexity, A\* may not be suitable for real-time applications where quick decisions are necessary.



## Topic : Artificial Intelligence

#Q. Answer the following questions about the search problem shown above. Break any ties alphabetically. For the questions that ask for a path, please give your answers in the form 'S - A - D - G'.



- (a) What path would breadth-first graph search return for this search problem?
- (b) What path would uniform cost graph search return for this search problem?
- (c) What path would depth-first graph search return for this search problem?
- (d) What path would A\* graph search, using a consistent heuristic, return for this search problem?



## Topic : Artificial Intelligence

#Q. Consider the heuristics for this problem shown in the table below.

- (i) Is  $h_1$  admissible? Yes No
- (ii) Is  $h_1$  consistent? Yes No
- (iii) Is  $h_2$  admissible? Yes No
- (iv) Is  $h_2$  consistent? Yes No

| State | $h_1$ | $h_2$ |
|-------|-------|-------|
| S     | 5     | 4     |
| A     | 3     | 2     |
| BH    | 6     | 6     |
| C     | 2     | 1     |
| D     | 3     | 3     |
| G     | 0     | 0     |



## Topic : Artificial Intelligence

#Q. Let  $h_1$  and  $h_2$  be two admissible heuristics used in A\* search.

Which ONE of the following expressions is always an admissible heuristic?

A

$$h_1 + h_2 \times$$

B

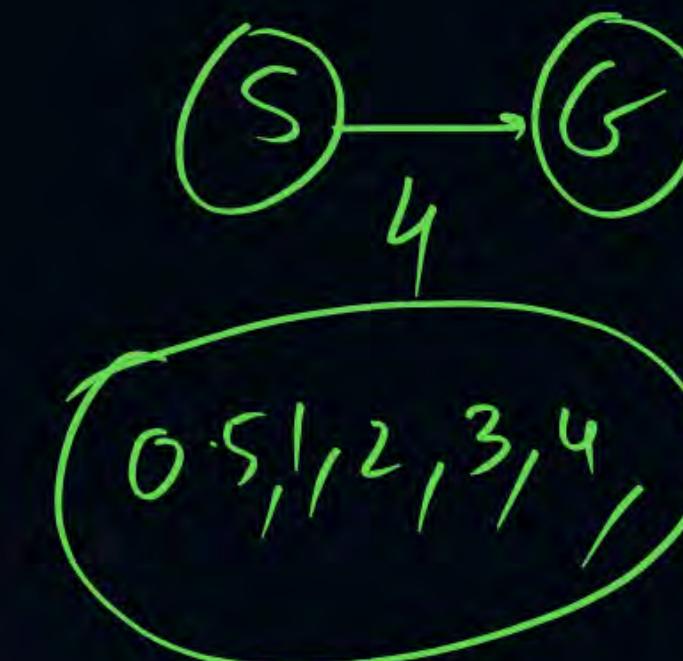
$$h_1 \times h_2 \times$$

C

$$h_1 / h_2, (h_2 \neq 0) \times$$

D

$$|h_1 - h_2| \checkmark$$



$$\frac{4}{0.5} = 8$$

$$9/12 \rightarrow$$

$$\frac{35}{1}$$



## Topic : Artificial Intelligence



#Q. If  $h_1$  and  $h_2$  are two admissible heuristics, then which of the following are guaranteed to be admissible?

- A  $\min(h_1, h_2)$
- B  $\max(h_1, h_2)$
- C  $(h_1 + h_2) / 2$
- D  $(h_1 \cdot h_2)^{0.5}$

$$h \leq h^*$$

$$h \leq 15$$

$$h = (1, 2, \dots, 14, 15)$$

$$\frac{14+15}{2} \leq 15$$

$$\sqrt{14 \times 15} \leq 15$$



## Topic : Artificial Intelligence

### **Weighted A\* search Algorithm:**

- Weighted A\* is a variant of the classic A\* algorithm used for faster pathfinding by introducing a weight factor to the heuristic. It trades optimality for speed, making it useful in time-critical applications.
- A\* guarantees optimal path (if heuristic is admissible and consistent) but can be slow. Weighted A\* speeds up the search by being more aggressive in exploring nodes closer to the goal, even if it sacrifices optimality.

1) GBFS  $\rightarrow$  fast

$$f = h$$

2)  $A^*$   $\rightarrow$  slow ( $f = \underline{h+g}$ )  $\Rightarrow$

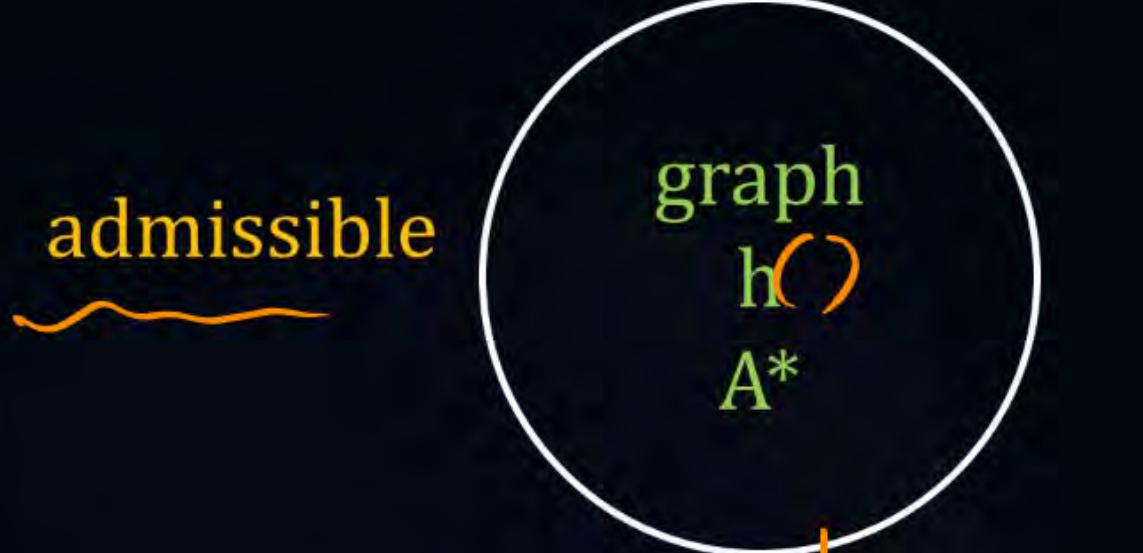
3) weighted  $A^*$   $\rightarrow$  fast  $\Rightarrow$   $f = \boxed{g + w \times h(n)}$

but may  
not be  
optimal

$$\underline{w}$$

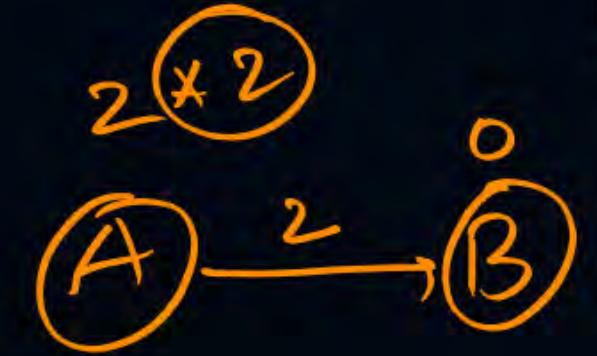
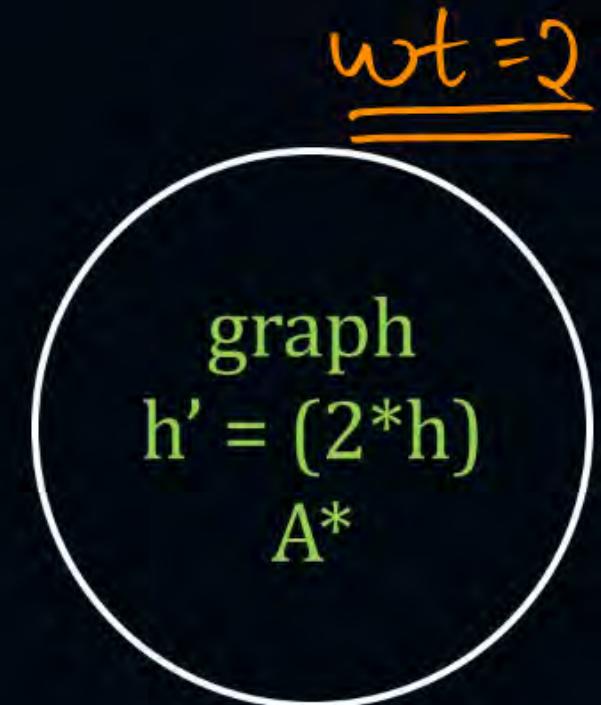


# Topic : Artificial Intelligence



$$h \leq h^*(C) \rightarrow$$

Simple  $A^*$  search  
Optimal  
Solut.



$h'$  = no more admissible  
So, no more optimal

Same as applying weighted  $A^*$  on original graph with weight = 2



# Topic : Artificial Intelligence

## Weighted A\* search Algorithm:

- Classic A\* uses:

$$f(n) = g(n) + h(n)$$

- Weighted A\* modifies this to:

$$f(n) = g(n) + \underline{w} \cdot h(n)$$

Where:

- $g(n)$  = cost from start to current node n
- $h(n)$  = heuristic estimate from n to goal
- $w \geq 1$  = weight factor



## Topic : Artificial Intelligence

### Weighted A\* search Algorithm:

- If  $w = 1$  : behaves like classic A\*, optimal and complete
- If  $w > 1$  : heuristic is over-emphasized, search is faster but not guaranteed to be optimal
- Higher  $\underbrace{w}_{\text{the more}}$  greedy the algorithm becomes (closer to Greedy Best-First Search)



$$F = g + \underbrace{w \cdot h}_{\underline{\underline{h}}} = \underline{\underline{g+h}}$$



# Topic : Artificial Intelligence

## Weighted A\* search Algorithm:

Summary

| Algorithm    | Modification                 | Notes                          |
|--------------|------------------------------|--------------------------------|
| A*           | $f(n) = g(n) + h(n)$         | Optimal                        |
| Weight A*    | $f(n) = g(n) + w \cdot h(n)$ | Faster, <u>suboptimal</u>      |
| Greedy BFS   | $f(n) = h(n)$                | Fastest, no optimality         |
| Uniform Cost | $f(n) = g(n)$                | Ignores heuristic, but optimal |



**THANK - YOU**



# DS & AI ENGINEERING



## Artificial Intelligence

### Informed search

Lecture No.- 04

By- Aditya sir



# Recap of Previous Lecture



Topic

Topic

Topic

Topic

*A\* Search*

*Weighted A\* Search*

# Topics to be Covered



Topic

Topic

Topic

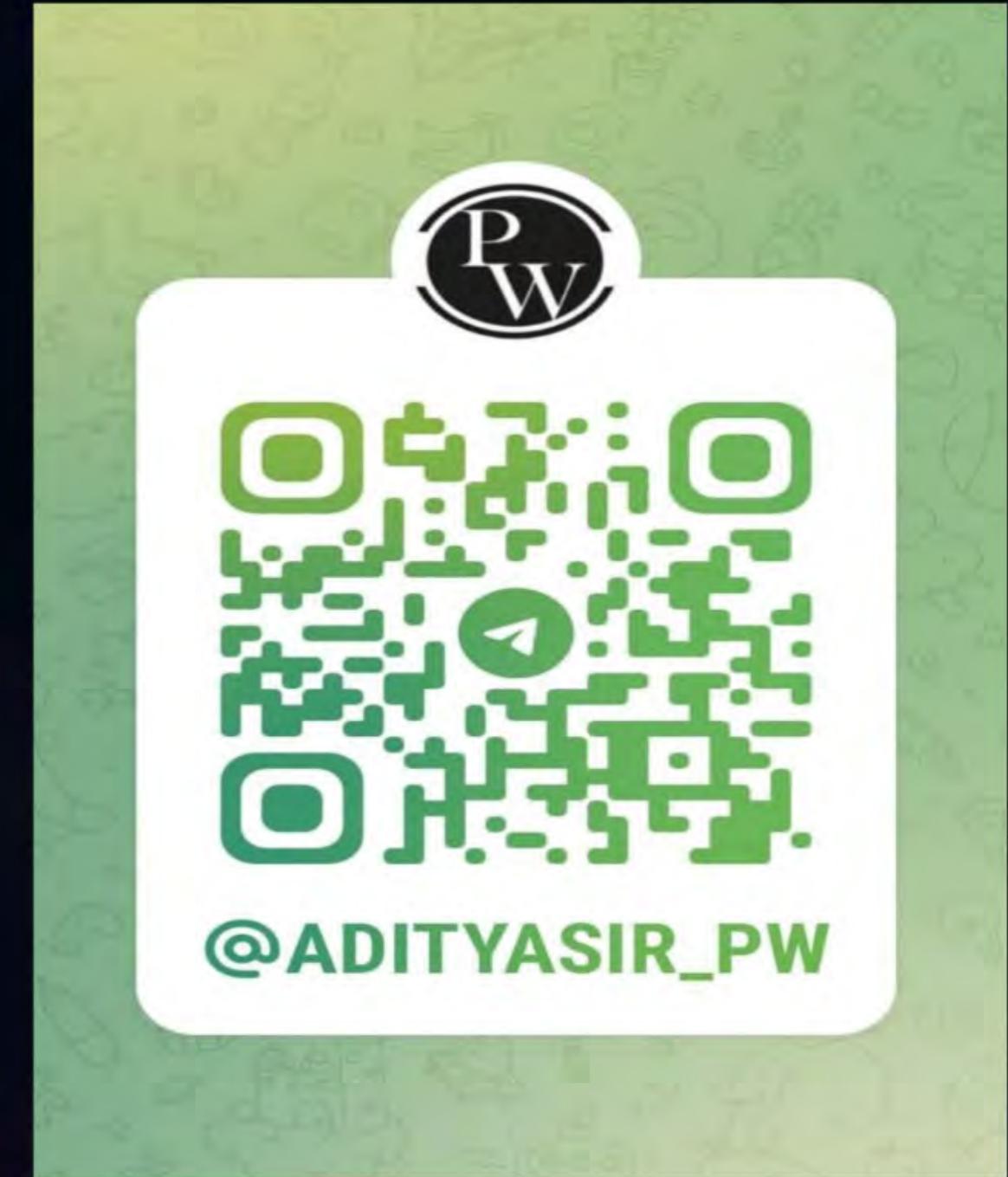
Practice Questions



## About Aditya Jain sir

1. Appeared for GATE during BTech and secured AIR 60 in GATE in very first attempt - City topper
2. Represented college as the first Google DSC Ambassador.
3. The only student from the batch to secure an internship at Amazon. (9+ CGPA)
4. Had offer from IIT Bombay and IISc Bangalore to join the Masters program
5. Joined IIT Bombay for my 2 year Masters program, specialization in Data Science
6. Published multiple research papers in well known conferences along with the team
7. Received the prestigious excellence in Research award from IIT Bombay for my Masters thesis in ML
8. Completed my Masters with an overall GPA of 9.36/10
9. Joined Dream11 as a Data Scientist
10. Have mentored 15,000+ students & working professionals in field of Data Science and Analytics
11. Have been mentoring & teaching GATE aspirants to secure a great rank in limited time
12. Have got around 27.5K followers on Linkedin where I share my insights and guide students and professionals.

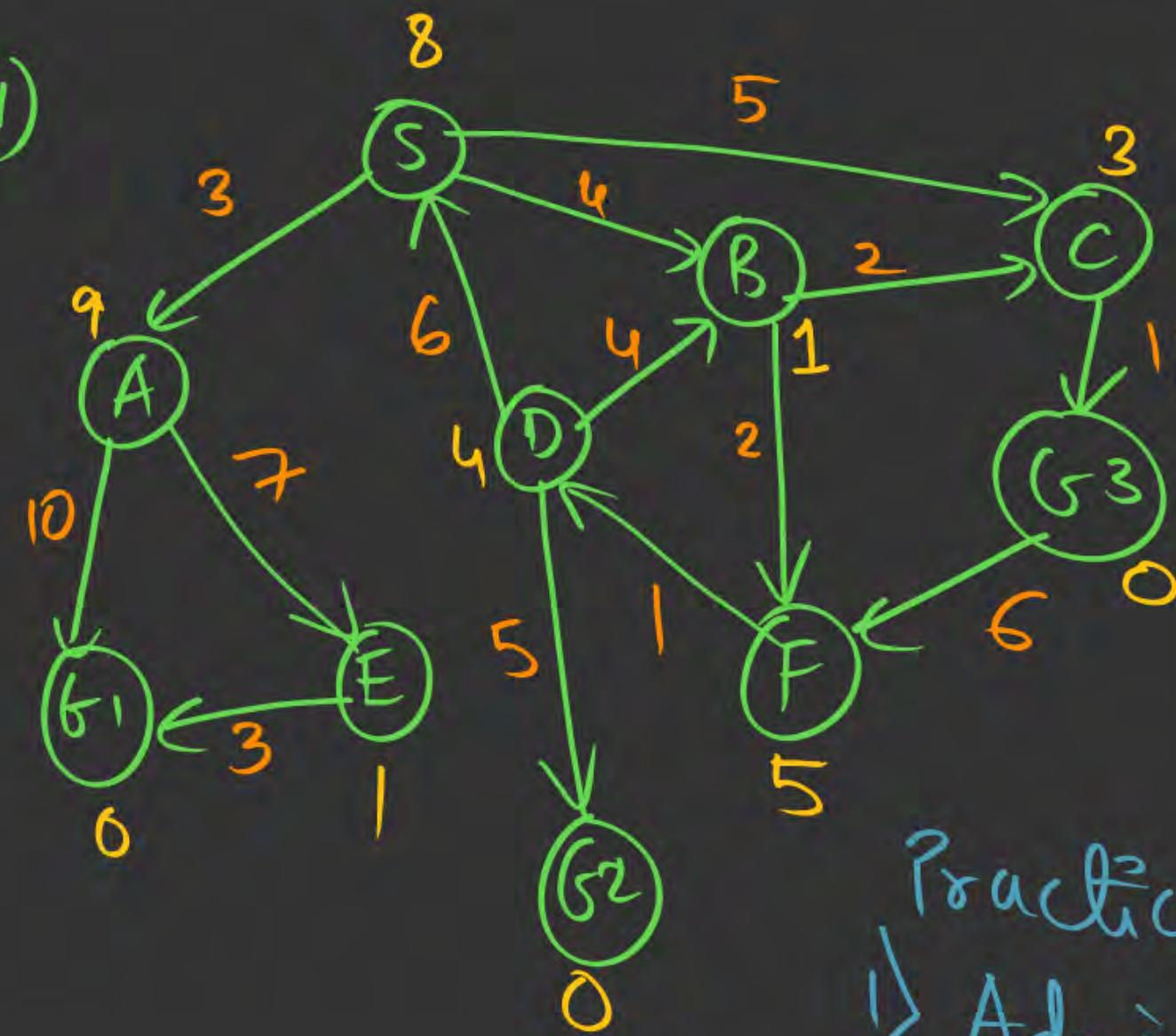




Telegram

**Telegram Link for Aditya Jain sir:**  
**[https://t.me/AdityaSir\\_PW](https://t.me/AdityaSir_PW)**

(Q.1)



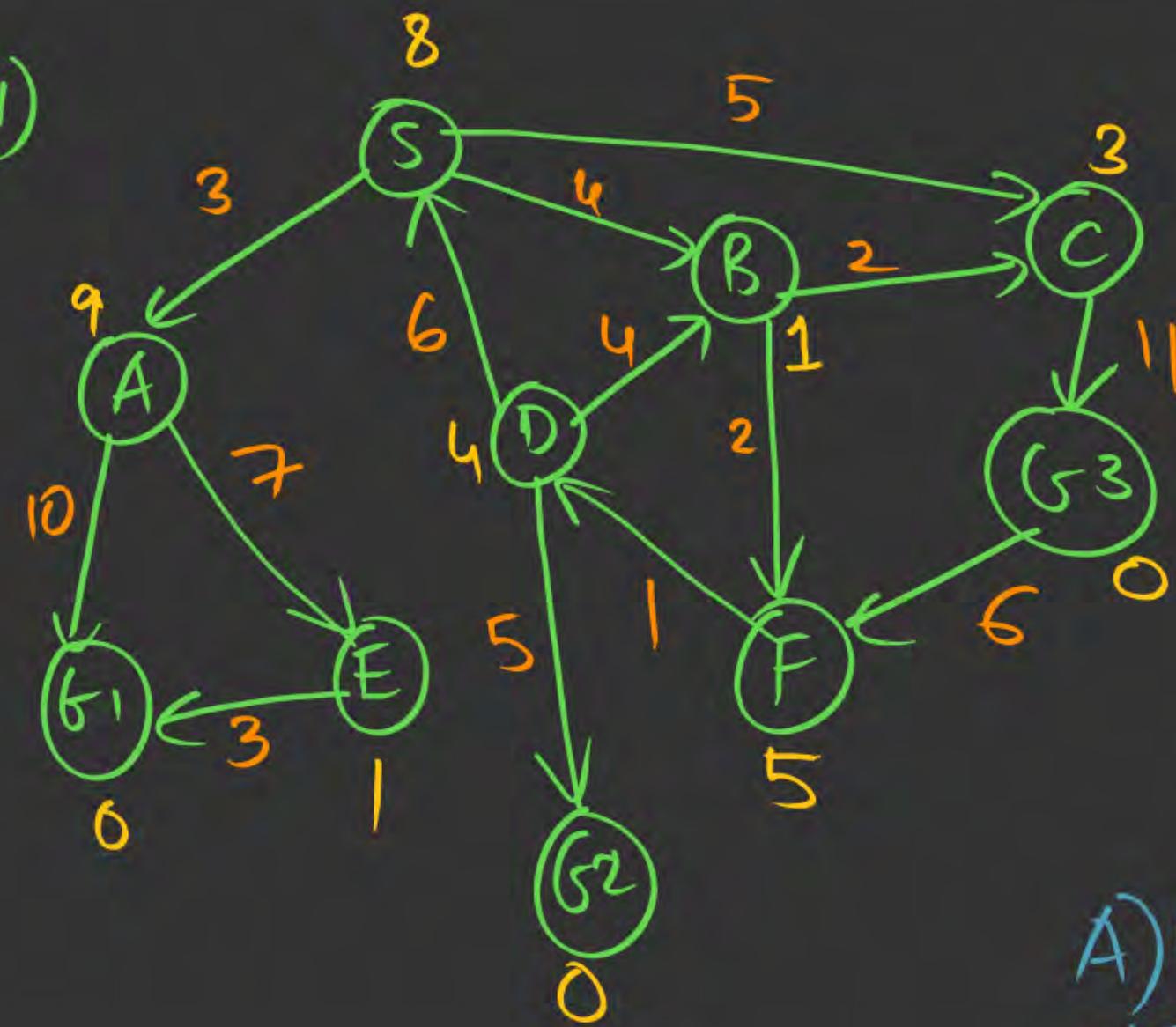
Practice  
1) Admissible?  
2) Consistent?

1) Apply A\* Graph Search  
to find which goal is  
reached first?

Start: S

if same cost:  $\frac{G_1 > G_2 > G_3}{}$

(Q.1)



- A) G1  
B) G2  
C) G3

✓ i) Apply A\* Graph Search  
to find which goal is  
reached first?

Start: S

if same cost  
among  
goals,

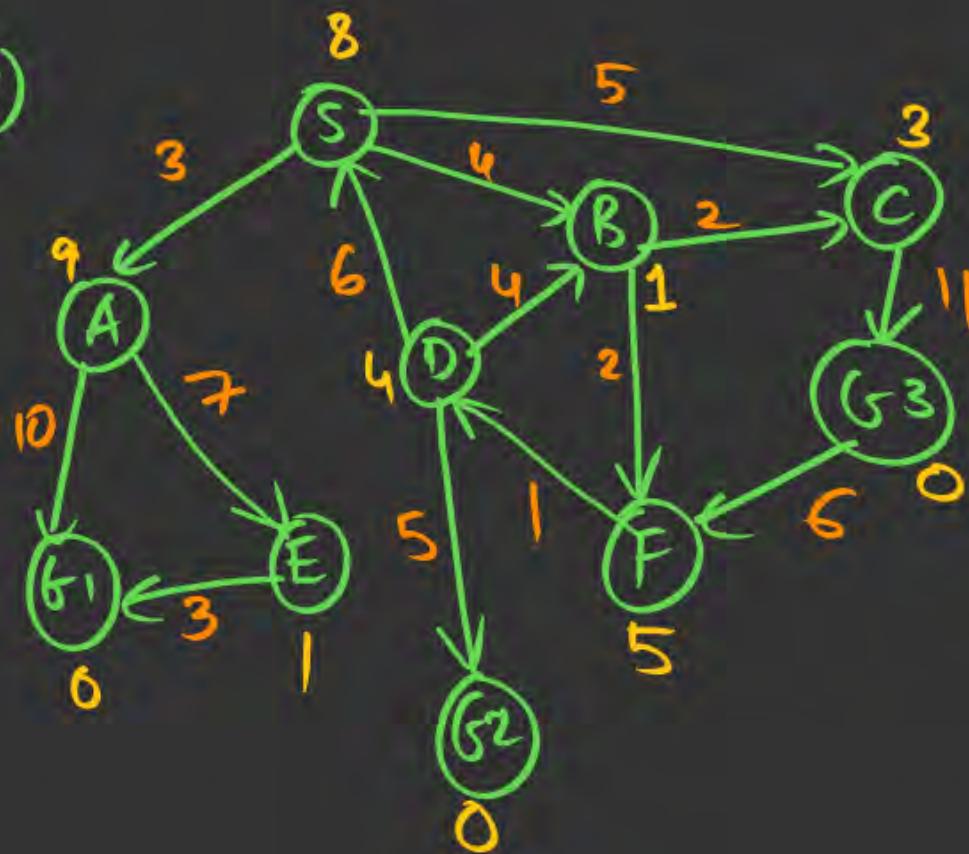
path:

cost: 12

$$\boxed{G_1 > G_2 > G_3}$$

S B F D G2

(Q.1)



Open | Closed

|                | S  | B | C  | F  | D  | A  | E  | G <sub>2</sub> |
|----------------|----|---|----|----|----|----|----|----------------|
| S              | x  |   |    |    |    |    |    |                |
| A              | 12 |   |    |    |    |    |    |                |
| B              | 5  | x |    |    |    |    |    |                |
| C              | 8  | 5 | x  |    |    |    |    |                |
| F              | x  | 8 | 8  | x  |    |    |    |                |
| G <sub>3</sub> | x  | x | 16 | 16 | 16 | 16 | 16 |                |
| D              | x  | x | x  | x  | x  | x  | x  |                |
| G <sub>2</sub> | x  | x | x  | x  | x  | x  | x  | x              |
| E              | x  | x | x  | x  | x  | x  | x  | x              |
| G <sub>1</sub> | x  | x | x  | x  | x  | x  | x  | x              |

Cost = 12  
path = [S, B, F, D, G<sub>2</sub>] ✓



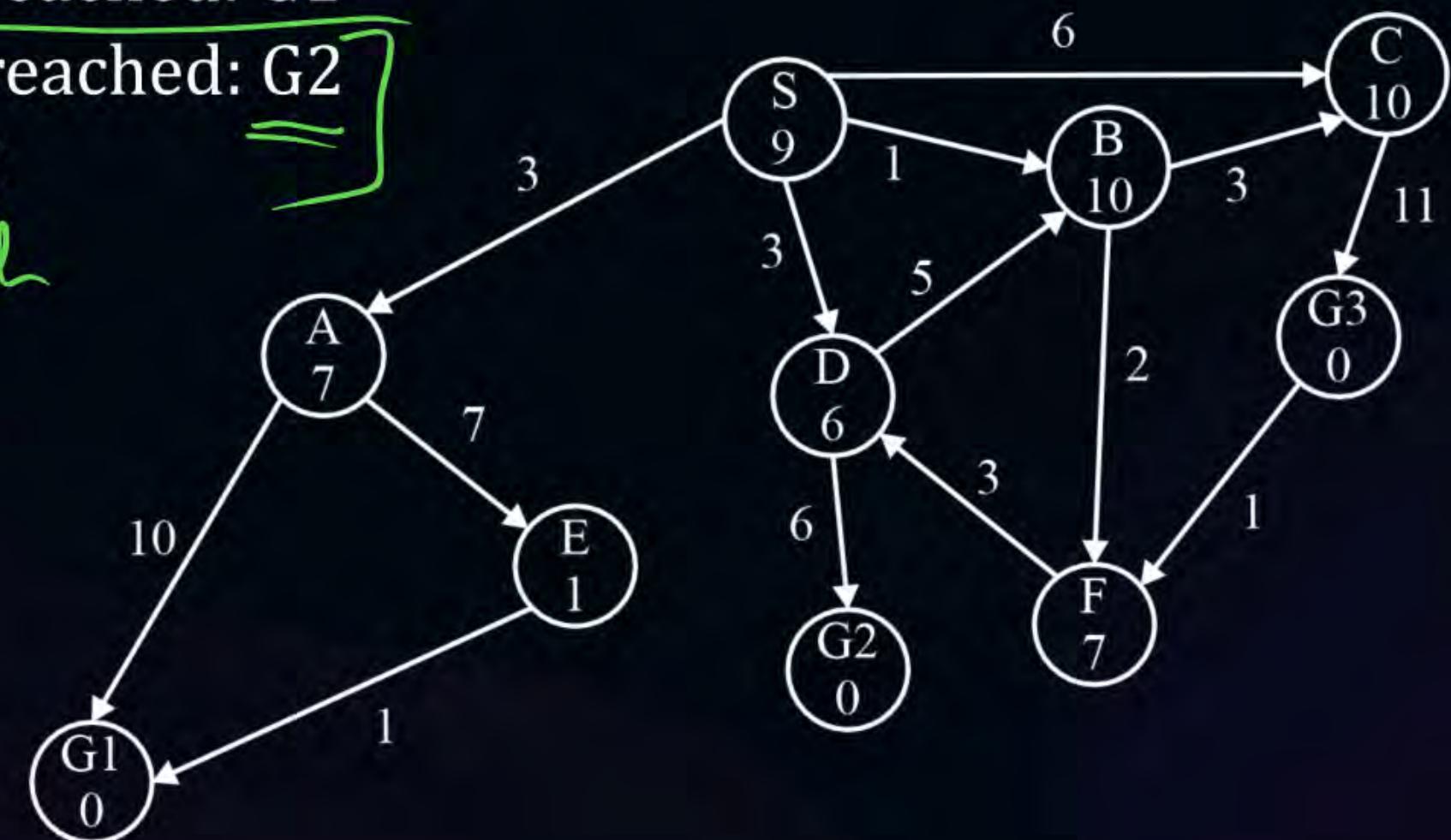
# Topic : Artificial Intelligence

#Q. Consider the search graph below, where S is the start node and G1, G2, and G3 are goal states. Arcs are labeled with the cost of traversing them and the heuristic cost to a goal is shown inside the nodes. For each of the three search strategies below, indicate which of the goal states is reached:

- (a) Breadth-first search. Goal reached: G1
- (b) Uniform cost search. Goal reached: G2
- (c) A\* search. Goal reached: G2

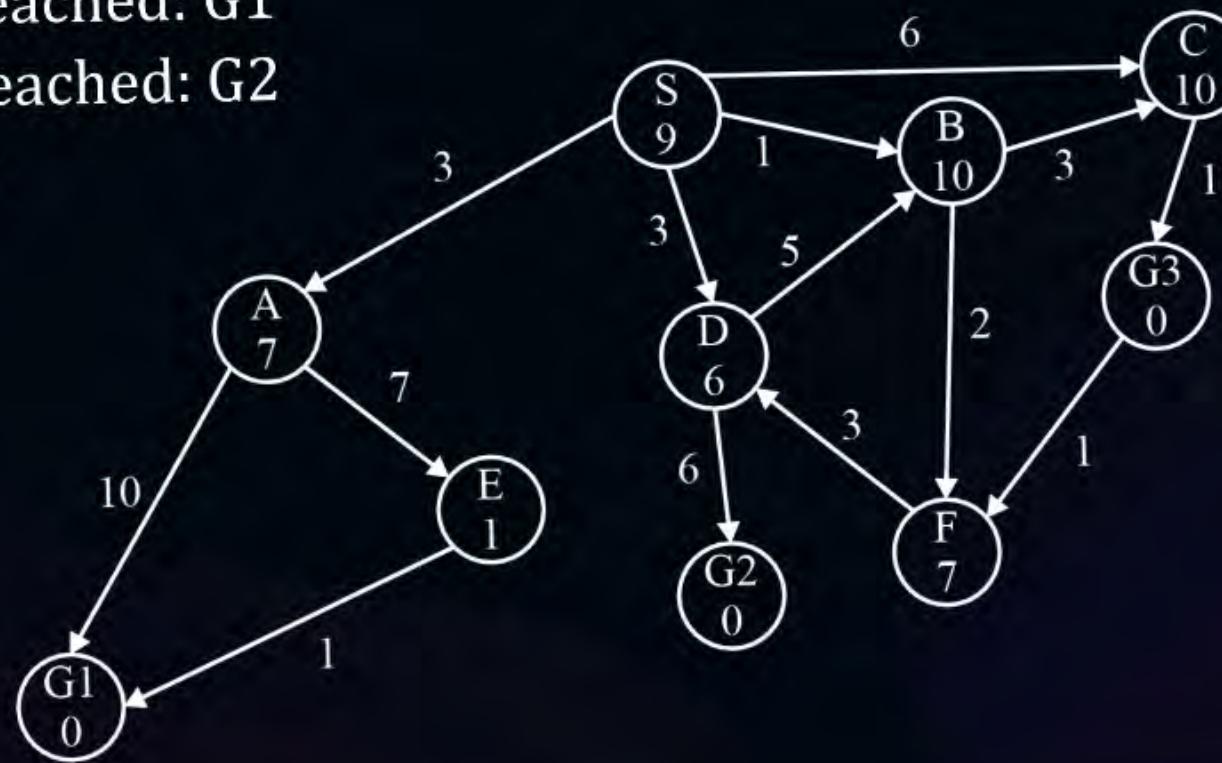
cost =  
path =

graph search



reached: G1  
reached: G2

2



SDC, 9

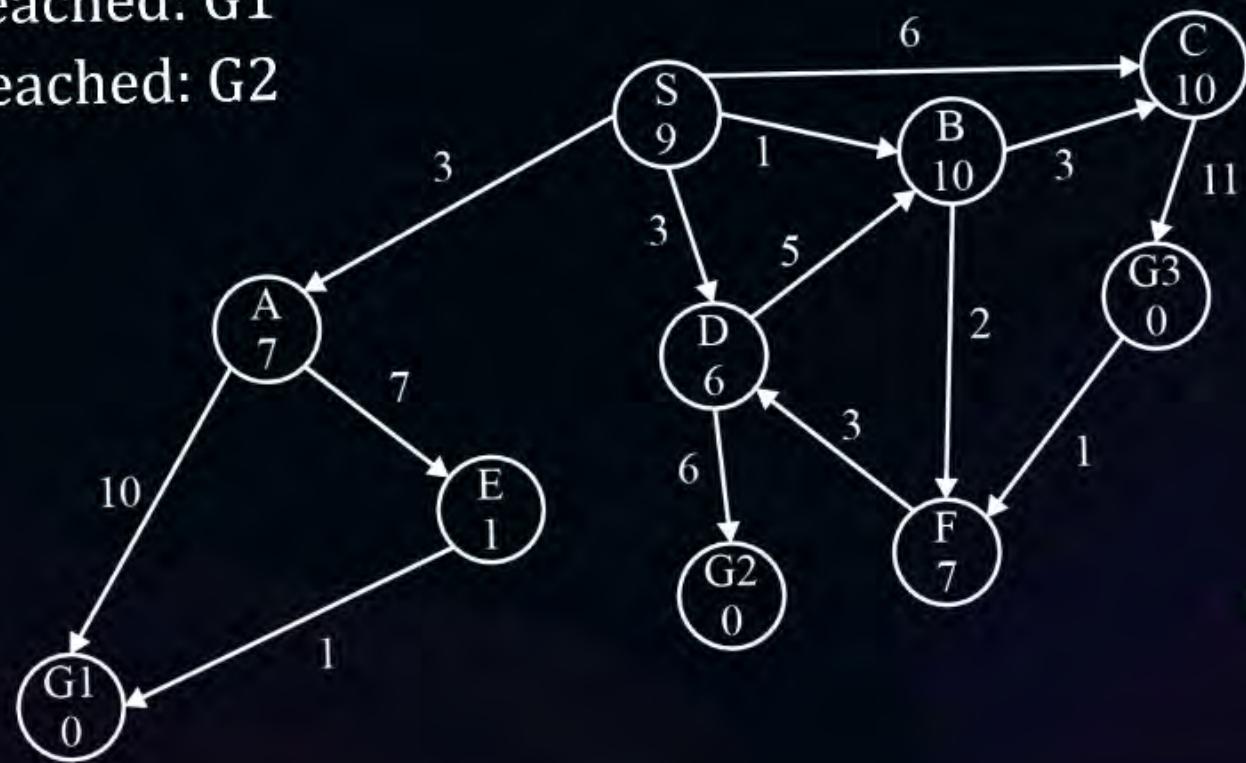
▷ UCS

|      | Open  | Closed |    |    |    |    |  |
|------|-------|--------|----|----|----|----|--|
| S X  | S     |        |    |    |    |    |  |
| A X  | 3     | 3      |    |    |    |    |  |
| B X  | 1     | 4      |    |    |    |    |  |
| C X  | 6 → 4 | 4      |    |    |    |    |  |
| D X  | 3     | 3      | 3  | 3  | 3  | 3  |  |
| F X  | 3     | 3      | 3  | 3  | 3  | 3  |  |
| E X  | X     | 10     | 10 | 10 | 10 | 10 |  |
| G1 X | X     | 13     | 13 | 13 | 13 | 13 |  |
| G2 X | X     | X      | 9  | 9  | 9  | 9  |  |
| G3 X | X     | X      | X  | X  | X  | X  |  |

Cost = 9  
Path = S D G2

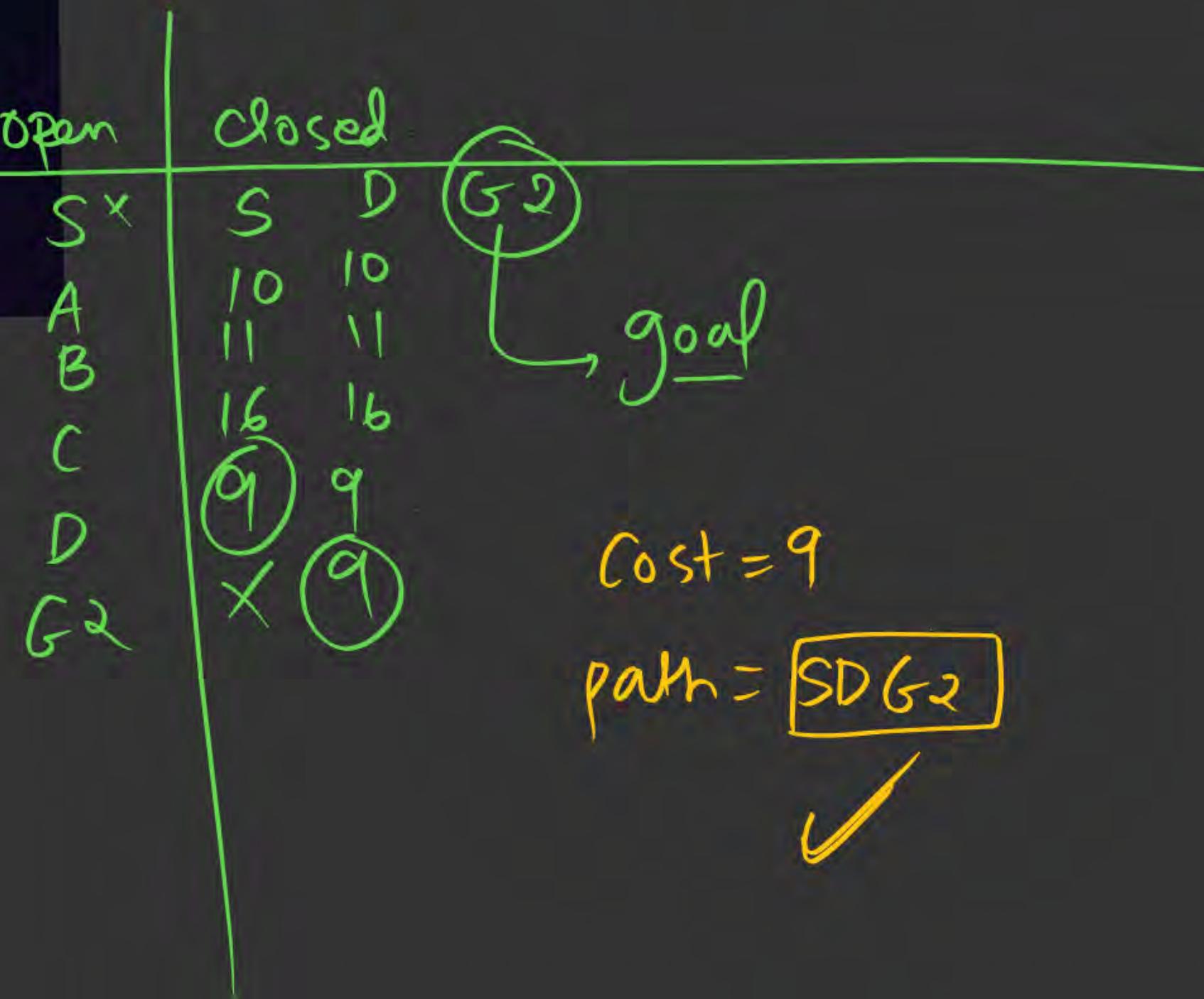
reached: G1  
reached: G2

2



2 > A\* graph Search

$$f = h + g$$





## Topic : Artificial Intelligence



### Time Complexity

THE TIME COMPLEXITY OF A\* LARGELY DEPENDS ON THE HEURISTIC FUNCTION AND THE STRUCTURE OF THE SEARCH SPACE. IN THE WORST CASE, The time complexity is  $O(b^d)$ , where:

- b is the branching factor (the average number of successors per state).
- d is the depth of the shallowest solution.

However, if the heuristic function is good (i.e., it is admissible and consistent), the time complexity can be significantly reduced, though it remains exponential in the worst case.



## Topic : Artificial Intelligence

### Space Complexity

The space complexity of A\* is generally  $O(b^d)$  as well. A\* keeps all generated nodes in memory, which can lead to high space usage, particularly in large or complex search spaces. This is because it stores the open list (the set of nodes that have been generated but not yet expanded) and the closed list (the set of nodes that have been fully explored).



## Topic : Artificial Intelligence

### Advantages

- **Optimality:** A\* is guaranteed to find the least-cost path to the goal if the heuristic used is admissible (never overestimates the true cost) and consistent (satisfies the triangle inequality).
- **Efficiency:** A\* is generally more efficient than uninformed search algorithms like Dijkstra's algorithm because it uses heuristics to guide the search. This often results in exploring fewer nodes, reducing the overall computation.
- **Flexibility:** A\* can be adapted to various problems by changing the heuristic function. This flexibility makes it applicable to a wide range of scenarios, from simple grid navigation to complex graph-based pathfinding.
- **Combines Advantages of Greedy and Uniform-Cost Search:** A\* combines the strengths of greedy best-first search (which is fast but not always optimal) and uniform-cost search (which is optimal but can be slow) by balancing exploration and exploitation.



## Topic : Artificial Intelligence



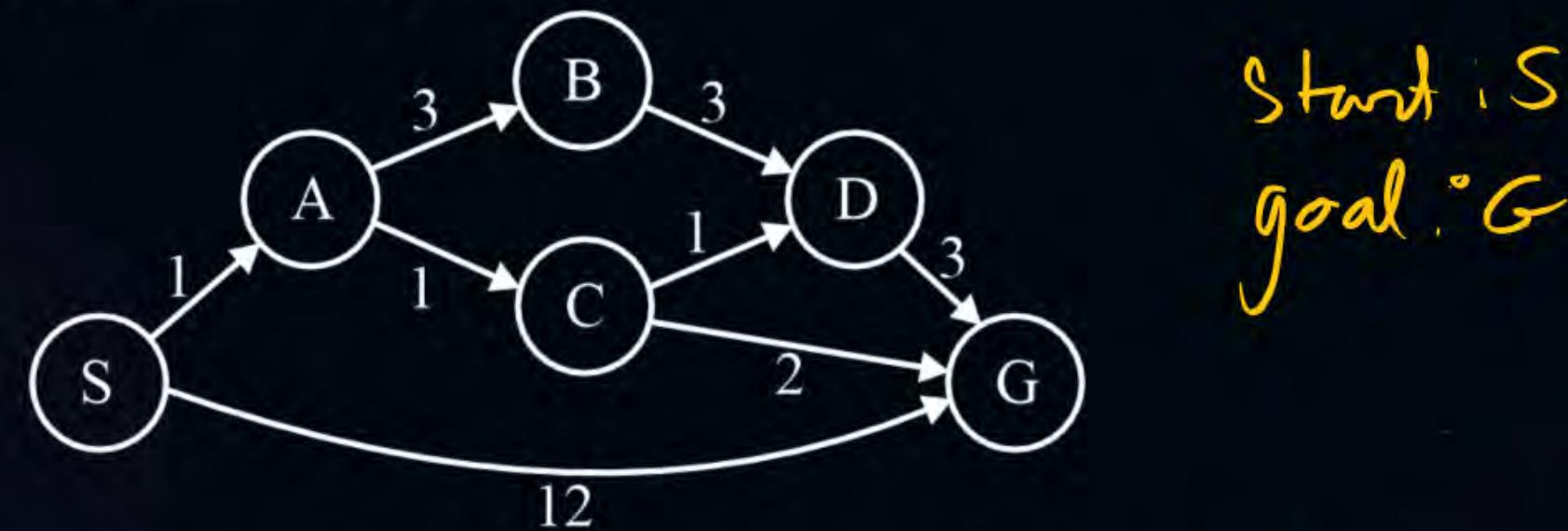
### Disadvantages

- **High Memory Usage:** A\* requires storing all generated nodes in memory, which can lead to high memory consumption, especially in large search spaces. This makes it less suitable for problems with very large state spaces.
- **Computationally Intensive:** In the worst case, A\* can be computationally expensive, especially if the branching factor is high or the heuristic is not well-optimized. The algorithm's time complexity is exponential in the worst case.
- **Heuristic Dependency:** The performance of A\* heavily depends on the quality of the heuristic. If the heuristic is poorly chosen, the algorithm can degrade to a less efficient search, potentially exploring many unnecessary nodes.
- **Not Always Practical for Real-Time Applications:** Due to its potential high time and space complexity, A\* may not be suitable for real-time applications where quick decisions are necessary.



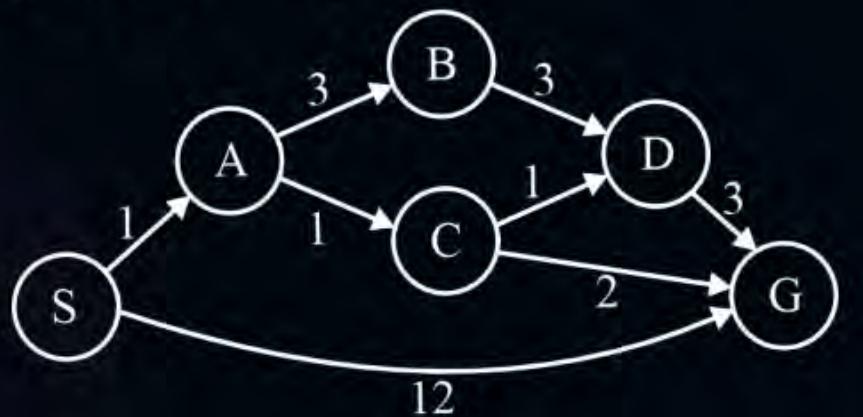
## Topic : Artificial Intelligence

#Q. Answer the following questions about the search problem shown above. Break any ties alphabetically. For the questions that ask for a path, please give your answers in the form 'S - A - D - G'.

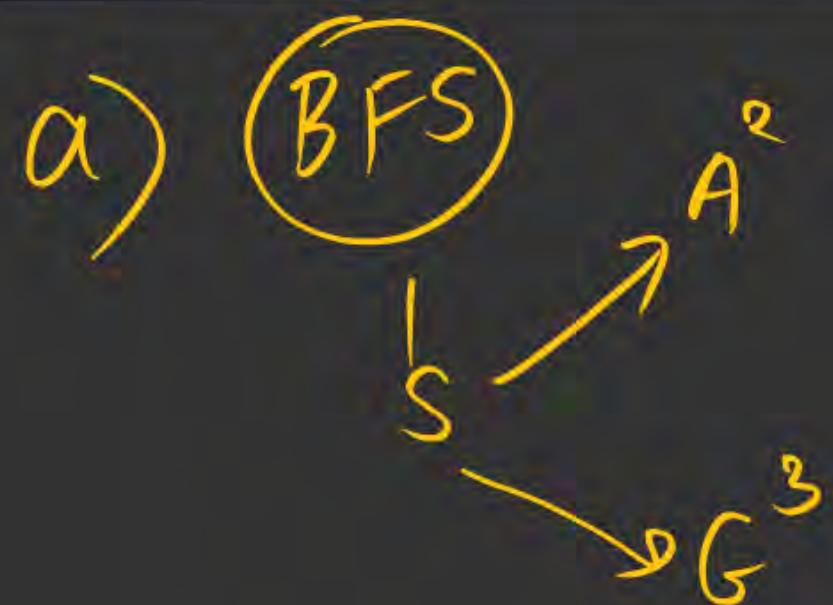


- (a) ✓ What path would breadth-first graph search return for this search problem?
- (b) ✓ What path would uniform cost graph search return for this search problem?
- (c) ✓ What path would depth-first graph search return for this search problem?
- (d) ✓ What path would A\* graph search, using a consistent heuristic, return for this search problem?

answers in the form S -> A -> ...

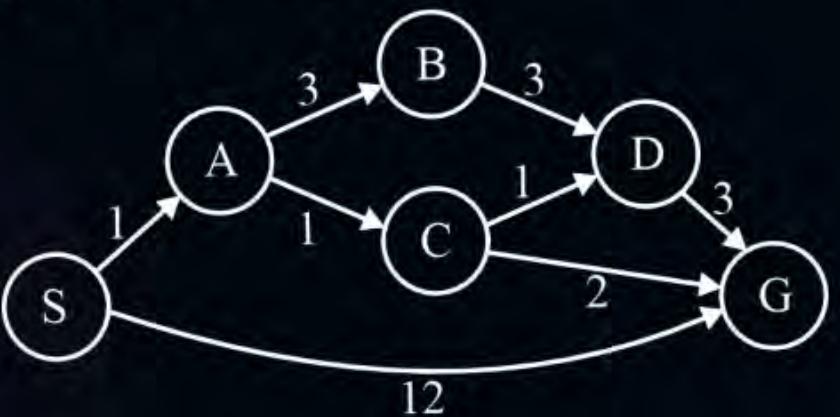


- (a) What path would breadth-first graph search return for this search problem?
- (b) What path would uniform cost graph search return for this search problem?
- (c) What path would depth-first graph search return for this search problem?
- (d) What path would A\* graph search, using a consistent heuristic, return for this search problem?



path: S-G

answers in the form S → A → B → C



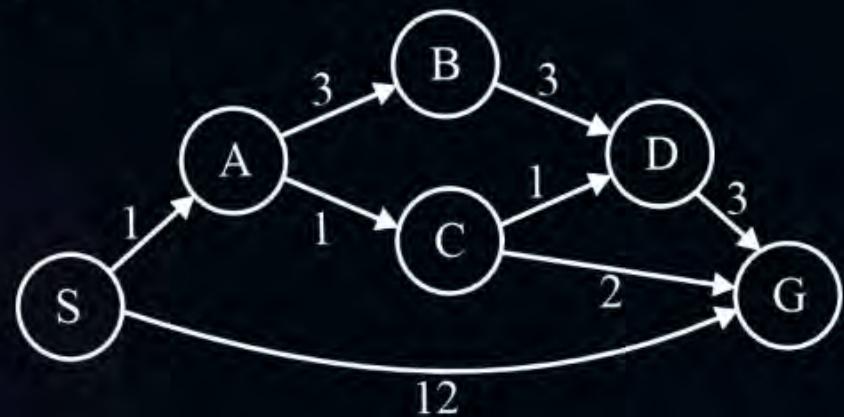
- (a) What path would breadth-first graph search return for this search problem?  
(b) What path would uniform cost graph search return for this search problem?  
(c) What path would depth-first graph search return for this search problem?  
(d) What path would A\* graph search, using a consistent heuristic, return for this search problem?

b) UCS

|    | open | closed |   |   |   |   |
|----|------|--------|---|---|---|---|
| S* |      | S      | A | C | D | B |
| A* | ①    |        |   |   |   | G |
| G* | 12   | 12     | 4 |   |   |   |
| B* | X    |        | 4 |   |   |   |
| C* | X    | X      | 2 | 2 | 2 |   |
| D* | X    | X      | 3 | 3 | 3 |   |

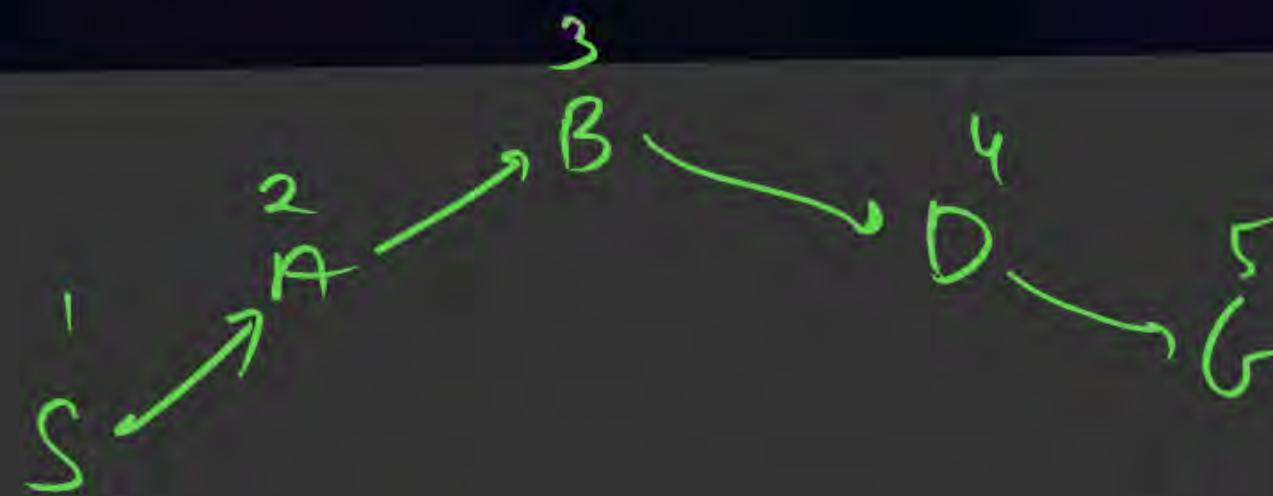
cost = 4  
path: SACG

answers in the form S -> A -> ...



- (a) What path would breadth-first graph search return for this search problem?
- (b) What path would uniform cost graph search return for this search problem?
- (c) What path would depth-first graph search return for this search problem?
- (d) What path would A\* graph search, using a consistent heuristic, return for this search problem?

c) DFS



path: SABDG



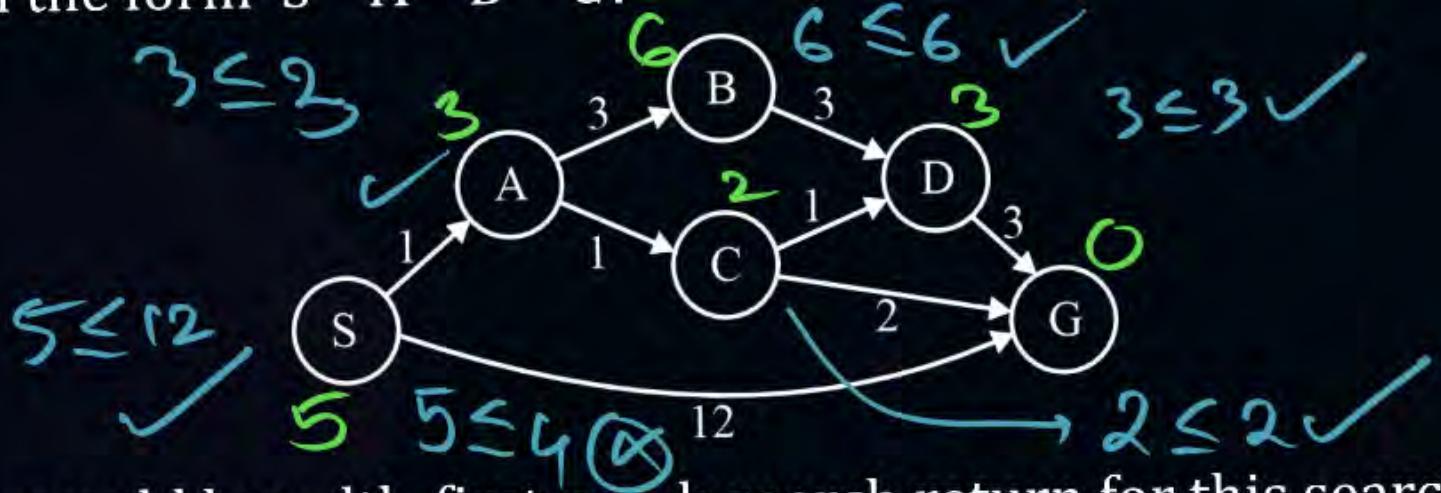
## Topic : Artificial Intelligence

#Q.

Consider the heuristics for this problem shown in the table below.

- (i) Is  $h_1$  admissible? Yes No
- (ii) Is  $h_1$  consistent? Yes No
- (iii) Is  $h_2$  admissible? Yes No
- (iv) Is  $h_2$  consistent? Yes No

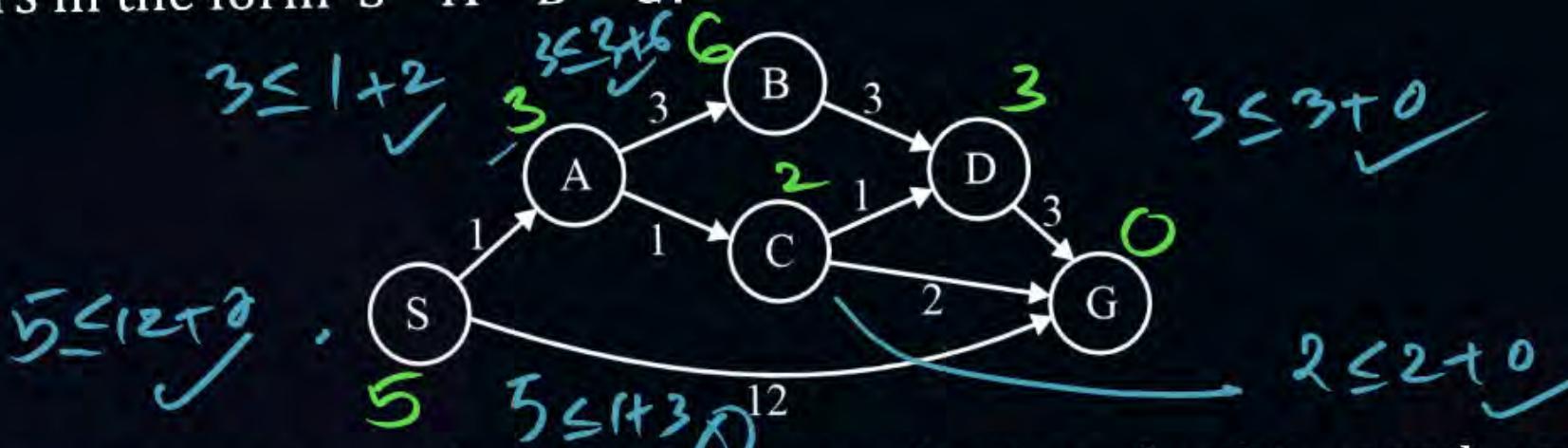
| State | $h_1$ | $h_2$ |
|-------|-------|-------|
| S     | 5     | 4     |
| A     | 3     | 2     |
| B     | 6     | 6     |
| C     | 2     | 1     |
| D     | 3     | 3     |
| G     | 0     | 0     |



- (a) What path would breadth-first graph search return for this search problem?
- (b) What path would uniform cost graph search return for this search problem?
- (c) What path would depth-first graph search return for this search problem?
- (d)  What path would A\* graph search, using a consistent heuristic, return for this search problem?

d)  $h$  → Not admissible

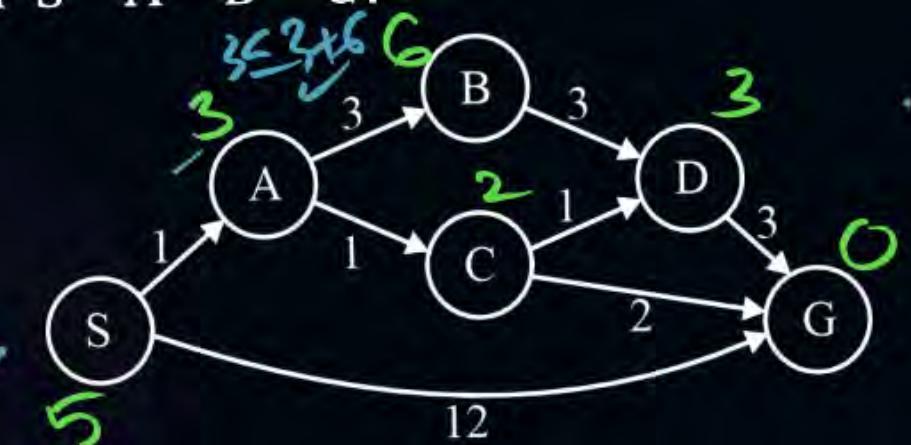
answers in the form  $S \xrightarrow{f} A \xrightarrow{f} B \xrightarrow{f} C$



- (a) What path would breadth-first graph search return for this search problem?
- (b) What path would uniform cost graph search return for this search problem?
- (c) What path would depth-first graph search return for this search problem?
- (d)  What path would A\* graph search, using a consistent heuristic, return for this search problem?

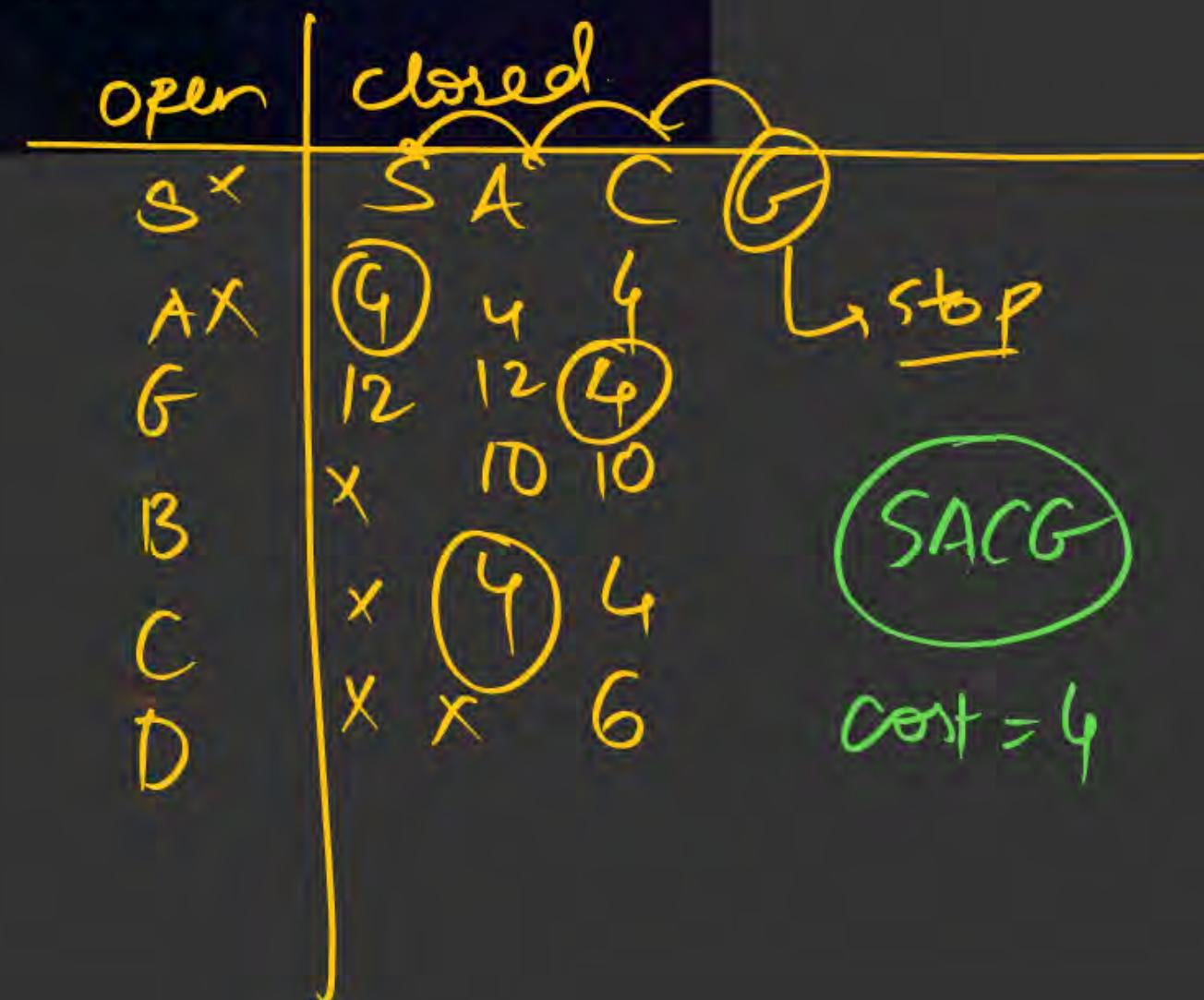
d)  $h$   $\rightarrow$  not consistent

answers in the form S -> A -> ...



- (a) What path would breadth-first graph search return for this search problem?
- (b) What path would uniform cost graph search return for this search problem?
- (c) What path would depth-first graph search return for this search problem?
- (d)  What path would A\* graph search, using a consistent heuristic, return for this search problem?

d) h1  $\rightarrow$  A^\* graph search





## Topic : Artificial Intelligence

#Q.

Consider the heuristics for this problem shown in the table below.

- (i) Is  $h_1$  admissible? Yes No
- (ii) Is  $h_1$  consistent? Yes No
- (iii) Is  $h_2$  admissible? Yes No
- (iv) Is  $h_2$  consistent? Yes No

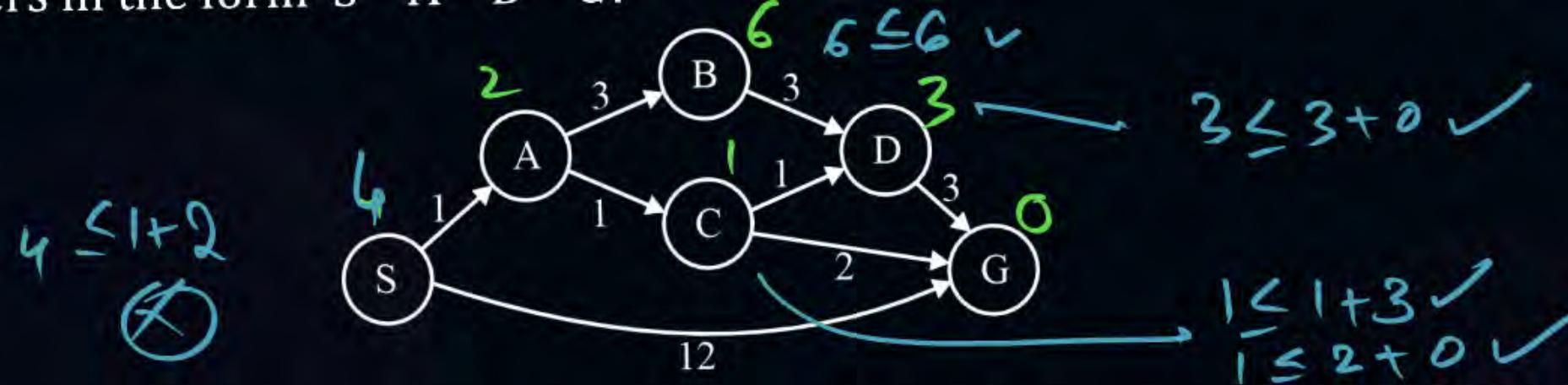
| State | $h_1$ | $h_2$ |
|-------|-------|-------|
| S     | 5     | 4     |
| A     | 3     | 2     |
| B     | 6     | 6     |
| C     | 2     | 1     |
| D     | 3     | 3     |
| G     | 0     | 0     |



- (a) What path would breadth-first graph search return for this search problem?
- (b) What path would uniform cost graph search return for this search problem?
- (c) What path would depth-first graph search return for this search problem?
- (d)  What path would A\* graph search, using a consistent heuristic, return for this search problem?

d)  $h_2$  → Admissible

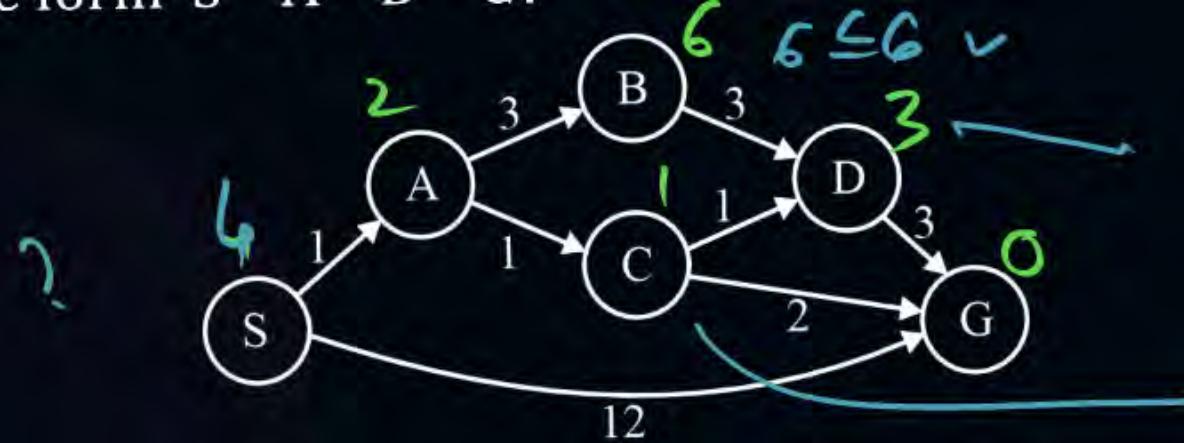
answers in the form S -> A -> ...



- (a) What path would breadth-first graph search return for this search problem?
- (b) What path would uniform cost graph search return for this search problem?
- (c) What path would depth-first graph search return for this search problem?
- (d) ✓ What path would A\* graph search, using a consistent heuristic, return for this search problem?

d)  $h_2$  → Not Consistent

answers in the form S-A-B-C



- (a) What path would breadth-first graph search return for this search problem?
- (b) What path would uniform cost graph search return for this search problem?
- (c) What path would depth-first graph search return for this search problem?
- (d)  What path would A\* graph search, using a consistent heuristic, return for this search problem?

d) h2 →

A\* Graph Search

|    | Open | Closed |  |
|----|------|--------|--|
| S* |      | S      |  |
| A* |      | A      |  |
| G  | 3    |        |  |
| B  | 12   |        |  |
| C* |      | C      |  |
| D  |      | G      |  |

stop  
SACGG  
Cost = 4



## Topic : Artificial Intelligence

#Q. Which of the following algorithms is/are guaranteed to give an optimal solution?

- A Greedy Best First Search GBFS X
- B A\* with zero heuristic  $F = h + g, h = 0 \rightarrow F = g$  UCS
- C A\* with consistent heuristic Consistent  $\rightarrow$  admissible good
- D Depth First Search X



## Topic : Artificial Intelligence

#Q. Given a search tree where A\* search is applied, the branching factor is 3, the depth of the optimal solution is 4, and the heuristic is perfect (i.e., it always predicts the exact remaining cost).

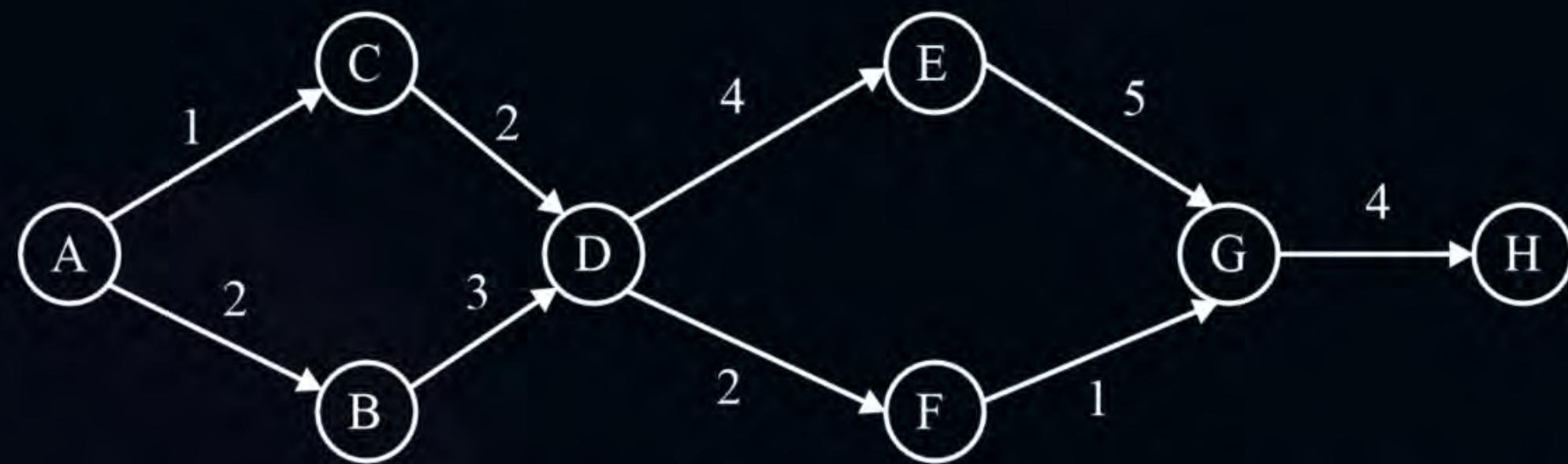
How many nodes are expanded by A\* in this tree?





## Topic : Artificial Intelligence

#Q. Consider the following directed graph.



The heuristic function for the nodes is defined as  $h(A) = 15$ ,  $h(B) = 10$ ,  $h(C) = 12$ ,  $h(D) = 7$ ,  $h(E) = 10$ ,  $h(F) = 6$ ,  $h(G) = 4$ ,  $h(H) = 0$ . The start node is A and the goal node is H. Assume that ties in selecting node for expansion from the fringe are resolved by choosing the alphabetically smaller node.

Which of the following statements are correct?



**THANK - YOU**



# DS & AI ENGINEERING



## Artificial Intelligence

### Informed search

Lecture No.- 05

By- Aditya sir



# Recap of Previous Lecture



Topic

Topic

Topic

Topic

Questions

# Topics to be Covered

P  
W



Topic

Topic

Topic

Questions

IDA\*



## About Aditya Jain sir

1. Appeared for GATE during BTech and secured AIR 60 in GATE in very first attempt - City topper
2. Represented college as the first Google DSC Ambassador.
3. The only student from the batch to secure an internship at Amazon. (9+ CGPA)
4. Had offer from IIT Bombay and IISc Bangalore to join the Masters program
5. Joined IIT Bombay for my 2 year Masters program, specialization in Data Science
6. Published multiple research papers in well known conferences along with the team
7. Received the prestigious excellence in Research award from IIT Bombay for my Masters thesis in ML
8. Completed my Masters with an overall GPA of 9.36/10
9. Joined Dream11 as a Data Scientist
10. Have mentored 15,000+ students & working professionals in field of Data Science and Analytics
11. Have been mentoring & teaching GATE aspirants to secure a great rank in limited time
12. Have got around 27.5K followers on Linkedin where I share my insights and guide students and professionals.





**Telegram Link for Aditya Jain sir:**  
**[https://t.me/AdityaSir\\_PW](https://t.me/AdityaSir_PW)**



## Topic : Artificial Intelligence

$$n \rightarrow n+1 \\ n \rightarrow n+2$$

33%

#Q. Consider a state space where the start state is number 1. The successor function for the state numbered a returns two states numbered  $n+1$  and  $n+2$ . Assume that the states in the unexpanded state list are expanded in the ascending order of numbers and the previously expanded states are not added to the unexpanded state list.

Which ONE of the following statements about breadth-first search (BFS) and depth-first search (DFS) is true, when reaching the goal state number 6?

A

BFS expands more states than DFS. X

B

DFS expands more states than BFS. X

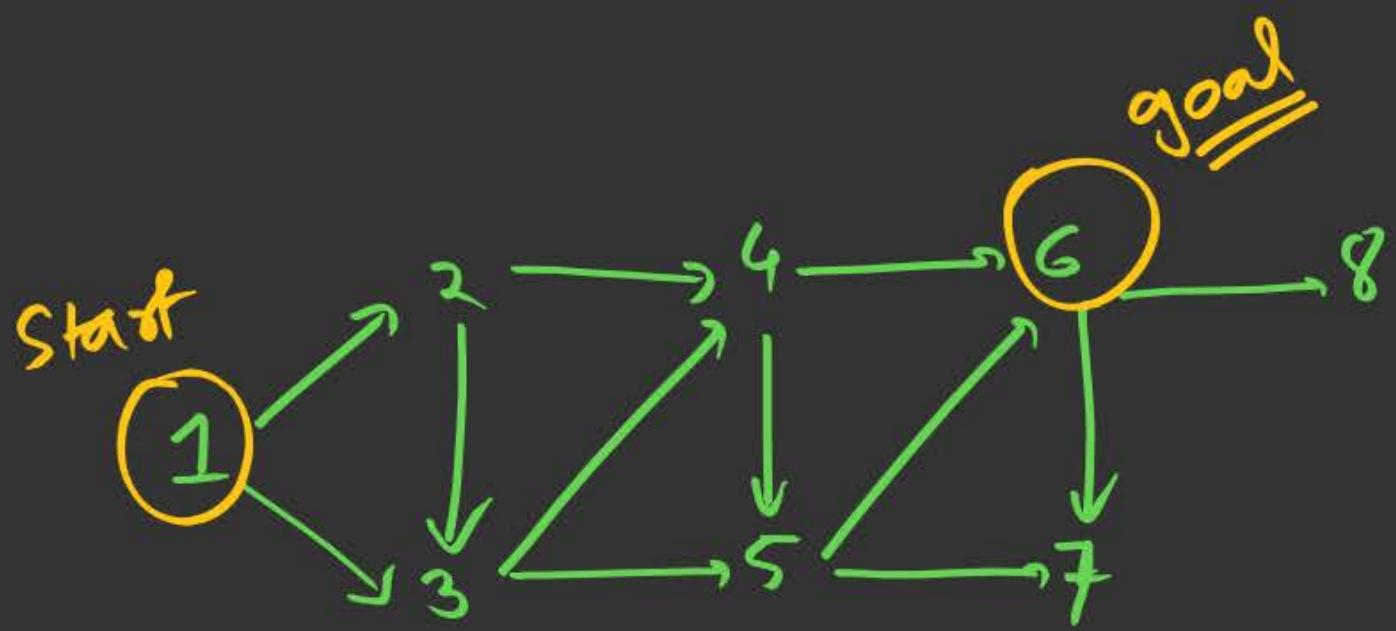
C

Both BFS and DFS expand equal number of states. ✓

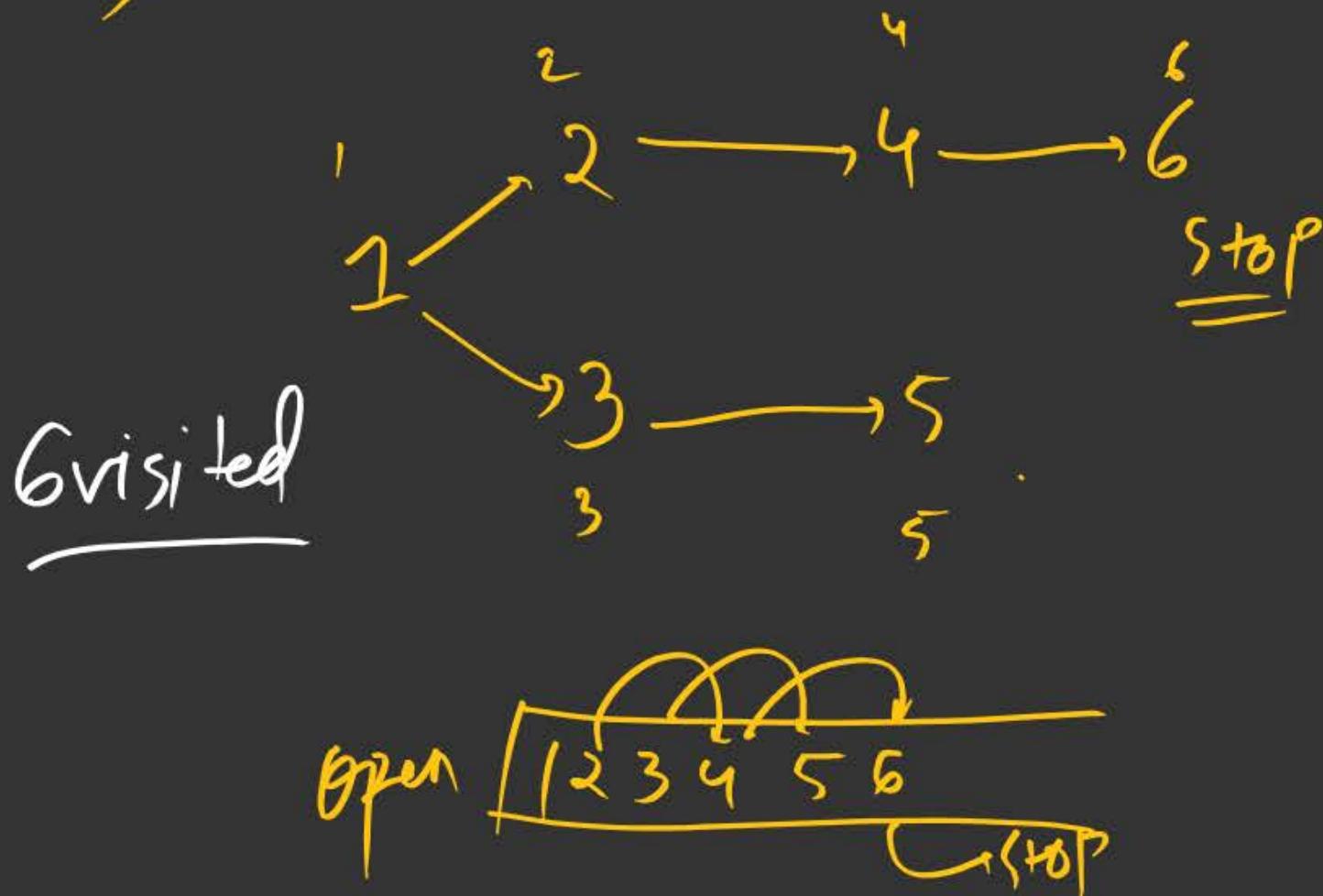
D

Both BFS and DFS do not reach the goal state number 6. X

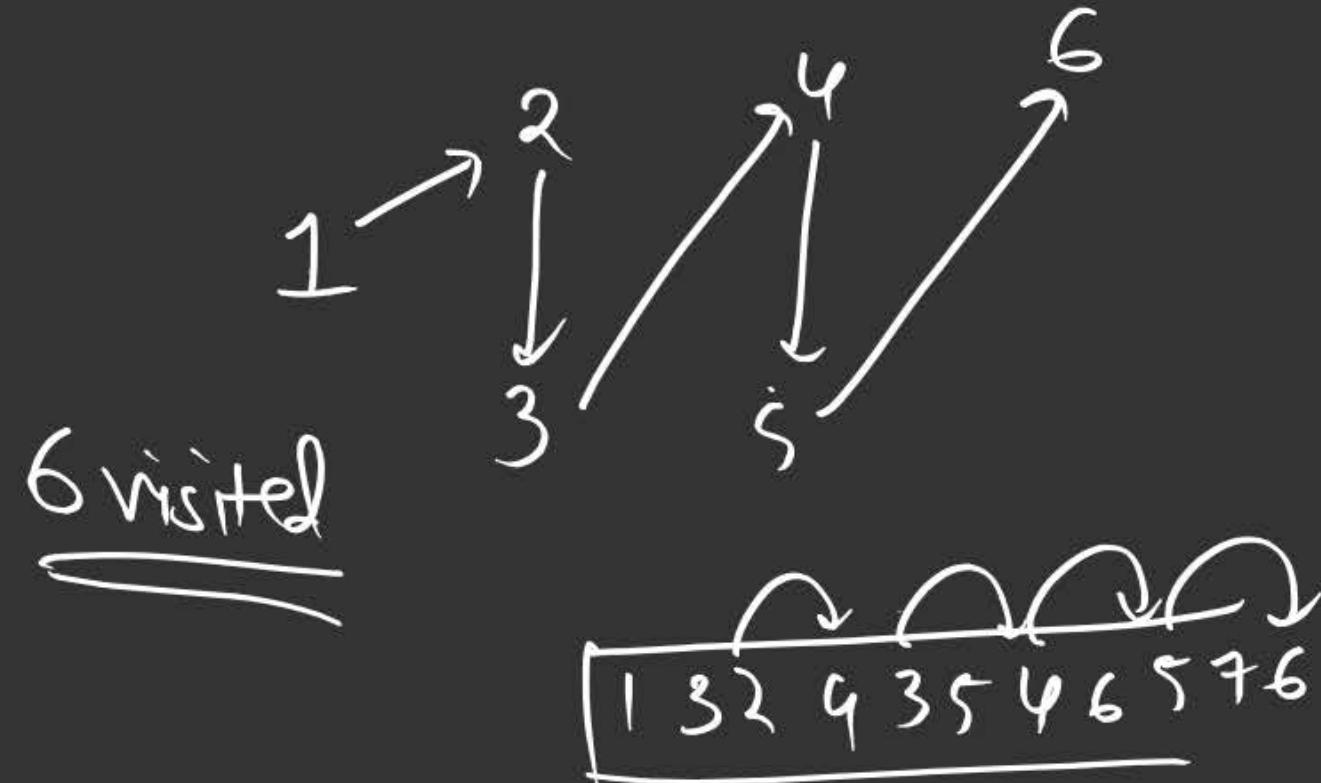
expand → visited



1) BFS



2) DFS





# Topic : Artificial Intelligence

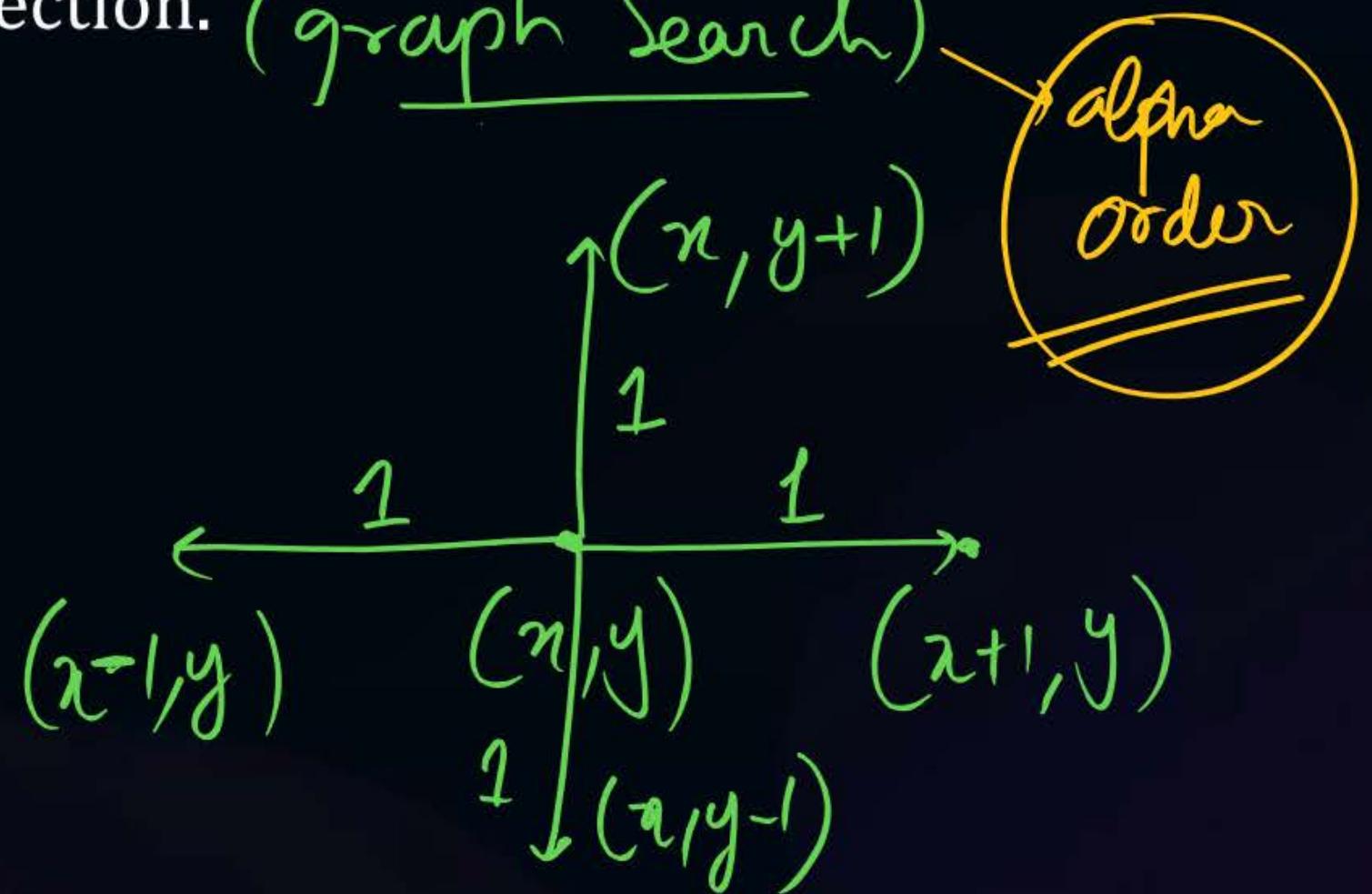
Adv

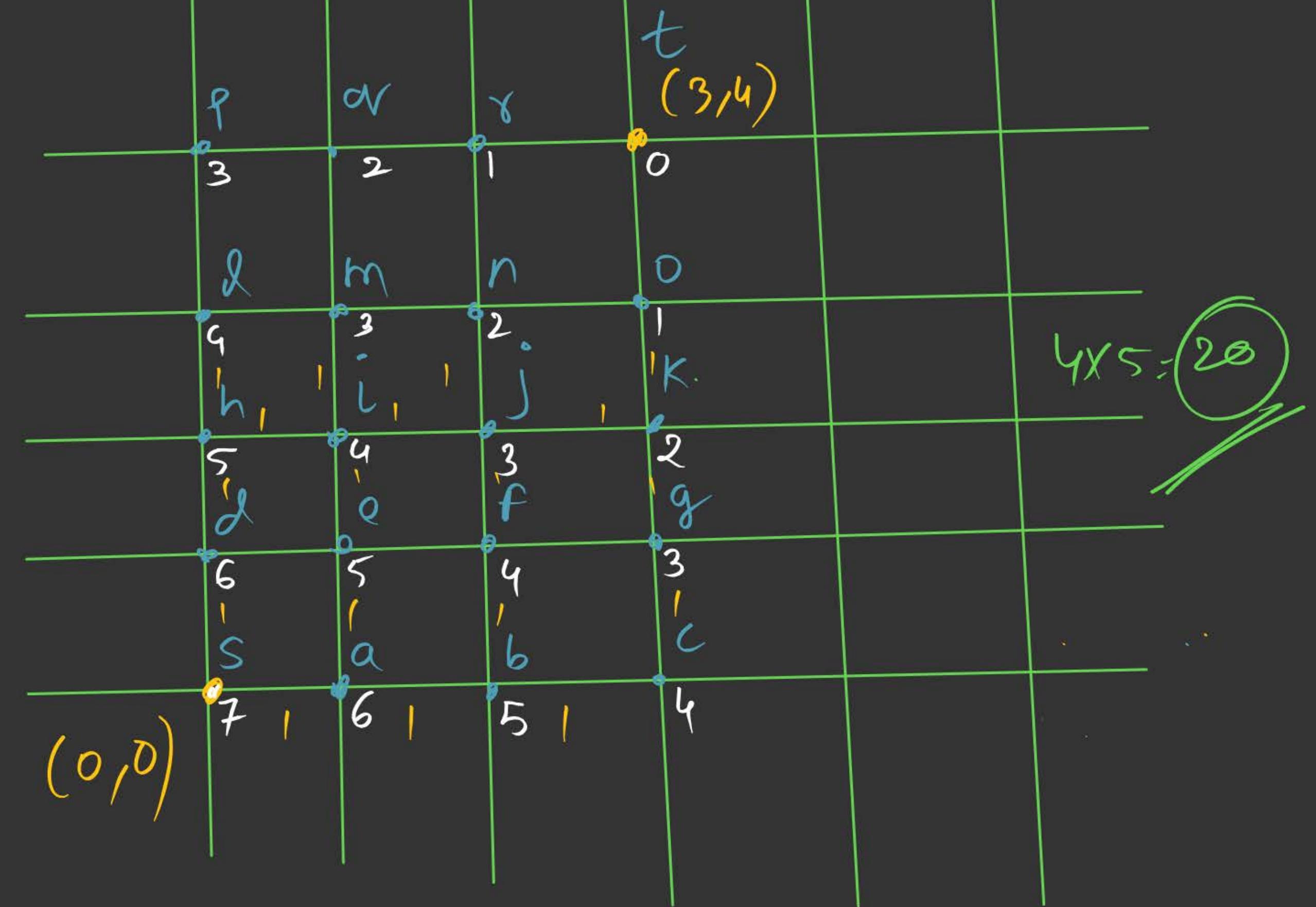
P  
W

#Q. Consider an infinite search space  $Z \times Z$ . The start state is at  $(0, 0)$  and the goal state is at  $(g_x, g_y)$ . Given that the agent can move from a state  $(x, y)$  to either of  $\{(x+1, y), (x-1, y), (x, y+1), (x, y-1)\}$  with a unit step cost, find the number of nodes explored using A\* tree search with Manhattan distance heuristic for  $(g_x, g_y) = (3, 4)$ . Assume no duplicate detection.

graph Search

$$\left( |x_1 - x_2| + |y_1 - y_2| \right)$$





## A\* Graph Search

|   | s* | a | b | c | d | ... | ... |
|---|----|---|---|---|---|-----|-----|
| s | ?  | ? | ? | ? | ? | ... | ... |
| a | ?  | ? | ? | ? | ? | ... | ... |
| d | ?  | ? | ? | ? | ? | ... | ... |
| b | x  | ? | ? | ? | ? | ... | ... |
| e | x  | ? | ? | ? | ? | ... | ... |
| c | x  | x | ? | ? | ? | ... | ... |
| F | x  | x | ? | ? | ? | ... | ... |
| g | x  | x | x | ? | ? | ... | ... |

All the nodes will be visited



## Topic : Artificial Intelligence

#Q.

We define an evaluation function for a heuristic search problem as:

$$f(n) = (w * g(n)) + ((1 - w) * h(n))$$

min

51 %

where  $g(n)$  is the cost of the best path found from the start state to state  $n$ ,  $h(n)$  is an admissible heuristic function that estimates the cost of a path from  $n$  to a goal state, and  $0.0 \leq w \leq 1.0$ .

What search algorithm do you get when  $w = 0$ ?

A

Breadth-First search



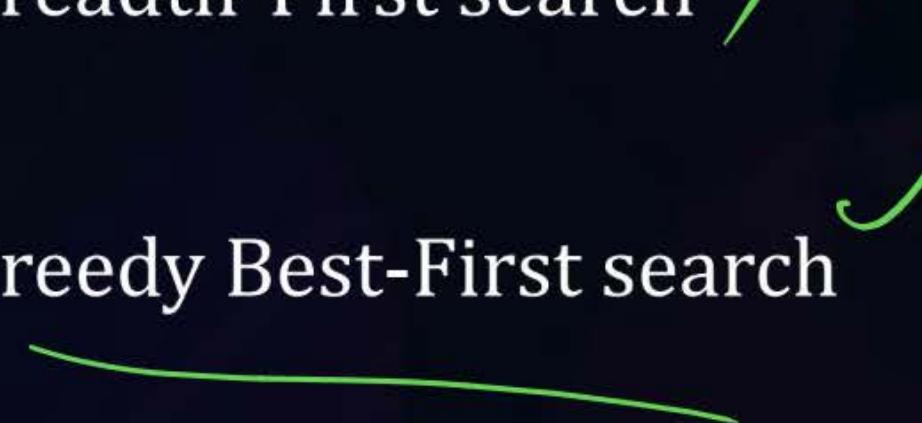
B

Uniform-Cost search



C

Greedy Best-First search



D

Algorithm A\* Search



$$\begin{aligned} w &= 1 & f &= g + 0 \cdot h \\ f &= g & \text{UCS} \end{aligned}$$





## Topic : Artificial Intelligence



#Q. We define an evaluation function for a heuristic search problem as:

$$f(n) = (w * g(n)) + ((1 - w) * h(n))$$

where  $g(n)$  is the cost of the best path found from the start state to state  $n$ ,  $h(n)$  is an admissible heuristic function that estimates the cost of a path from  $n$  to a goal state, and  $0.0 \leq w \leq 1.0$ .

What search algorithm do you get when  $w = 1.0$ ?

**A**

Breadth-First search

**B**

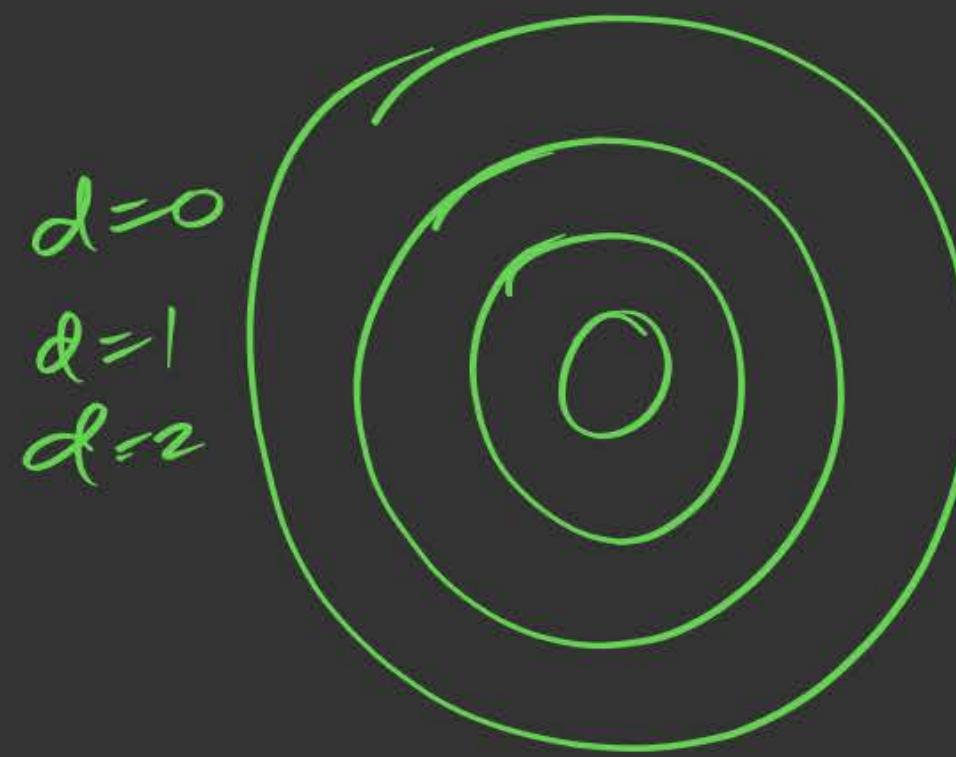
Uniform-Cost search

**C**

Greedy Best-First search

**D**

Algorithm A\* Search





# Topic : Artificial Intelligence

## Iterative Deepening A\* algorithm (IDA\*) - Artificial intelligence

- The Iterative Deepening A\* (IDA\*) algorithm is an extension of the A\* search algorithm designed to use less memory, making it more suitable for large search spaces where A\* would require too much memory to store all nodes in the open and closed lists.
- IDA\* combines the space efficiency of Iterative Deepening Depth-First Search (IDDFS) with the optimality of A\*. Instead of maintaining a priority queue like A\*, IDA\* performs a series of depth-limited searches, where the depth limit is determined by a threshold that is increased iteratively. Each iteration explores paths that do not exceed the current threshold, and the threshold is based on the f-cost (sum of g-cost and h-cost).





# Topic : Artificial Intelligence

Algo/Idea

## Iterative Deepening A\* algorithm (IDA\*) - Artificial intelligence

- **Step 1:** Initialization  
Set the root node as the current node, and find the f-score.
- **Sep 2:** Set threshold  
Set the cost limit as a threshold for a node i.e the maximum f-score allowed for that node for further explorations.
- **Step 3:** Node Expansion  
Expand the current node to its children and find f-scores.
- **Step 4: Pruning**  
If for any node the f-score > threshold, prune that node because it's considered too expensive for that node, and store it in the visited node list.
- **Step 5: Return Path**  
If the Goal node is found then return the path from the start node Goal node.



## Topic : Artificial Intelligence

Iterative Deepening A\* algorithm (IDA\*) - Artificial intelligence

- **Step 6:** Update the Threshold

If the Goal node is not found then repeat from step 2 by changing the threshold with the minimum pruned value from the visited node list. And Continue it until you reach the goal node.



# Topic : Artificial Intelligence



It 1:  $f_{th} = 2$ , AB pruned node

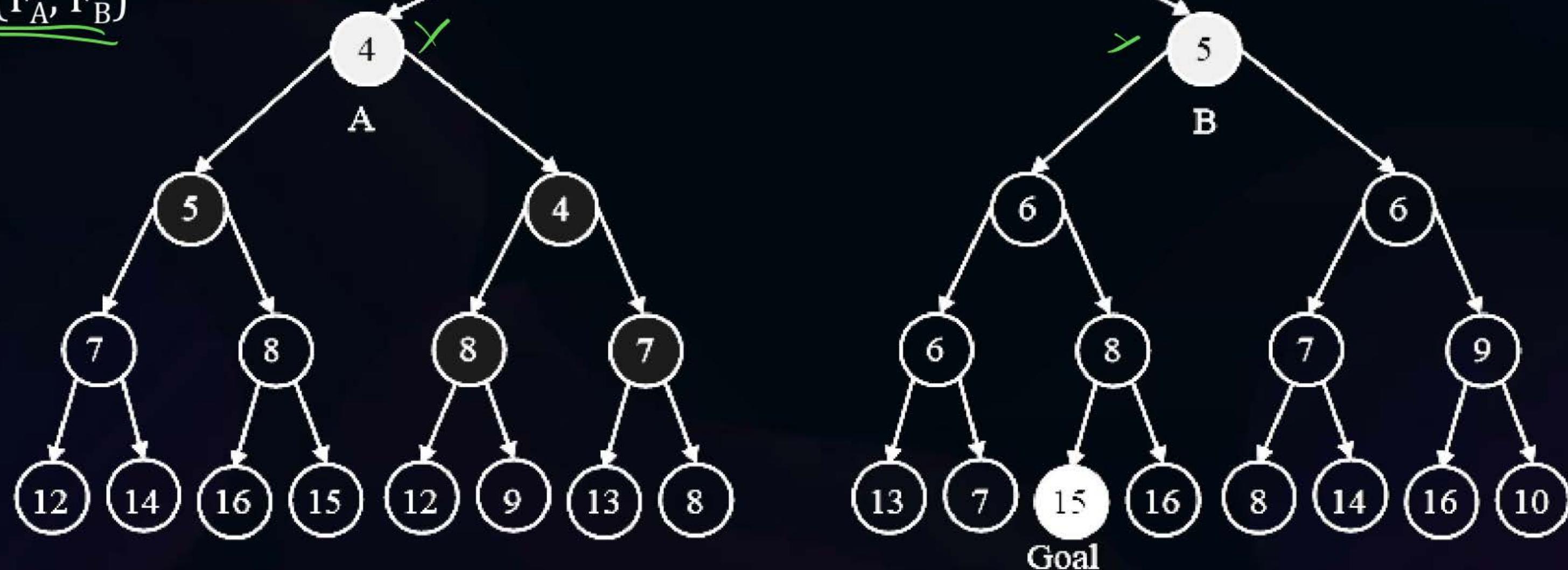
Closed list S

Prune list A,B

- $F_{th}$  for iteration 2

$\Rightarrow \min(F_A, F_B)$

$\Rightarrow 4$





# Topic : Artificial Intelligence



It 1:  $f_{th} = 2$ , AB pruned node

It 2:  $f_{th} = 4$

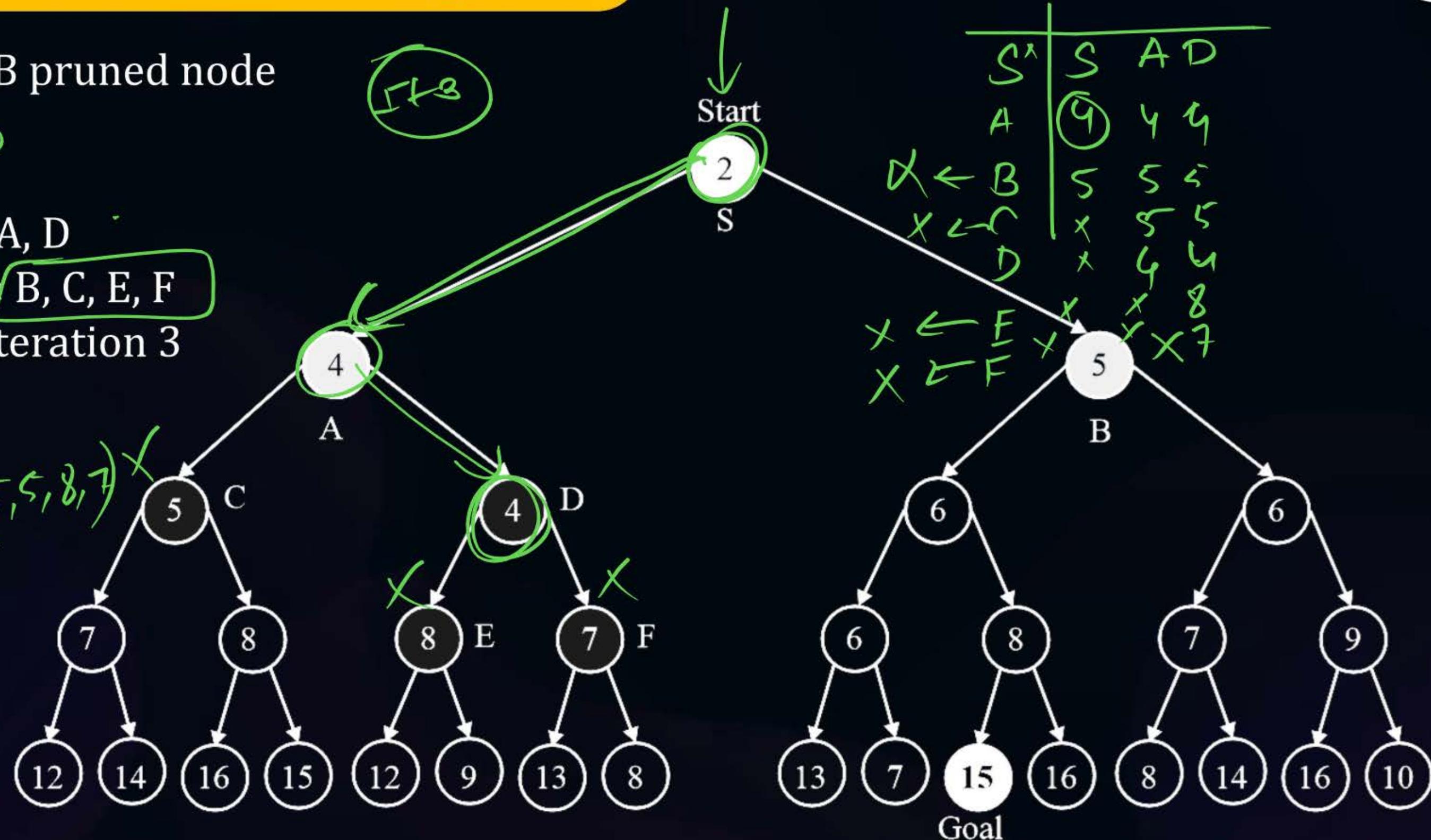
Closed list S

~~Prune~~ list S, A, D

Pruned node B, C, E, F

New  $F_{th}$  for iteration 3  
 $\Rightarrow 3$

$$f_{th} = \min(5, 5, 8, 7) \\ = 5$$





# Topic : Artificial Intelligence



It: 3

Fth = 5

Close list: S,A,B,C, D

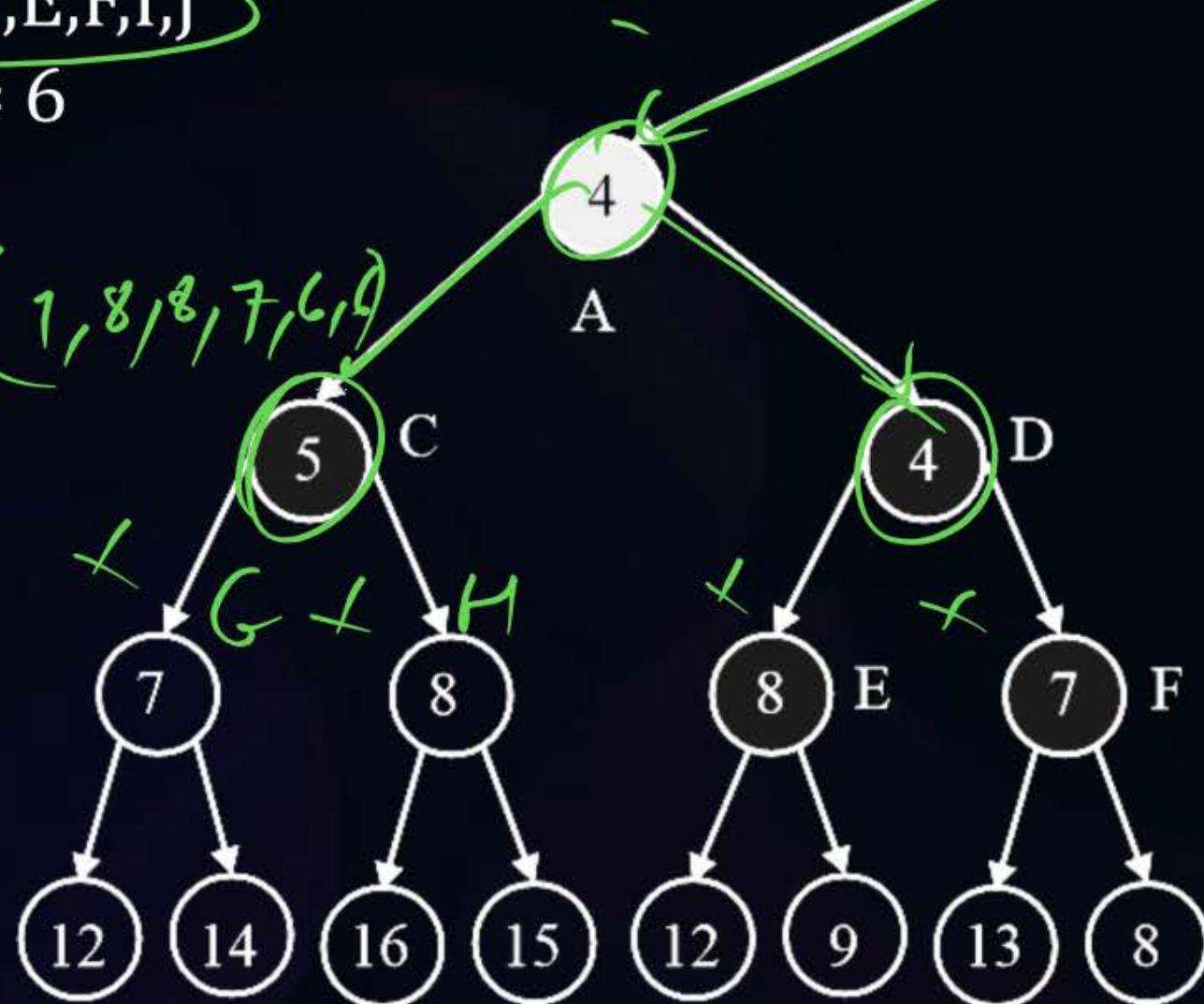
Prune G,H,E,F,I,J

It 4: Fth = 6

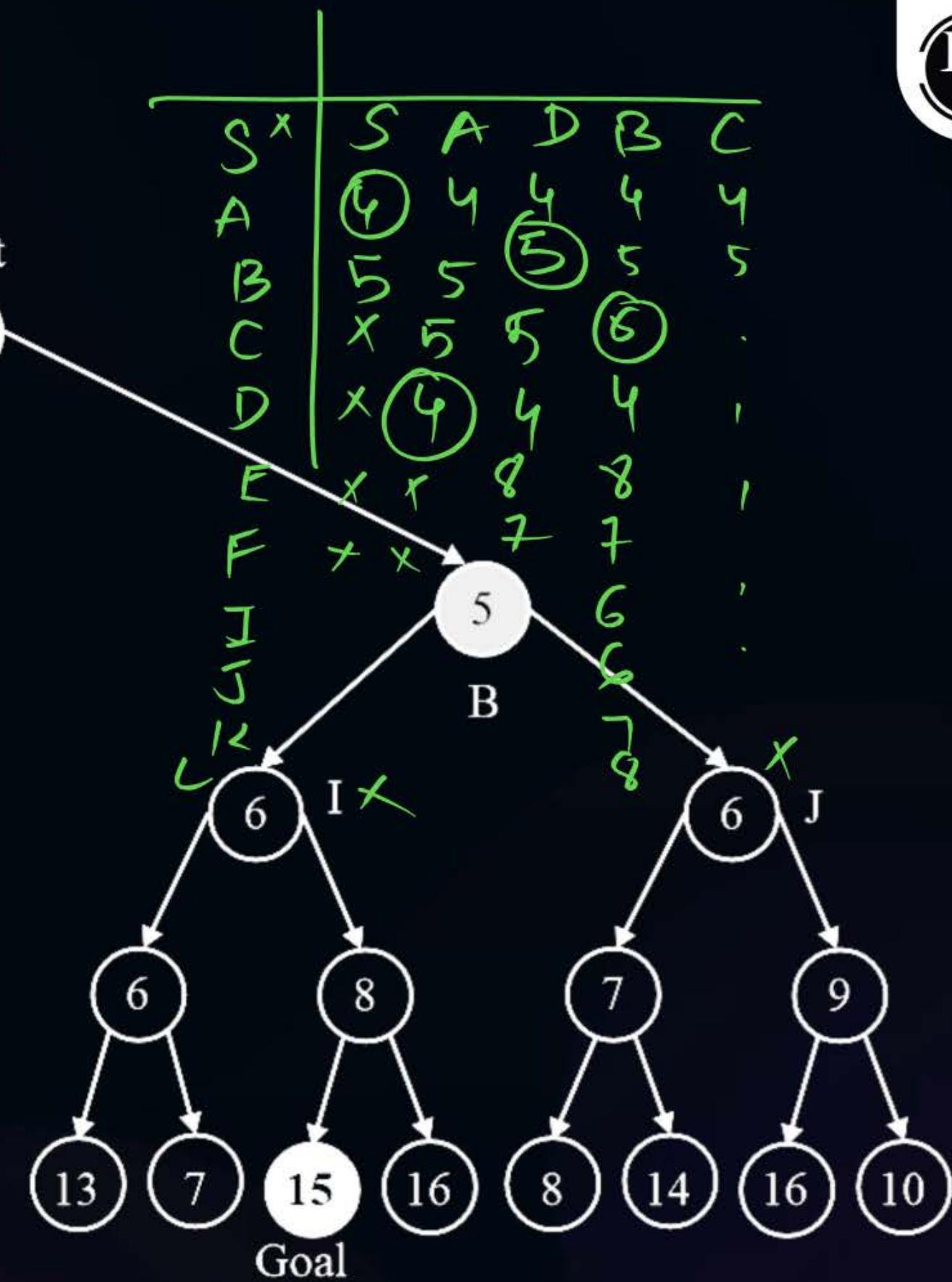
$$f_{th} = \min(1, 8, 8, 7, 6, 9)$$

$$= 6$$

HV



ss✓





# Topic : Artificial Intelligence



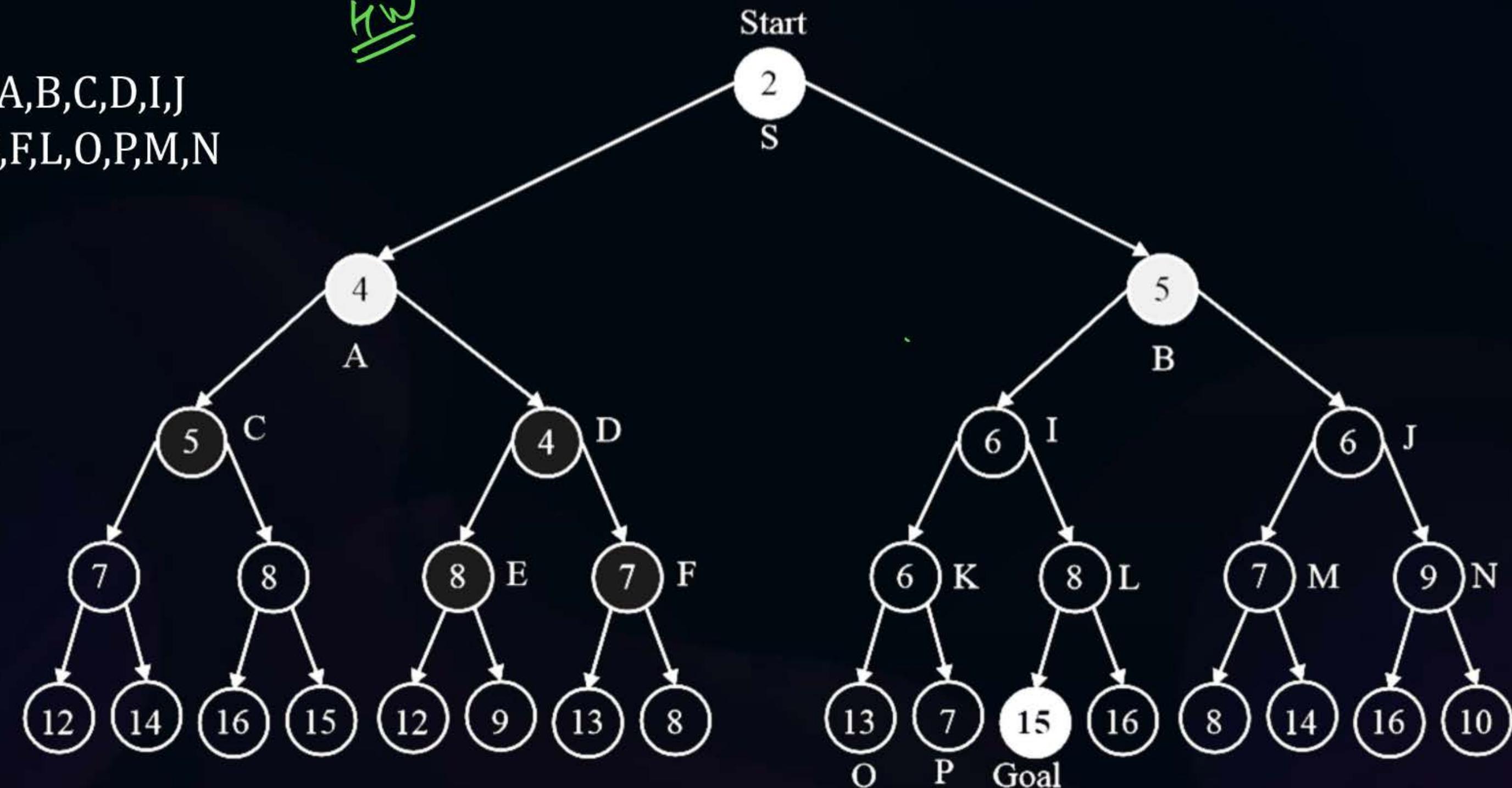
It: 4

$F_{th} = 6$

Close list: S,A,B,C,D,I,J

Prune G,H,E,F,L,O,P,M,N

$F_{th} = 7$





# Topic : Artificial Intelligence



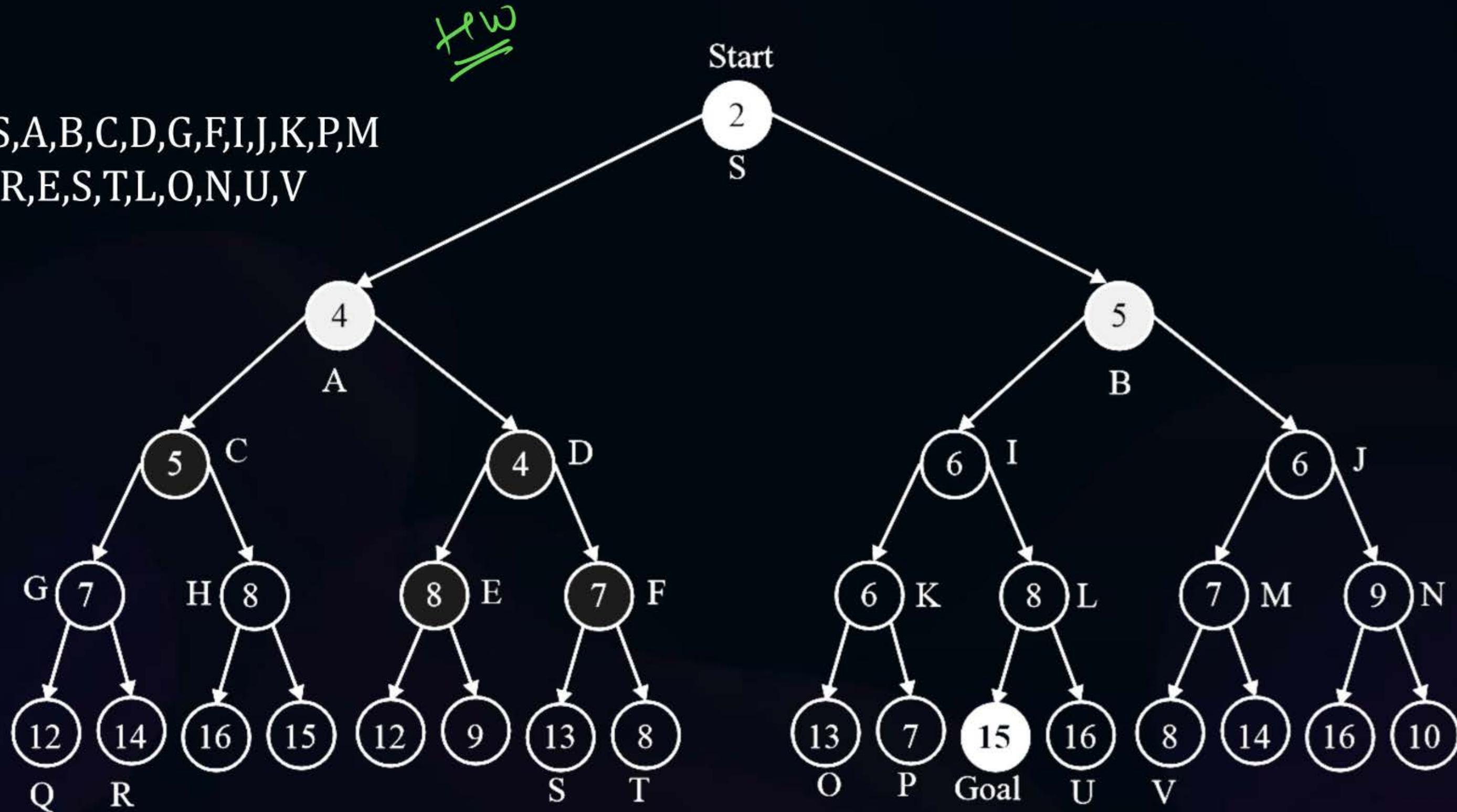
It: 5

$F_{th} = 7$

Close list: S,A,B,C,D,G,F,I,J,K,P,M

Prune H,Q,R,E,S,T,L,O,N,U,V

$F_{th} = 8$





# Topic : Artificial Intelligence

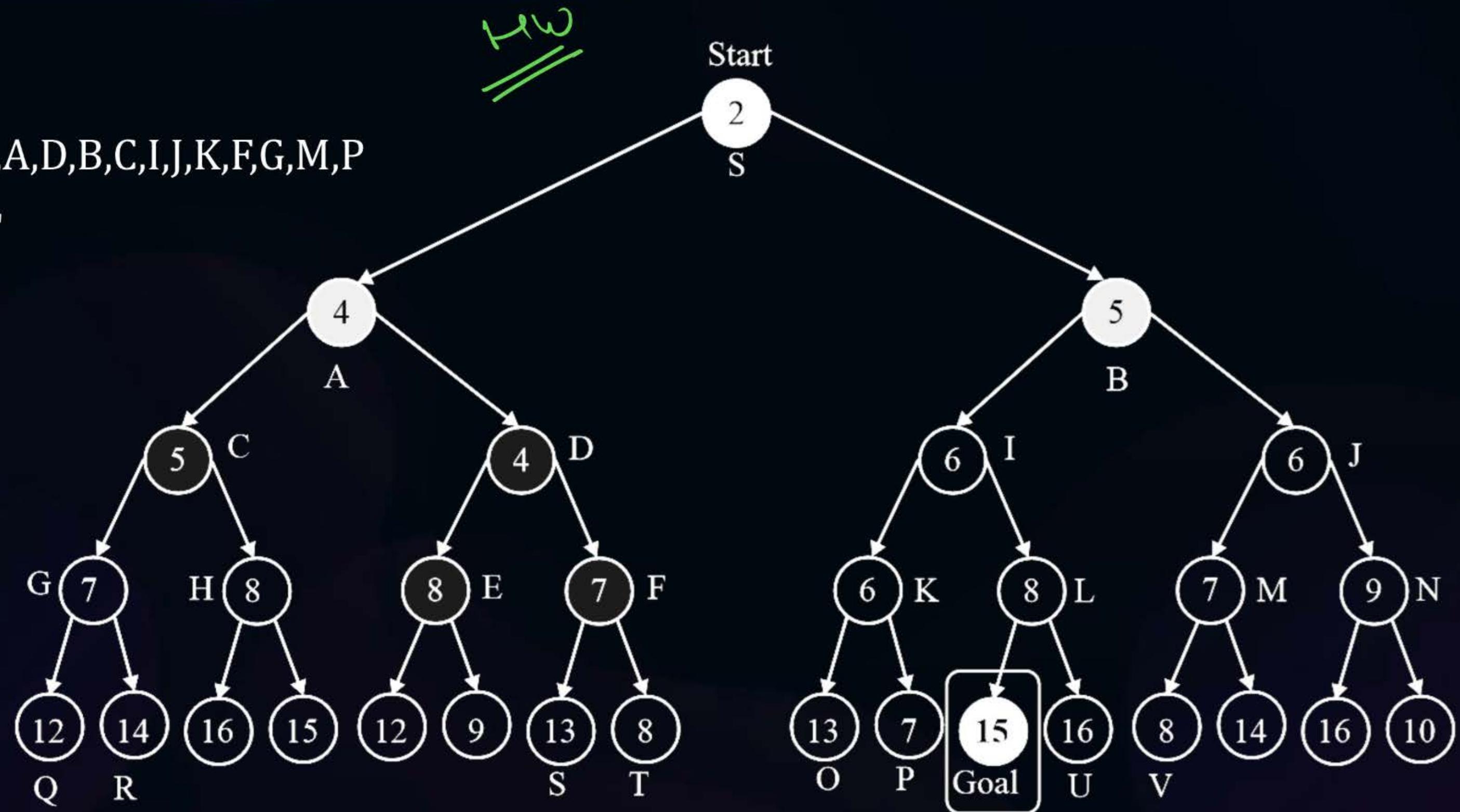


It: 6

$F_{th} = 8$

Close list: S,A,D,B,C,I,J,K,F,G,M,P

Prune E,H,L



## IDA<sup>\*</sup> Search

Algo is moving like DFS

But we are applying A<sup>\*</sup> Search in each iteration

| <u>A*</u>            | <u>IDA*</u>           |
|----------------------|-----------------------|
| 1) SC: $O(b^d)$      | 1) SC: $O(b \cdot d)$ |
| 2) TC: $O(b^d)$      | 2) TC: $O(b^d)$       |
| 3) Optimal           | 3) Optimal            |
| 4) Complete          | 4) Complete           |
| 5) Priority Queue    | 5) No priority Queue  |
| 6) BFS version of A* | 6) DFS version of A*  |

# Topic : Artificial Intelligence

IDA\*

⇒ Iterative deepening A\*

A\* ⇒ Space  $O(b^d)$

Time  $O(b^d)$

IDA\* ⇒ Space  ~~$O(b^{d+1})$~~

Time  $O(b^d)$

⇒ Iterative A\*

⇒ Iterative Run on the f core

IDDFS

Depth = 0,1,2,3....

IDFS

Space comp.  $O(b \cdot d)$

Time comp.  $O(b^d)$



**THANK - YOU**



# DS & AI ENGINEERING

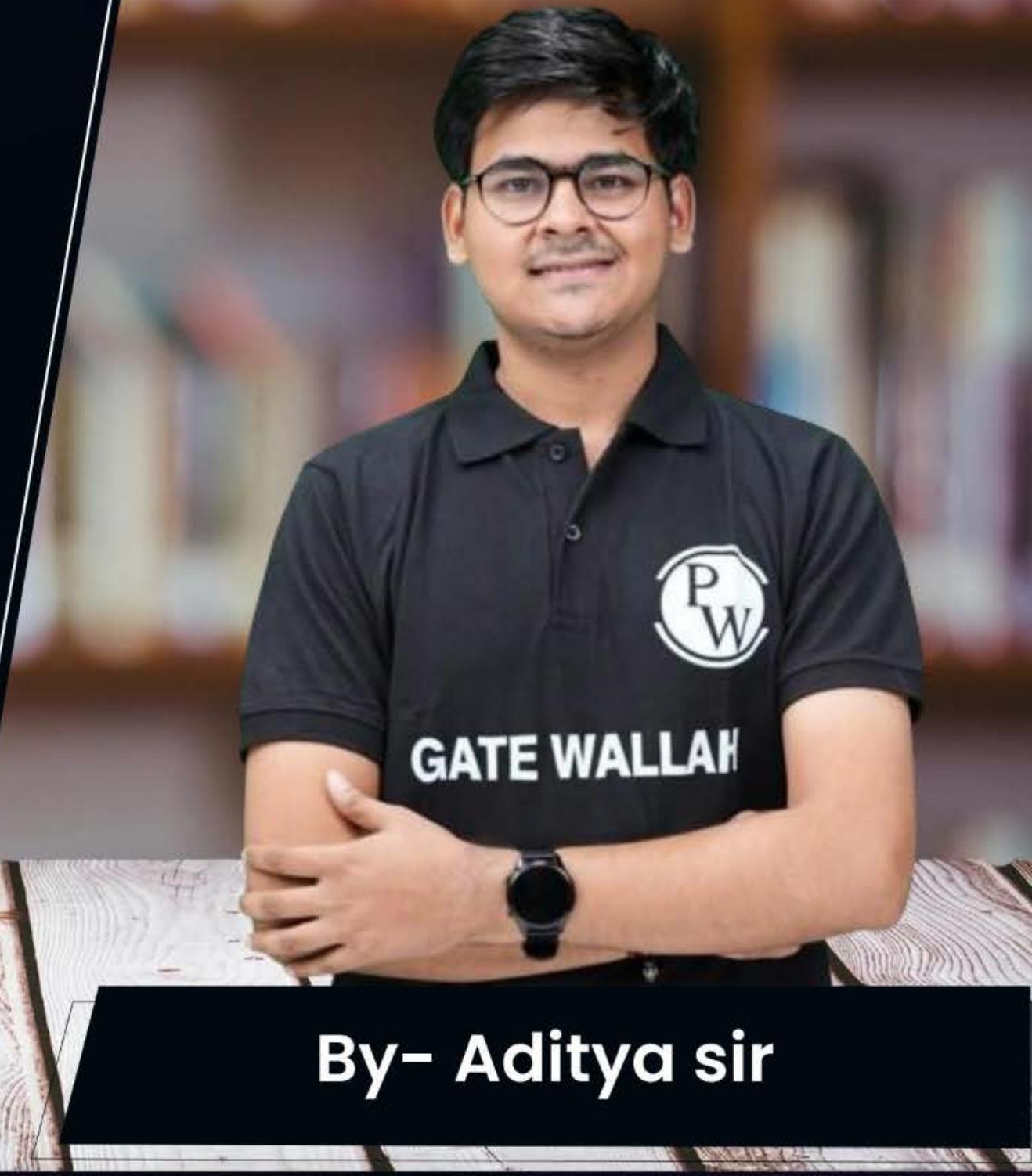


## Artificial Intelligence

### Informed search

Lecture No.- 6

By- Aditya sir



# Recap of Previous Lecture



Topic

IDA\*

Topic

Questions

Topic

Topic

# Topics to be Covered



Topic

Topic

Topic

Practice  
Concepts





## About Aditya Jain sir

1. Appeared for GATE during BTech and secured AIR 60 in GATE in very first attempt - City topper
2. Represented college as the first Google DSC Ambassador.
3. The only student from the batch to secure an internship at Amazon. (9+ CGPA)
4. Had offer from IIT Bombay and IISc Bangalore to join the Masters program
5. Joined IIT Bombay for my 2 year Masters program, specialization in Data Science
6. Published multiple research papers in well known conferences along with the team
7. Received the prestigious excellence in Research award from IIT Bombay for my Masters thesis in ML
8. Completed my Masters with an overall GPA of 9.36/10
9. Joined Dream11 as a Data Scientist
10. Have mentored 15,000+ students & working professionals in field of Data Science and Analytics
11. Have been mentoring & teaching GATE aspirants to secure a great rank in limited time
12. Have got around 27.5K followers on Linkedin where I share my insights and guide students and professionals.





Telegram Link for Aditya Jain sir:  
[https://t.me/AdityaSir\\_PW](https://t.me/AdityaSir_PW)

#Q. The heuristic path algorithm is a best-first search in which  $f(n) = (2-w)g(n) + w \cdot h(n)$ .

Select the correct statement(s).

$$f = (2 \cdot w)g + w \cdot h$$

- A For  $w = 1$ ,  $f(n)$  represents the  $A^*$  algorithm.  $\rightarrow f = g + h$
- B For  $w = 2$ ,  $f(n)$  is complete.  $\rightarrow f = 2 \cdot h \times$  | 1) UCS:  $f = g$
- C For  $w = 2$ ,  $f(n)$  is optimal.  $\rightarrow f = 2 \cdot h \times$  | 2) GBFS  $\rightarrow f = h$
- D For  $w = 0$ ,  $f(n)$  represents UCS.  $\rightarrow f = 2 \cdot g \times$  | 3)  $A^* \rightarrow f = g + h$

$$\begin{array}{r} a+a=2a \\ \hline a+a=10a \end{array}$$

?X

$$\begin{array}{r} f=g \\ f=2g \end{array}$$

f  $\propto$  g

#Q. Consider the below graph with heuristic values.

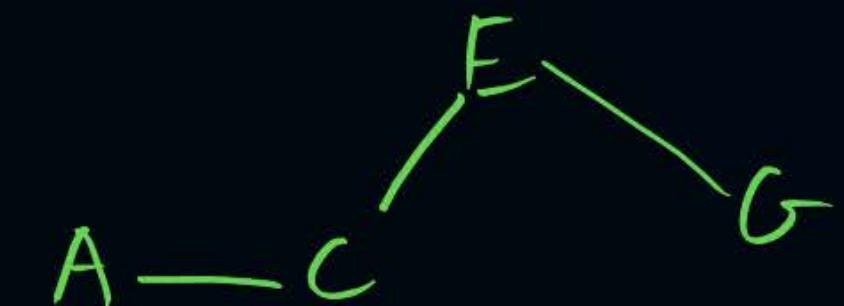
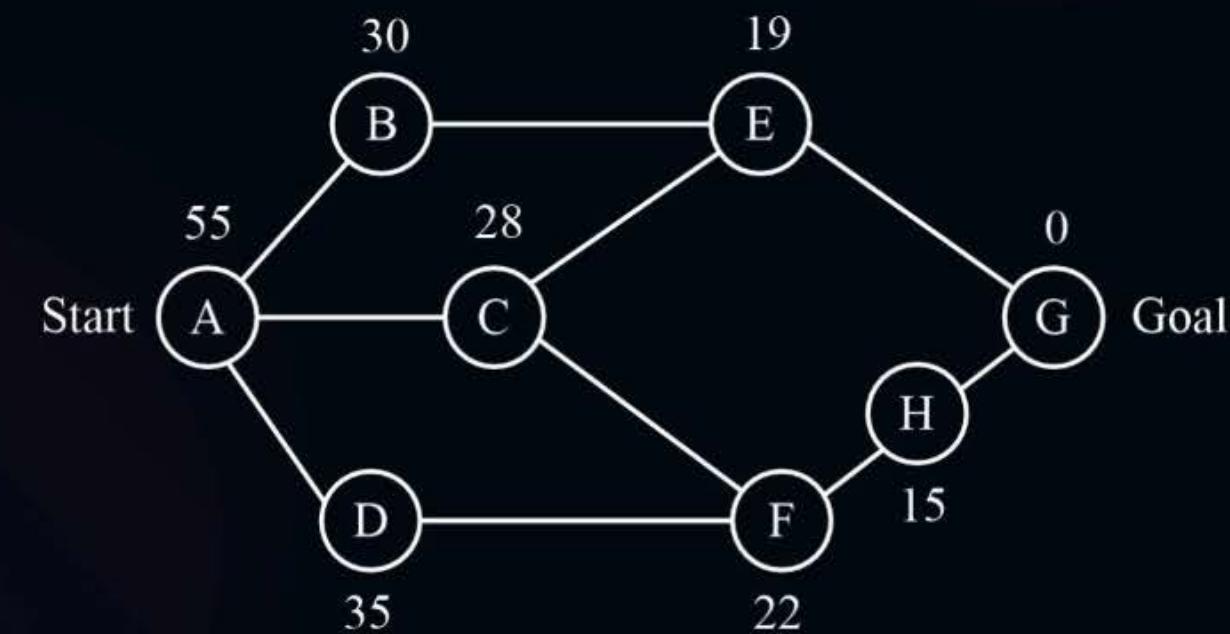
| State | Heuristic: $h(n)$ |
|-------|-------------------|
| A     | 366               |
| B     | 374               |
| C     | 329               |
| D     | 244               |
| E     | 253               |
| F     | 178               |
| G     | 193               |
| H     | 98                |
|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

- A) 178 + 211  
 B) 253 + 99 + 211  
~~C) 178~~  
 D) 253  
 GBFS  $\Rightarrow f = h$



What will be the value of evaluation function for the node F, if we apply Greedy Search algorithm?

#Q. Consider the following graph use the best first search algorithm to find the path from node A to node G (Assume node A is start node and node G is goal node). Which of the following path sequence is correct?



**B** A → C → E → G

**A**

A → D → C → G

**D**

A → D → F → G

**C**

A → C → F → H → G

#Q. Which of the following is/are true about A\* Search?

- A A\* search is both complete and optimal under certain conditions for the heuristic function  $h(n)$
- B A\* search is complete but not optimal under certain conditions for the heuristic function  $h(n)$
- C A search is similar to uniform-cost search, it uses the sum of  $g$  and  $h$  values instead of  $g$
- D A search is similar to uniform-cost search, as it uses the cumulative cost of node ' $n$ ' from a start node

A, B, C

#Q. In Greedy Best First Search, what is the evaluation function used to select the next node for expansion?

$$f = h$$

85%

- A  $\times$   $f(n) = g(n) + h(n)$   $\rightarrow A^*$
- B  $\times$   $f(n) = g(n) - h(n)$   $\times$
- C  $\checkmark$   $f(n) = h(n)$
- D  $\times$   $f(n) = g(n)$   $\rightarrow$  OCS

#Q. Consider the graph given below:

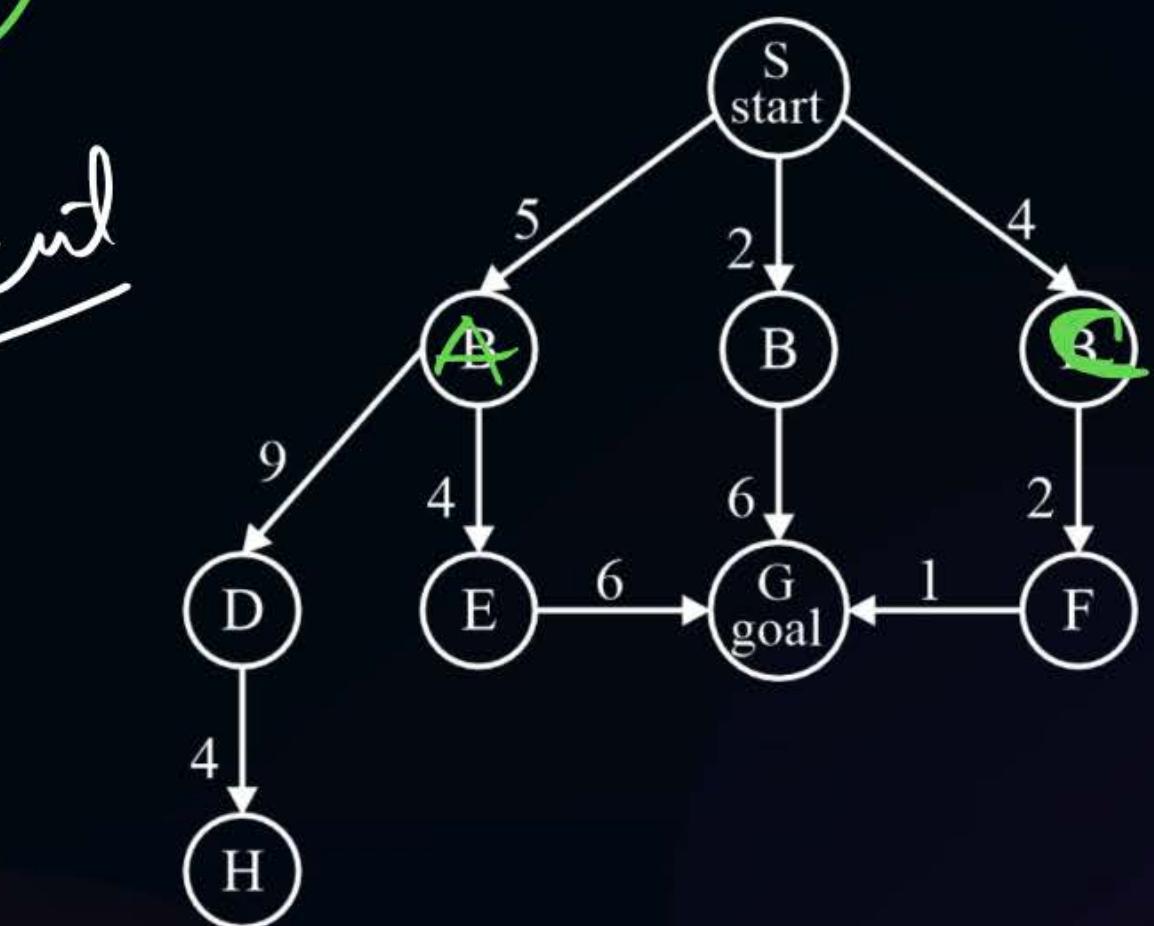
UCS

50%

If we apply Uniform Cost Search Algorithm, then what will be the path to reach node G from node S?

- A S → C → F → G  $\Rightarrow 4 + 2 + 1 = 7$
- B S → B → G  $\Rightarrow 2 + 6 = 8$
- C S → A → E → G  $\Rightarrow 5 + 4 + 6 = 15$
- D S → A → B → C → G  $\Rightarrow \times$

*shortest*



#Q. Consider the graph given below:

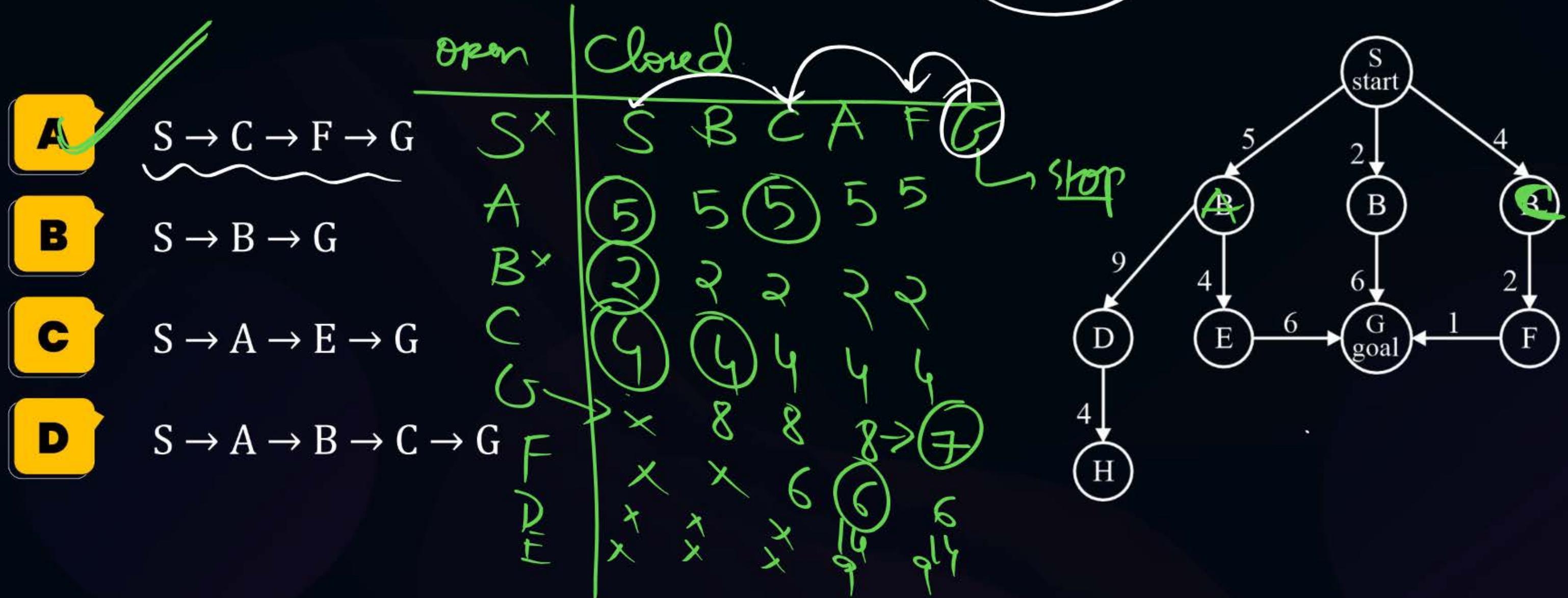
UCS

50%

If we apply Uniform Cost Search Algorithm, then what will be the path to reach node G from node S?

SCFG

S → G



#Q. In the problem the start state is S, and the goal state is G. The transition costs are next to the edges, and the heuristic estimate,  $h$ , of the distance from the state to the goal is in the state's node. Assume ties are always broken by choosing the state which comes first alphabetically.

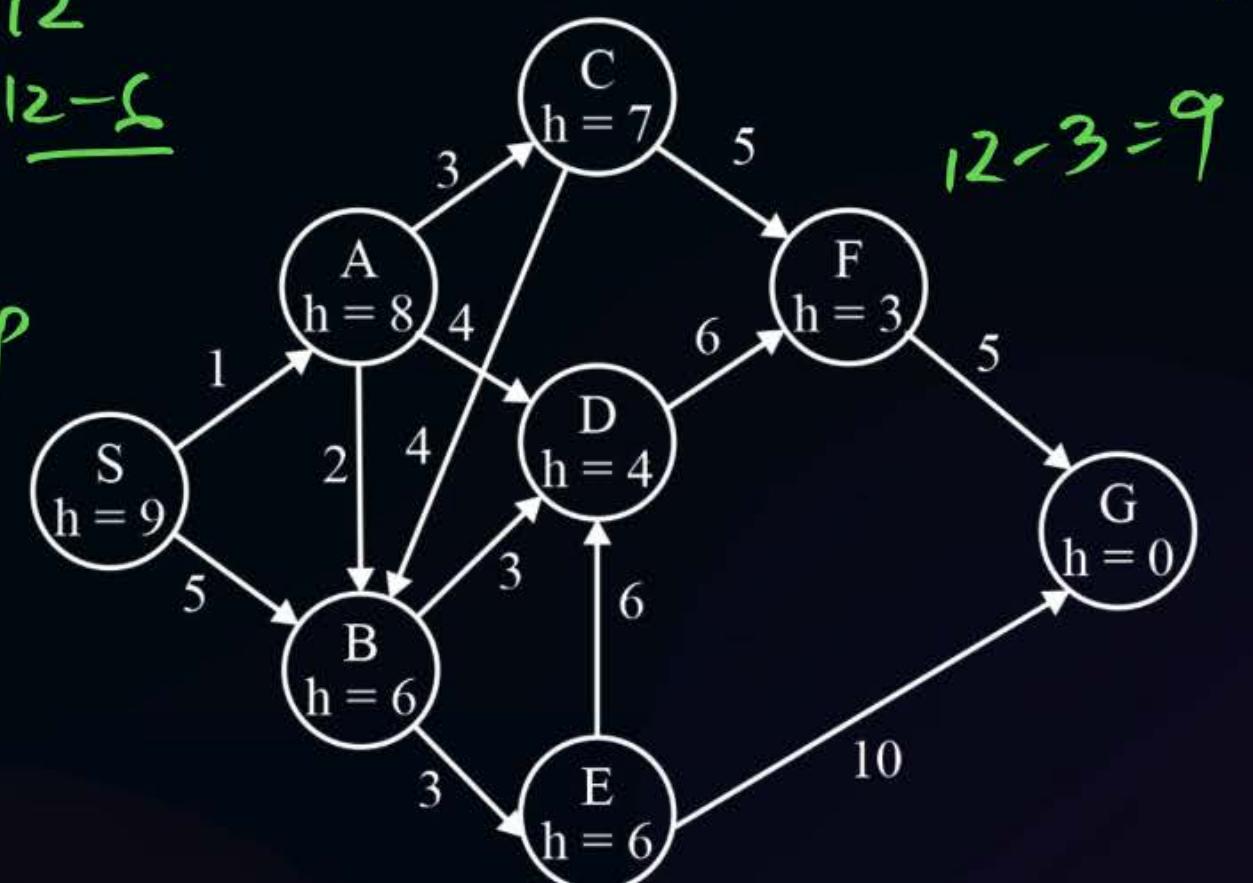
- What is the order of states expanded using A\* search?

|   | open | closed            |
|---|------|-------------------|
| S | X    | S A B D C E F G   |
| A | X    | 9 9 9 9 9 9 9     |
| B | X    | 11 11 11 11 11 11 |
| C | X    | 9 9 9 9 9 9 9     |
| D | X    | 12 12 12 12 12 12 |
| E | X    | 14 14 14 14 14 14 |
| F | X    | 16 16 16 16 16 16 |
| G | X    | 18 18 18 18 18 18 |

$$\alpha + 6 = 12$$

$$\alpha = 12 - 6$$

Stop



(Graph Search)

#Q. Consider a graph where each edge has a different cost. If you are using the Uniform Cost Search algorithm to find the shortest path from a start node A to a goal node G, which of the following statements is TRUE regarding the characteristics of UCS?

A

UCS is complete, but not optimal.

B

UCS is both complete and optimal

C

UCS can only be used in graphs with uniform edge costs.

D

UCS uses a depth-first strategy

UCS ↳ Complete ✓  
Optimal ✓

80%

#Q. Let  $h_1$  and  $h_2$  be two admissible heuristics used in A\* search.

Which ONE of the following expressions is always an admissible heuristic?

A

$$h_1 + h_2 \times$$

$$9 + 10 \leq 10 \times$$

B

$$h_1 \times h_2$$

$$9 \times 10 \leq 10 \times$$

C

$$|h_1 - h_2|$$



D

$$h_1 / h_2 \quad (h_1 \neq h_2)$$



$$\frac{10}{0.1} = 100 \leq 10 \times$$

$$\left\{ 0.1, 1, 2, \dots, 10 \right\}$$

$$h_1, h_2$$



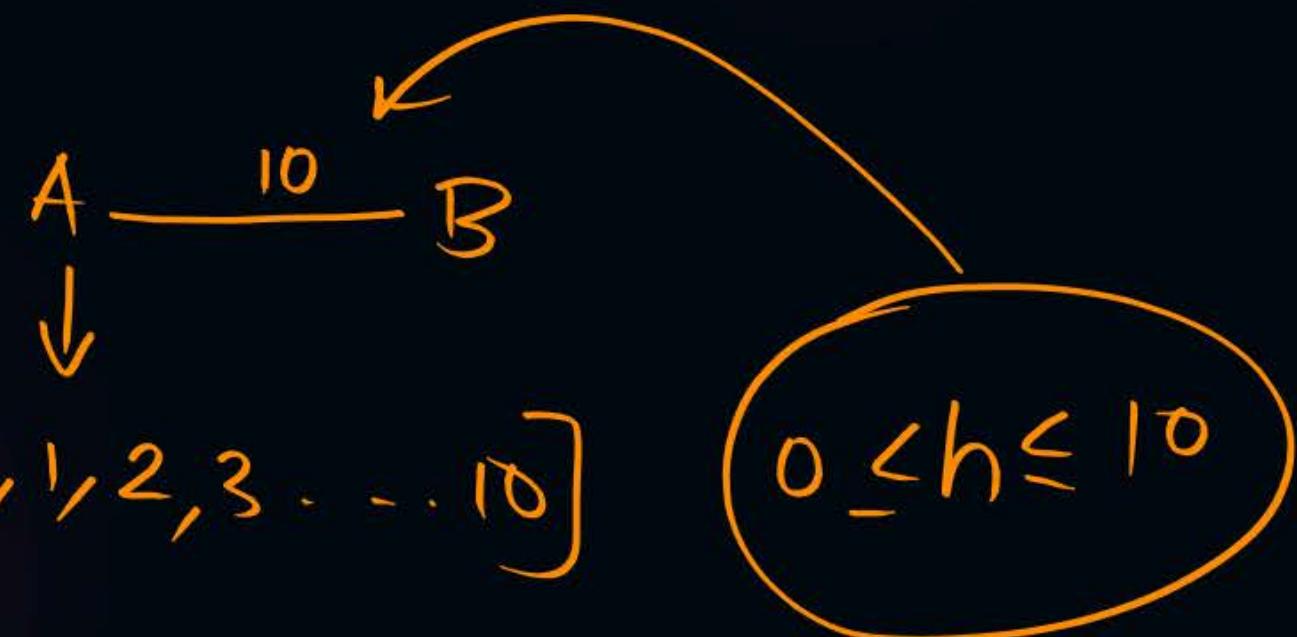
$$100 /$$

$$h \leq h^*$$

## [MCQ]



#Q. If  $h$  is an admissible heuristic (non-negative), which of the following can never be an admissible heuristic?



47%

A

$$h + 1 \Rightarrow h=1, |h| \leq 10$$

$\boxed{0, 1, 2, 3, \dots, 10}$

B

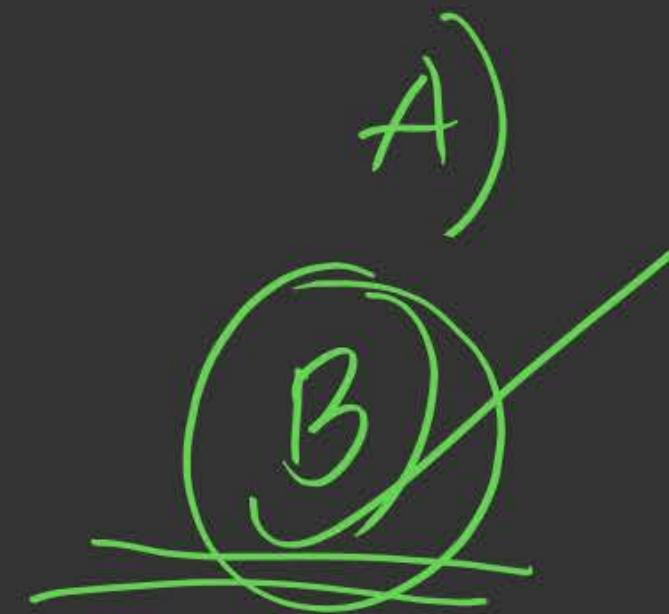
$$2h \Rightarrow h=1, 2x \leq 10$$

C

$$\sqrt{h} \Rightarrow h=1, \sqrt{1}=1 \leq 10$$

D

They all can be admissible under some situation.



77.33/100



#Q. If  $h_1$  and  $h_2$  are admissible heuristics, which of the following are guaranteed to be admissible?

A

$$h_1 + h_2 \times 10 + 9 = 19$$

B

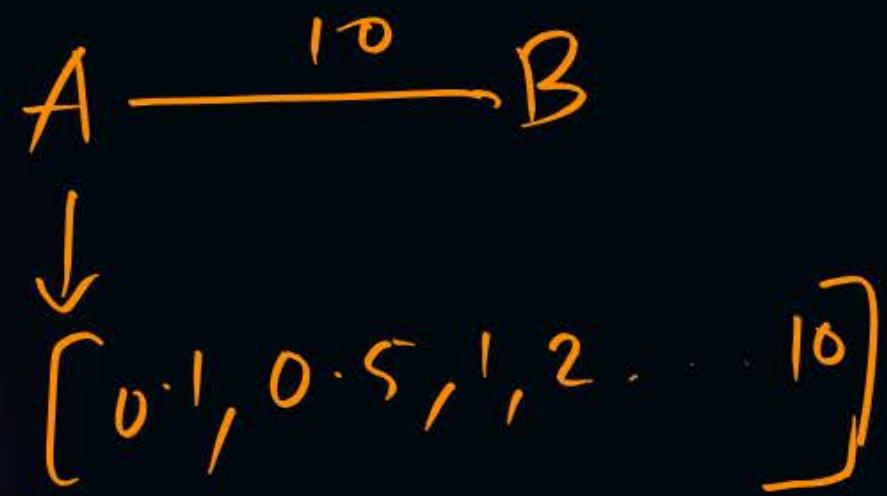
$$\min(h_1, h_2)$$

C

$$\max(h_1, h_2) \checkmark \quad \max(9, 10) = 10$$

D

$$\alpha h_1 + (1 - \alpha) h_2 \text{ for } \alpha \in [0, 1]$$

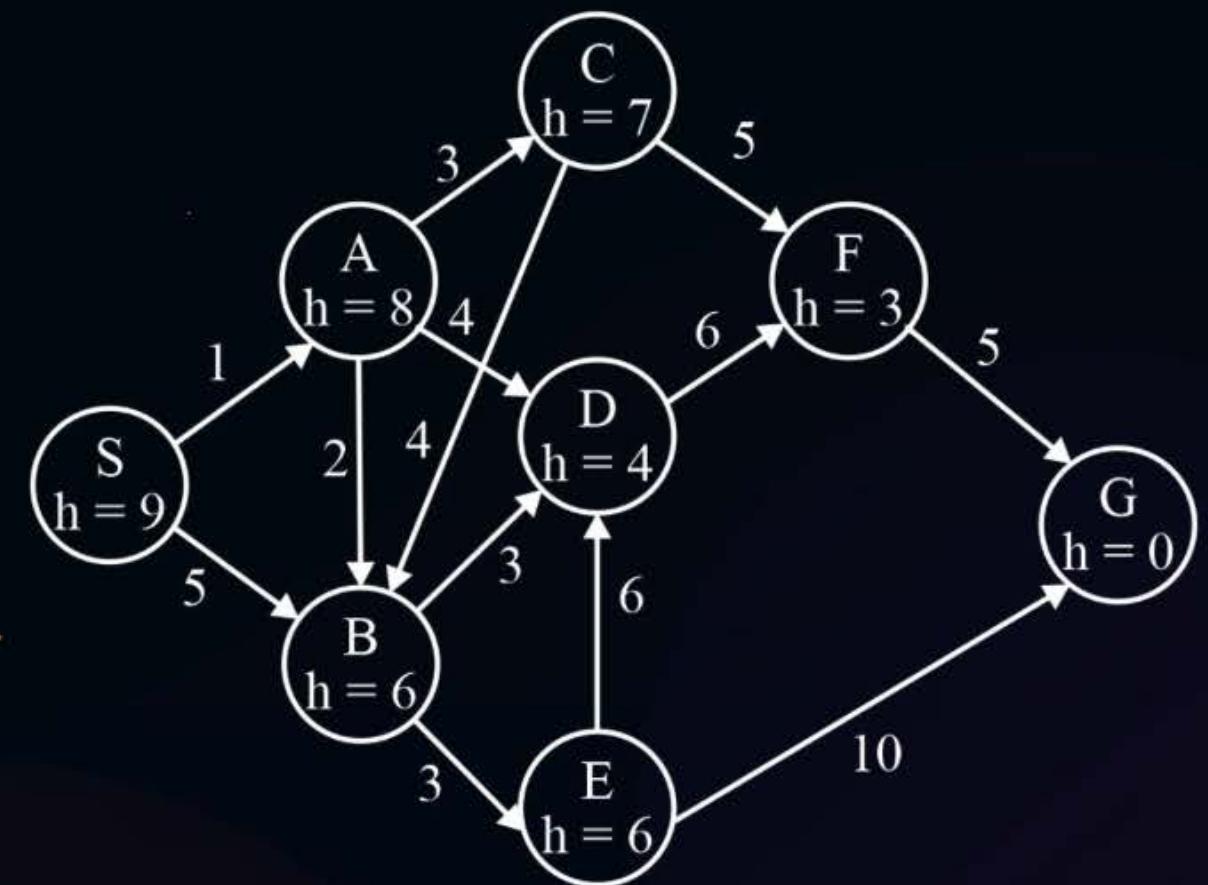
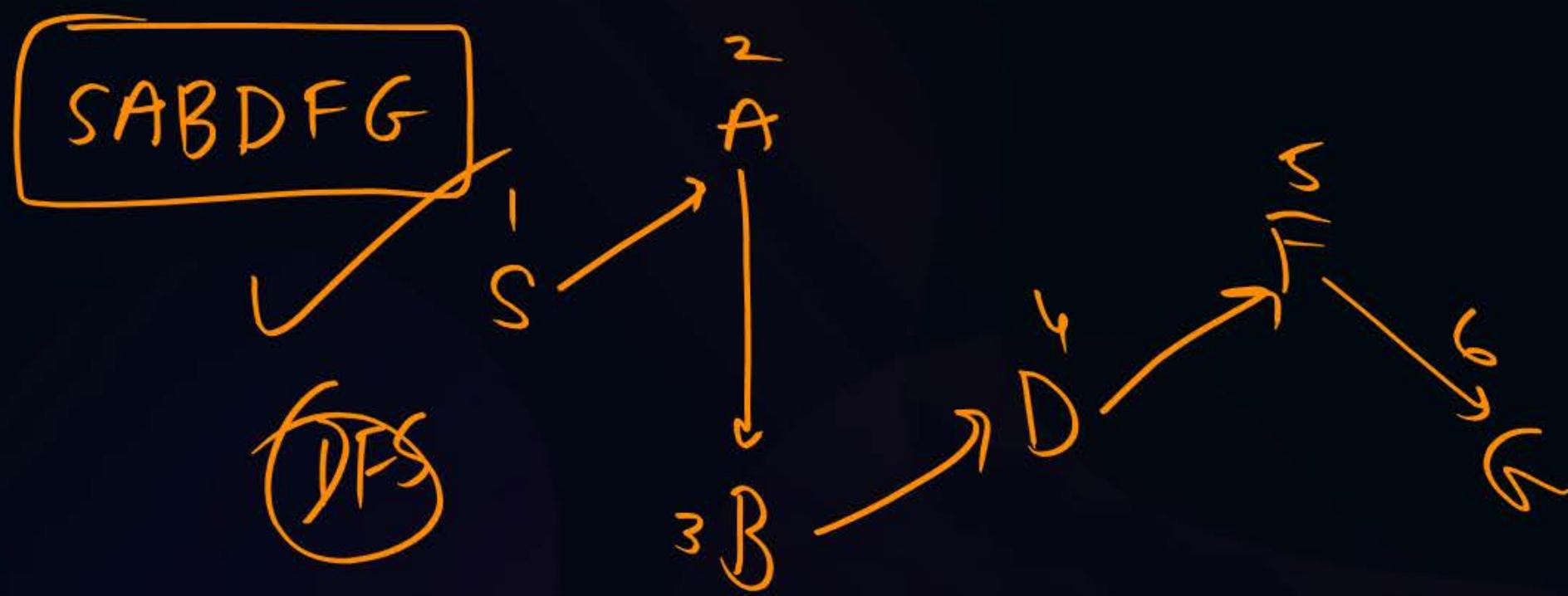


Tricky

$$\begin{aligned}
 & \alpha = 0, \\
 & h = 0 \cdot h_1 + 1 \cdot h_2 \\
 & h = h_2 \\
 & \alpha = 1 \\
 & h = h_1 \\
 & \alpha > 0.5 \\
 & = 0.5h_1 + 0.5h_2 \\
 & = \frac{9+10}{2} \leq 10
 \end{aligned}$$

#Q. In the problem the start state is S, and the goal state is G. The transition costs are next to the edges, and the heuristic estimate,  $h$ , of the distance from the state to the goal is in the state's node. Assume ties are always broken by choosing the state which comes first alphabetically.

- What is the order of states expanded using Depth First Search? Assume DFS terminates as soon as it reaches the goal state.



#Q. Which of the following evaluation functions will result in identical behaviour to greedy best-first search (assume all edge costs are positive)?

Adv

A

$$f(n) = 100 * h(n)$$

B

$$f(n) = \underbrace{g(n)}_{\text{fixed}} * h(n)$$

C

$$f(n) = \max \left( h(n) \right)^2$$

D

$$f(n) = 1/h(n)$$

GBFS

$$f = h$$

$$A, C$$

Idea: take  $\min h$   
only depend on h

| node | $h(a)$ | $100 \times h$ | $V_h$ | $b^2$   | $h \times g$ |
|------|--------|----------------|-------|---------|--------------|
| P    | 1      | 100 ✓          | $V_1$ | $b_1^2$ | $g$<br>1000  |
| Q    | 5      | 500            | $V_5$ | $b_5^2$ | 2            |
| R    | 2      | 200            | $V_2$ | $b_2^2$ | 5            |
| S    | 7      | 700            | $V_7$ | $b_7^2$ | 1            |



# Topic : Artificial Intelligence



**BFHS vs GBFS\* → GBFS (not Complete)**

**BFHS (Complete)**

- BFHS is a hybrid algorithm that combines aspects of Breadth-First Search (BFS) and heuristic guidance. It systematically explores the search space level by level, but with a preference for nodes with better heuristic values.
- GBFS is an algorithm that always expands the node that appears to be closest to the goal according to a heuristic. It focuses purely on the heuristic value ( $h(n)$ ), without considering the cost to reach the node ( $g(n)$ ).
- BFHS is generally less efficient than GBFS because it explores all nodes at a particular depth before moving on, potentially expanding unnecessary nodes.
- BFHS need more memory.
- GBFS need less memory.
- BFHS expand more nodes than GBFS.



**THANK - YOU**



# DS & AI ENGINEERING



## Artificial Intelligence

### Informed search

Lecture No.- 07

By- Aditya sir



# Recap of Previous Lecture



Topic

Questions

# Topics to be Covered

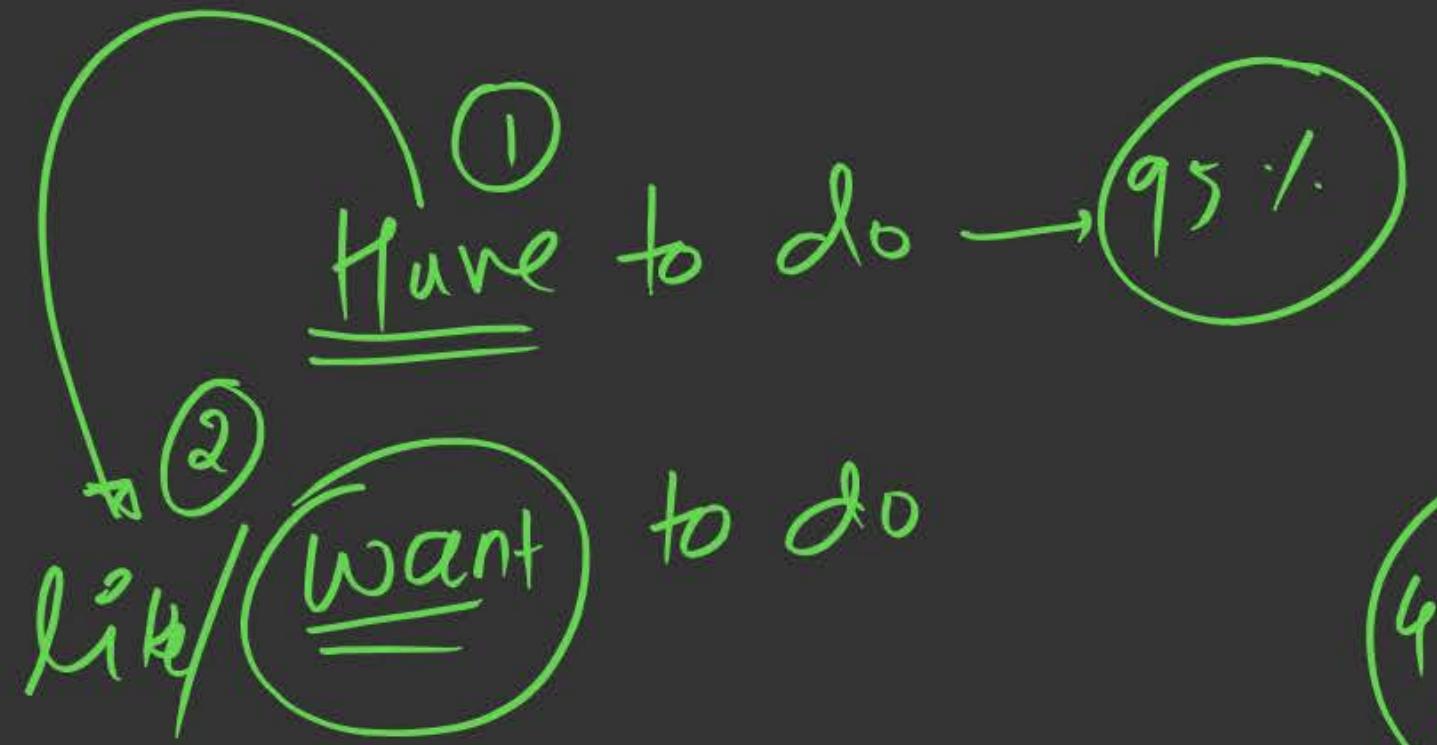
P  
W



Topic

Topic

Hill climbing  
Practice Questions



4 - 7 yrs old

① Love what you do  
② Do what you love



## Topic: Beam Search



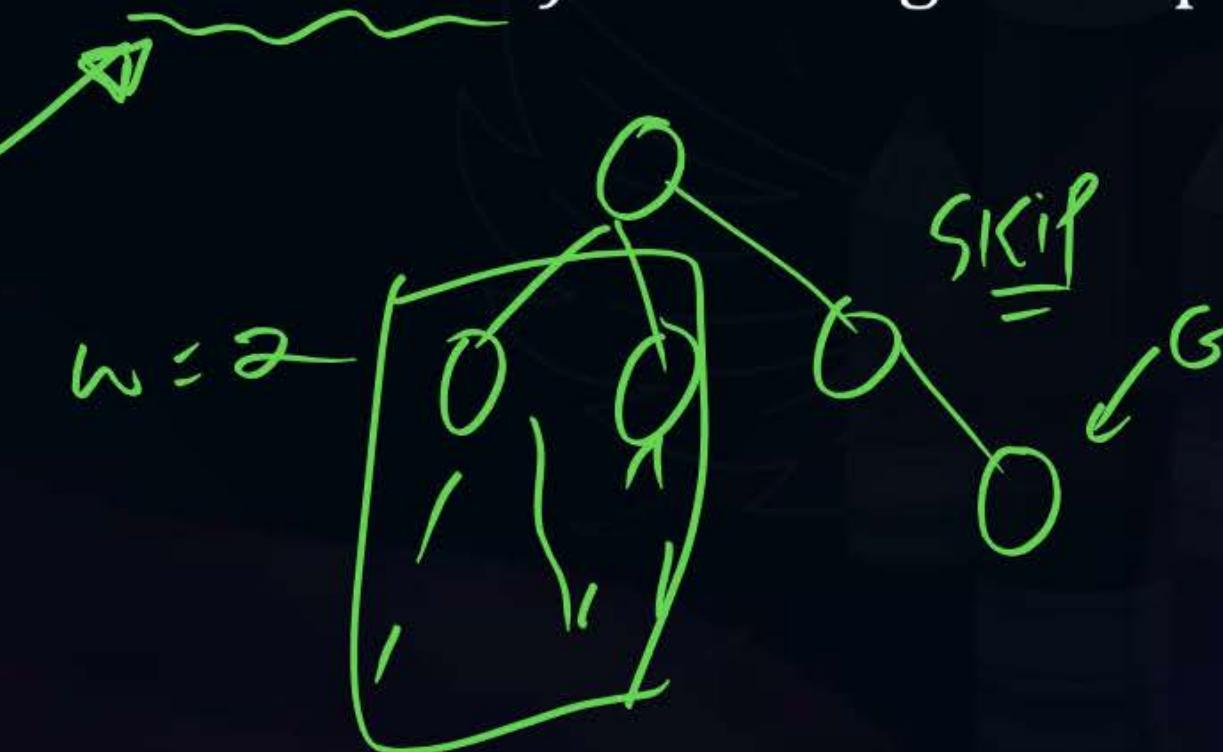
Beam Search  $\Rightarrow$  Complete  $\times$  Optimal



Concept of Beam width

- Whenever any node is Expanded, then we do not bring all nodes into the open list
- We define a Beam width ( $W$ ) and then whenever any node is visited, best " $W$ " neighbours (neighbours with min f value) are brought to open list.

In GBFS  $\Rightarrow f = \text{Heuristic Value}$





## Topic: Beam Search

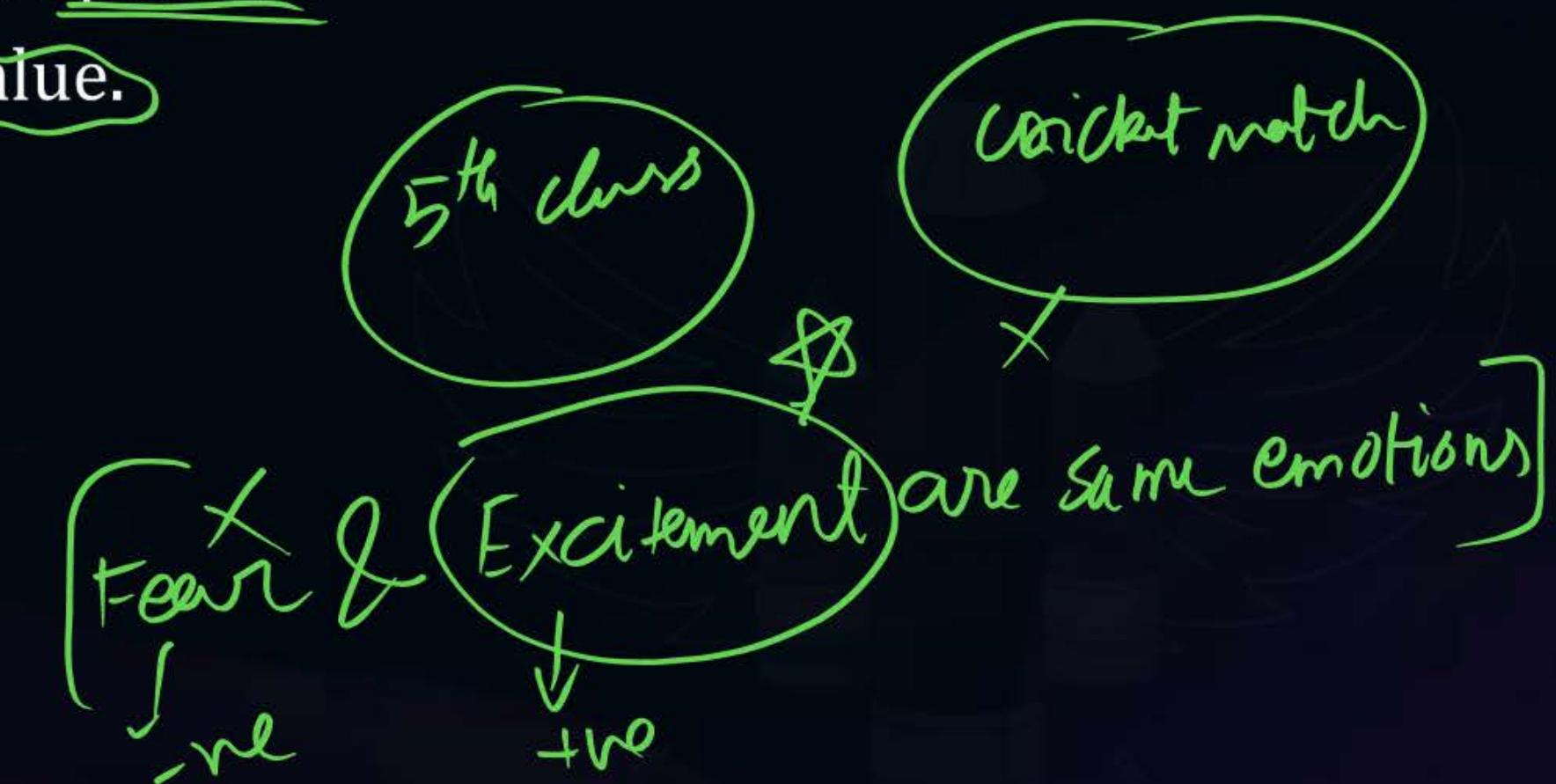
So, Steps in Beam Search  $\Rightarrow$  (Beam Width "w")

1. Start node visit.

Find  $f = h$  value for all neighbours.

2. Bring all the best "w" neighbours in open list.
3. Visit node in open list with min f value.
4. Keep repeating step 1, 2, 3.

(neither Complete nor Optimal).





## Topic: Hill Climbing Search



GBFS

V.VIMP

- Use an evaluation function  $f(n) = h(n)$ , but the maximum size of the nodes list is  $w$ , a fixed constant.

- Only keeps  $w$  best nodes as candidates for expansion and throws the rest away. More space-efficient than greedy search but may throw away a node that is on a solution path.

w = 1  $\Rightarrow$  Hill climbing search

w = define  $\Rightarrow$  Beam search

w =  $\infty$   $\Rightarrow$  GBFS

} Here f value of each node  $\Rightarrow \underline{\underline{h(n)}}$

- Space complexity  $\Rightarrow O(b^d)$
- Time complexity  $\Rightarrow O(b^d)$
- No beam width defined, we observe all the neighbours.

- If we define 'w' then
- Space complexity =  $O(w^d)$
- Time complexity =  $O(w^d)$

GBFS

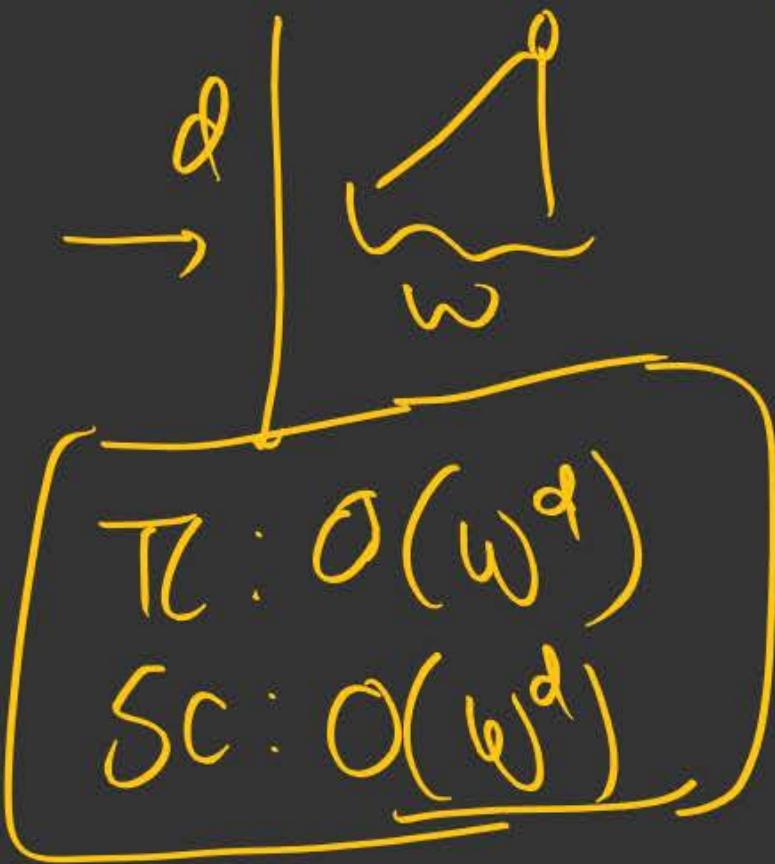
TC:  $O(b^d)$

SC:  $O(b^d)$



$\omega \leq b$

Beam Search:  $\xrightarrow{\omega}$

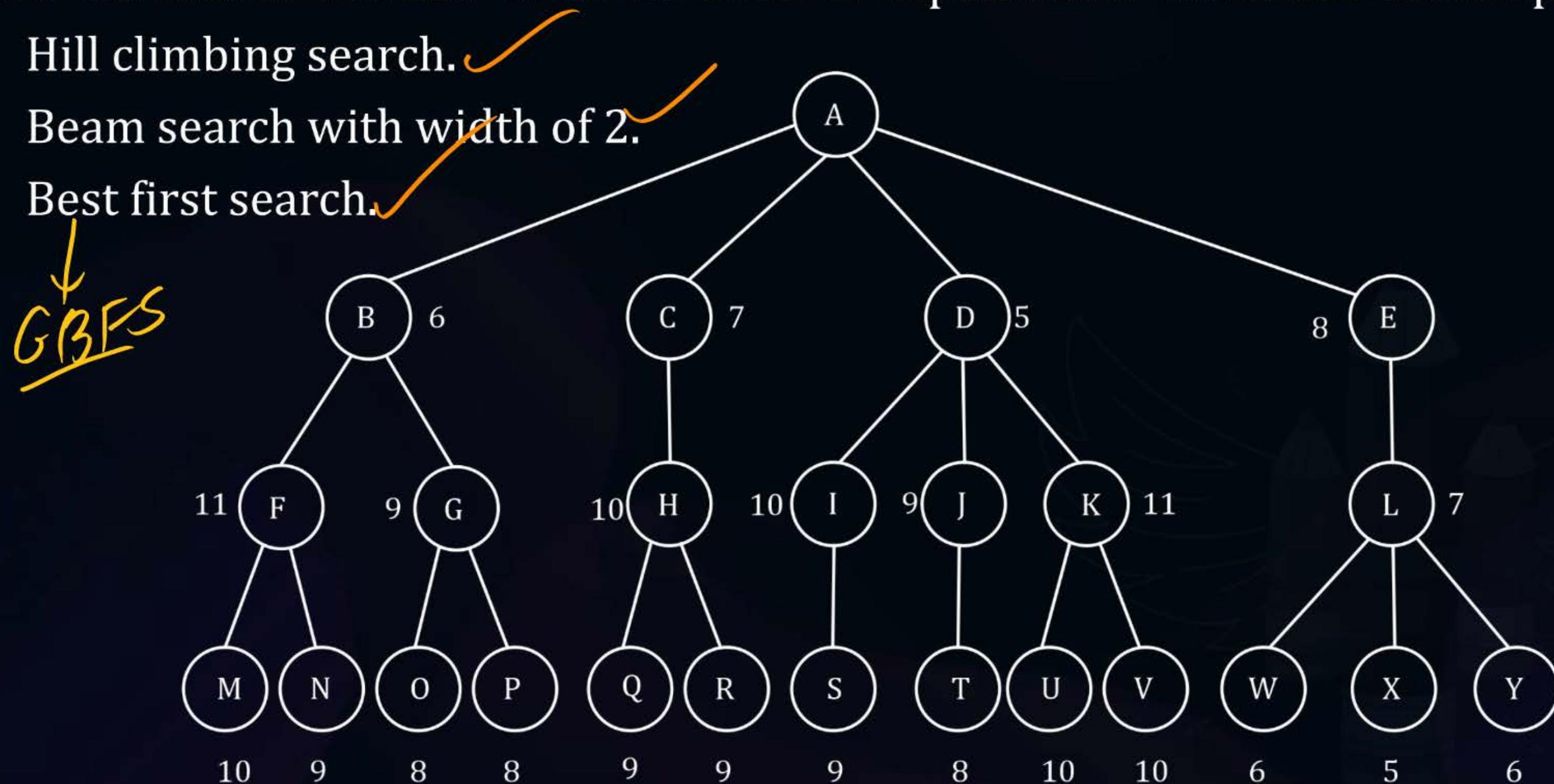




## Topic: Hill Climbing Search

Draw the search tree and write the order of expansion of the nodes when applying:

- Hill climbing search.
- Beam search with width of 2.
- Best first search.



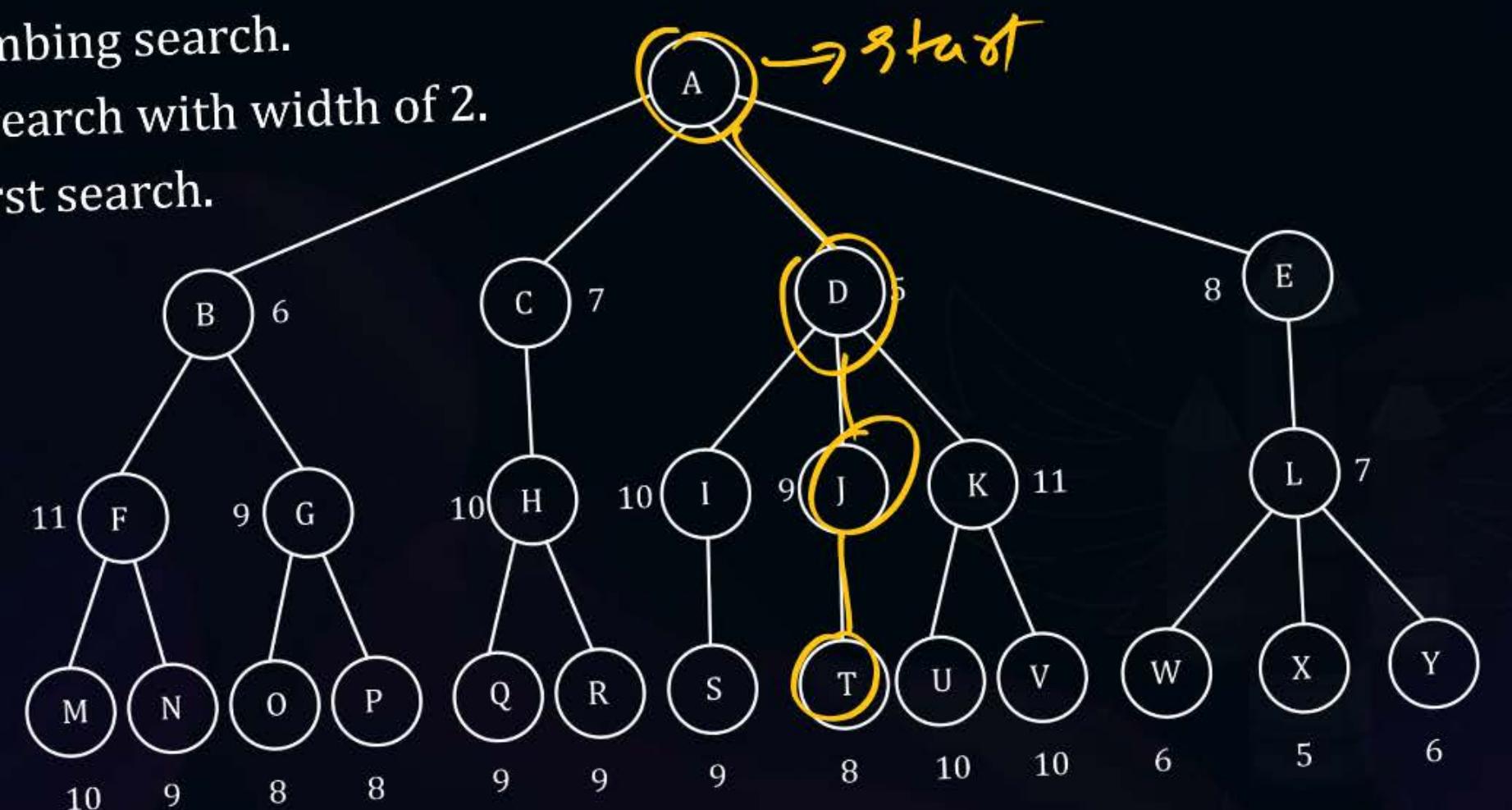
## c: Hill Climbing Search

search tree and write the order of expansion of the nodes when applying:

hilling search.

earch with width of 2.

st search.



Sequence of  
node visited

① Hill climbing .  $w = 1$

open list

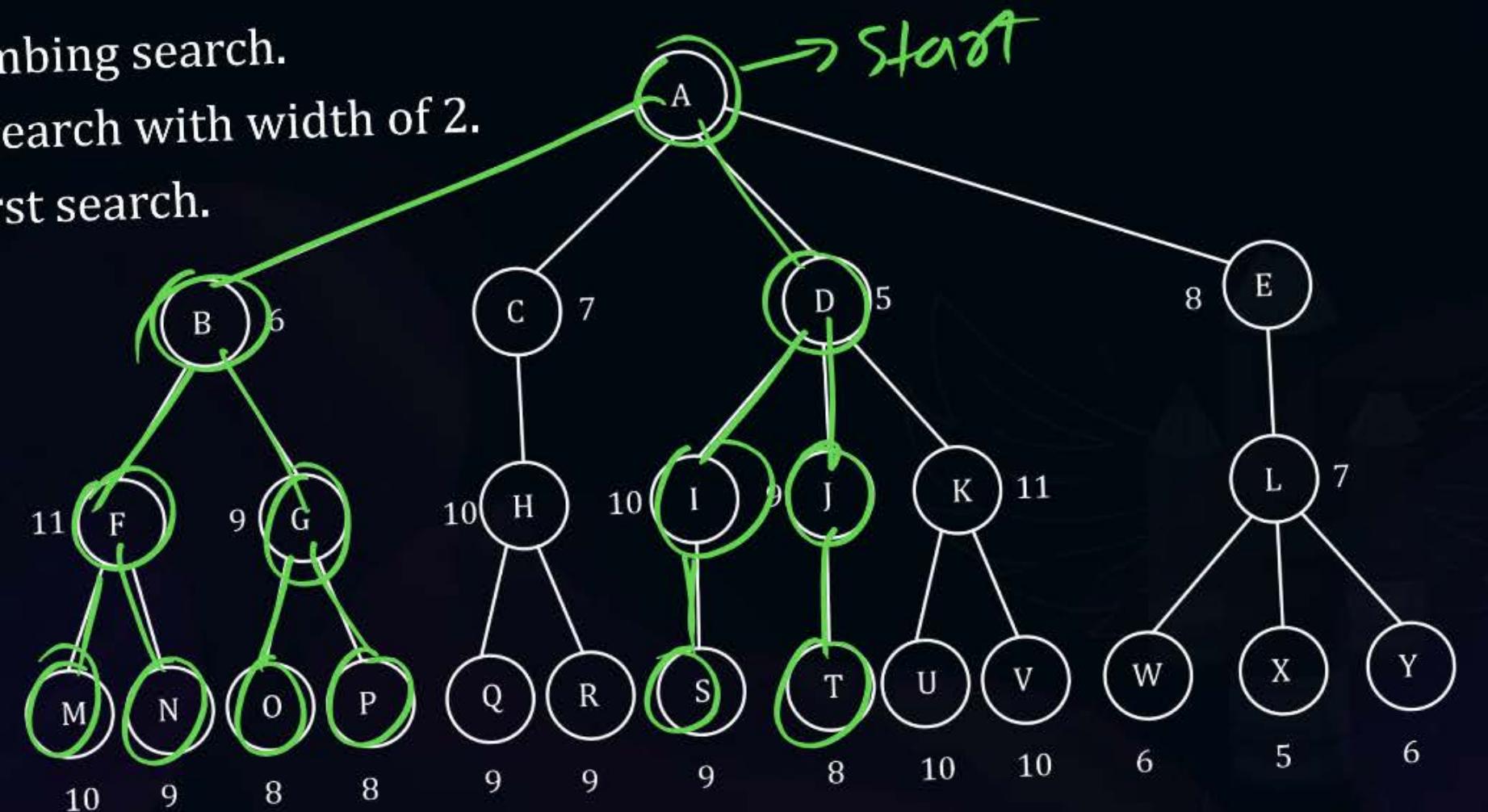
A D J T

Stop

## c: Hill Climbing Search

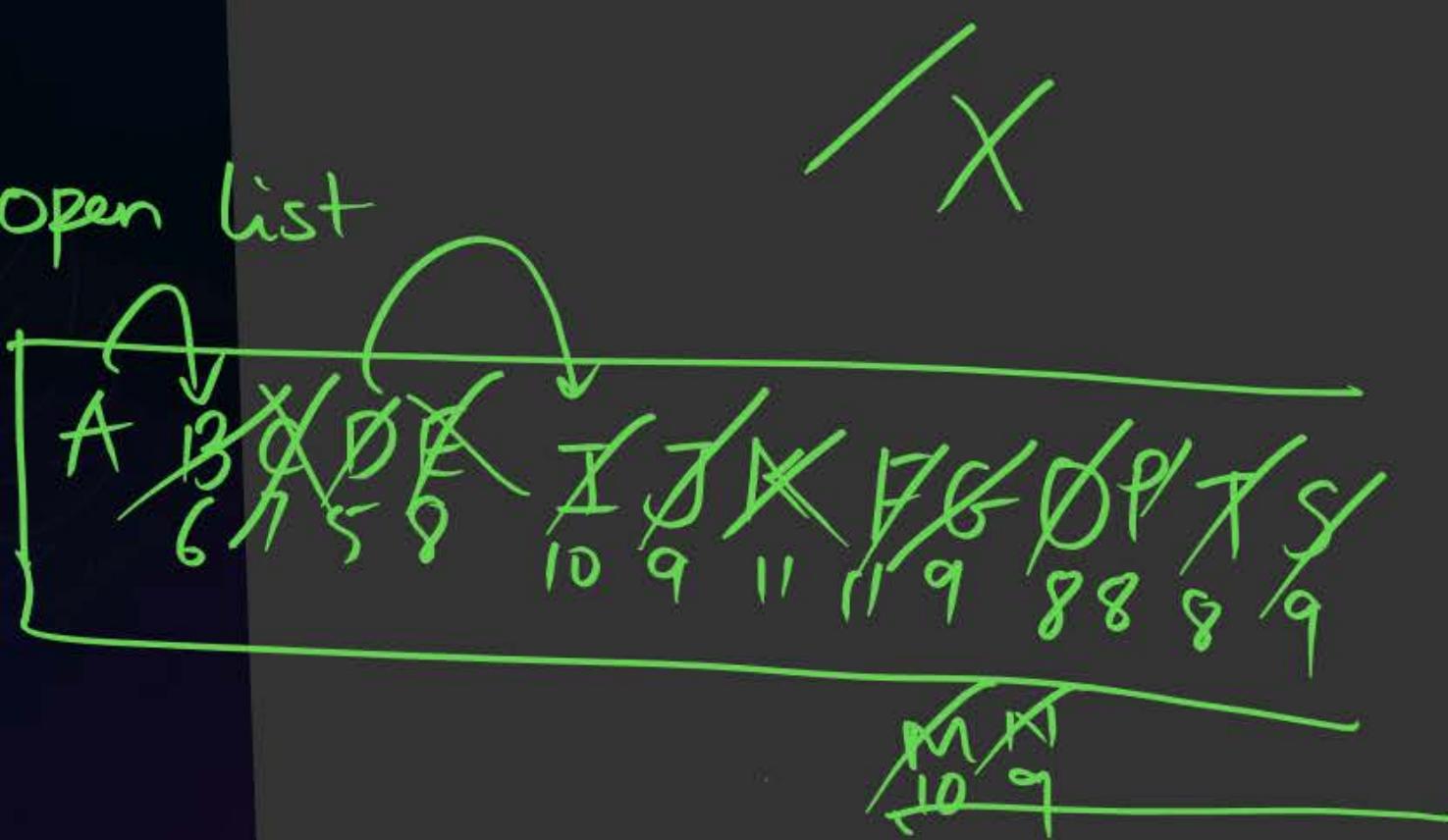
search tree and write the order of expansion of the nodes when applying:  
Hill Climbing search.

Search with width of 2.  
First search.



Sequence of visited nodes : (ADBFGOPJTIISFN)  
Stop

② Beam Search,  $W=2$

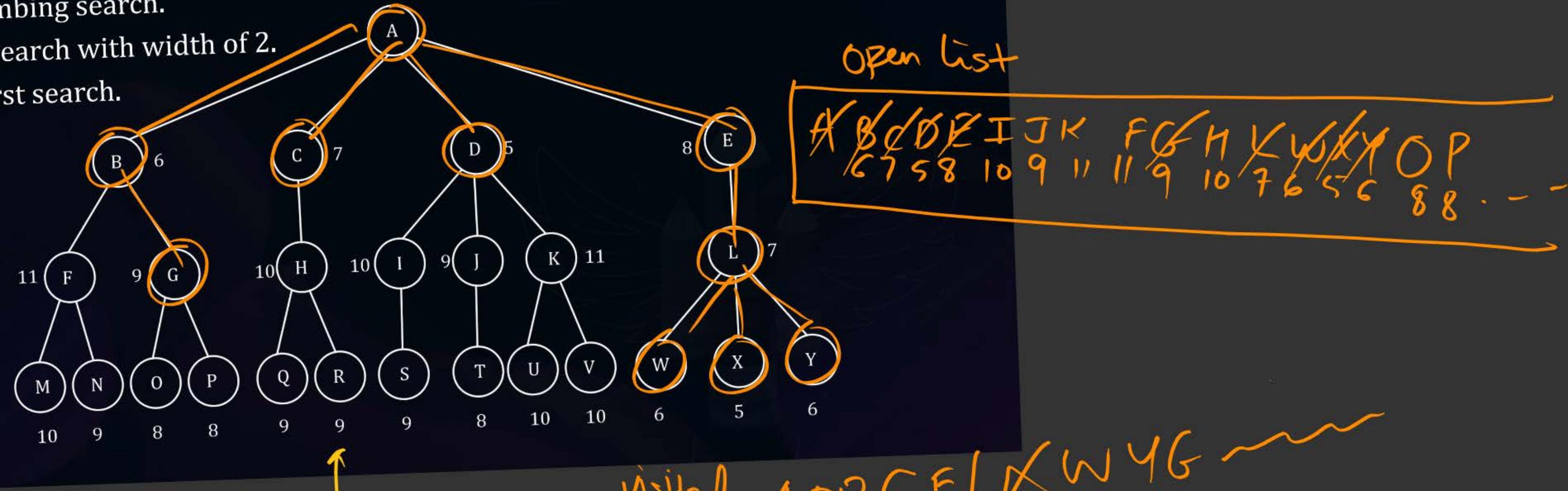


## c: Hill Climbing Search

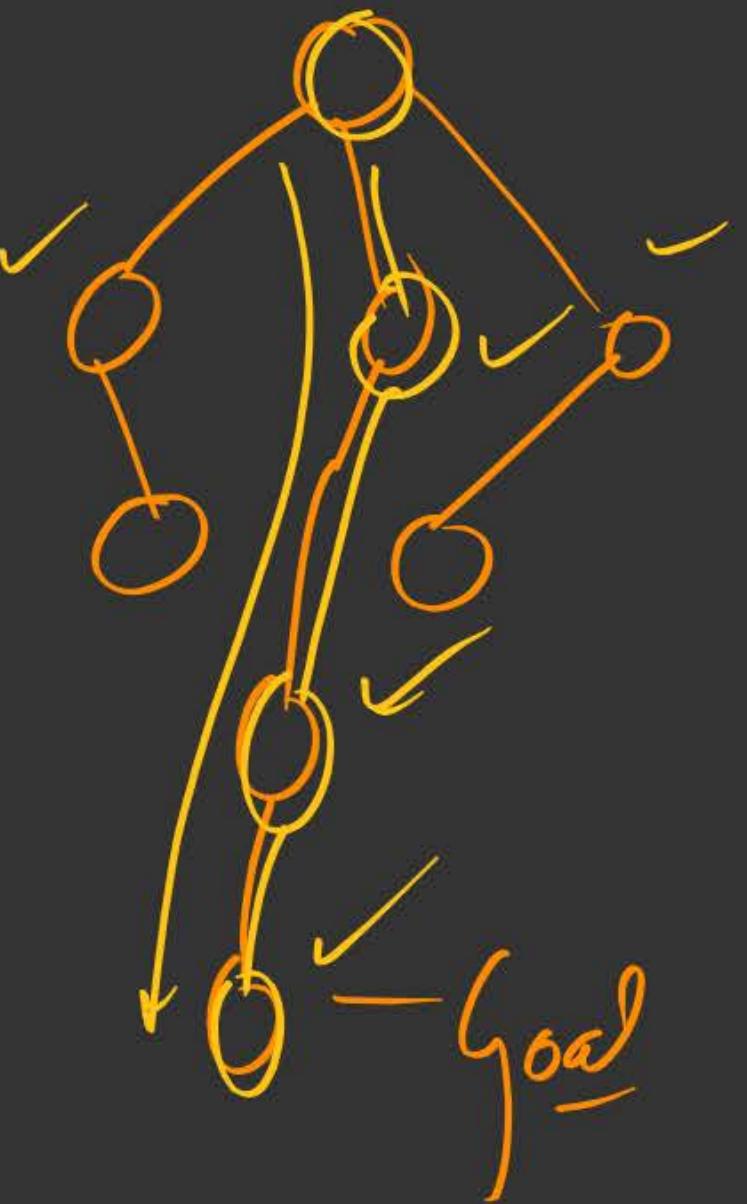
search tree and write the order of expansion of the nodes when applying:  
nbing search.

earch with width of 2.

st search.



Visited - A D B C E [X W Y G]  
Seq





## Topic: Hill Climbing Search

Hill climbing, beam search: Optimal X, Complete X

- because work ~~on~~ at heuristics only

because it delete or throw away some of nodes, because of this detection it may throw nodes that lead to goal.



Not complete:

because does not search whole graph

A yellow circle containing the equation  $f = h$ , with two diagonal lines through it, indicating it is incorrect or irrelevant.



## Topic: Hill Climbing Search

P  
W

GBFS

- If we have a Graph Search i.e. we create the visited node list i.e. once a node is visited it will not be visited again.

BFS:  $w = \infty$

- If we have a ~~trace~~ search no visited list

BFS:  $w = \infty$

Complete ✓ (slow)

Not Complete

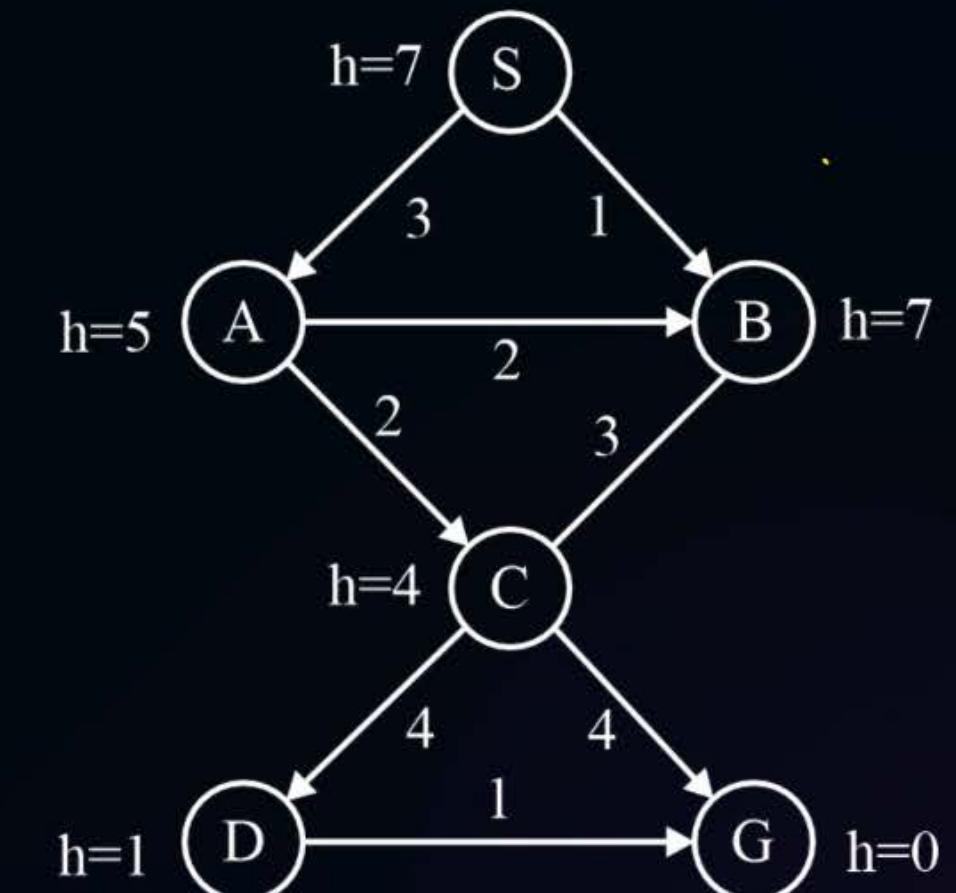
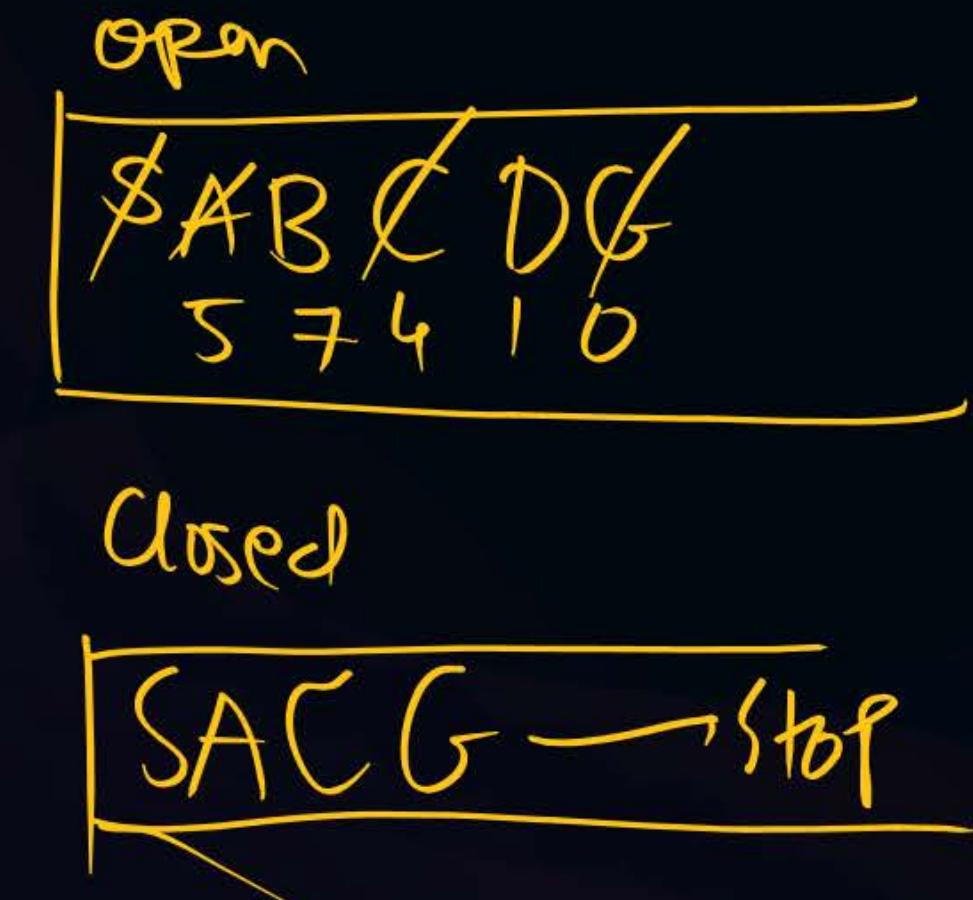
GBFS



#Q. We will investigate various search algorithms for the following graph. Edges are labeled with their costs, and heuristic values  $h$  for states are labeled next to the states. S is the start state, and G is the goal state. In all search algorithms, assume ties are broken in alphabetical order.

What path is returned by greedy graph search?

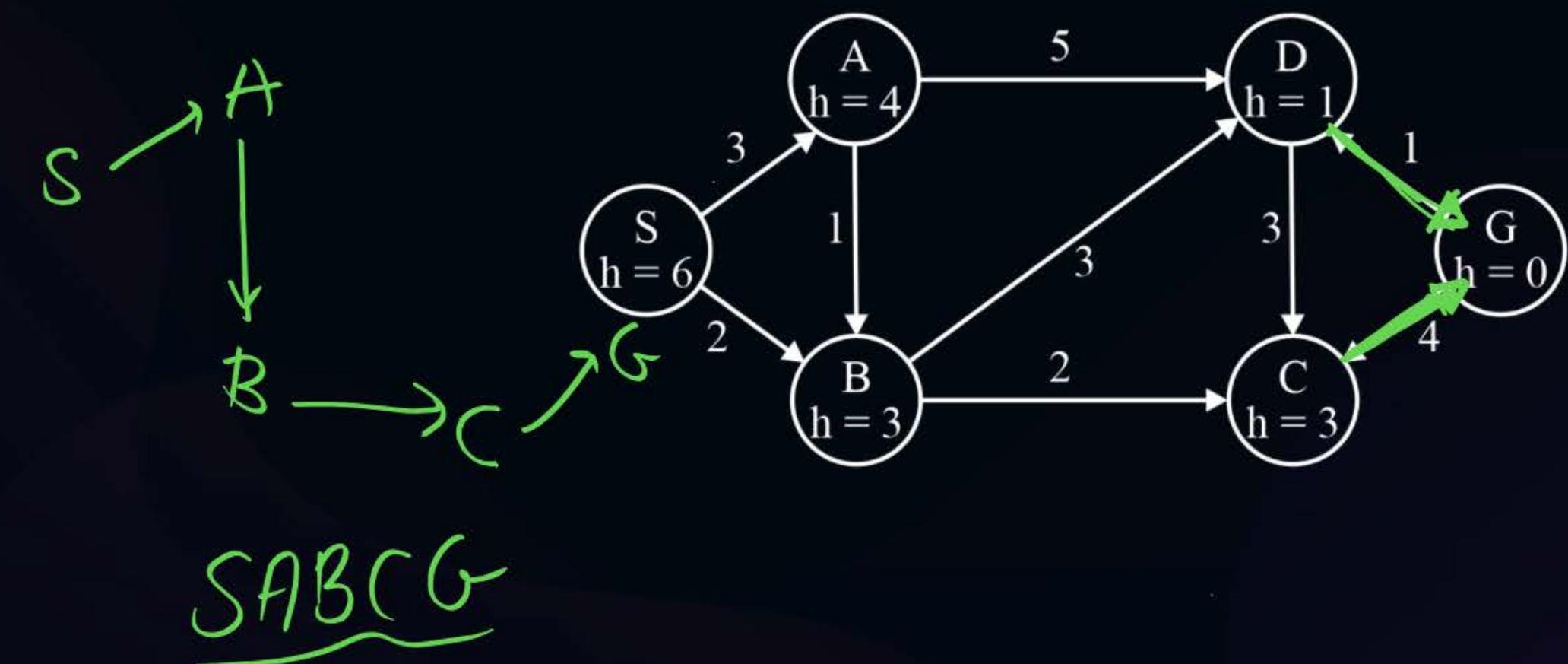
- A**  $S \rightarrow A \rightarrow C \rightarrow D \rightarrow G$
- B**  $S \rightarrow A \rightarrow C \rightarrow G$
- C**  $S \rightarrow A \rightarrow C \rightarrow D \rightarrow G$
- D** None of the above



#Q. Consider the following graph. For the following sub-questions, ties are broken in alphabetical order. *from* S  $\xrightarrow{+2}$  G  
Order of node visit is:

DFS

- A S A B D C G
- B S A B C D G
- C S A D G
- D None of the above



[NAT]

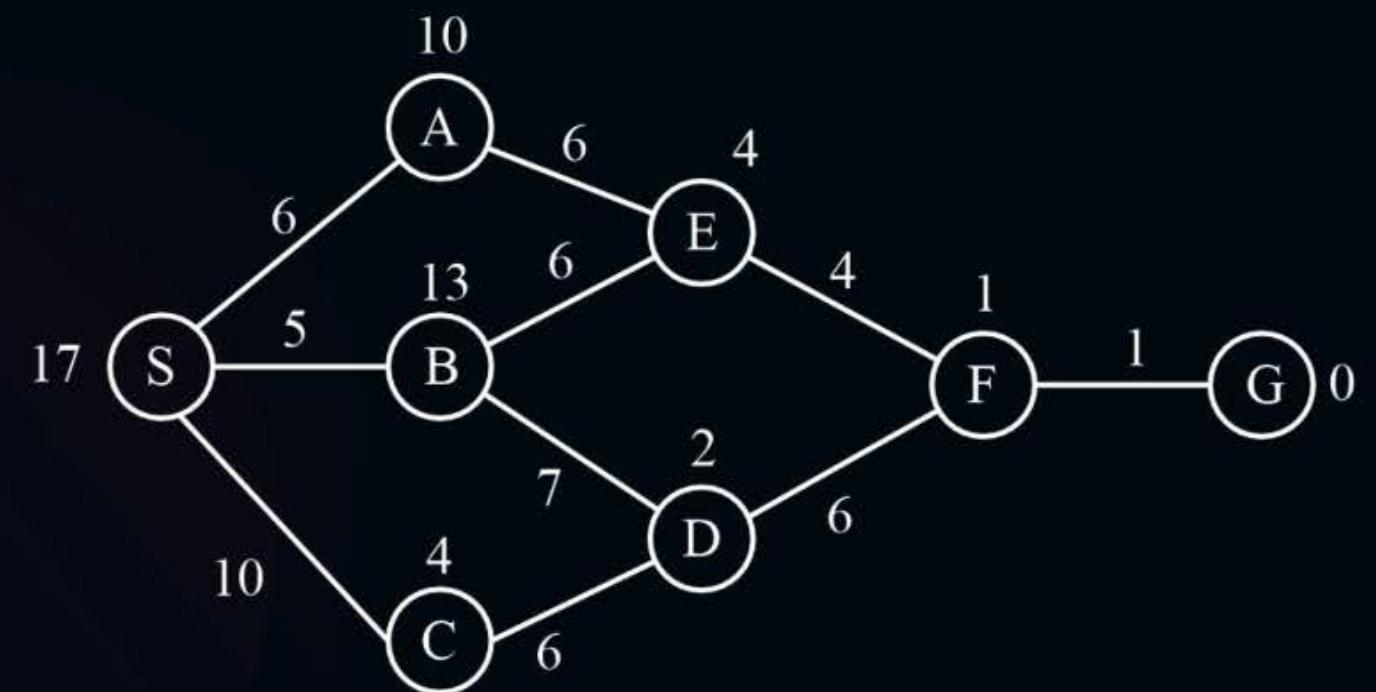
P  
W

Graph.

#Q. Perform the A\* algorithm on the following figure. Explicitly write down the queue at each step.

S → G  
Cost = ?

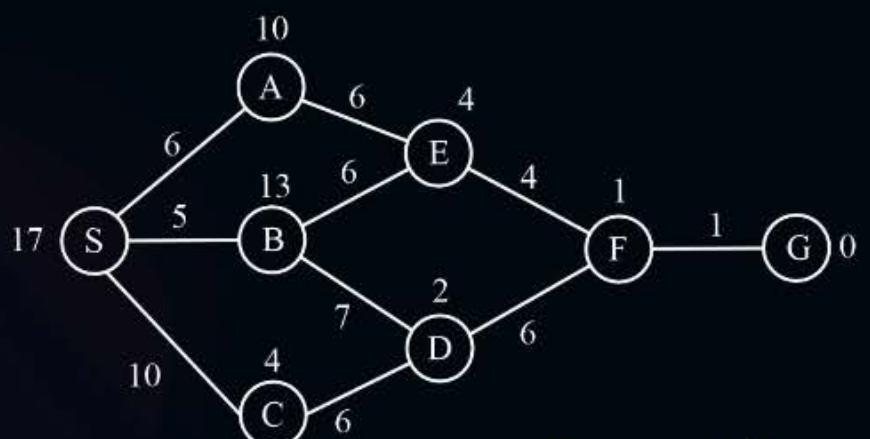
58%.



[NAT]

graph

#Q. Perform the A\* algorithm on the following figure. Explicitly write down the queue at each step.

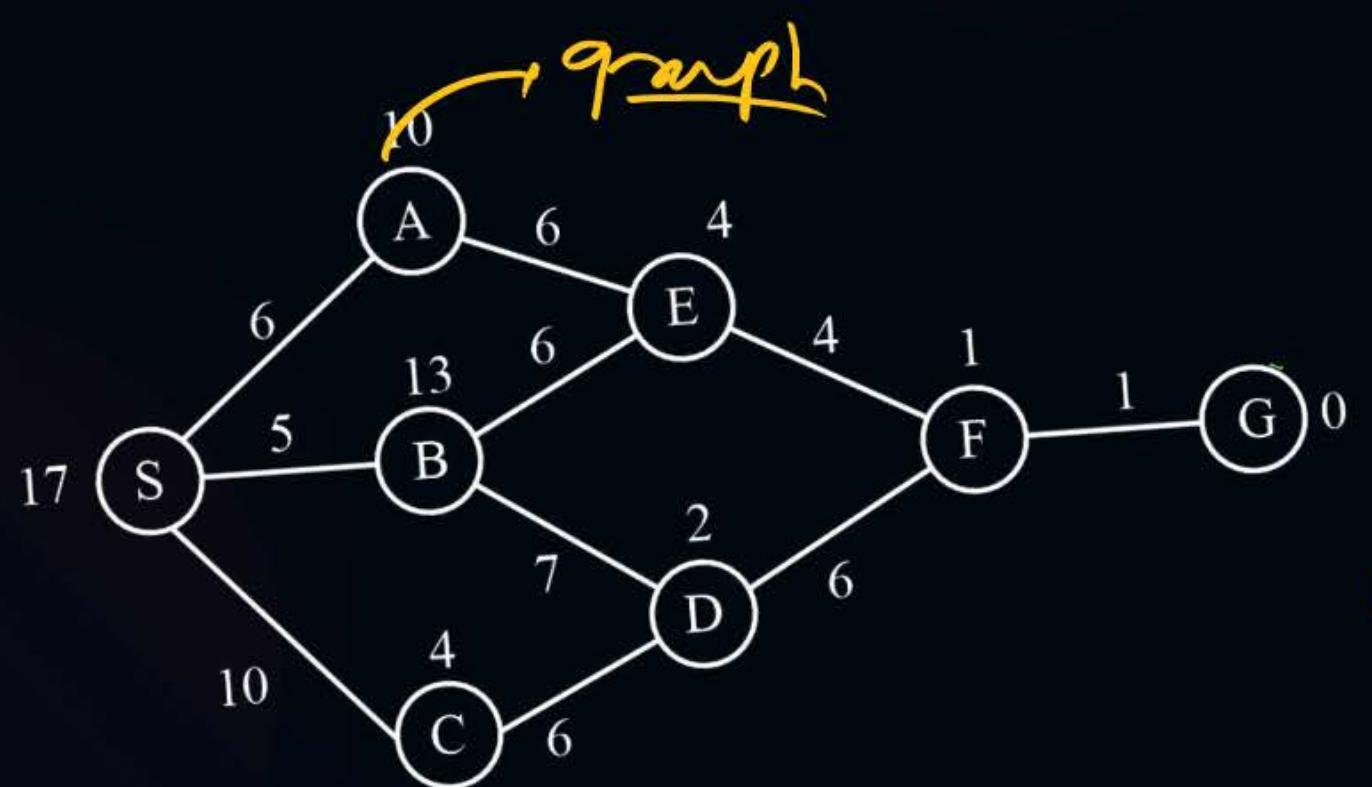


S → G

Cost = 17  
 Path [SAEFG] ✓

|    | open | closed |    |    |    |   |
|----|------|--------|----|----|----|---|
| S* | S    | C      | A  | E  | F  | G |
| A* | 16   | (16)   | 16 | 16 | 15 |   |
| B  | 18   | 18     | 18 | 18 | 18 |   |
| C* | 14   | 14     | 14 | 14 | 14 |   |
| D  | X    | 18     | 18 | 18 | 18 |   |
| E* | X    | X      | 16 | 16 | 16 |   |
| F* | XX   | X      | 17 | 17 | 17 |   |
| G* | XX   | X      | 17 | 17 | 17 |   |

stop



S → G  
Cost = ?

Apply UCS:

path  
SBEFG  
cost = 16  
90%

|   | open | closed |      |
|---|------|--------|------|
| S | X    |        |      |
| X | A    |        |      |
| A | X    | B      |      |
| B | X    | X      |      |
| C | X    |        |      |
| X | E    |        |      |
| E | X    | F      |      |
| F | X    | G      |      |
| G |      |        | Stop |

Cost values in circles:

- S: 0
- B: 6
- E: 10
- F: 11
- G: 12
- D: 12
- A: 6
- C: 6
- H: 16
- I: 11
- J: 12
- K: 15
- L: 15
- M: 16



**THANK - YOU**

# DS & AI ENGINEERING



## Artificial Intelligence

### Informed search

Lecture No.- 08

By- Aditya sir



# Recap of Previous Lecture



Topic

Topic

Topic

Topic

Hill climbing

Beam Search

GBFS



# Topics to be Covered



Topic

Topic

Topic

IDA\* Numerical  
AO\*  
Practice

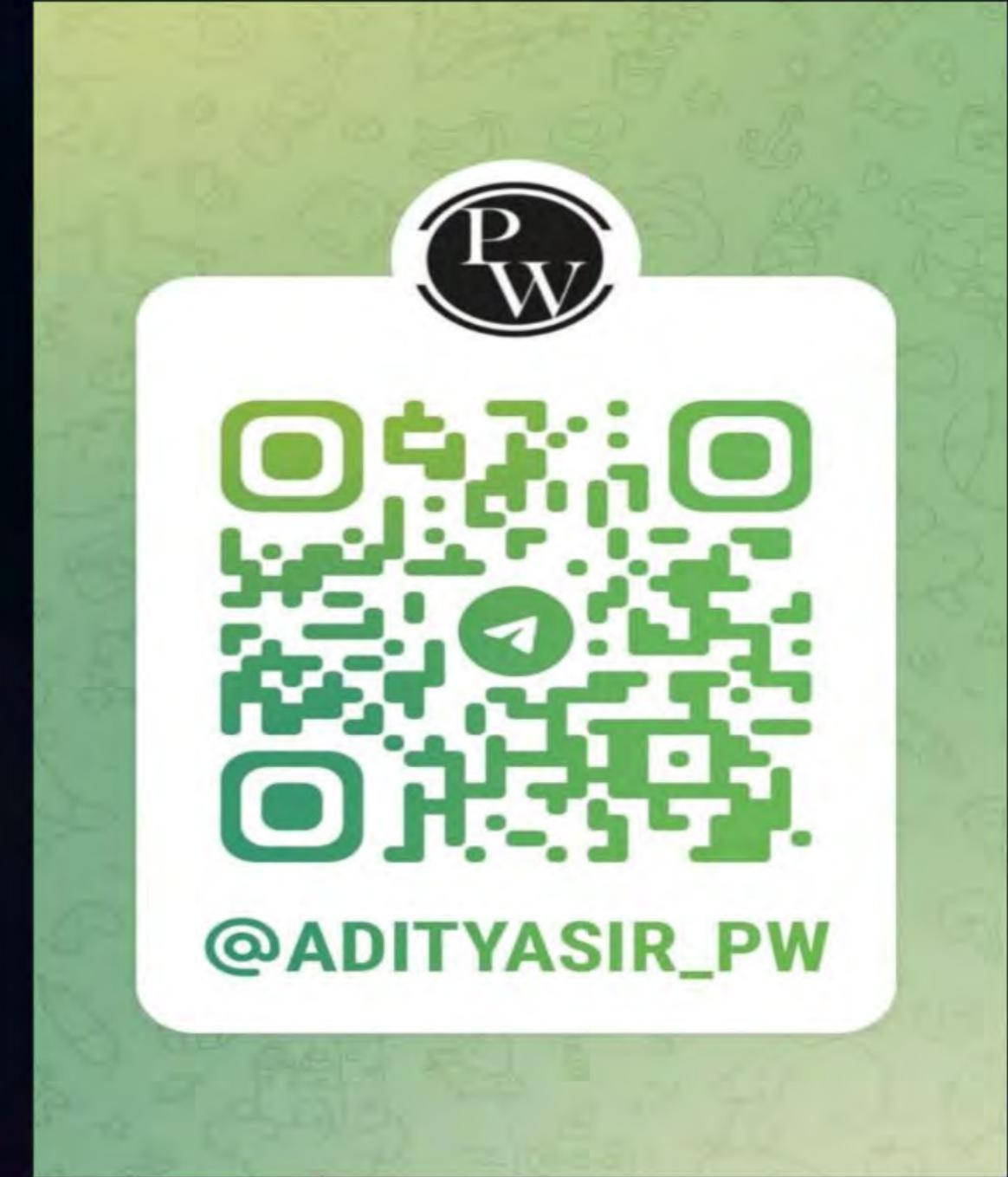




## About Aditya Jain sir

1. Appeared for GATE during BTech and secured AIR 60 in GATE in very first attempt - City topper
2. Represented college as the first Google DSC Ambassador.
3. The only student from the batch to secure an internship at Amazon. (9+ CGPA)
4. Had offer from IIT Bombay and IISc Bangalore to join the Masters program
5. Joined IIT Bombay for my 2 year Masters program, specialization in Data Science
6. Published multiple research papers in well known conferences along with the team
7. Received the prestigious excellence in Research award from IIT Bombay for my Masters thesis in ML
8. Completed my Masters with an overall GPA of 9.36/10
9. Joined Dream11 as a Data Scientist
10. Have mentored 15,000+ students & working professionals in field of Data Science and Analytics
11. Have been mentoring & teaching GATE aspirants to secure a great rank in limited time
12. Have got around 27.5K followers on Linkedin where I share my insights and guide students and professionals.





**Telegram Link for Aditya Jain sir:**  
[https://t.me/AdityaSir\\_PW](https://t.me/AdityaSir_PW)



## Topic : Artificial Intelligence

### Iterative Deepening A\* algorithm (IDA\*) - Artificial intelligence

- **Step 1:** Initialization  
Set the root node as the current node, and find the f-score.
- **Step 2:** Set threshold  
Set the cost limit as a threshold for a node i.e the maximum f-score allowed for that node for further explorations.
- **Step 3:** Node Expansion  
Expand the current node to its children and find f-scores.
- **Step 4:** Pruning  
If for any node the **f-score > threshold**, prune that node because it's considered too expensive for that node, and store it in the visited node list.
- **Step 5:** Return Path  
If the Goal node is found then return the path from the start node to Goal node.



## Topic : Artificial Intelligence

Iterative Deepening A\* algorithm (IDA\*) - Artificial intelligence

- Step 6: Update the Threshold

If the Goal node is not found then repeat from step 2 by changing the threshold with the minimum pruned value from the visited node list. And Continue it until you reach the goal node.



# Topic : Artificial Intelligence



It 1:  $f_{th} = 2$ , AB pruned node

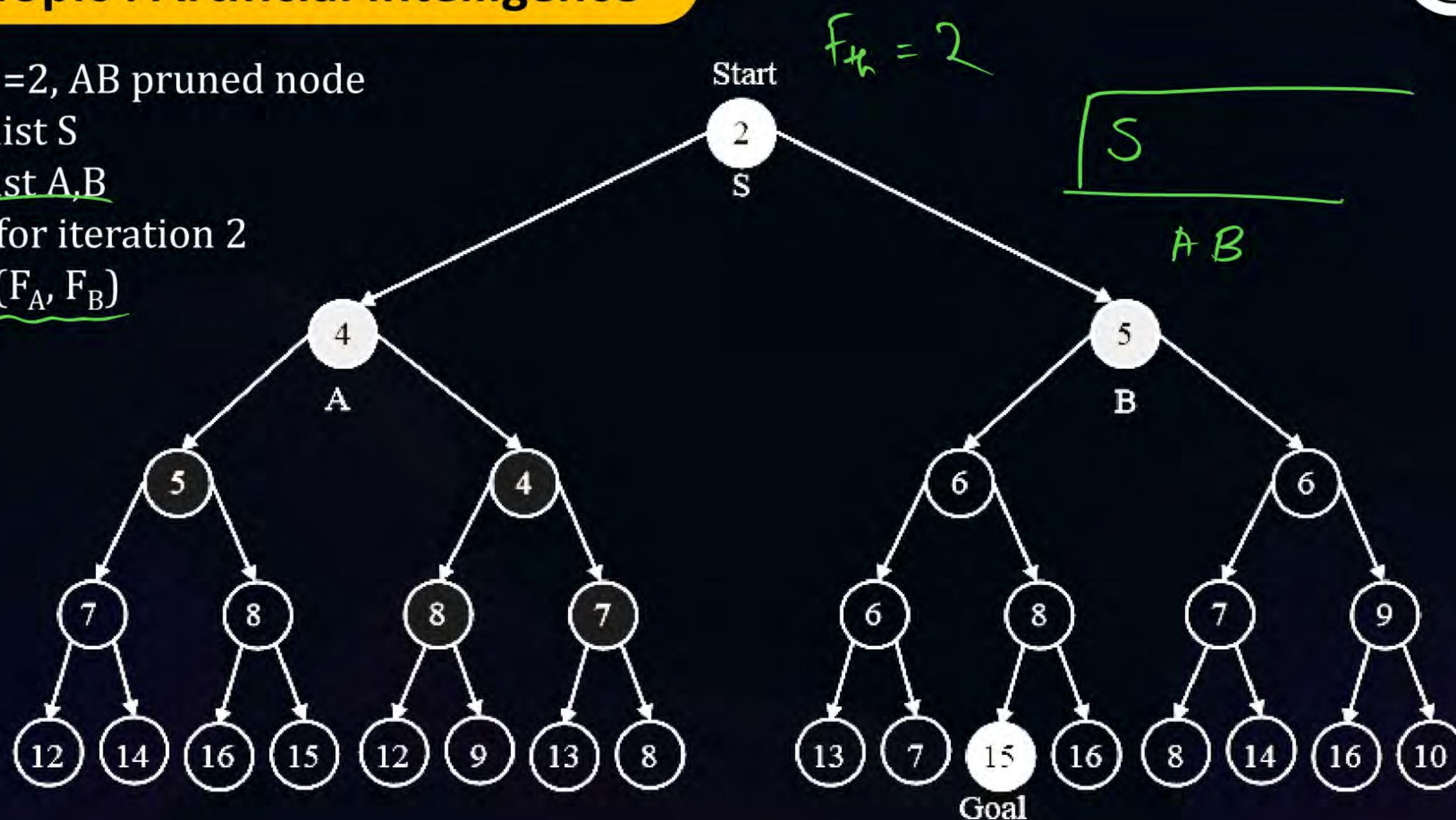
Closed list S

Prune list A,B

- $F_{th}$  for iteration 2

$$\Rightarrow \min (\underline{F_A, F_B})$$

$$\Rightarrow 4$$





# Topic : Artificial Intelligence



It 1:  $f_{th} = 2$ , AB pruned node

It 2:  $f_{th} = 4$

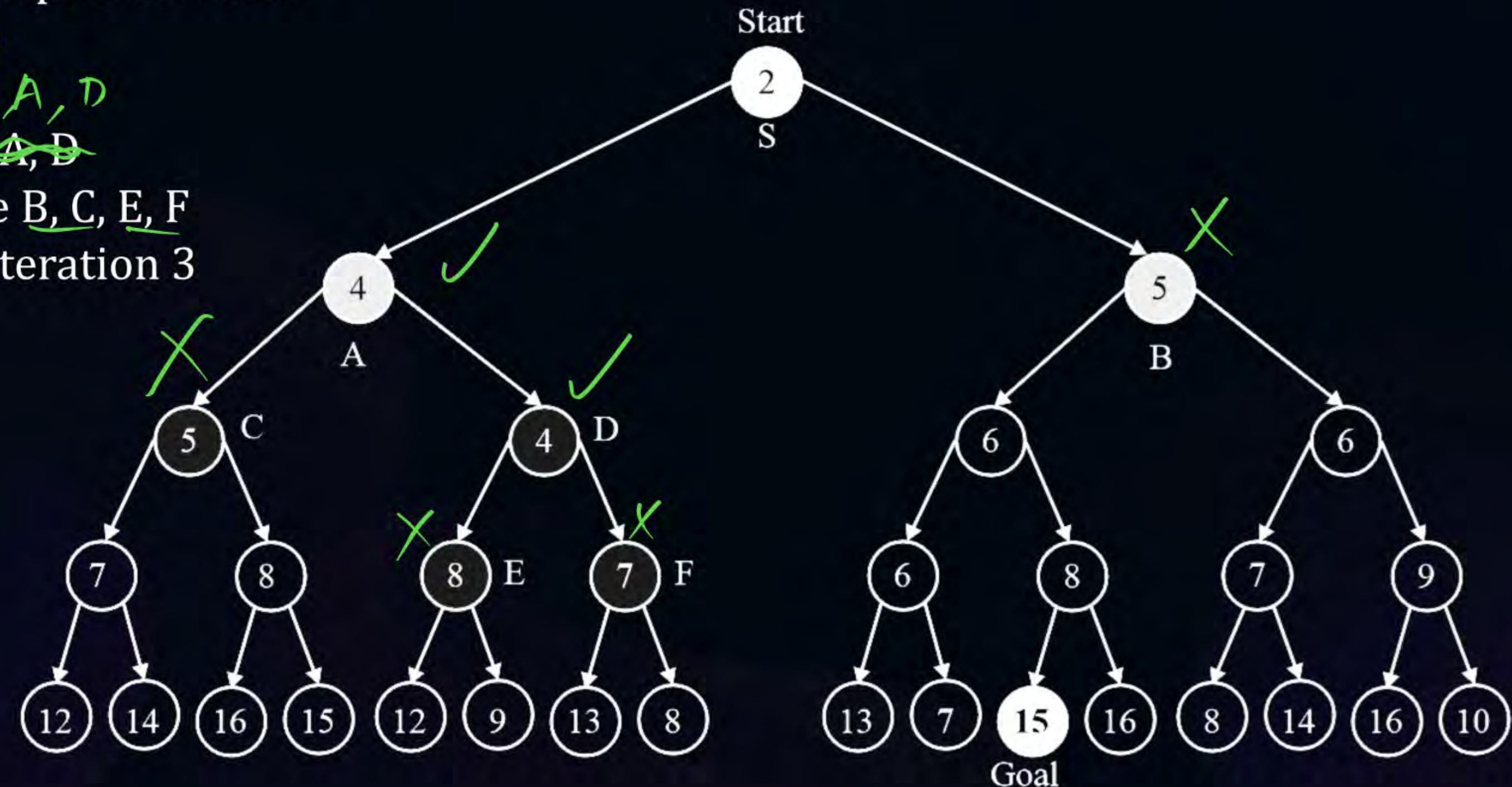
Closed list S A, D

Pruned list S, A, D

Pruned node B, C, E, F

New  $F_{th}$  for iteration 3

$\Rightarrow$  5





# Topic : Artificial Intelligence



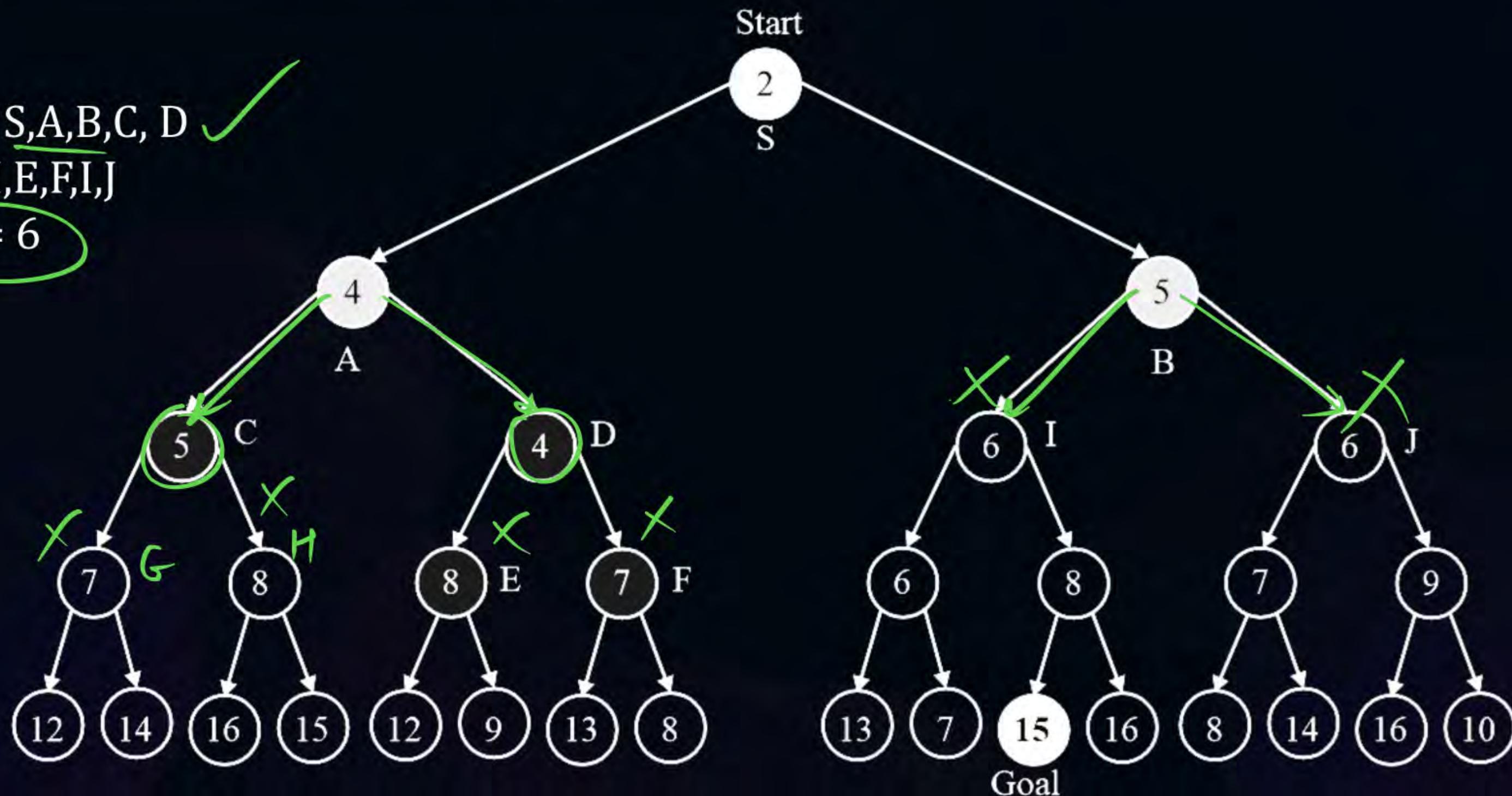
It: 3

Fth = 5

Close list: S, A, B, C, D ✓

Prune G, H, E, F, I, J

It 4: Fth = 6





# Topic : Artificial Intelligence



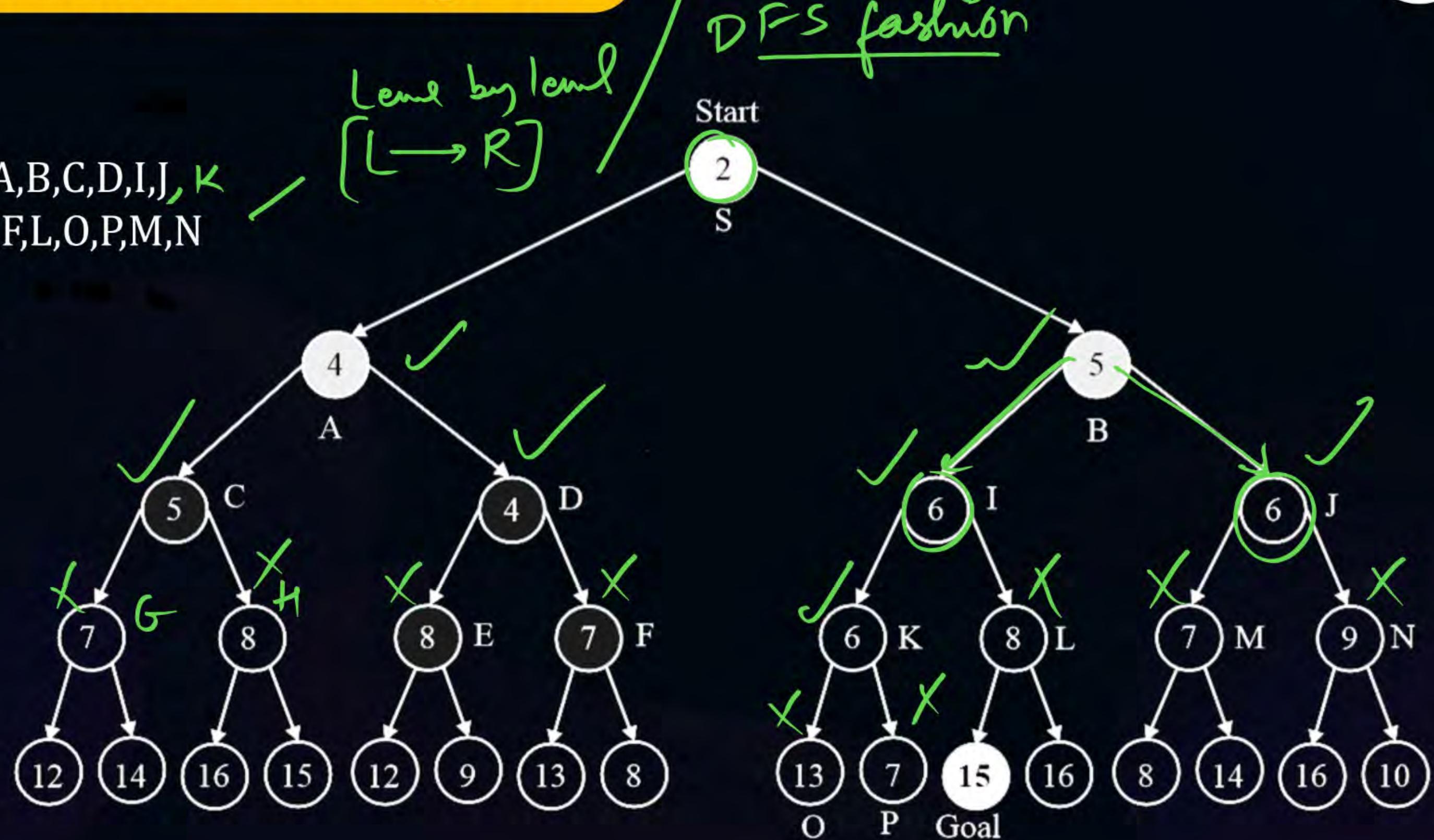
It: 4

$F_{th} = 6$

Close list: S,A,B,C,D,I,J,**K**

Prune G,H,E,F,L,O,P,M,N

$F_{th} = 7$





# Topic : Artificial Intelligence



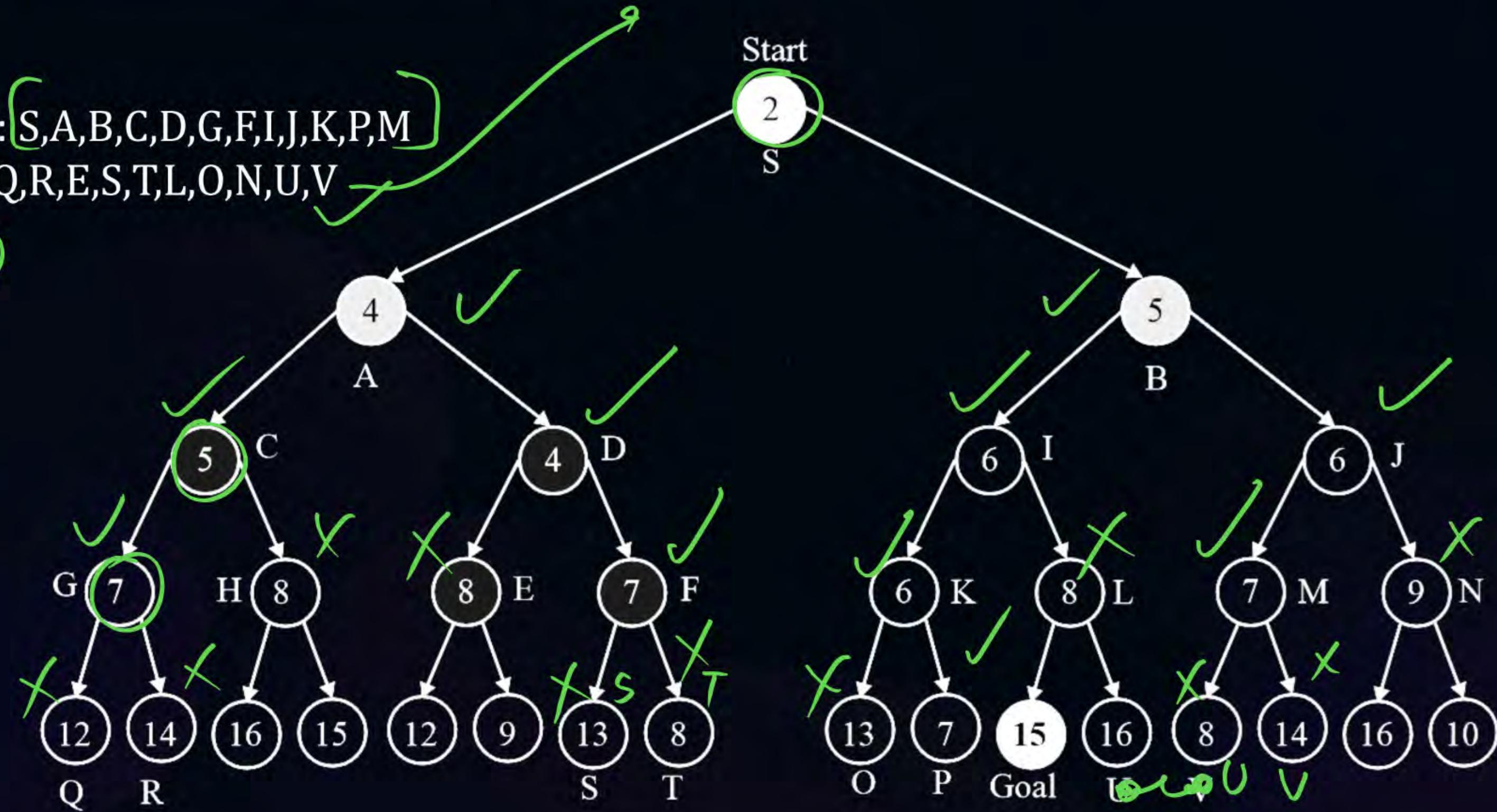
It: 5

$F_{th} = 7$

Close list: [S,A,B,C,D,G,F,I,J,K,P,M]

Prune H,Q,R,E,S,T,L,O,N,U,V

$F_{th} = 8$





# Topic : Artificial Intelligence

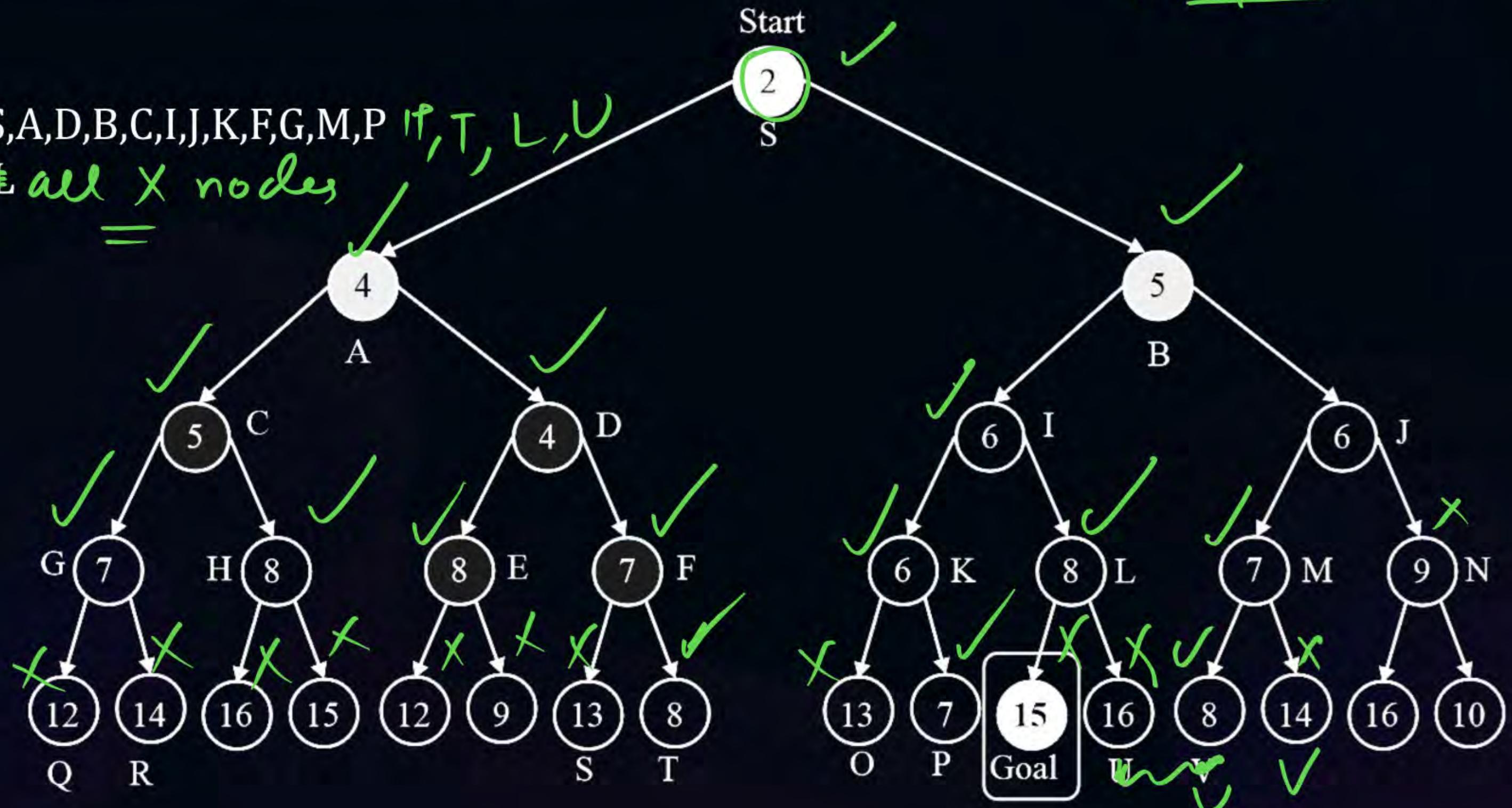


It: 6

$F_{th} = 8$

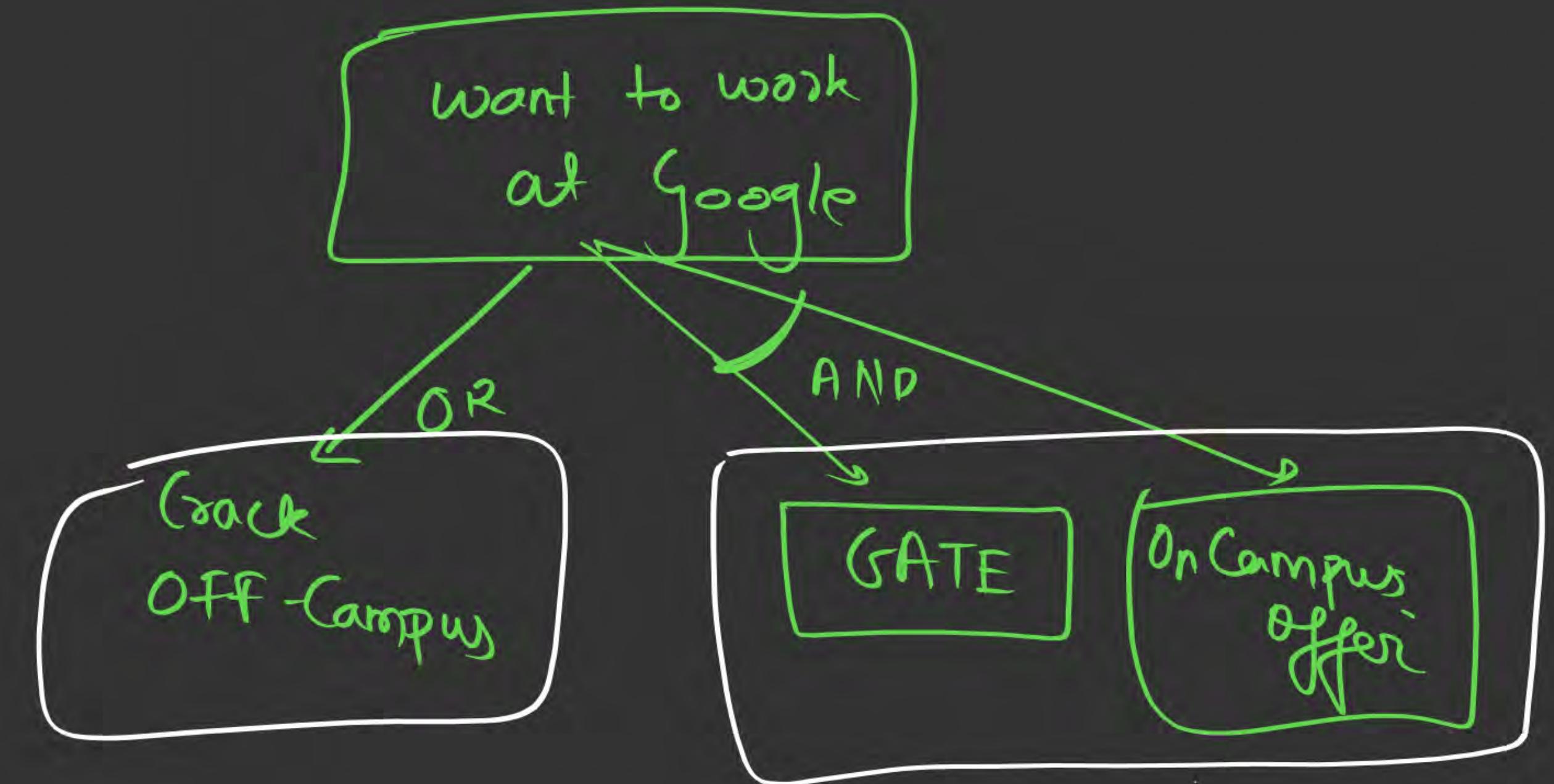
Close list: S,A,D,B,C,I,J,K,F,G,M,P ~~I<sup>P</sup>, T, L, U~~

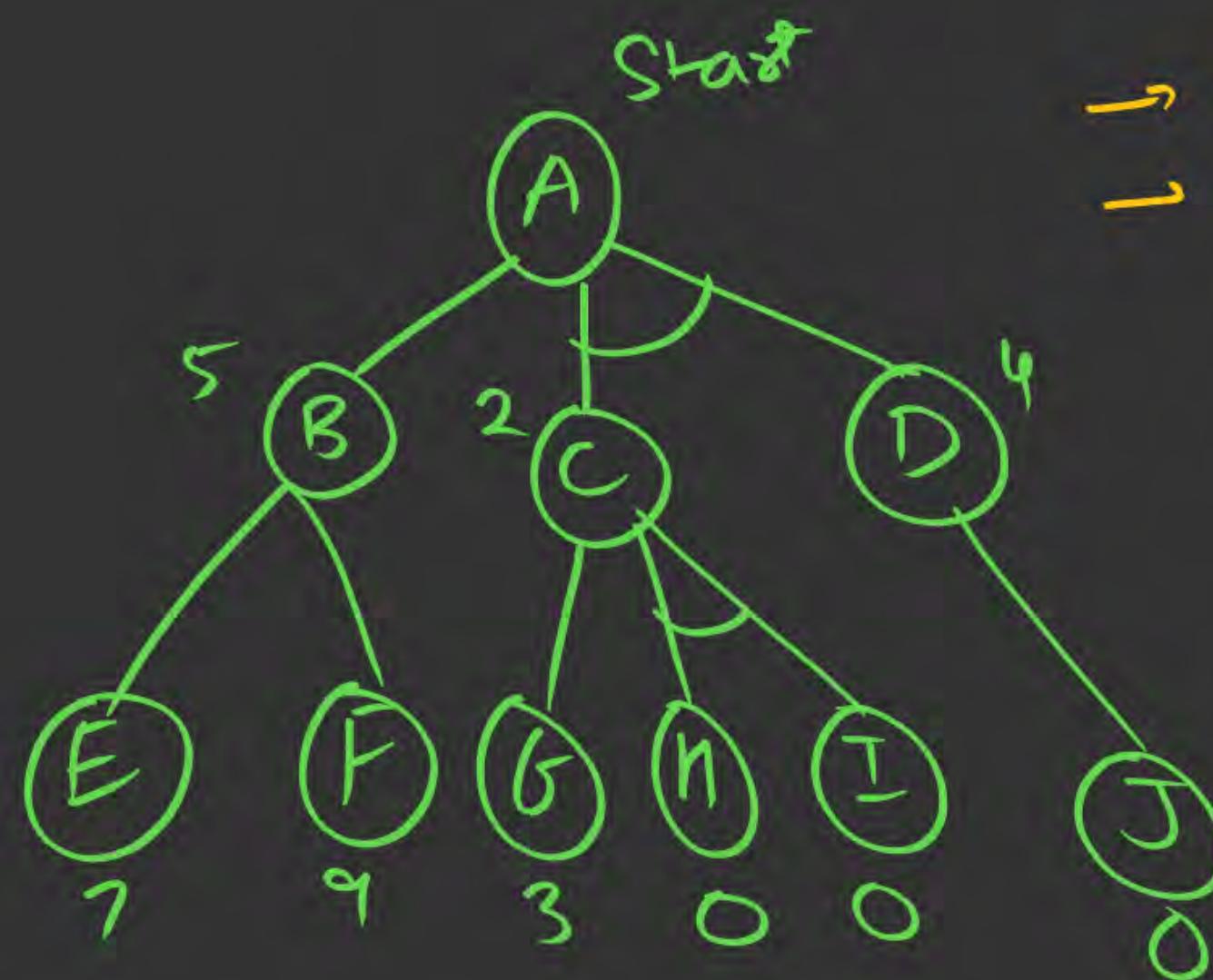
Prune E,H,L ~~all X nodes~~



$AO^*$  → "And-Or-Star"  
on DAG

preferred for AND-OR graphs  
Back-propagate updated Heuristics

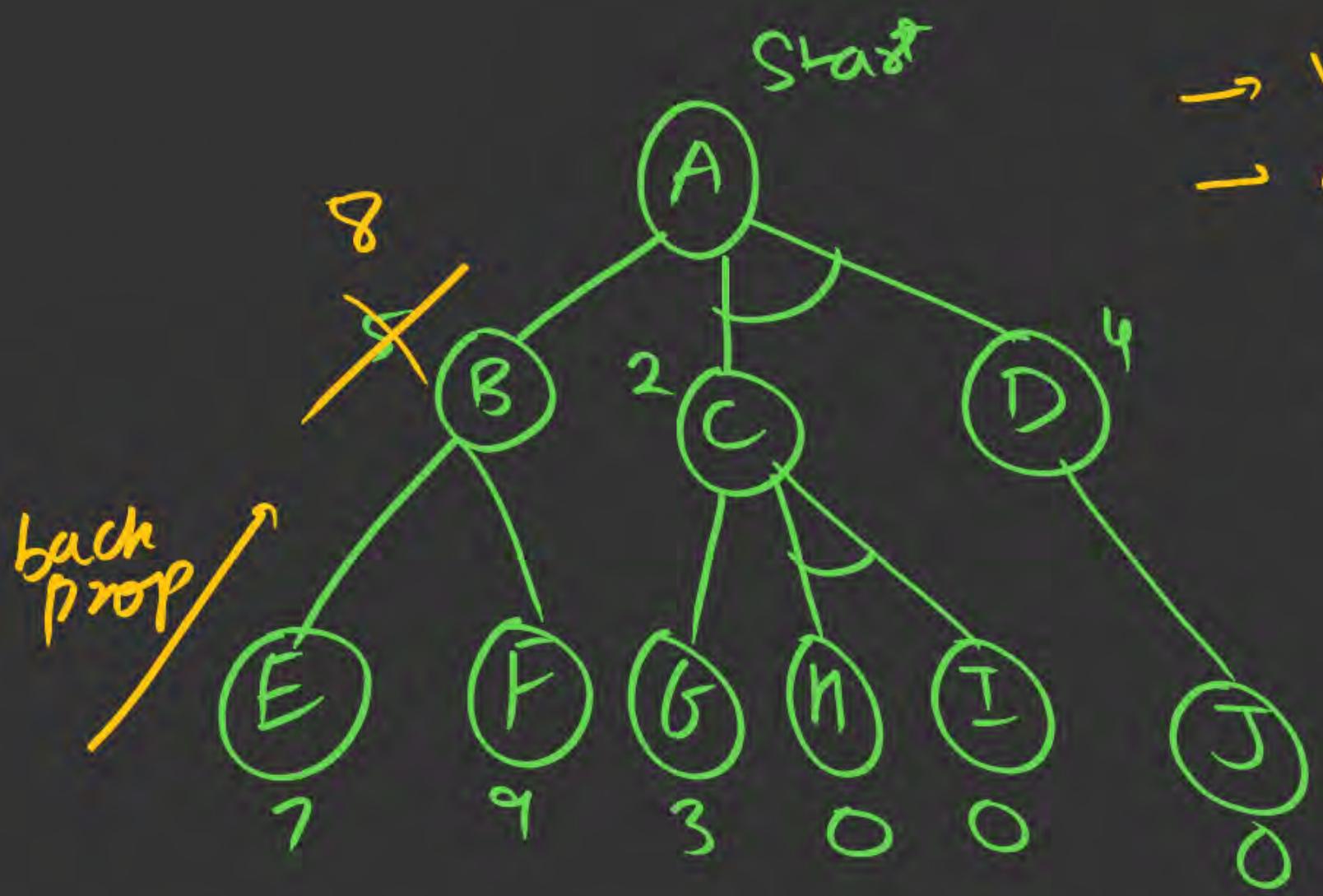


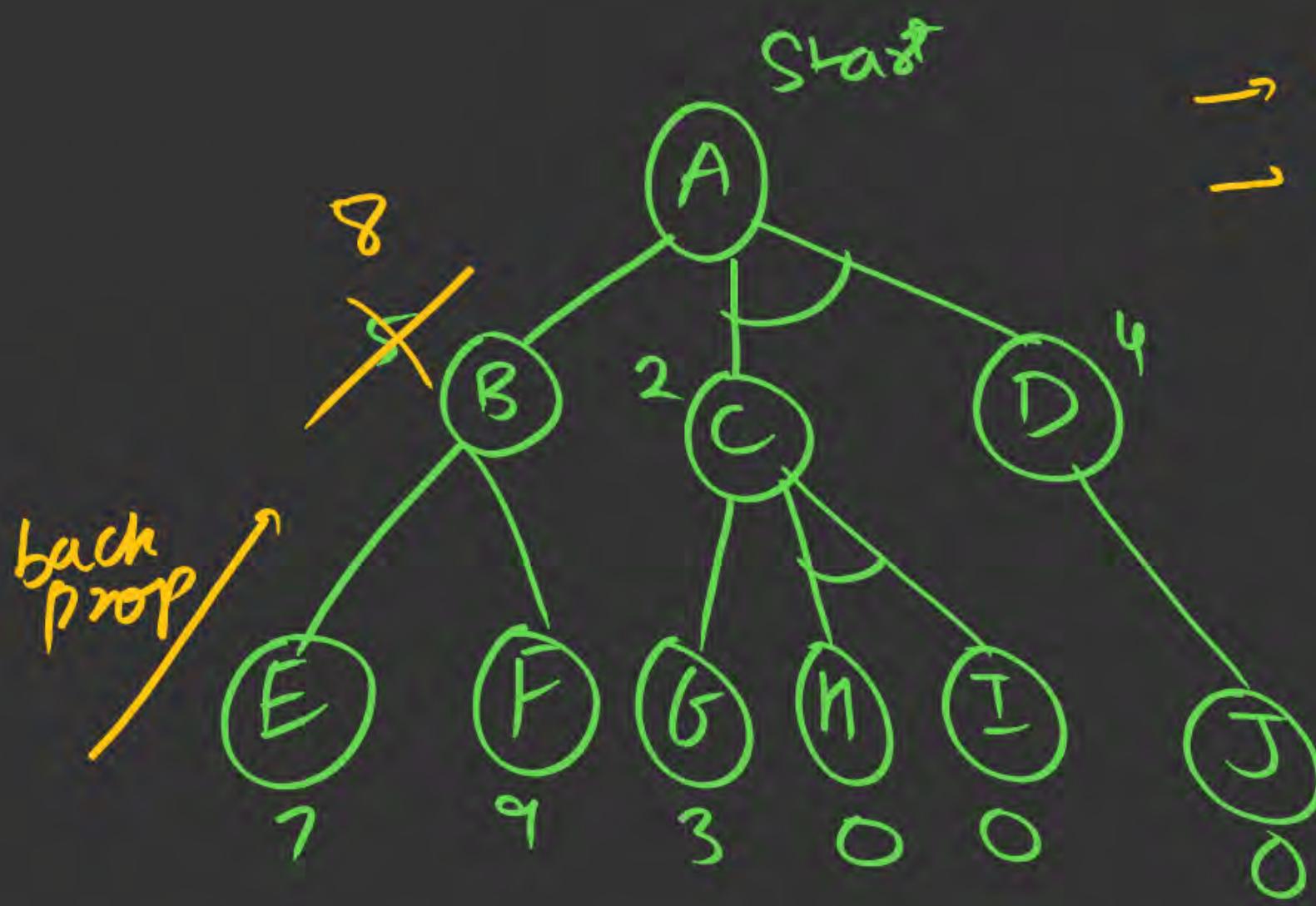


→ values on node → Heuristics  
 → edge costs = 1

Step 1: visit A.

- B →  $1+5=6 \checkmark$
- C and D →  $(1+2)+(1+4)=8$

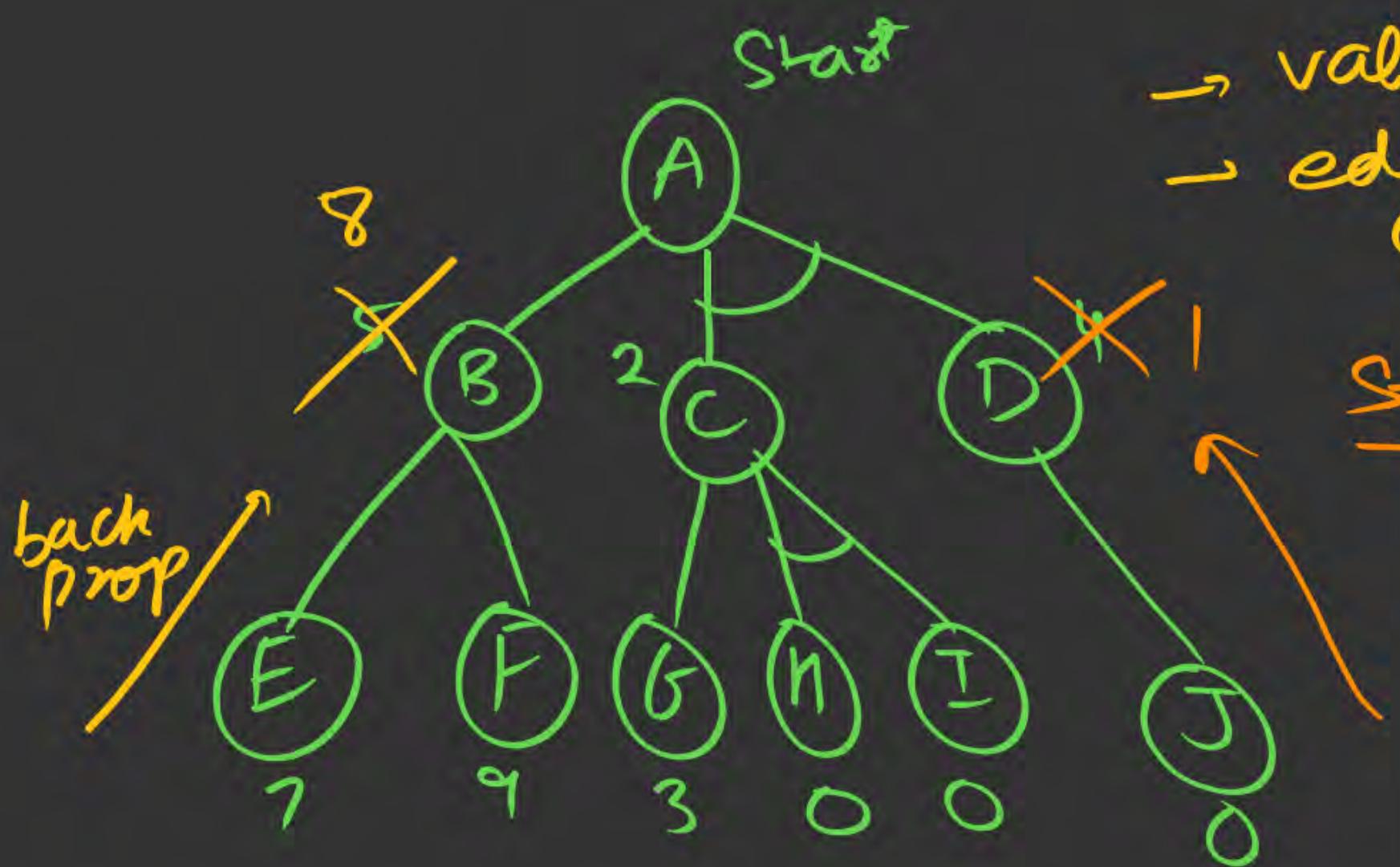




→ values on node → Heuristics  
 → edge costs = 1

Step 3: visit A. (with updated B)

- B →  $1+8=9$
- C and D →  $(1+2)+(1+4)=8$

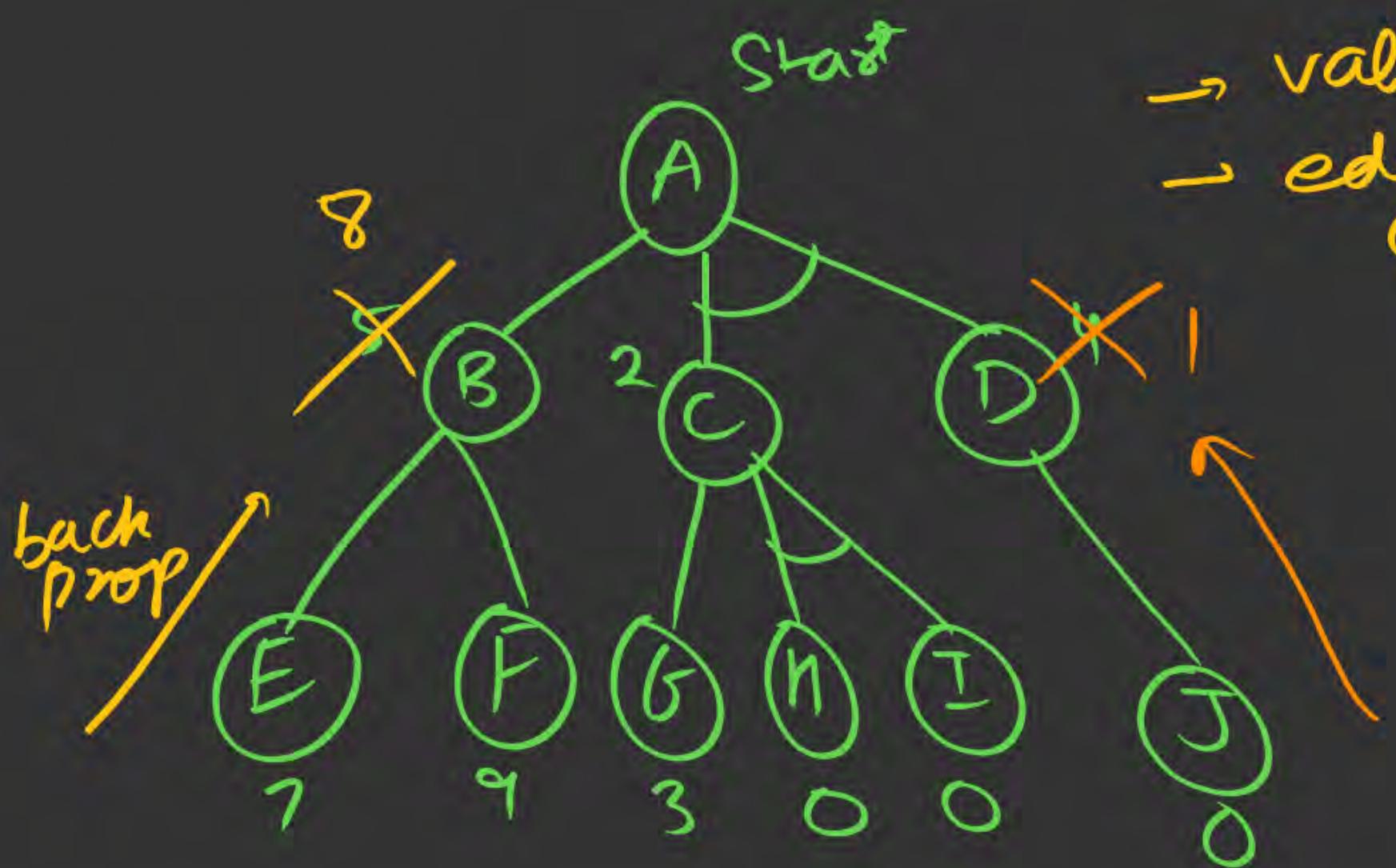


→ values on node → Heuristics  
 → edge costs = 1

Step 4: visit C,D

- C (after C)
  - ↳ G →  $(1+3)=4$
  - ↳ H and I →  $(1+0)+(1+0)=2$   
 $\Leftrightarrow \min(2,4)=2$

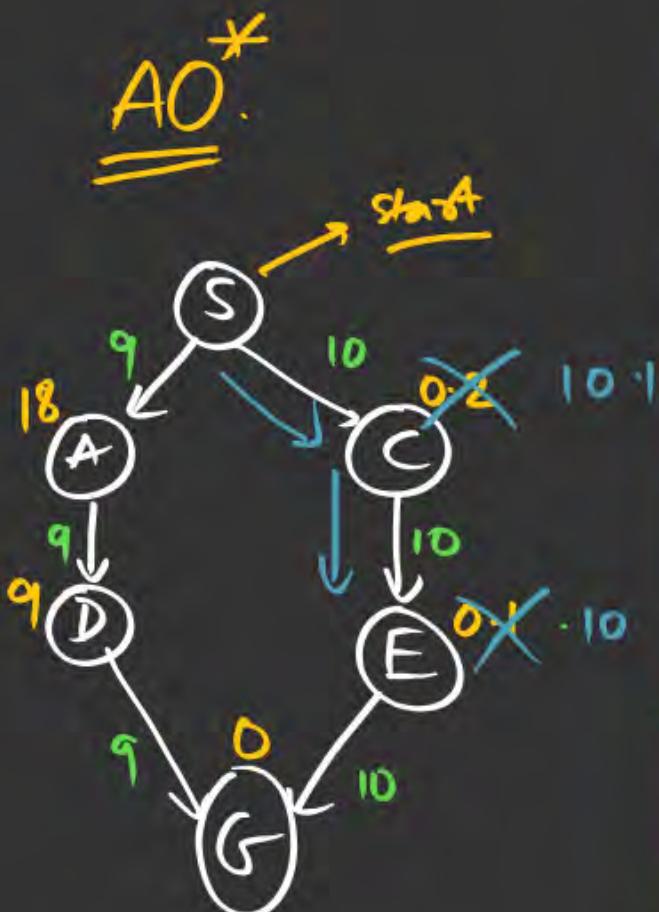
- D
  - ↳ J →  $(1+0)=1$   
 update  $h(D)=1$



→ values on node → Heuristics  
 → edge costs = 1

Step 5: Update A.

$$\begin{aligned}
 A &\rightarrow C \text{ and } D \quad \checkmark \\
 \hookrightarrow (1+2) + (1+1) \\
 = 3+2 &= \underline{\underline{5}}
 \end{aligned}$$



S1:

$$\begin{aligned} S \rightarrow \\ \hookrightarrow A = (9 + 18) = 27 \end{aligned}$$

$$\hookrightarrow C = (10 + 0 \cdot 2) = 10 \cdot 2$$

S2: visit C

$$\hookrightarrow E = (10 + 0 \cdot 1) = 10 \cdot 1$$

update  $h(C) = 10 \cdot 1$

S3: visit E

$$\hookrightarrow G: 10 + 0 = 10$$

update  $h(E) = 10$

S  $A \rightarrow 27$  ✓

$$C = (10 + 20) = \underline{\underline{30}}$$

↓

visit A.

$$D = (9 + 9) = 18$$

↓

visit D

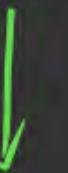
$$G = (9 + 0) = 9$$

↓

visit G → STOP

Both BFS & DFS discussed in Chap 1  
were graph search.

Graph Search

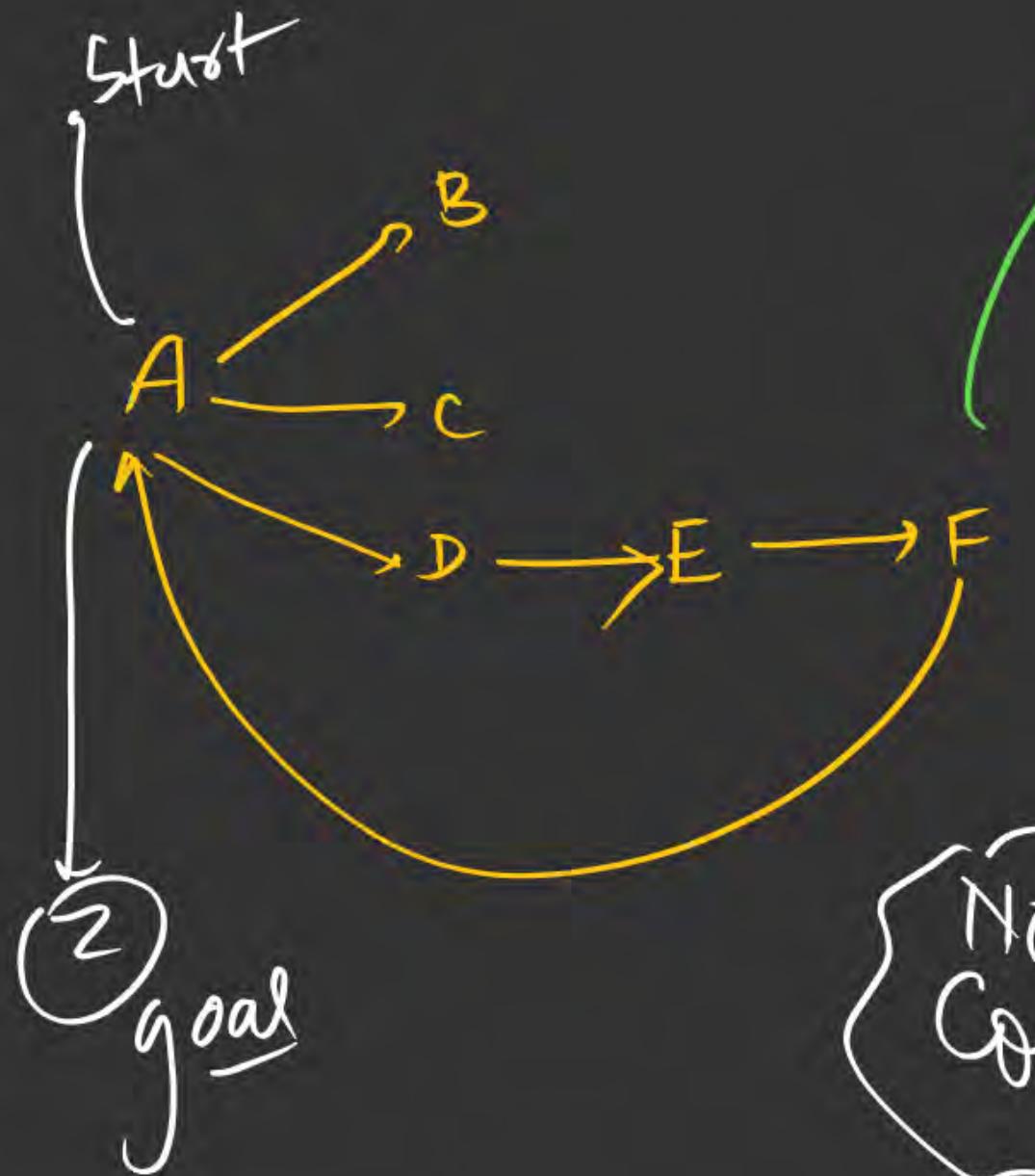


closed node  
not visited  
again

Tree Search



can visit closed  
node again



Alphabetical

① DFS Graph Search

A B C D E F

stop

② DFS Tree Search

A B C D E F A B C D . . .

NOT  
Complete

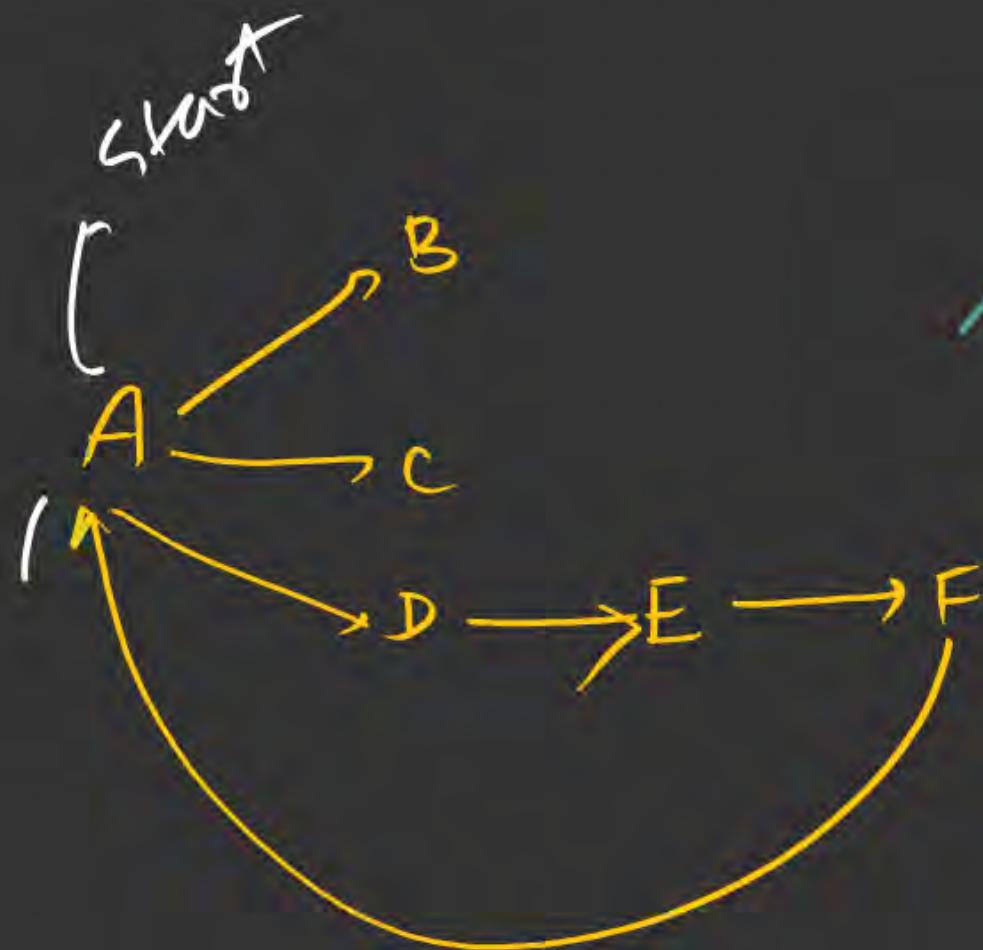
Repeating  
(DFS stuck in loop)

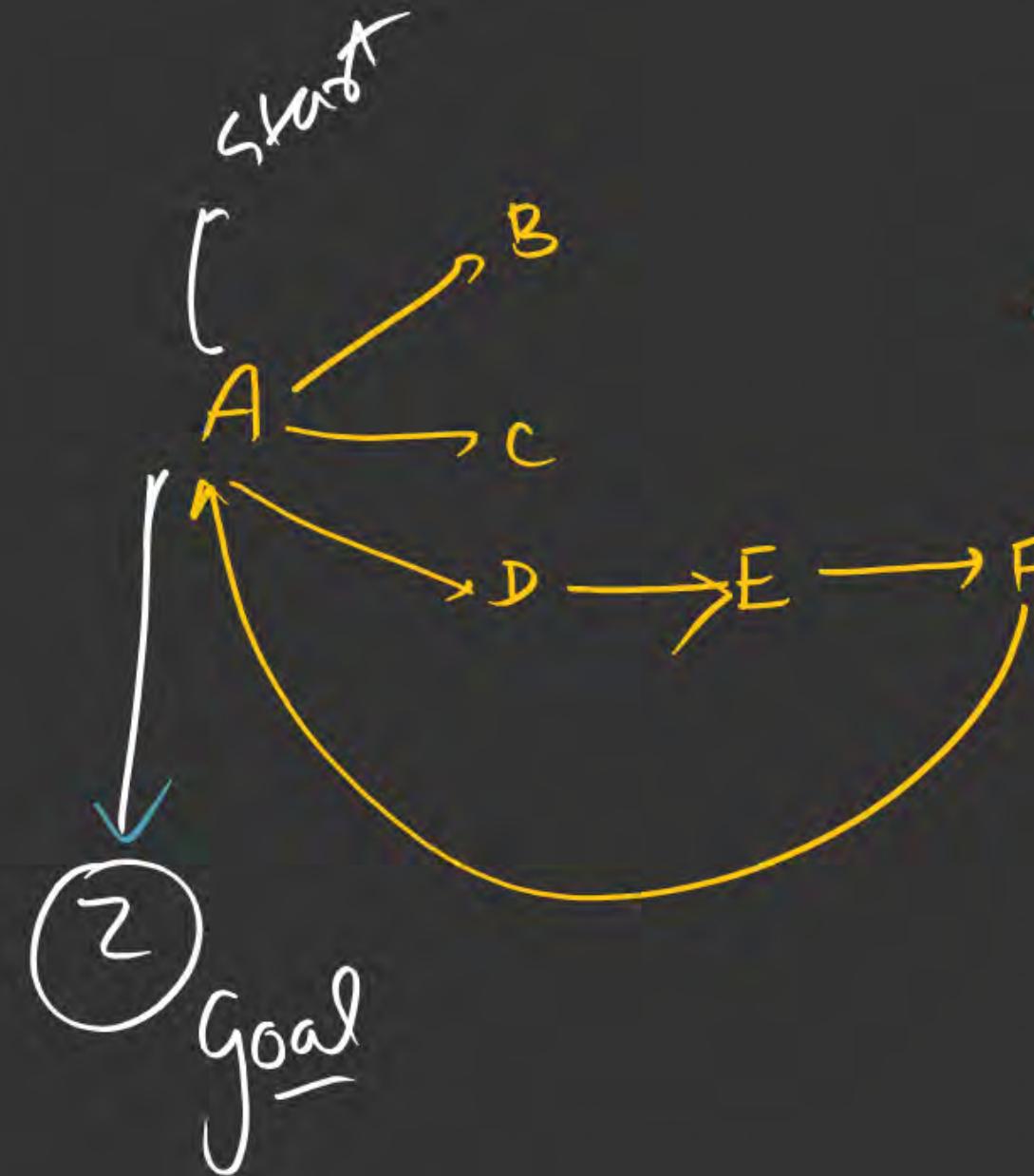
Alphabetical

① BFS Graph Search

Open [ A B C D E F ]

Closed [ A B C D E F ]  
STOP

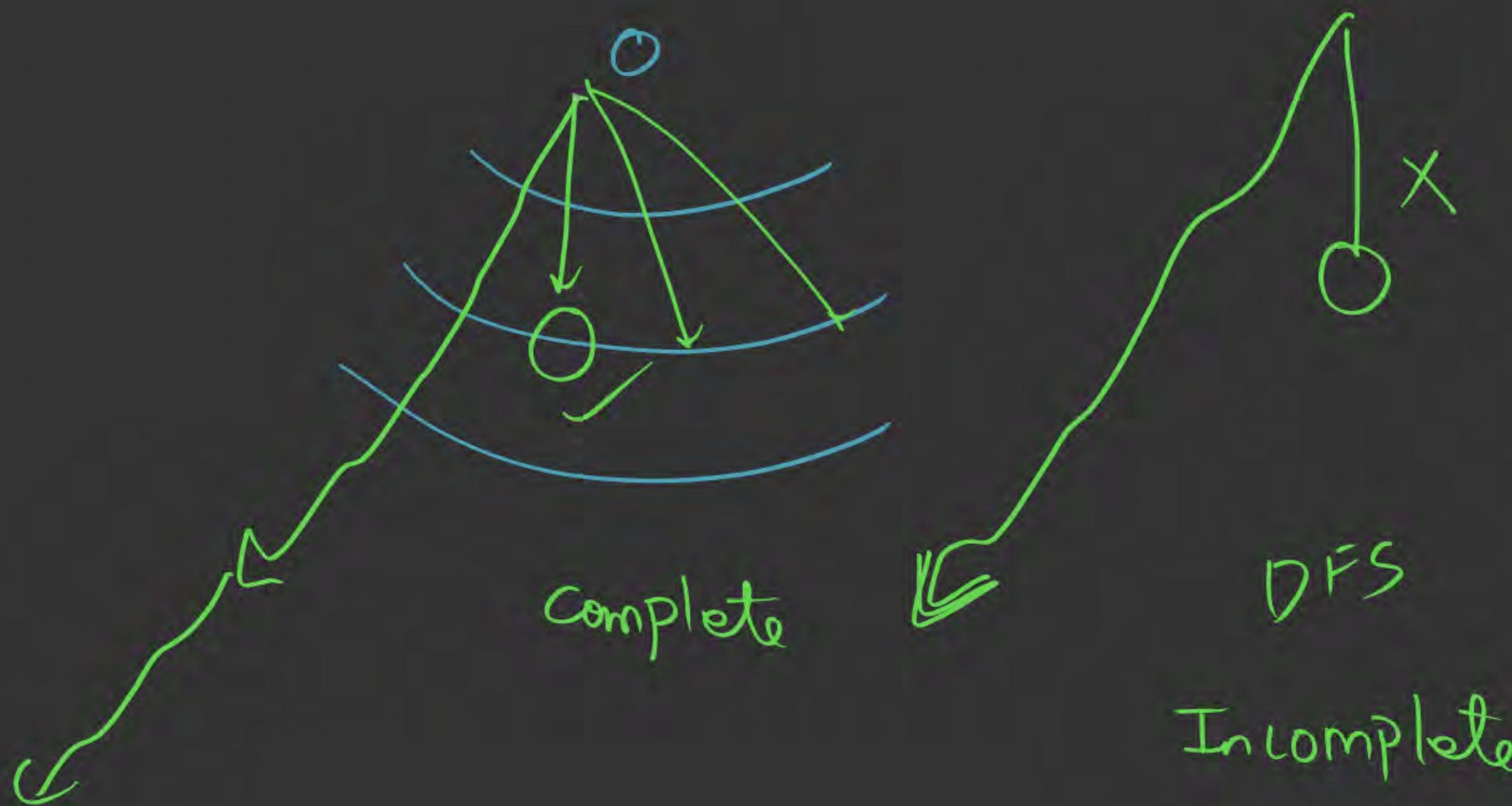




② BFS Tree Search



Alphabetical



#Q. Which of the following is/are true?

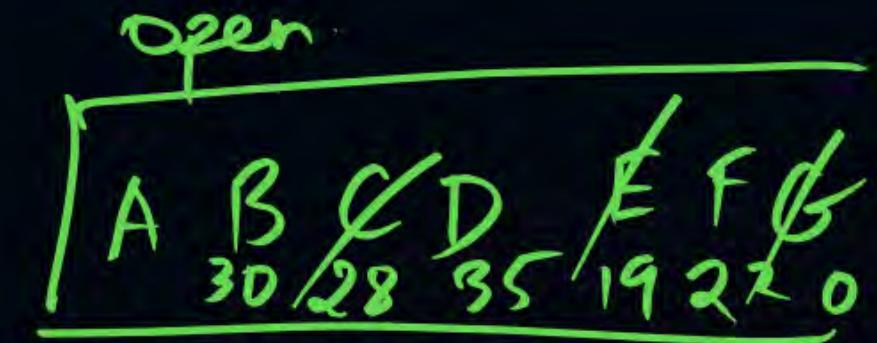


- A** Breadth First Search explores equally in all directions.
- B** <sup>UCS</sup> Dijkstra's Algorithm lets us prioritize which paths to explore.
- C** <sup>X</sup> A\* is a modification of Dijkstra's Algorithm that can find paths to all the nodes.
- D** <sup>/</sup> A\* is a modification of Dijkstra's Algorithm that is optimized for a single destination.

#Q. Consider the following graph use the best first search algorithm to find the path from node A to node G (Assume node A is start node and node G is goal node). Which of the following path sequence is correct?



GBFS



**A**

$A \rightarrow D \rightarrow C \rightarrow G$



$A \rightarrow C \rightarrow E \rightarrow G$

A C E G

**C**

$A \rightarrow C \rightarrow F \rightarrow H \rightarrow G$



$A \rightarrow D \rightarrow F \rightarrow G$



**THANK - YOU**



# DS & AI ENGINEERING



## Artificial Intelligence

### Adversarial Search

Lecture No.- 01

By- Aditya sir



# Recap of Previous Lecture



Topic



Informed Search

# Topics to be Covered



Topic

Topic

Adversarial Search





## Topic: Adversarial Search

### Adversarial Search:

- Game play Search.
- 2 player Game
- 1st player want to maximize Score
- Second player want to minimize Score
- We assume that both the players play optimally \*

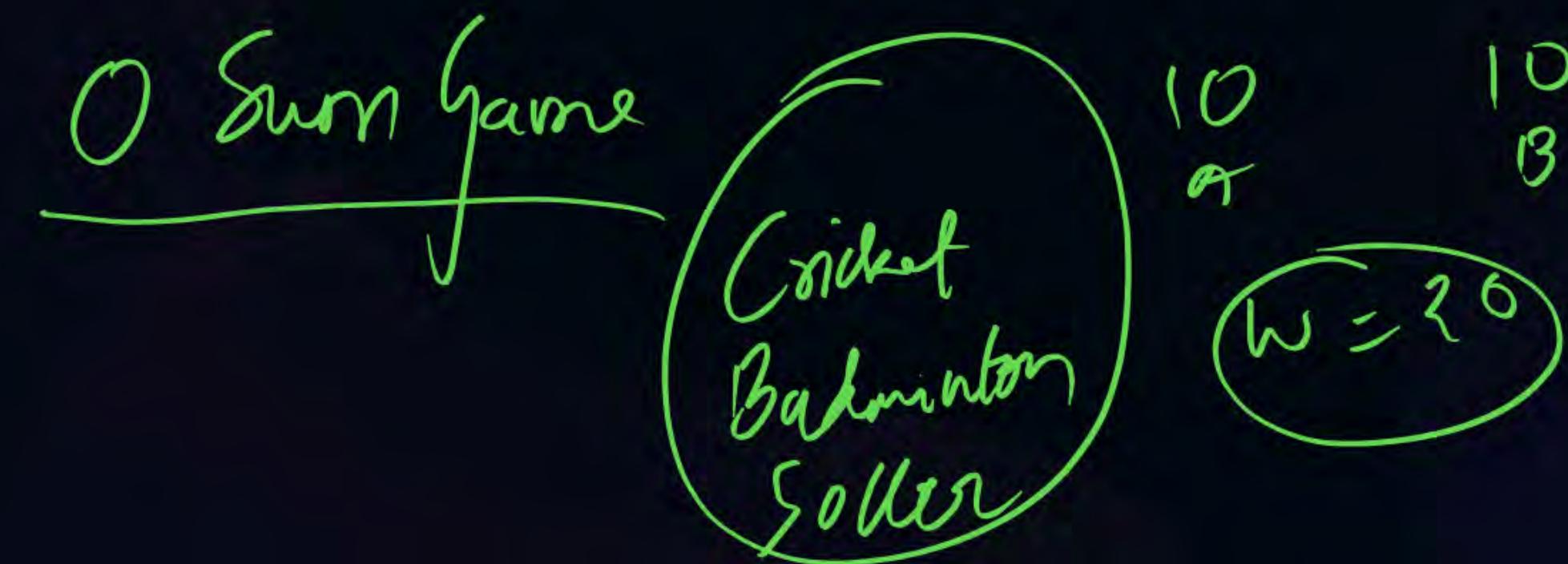
So, here a graph is given and we simply have to find the moves for max/min player to get best results.



## Topic: Adversarial Search

### Adversarial Search

- **Adversarial Search:** A search in scenarios where multiple agents (players) have **conflicting** objectives, and each agent's success is often measured against the other's failure. Unlike traditional search problems (like pathfinding), adversarial search considers the presence of an opponent actively working **against** the agent's goal.





## Topic: Adversarial Search



### Minimax Algorithm

- The Minimax algorithm is a recursive, decision-making algorithm used in two-player games, particularly zero-sum games where one player's gain is another player's loss. It systematically explores the game tree to determine the optimal move for a player, assuming that the opponent also plays optimally.
- It is a specialized Search Algo that returns optimal Sequence move for a player in a Zero-Sum Game.
- Recursive/ Back tracking algo which is used in decision making and Game theory and uses recursion to Search through Game Tree.
- Algo Computes minimax decision for current state.
- We have 2 players: max and min player. Max player select the maximum value and min player select the minimum value.
- Depth-First Search Algo is used for exploration of Complete Game Tree.
- Used in Chess, Tic Tac Toe and other 2 player Games

DFS



## Topic: Adversarial Search

Step

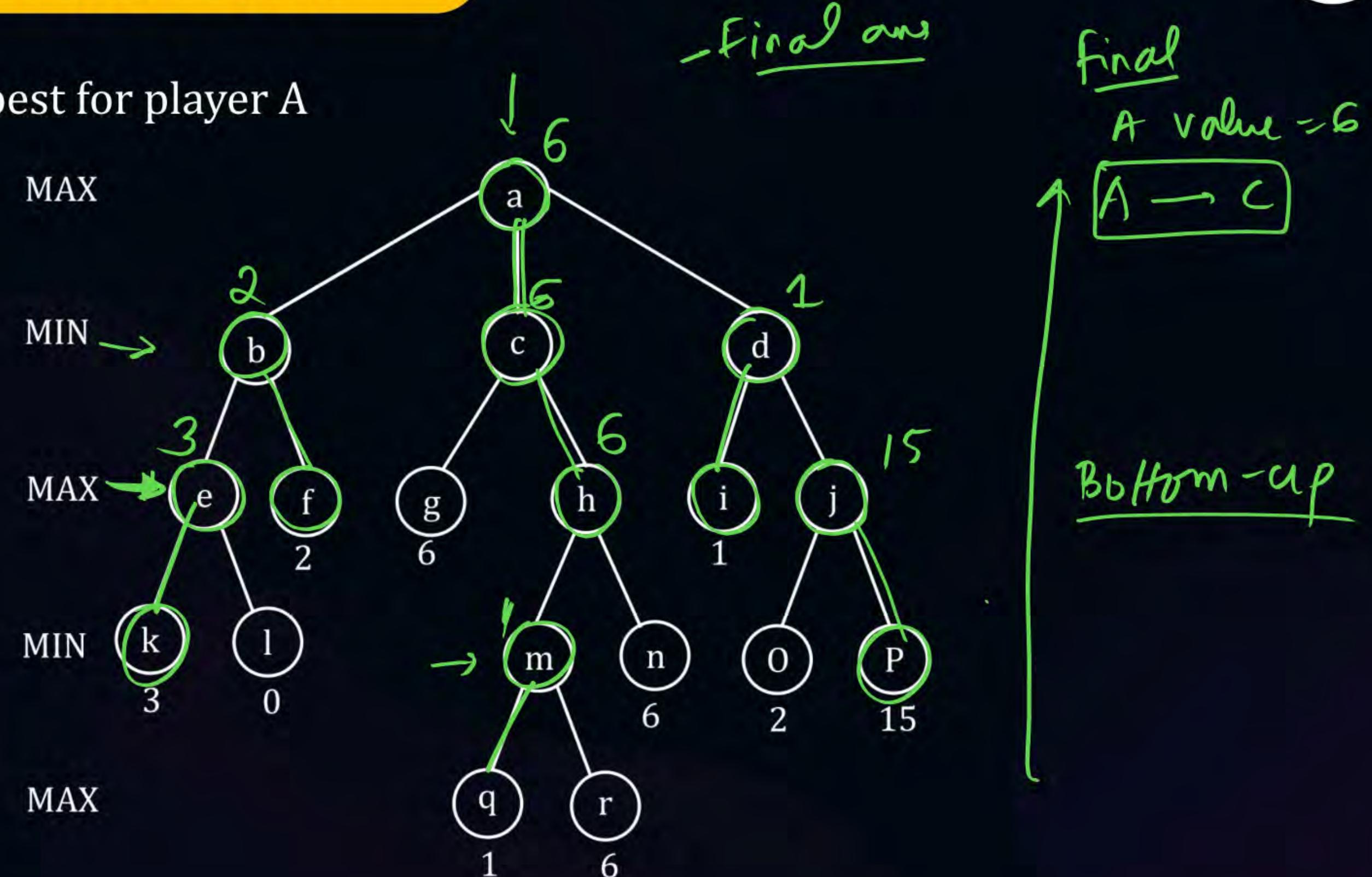
### Minimax Algorithm

- Initial values for Max and Min player: + infinity and - Infinity
- This is the backtracking algorithm
- Complete and optimal solution
- Time and Space complexity
- In Minimax algorithm we use DFS



## Topic: Adversarial Search

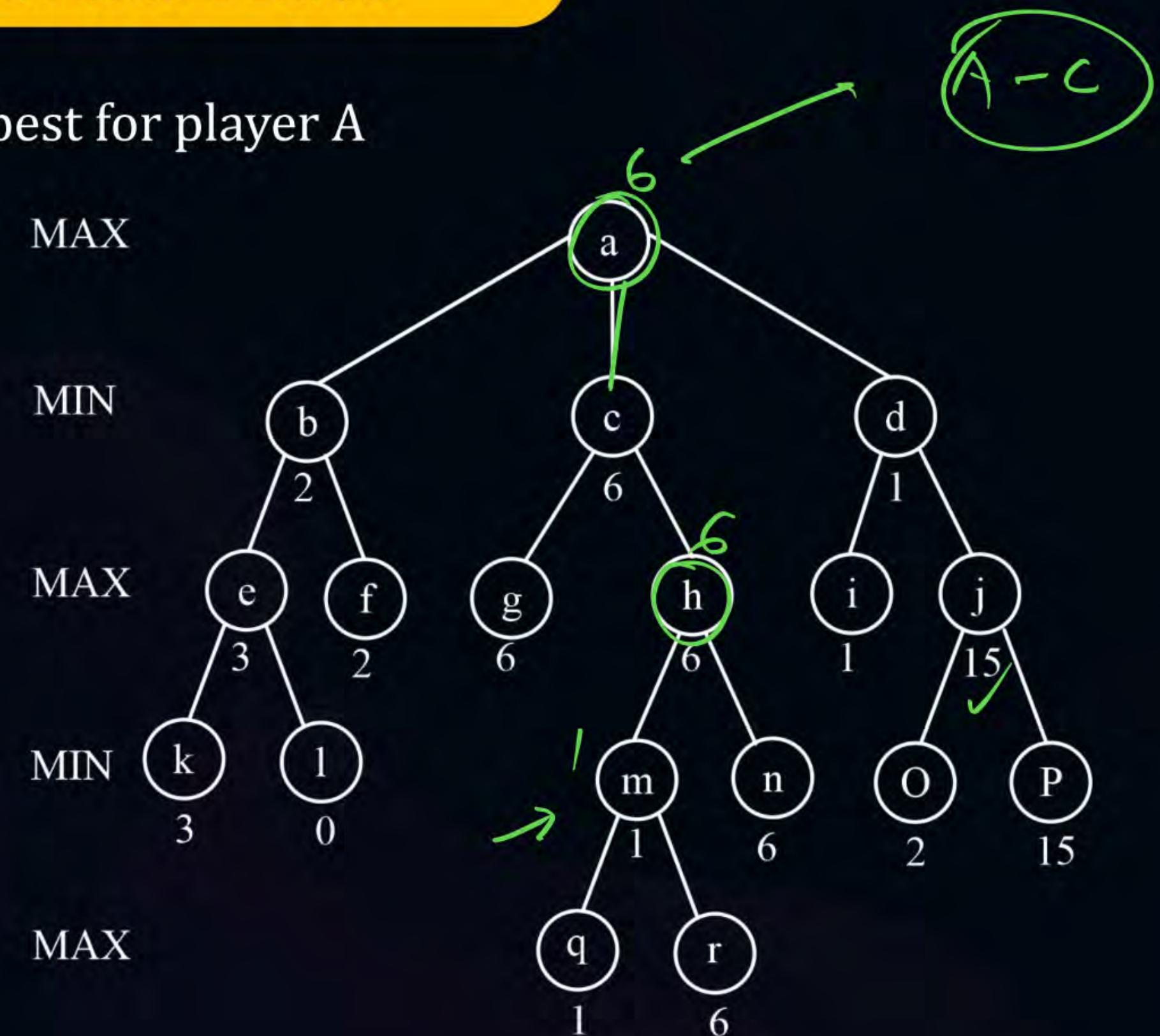
- Find the move best for player A





## Topic: Adversarial Search

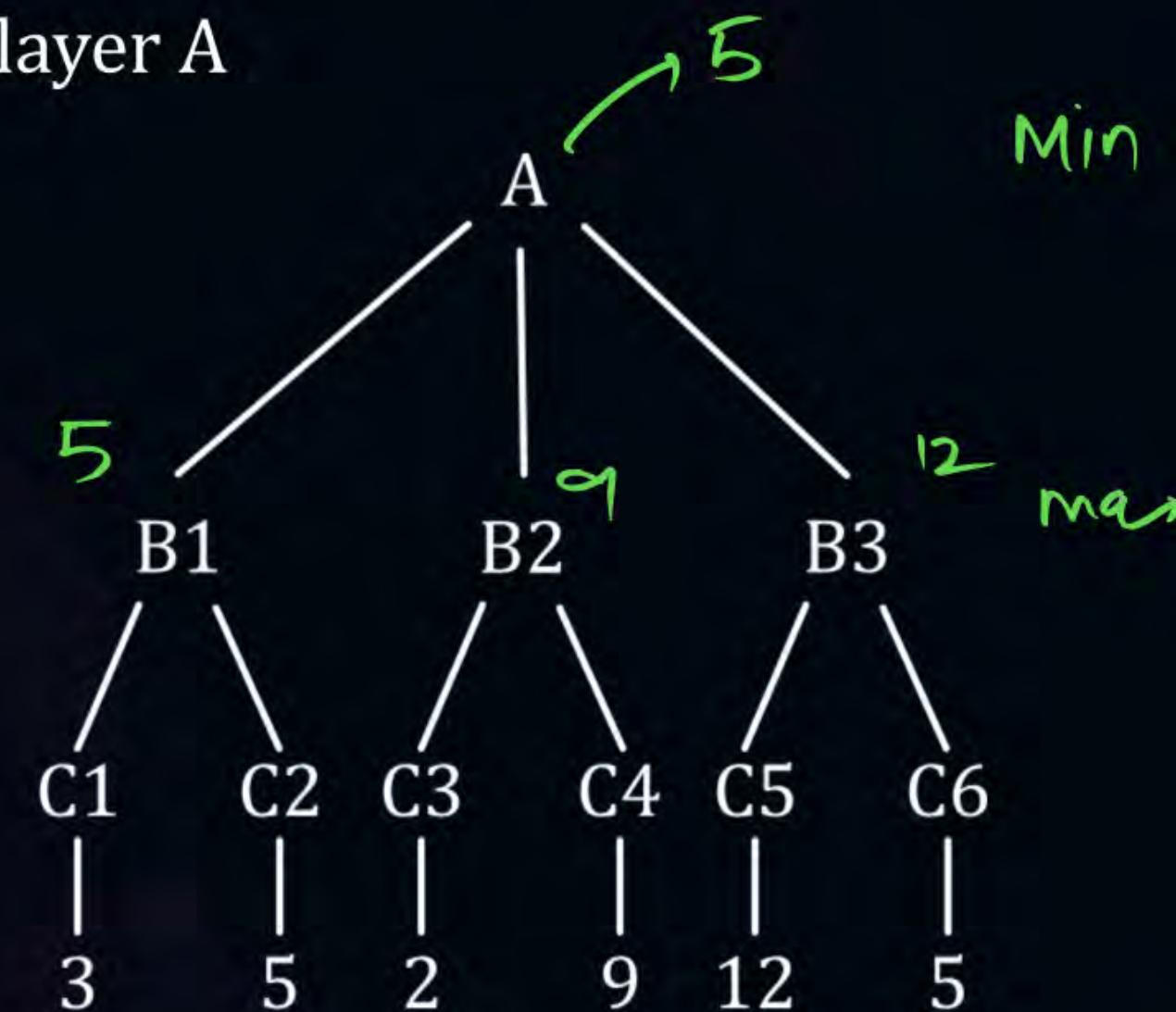
- Find the move best for player A





## Topic: Adversarial Search

- Find the move best for player A

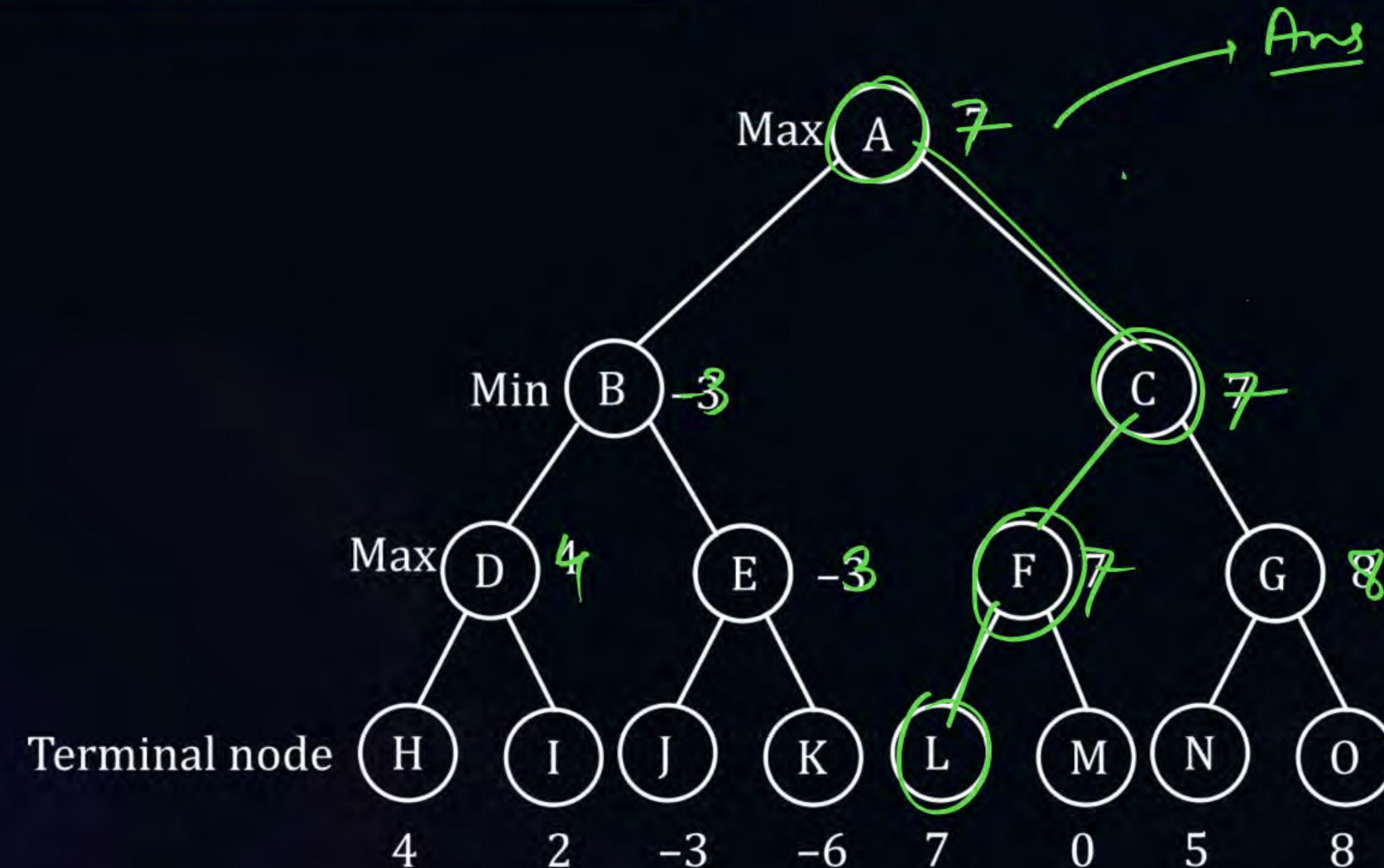


$A \Rightarrow 5$

Decision:  $(A - B1)$



## Topic: Adversarial Search

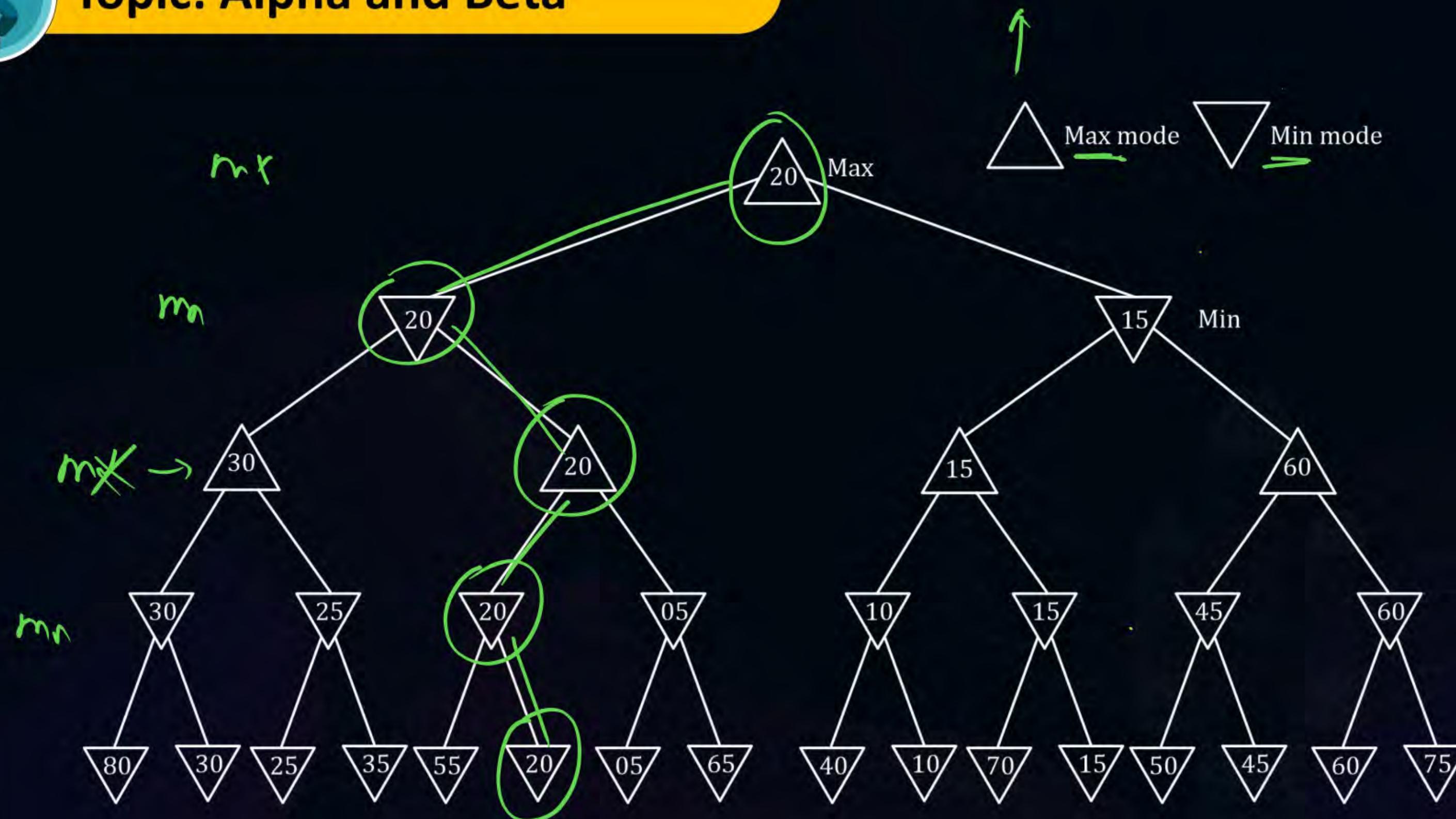


max node = 7, best move A → C → F → L



## Topic: Alpha and Beta

P  
W





## Topic: Alpha and Beta

### Properties of Minimax

Complete?

Yes (if tree is finite)

Optimal?

- Yes (against an optimal opponent) ✅
- No (does not exploit opponent weakness against suboptimal opponent)

Time complexity?

$O(b^m)$

$m \rightarrow \text{depth}$

Space complexity?

$O(b^m)$  (depth-first exploration)



## Topic: Alpha and Beta

- If we search the normal Game tree it becomes very Complex because of huge time and space complexity.
- Because here we have to traverse the whole tree to find best move.
- A Method called alpha-beta pruning, to solve this Game tree.





## Topic: Alpha and Beta



### Some Basic Rules

1. Start  $\alpha = -\infty, \beta = \infty$ .
2.  $\alpha \leftarrow$  update @ the max Nodes.
3.  $\beta \leftarrow$  update @ the min Nodes.
4.  $\alpha \geq \beta \Rightarrow$  **Pruning** i.e. whenever we get  $\alpha \geq \beta$  then that portion of tree is pruned because it will not effect the final ans.

dmP

DPS fashion



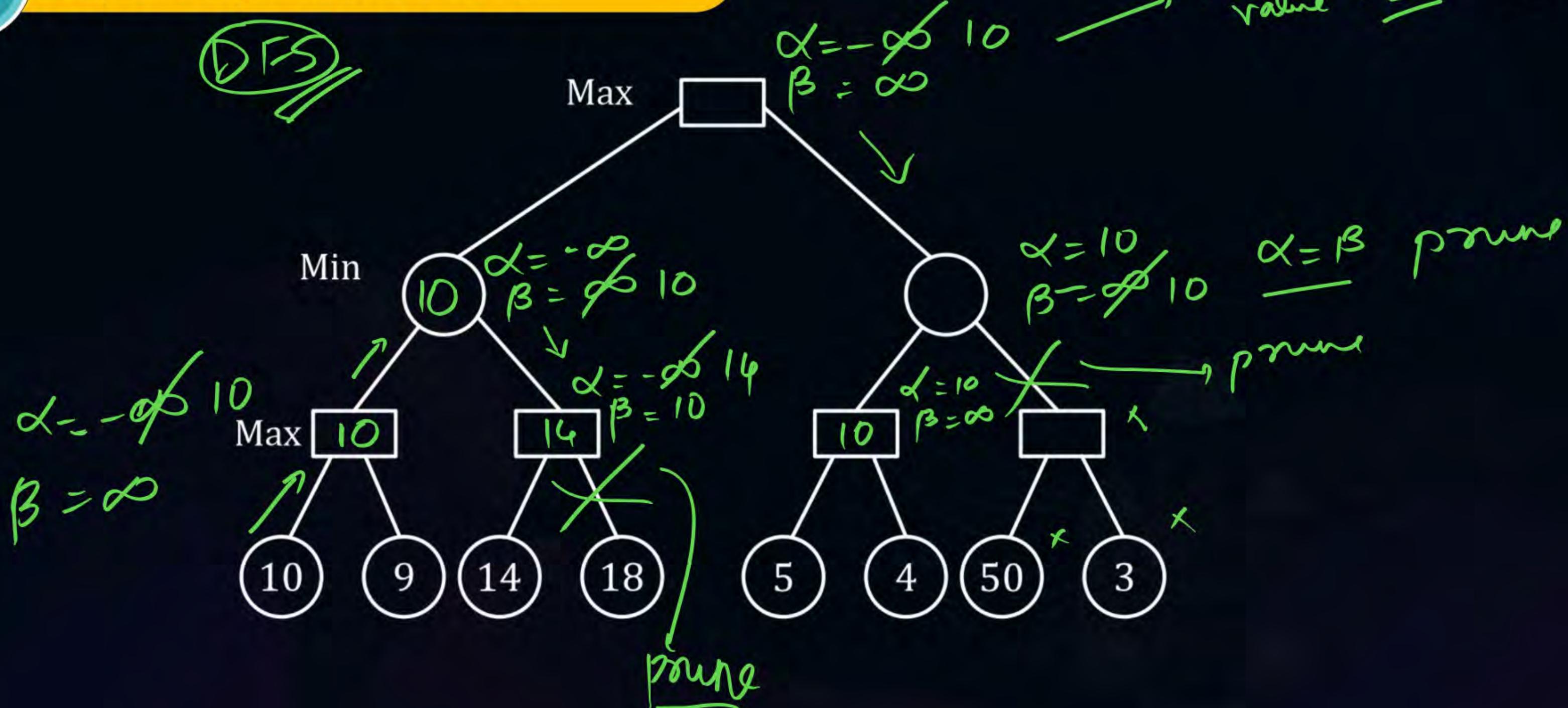
# Topic: Alpha and Beta

The logo consists of a stylized lowercase 'p' positioned above a lowercase 'w', all contained within a circular border.

ans

$$\text{Final value} = \underline{\underline{10}}$$

The logo consists of the letters 'DFS' in a stylized, italicized font, enclosed within a thick oval border.

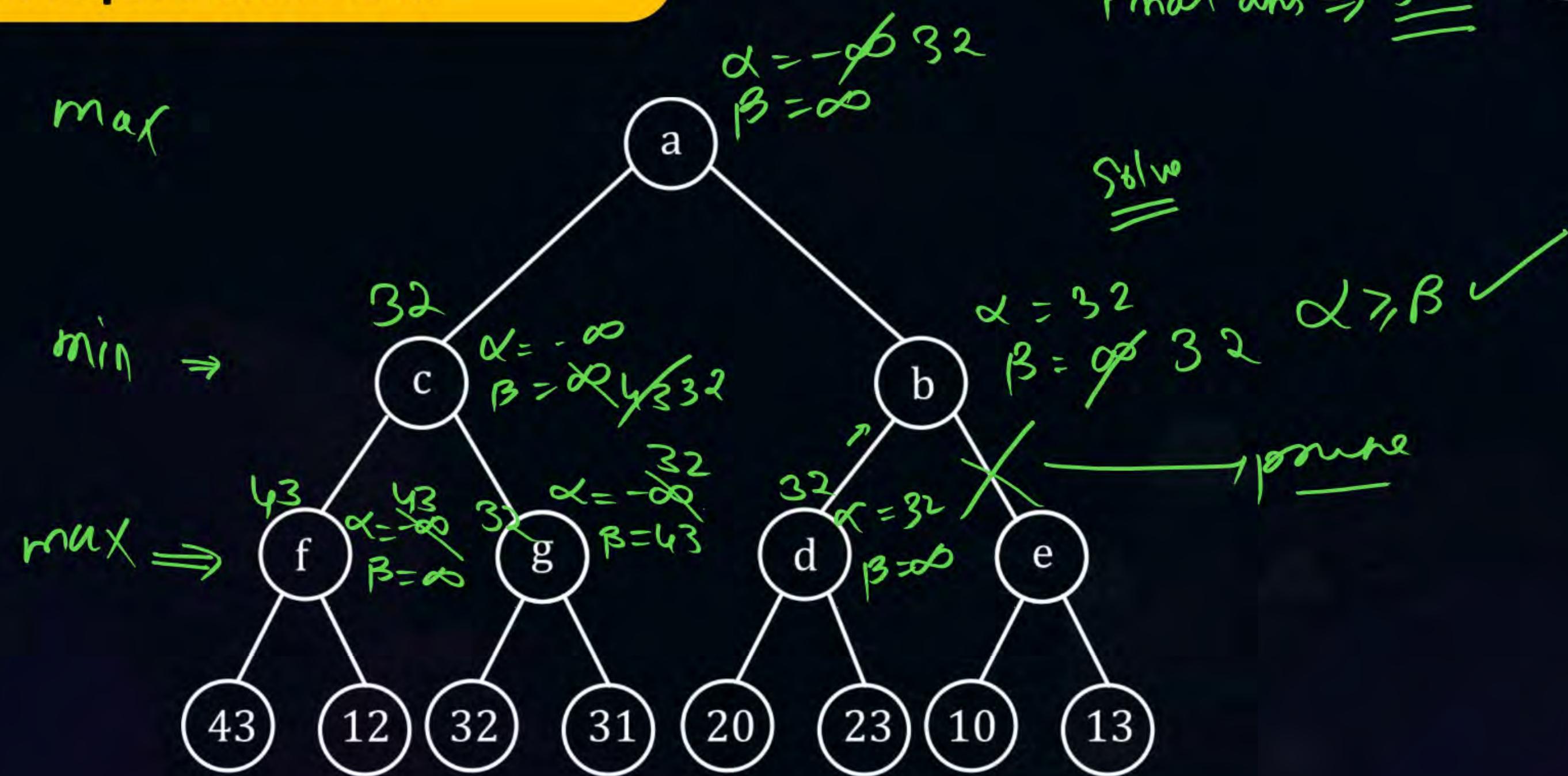




## Topic: Alpha and Beta

P  
W

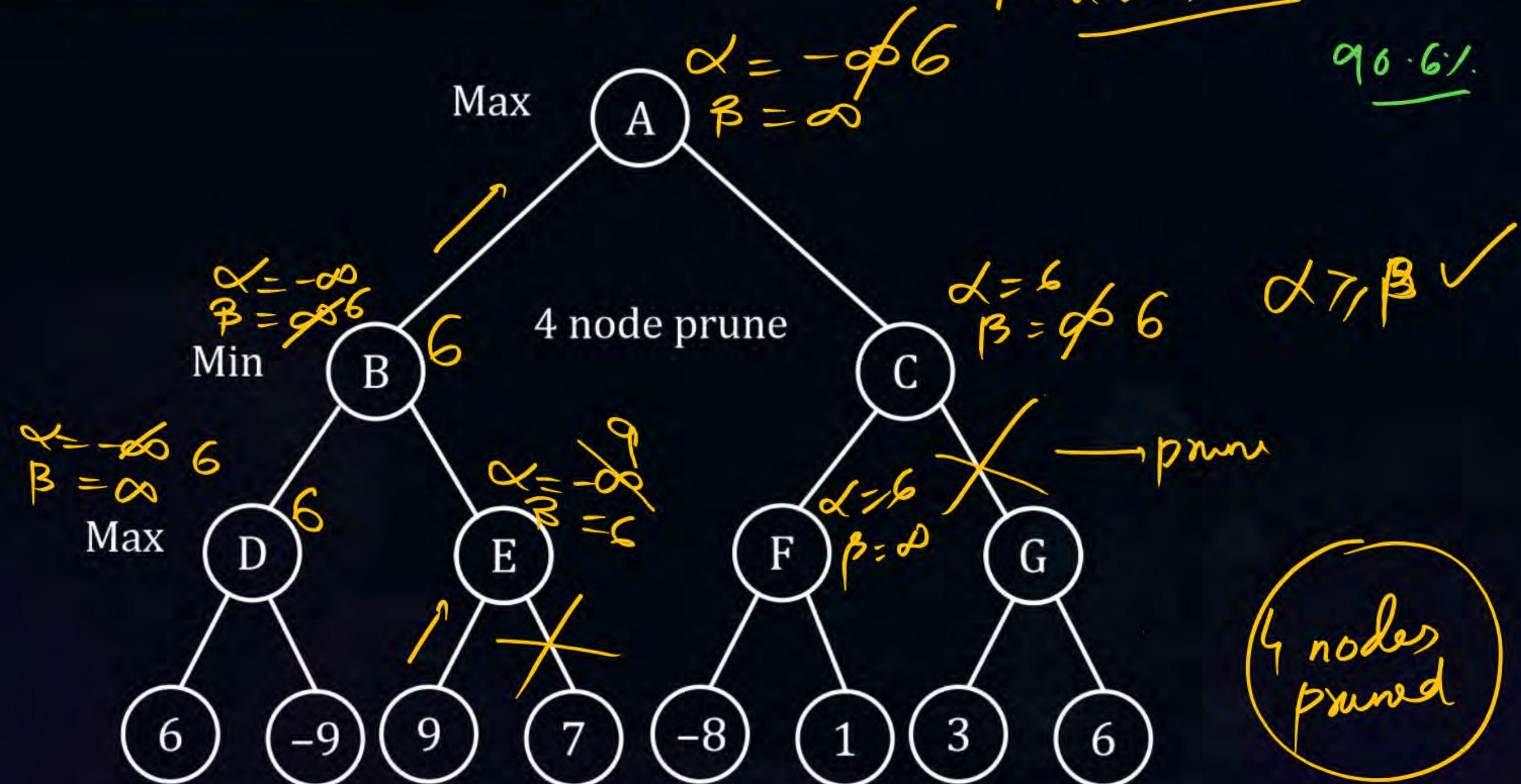
Final ans  $\Rightarrow \underline{\underline{32}}$





## Topic: Alpha and Beta

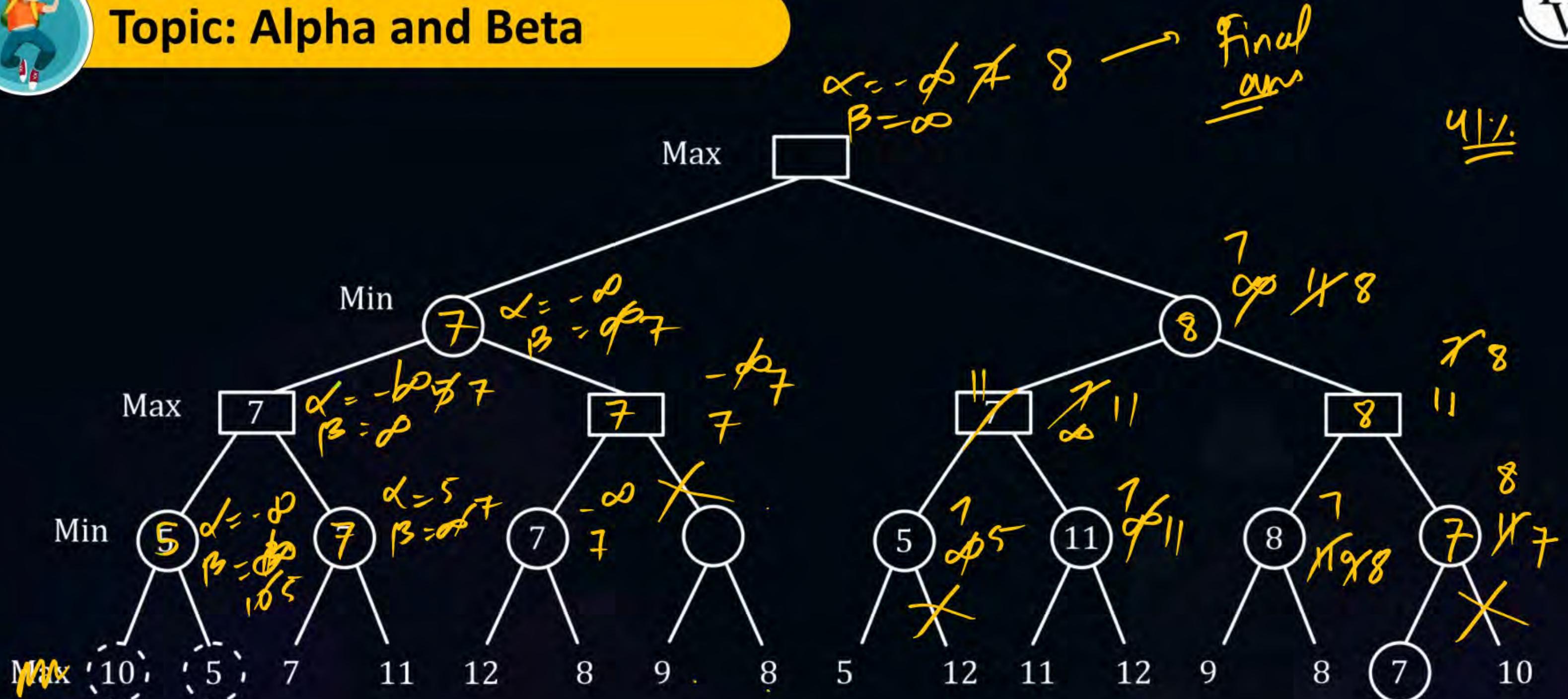
P  
W





## Topic: Alpha and Beta

P  
W



node prune final answer ~~411~~

5 nodes pruned



THANK - YOU



# DS & AI ENGINEERING

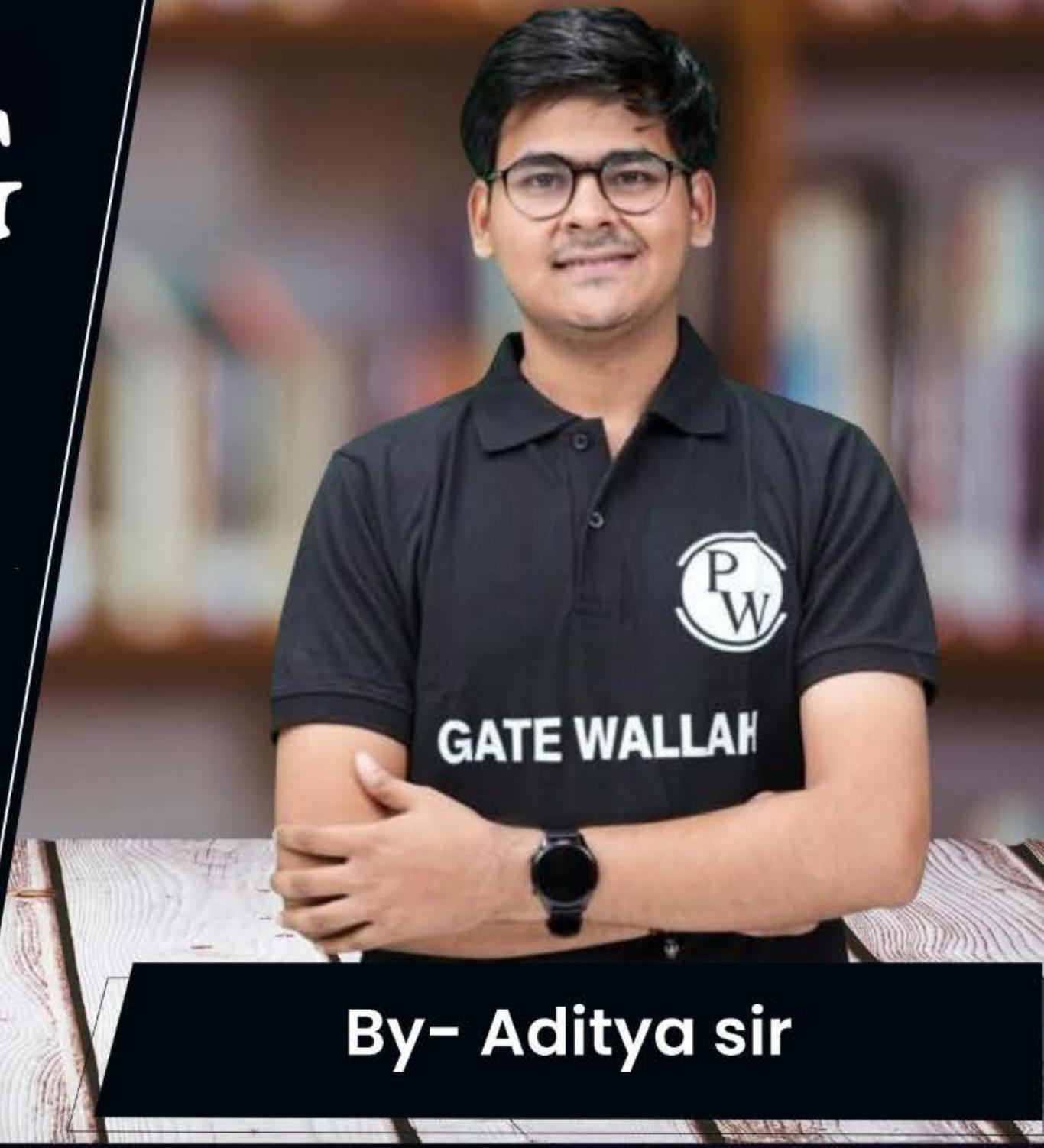


## Artificial Intelligence

### Adversarial Search

Lecture No.- 02

By- Aditya sir



# Recap of Previous Lecture



Topic

Topic

Topic

Topic

Intro to Adversarial Search.

Minimax

# Topics to be Covered



Topic

Topic

Topic

Questions on Minimax

$\alpha$ - $\beta$  Pruning



## About Aditya Jain sir

1. Appeared for GATE during BTech and secured AIR 60 in GATE in very first attempt - City topper
2. Represented college as the first Google DSC Ambassador.
3. The only student from the batch to secure an internship at Amazon. (9+ CGPA)
4. Had offer from IIT Bombay and IISc Bangalore to join the Masters program
5. Joined IIT Bombay for my 2 year Masters program, specialization in Data Science
6. Published multiple research papers in well known conferences along with the team
7. Received the prestigious excellence in Research award from IIT Bombay for my Masters thesis in ML
8. Completed my Masters with an overall GPA of 9.36/10
9. Joined Dream11 as a Data Scientist
10. Have mentored 15,000+ students & working professionals in field of Data Science and Analytics
11. Have been mentoring & teaching GATE aspirants to secure a great rank in limited time
12. Have got around 27.5K followers on Linkedin where I share my insights and guide students and professionals.



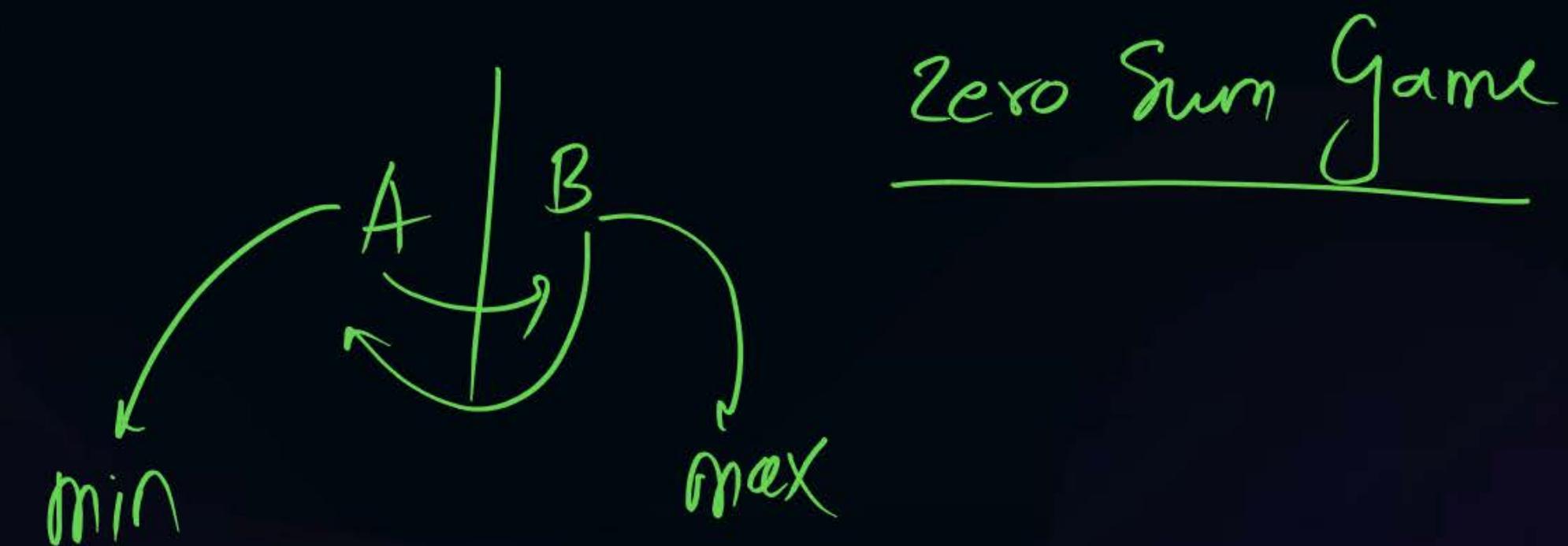


**Telegram Link for Aditya Jain sir:**  
**[https://t.me/AdityaSir\\_PW](https://t.me/AdityaSir_PW)**



## Topic: Adversarial Search

**Adversarial Search:** A search in scenarios where multiple agents (players) have conflicting objectives, and each agent's success is often measured against the other's failure. Unlike traditional search problems (like pathfinding), adversarial search considers the presence of an opponent actively working against the agent's goal.





## Topic: Minimax Algorithm

- The Minimax algorithm is a recursive, decision-making algorithm used in two-player games, particularly zero-sum games where one player's gain is another player's loss. It systematically explores the game tree to determine the optimal move for a player, assuming that the opponent also plays optimally.
- It is a specialized Search Algo that returns optimal Sequence move for a player in a Zero-Sum Game.  
Recursive/ Back tracking algo which is used in decision making and Game theory and uses recursion to Search through Game Tree.
- Algo Computes minimax decision for current state.
- We have 2 players: max and min player. Max player select the maximum value and min player select the minimum value
- Depth-First Search Algo is used for exploration of Complete Game Tree.
- Used in Chess, Tic Tac Toe and other 2 player Games



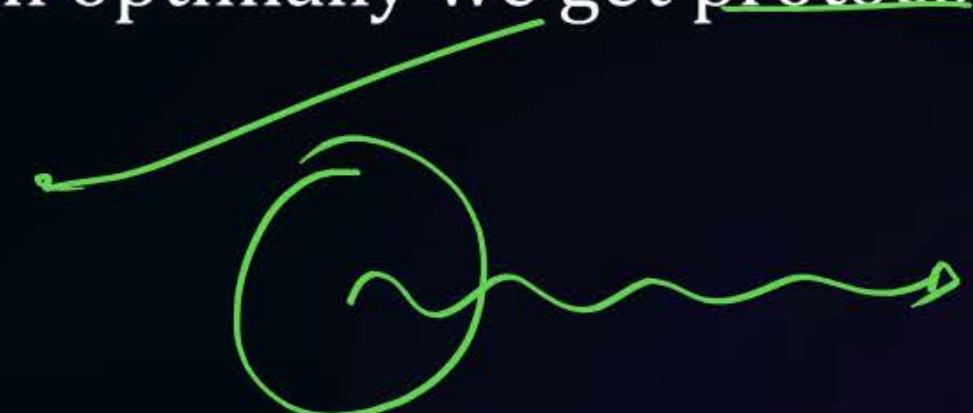
## Topic: Minimax Algorithm

- This is the backtracking algorithm
- Complete and optimal solution
- Time and Space complexity → Time complexity  $\underline{\underline{O(b^d)}}$   
Space complexity →  $\underline{\underline{O(b \times d)}}$
- In Minimax algorithm we use DFS

**Completeness:** Complete Algorithm, but only in finite trees

Optimal: Not always optimal because if any player play non-optimally we get protocol.

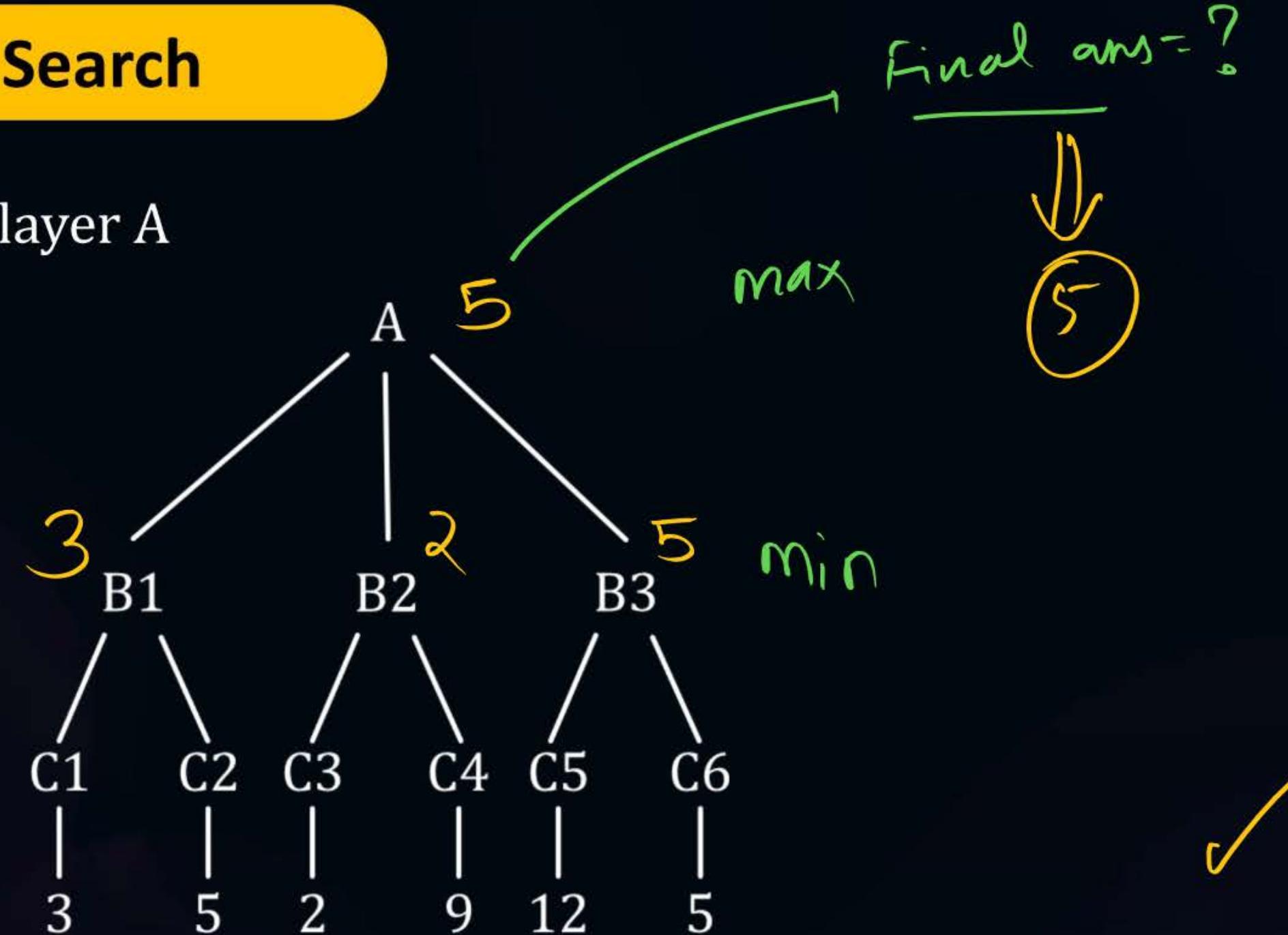
suboptimal  
Soln.





## Topic: Adversarial Search

- Find the move best for player A





## Topic: Adversarial Search

### Minimax Algorithm

- Minimax Principal
- The algorithm assumes that both players play optimally.
- It simulates all possible moves from the current state down to terminal states, then back-propagates the values up the tree to make the best decision at the root.

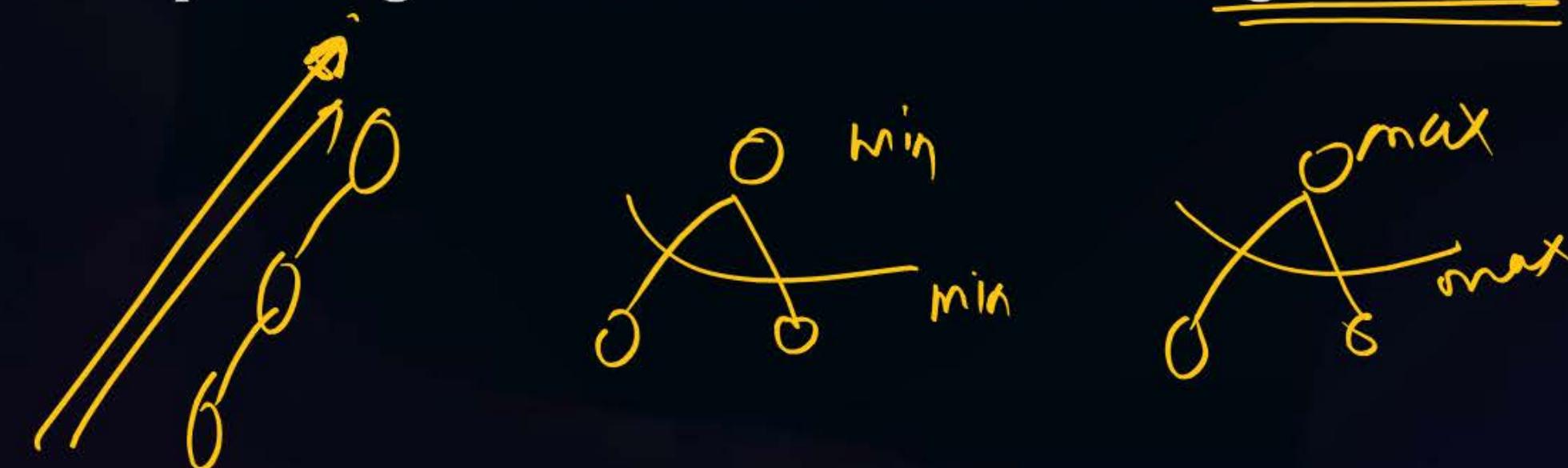




## Topic: Adversarial Search

### Minimax Algorithm Steps

- Backpropagate the Values:
  - Backpropagate the values from the terminal nodes up to the root.
  - At Maximizer nodes (Max nodes), select the maximum value from the child nodes.
  - At Minimizer nodes (Min nodes), select the minimum value from the child nodes.
  - Select the Best Move: At the root node (current game state), the Maximizer chooses the move corresponding to the child node with the highest value.





## Topic: Adversarial Search



### Problem in Min Max algorithm.

- TC reduces
- More efficient
- Less computation
- ↗  $\alpha - \beta$  pruning algo.

Space complexity remain some

Huge Time (as all nodes always explored)

$\alpha - \beta$  Pruning

- Both algo are used to solve game tree.
  1. same results of both: final result is exactly same.
  2. Benefit of pruning: Time complexity reduces  $O(b^{d/2})$ .
- Because now we do not visit all nodes.

minimax Algo

$\alpha - \beta$  pruning



## Topic: Alpha-Beta Pruning

### Alpha-Beta Pruning

Alpha-beta pruning is an optimization technique for the minimax algorithm, commonly used in decision-making for two-player games like chess, tic-tac-toe, and other adversarial games. The main purpose of alpha-beta pruning is to reduce the number of nodes evaluated by the minimax algorithm, thereby speeding up the decision-making process without affecting the final decision.



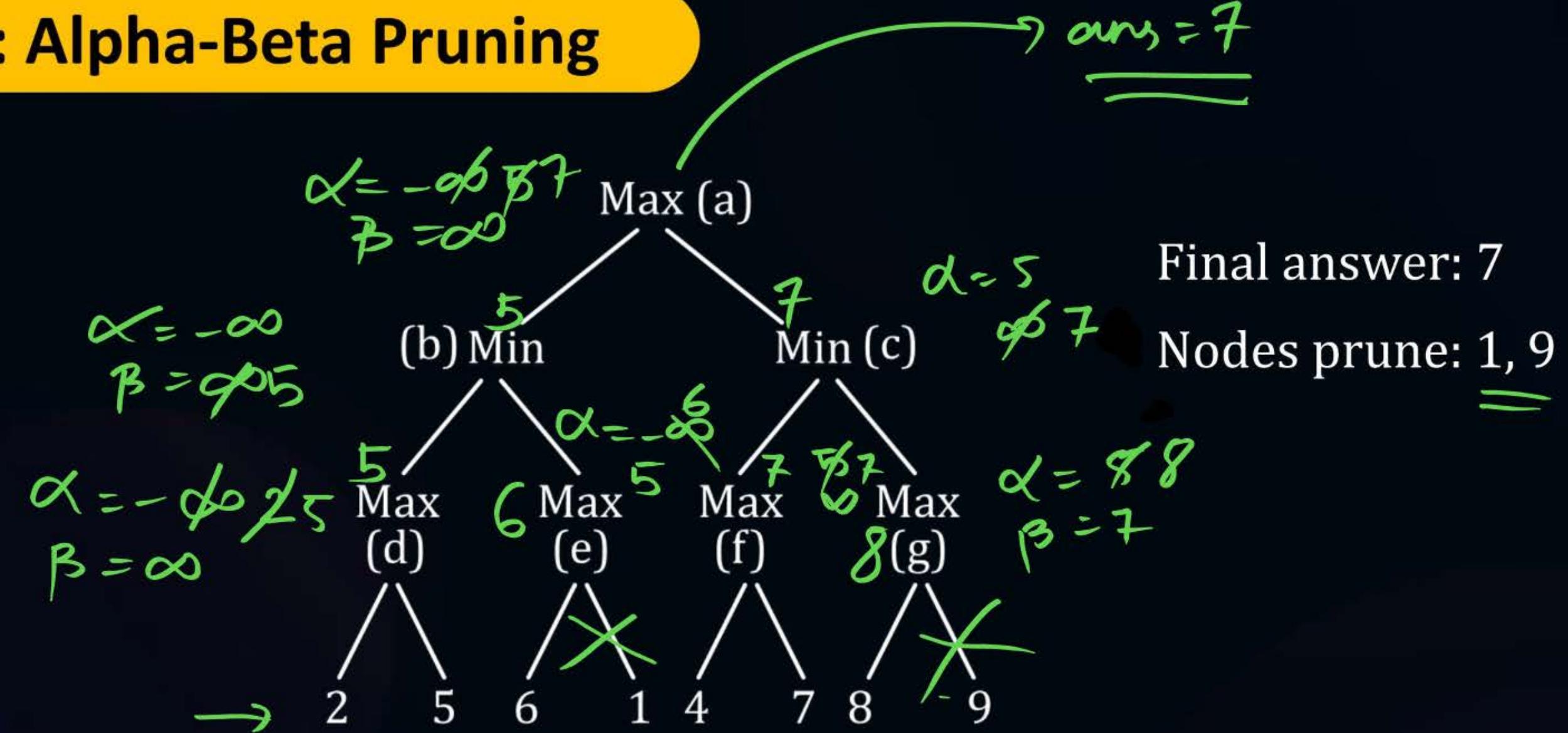
## Topic: Alpha-Beta Pruning

### Alpha-Beta Pruning

- This is advanced version of minimax algorithm.
- Because minimax algorithm has to explore all the nodes thus time complexity was  $\mathcal{O}(b^d)$
- But in this new algorithm we explore less number of nodes.
- Alpha is for max player.
- Beta is for min player.

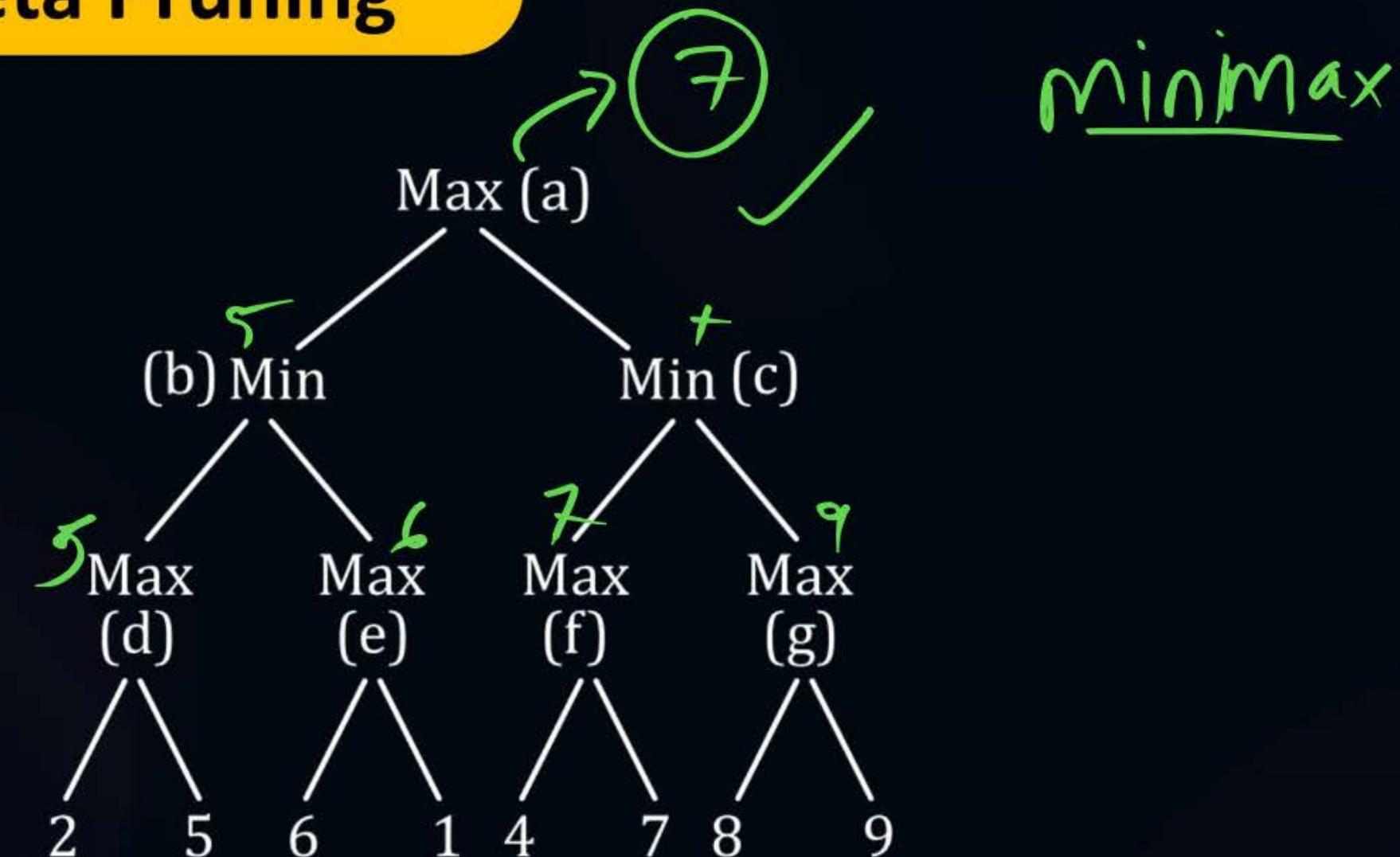


# Topic: Alpha-Beta Pruning





# Topic: Alpha-Beta Pruning



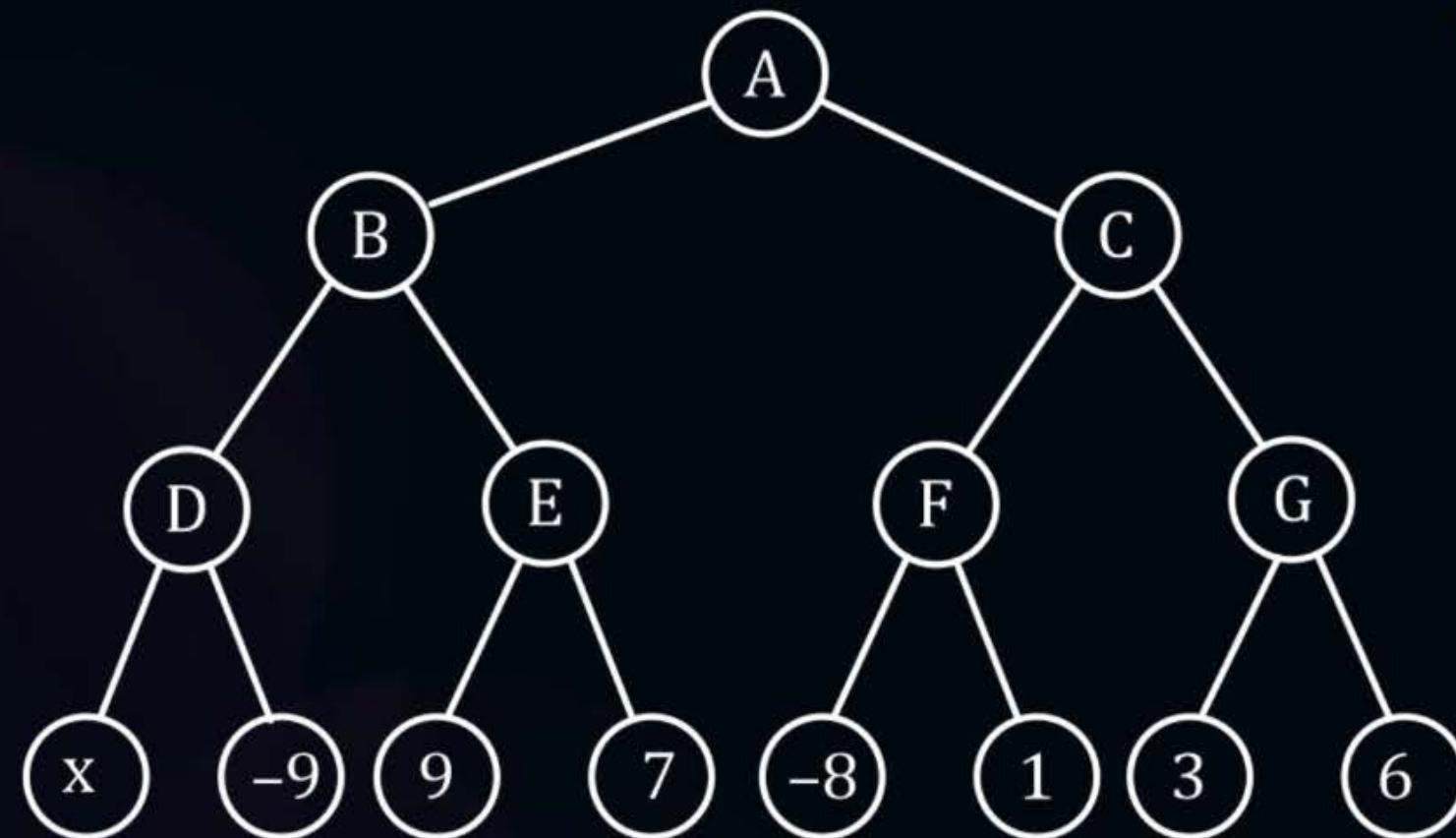


## Topic: Alpha-Beta Pruning

Adv

P  
W

#Q. Player 1 (max node) chooses the left action for which of the following value(s) of  $\chi_k$ . Assume optimal play and Player 1's turn at Node A.



ans  $n > 1$   
=====

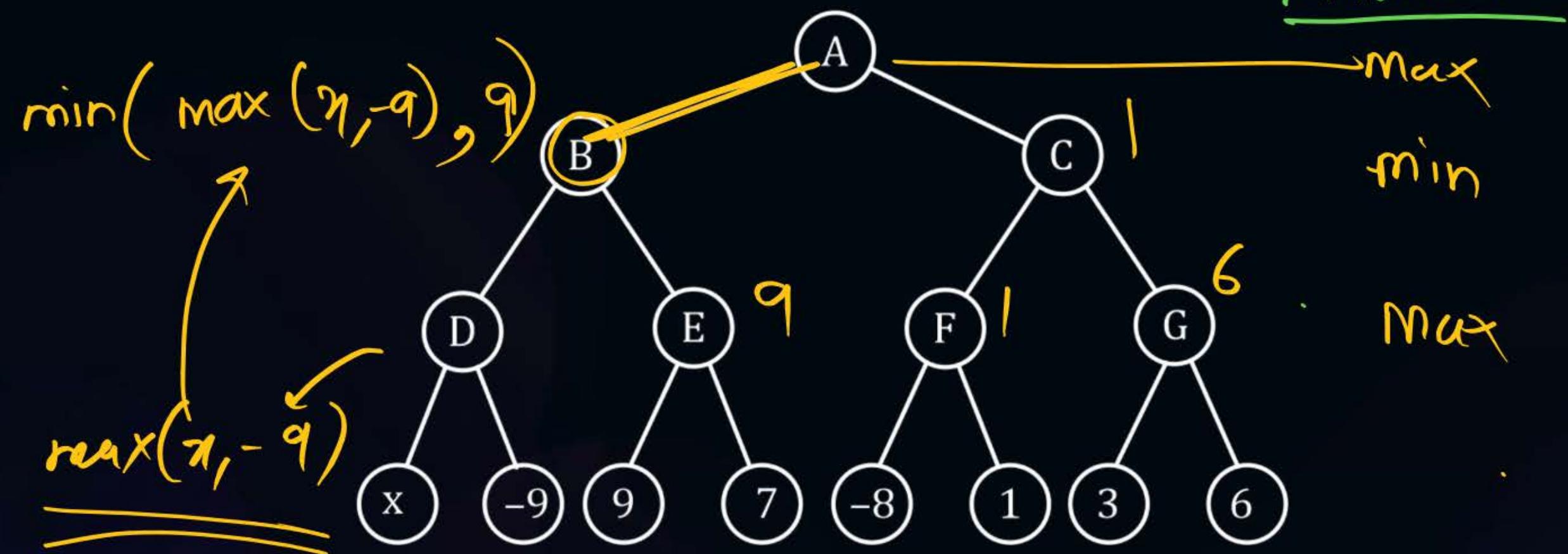


## Topic: Alpha-Beta Pruning

Adv

P  
W

#Q. Player 1 (max node) chooses the left action for which of the following value(s) of  $\chi_k$ . Assume optimal play and Player 1's turn at Node A.



$$A \Rightarrow \max\left(\min\left(\max(n, -9), 9\right), 1\right)$$

L                                    R

$$\min\left(\max(n, -9), 9\right) > 1$$

$$n < -9$$

$$n > 9$$

ans

$$n > 1$$



$$n > 9$$

X  $\min(-9, 9) = -9 > 1$  NO

$$\min(9, 9) > 1$$

$$\frac{x > 9}{9 > 1}$$

$$\frac{n < 9}{x > 1}$$

$$\min(7, 9) > 1$$

True



This is True for any  $x > 1$



## Topic: Alpha-Beta Pruning



#Q. Player 1 (max node) chooses the left action for which of the following value(s) of k. Assume optimal play and Player 1's turn at Node A.



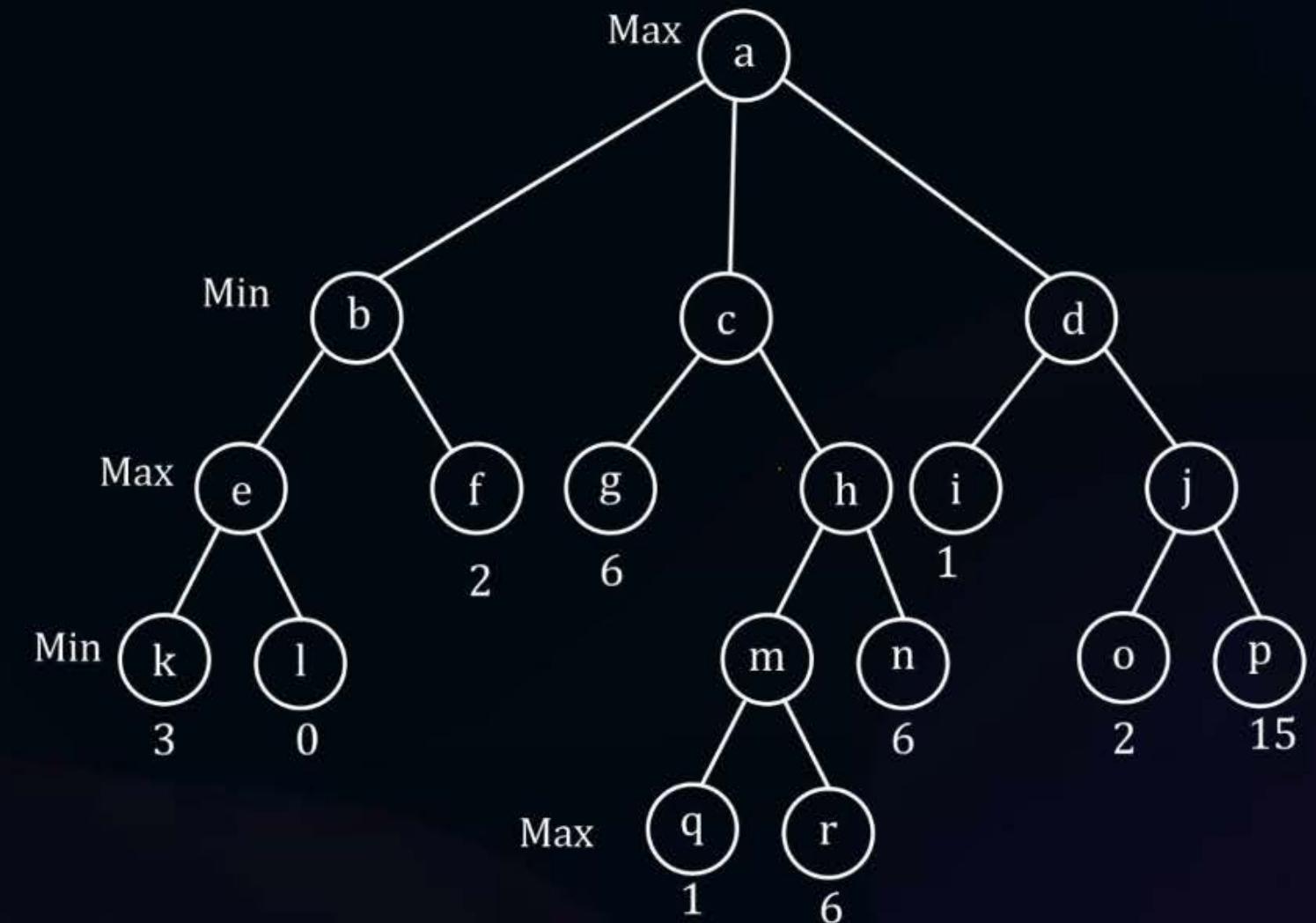


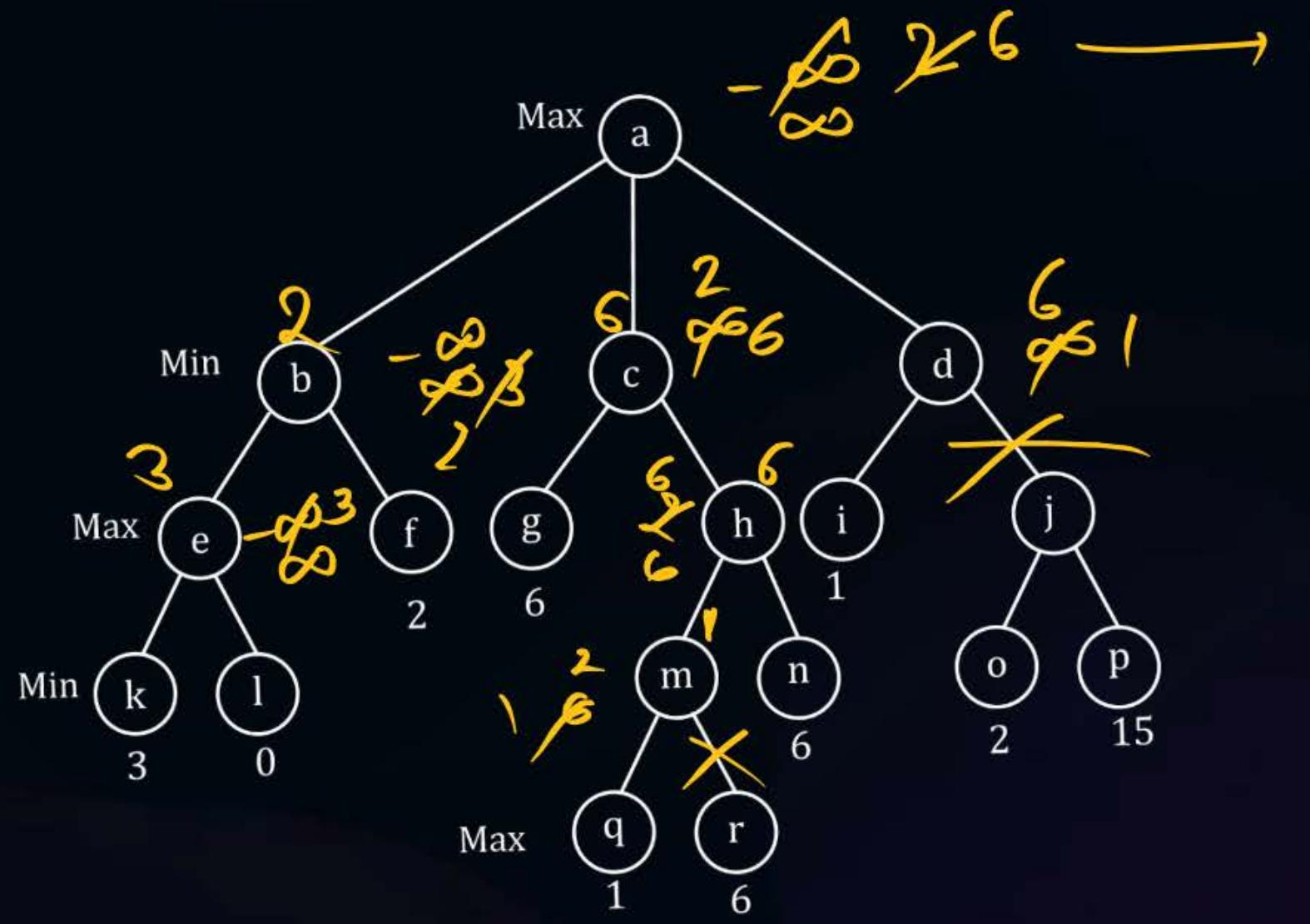
## Topic: Alpha-Beta Pruning



#Q. Consider the game tree shown below. The value below each node is the output of the utility function. The subtrees rooted at which of these nodes will be pruned because of alpha-beta pruning?

- A** m and j ~~X~~
- B** r and j ~~X~~
- C** h and p ~~X~~
- D** no nodes pruned ~~X~~





$\infty \rightarrow \infty^6 \rightarrow \underline{\text{ans} = 6}$

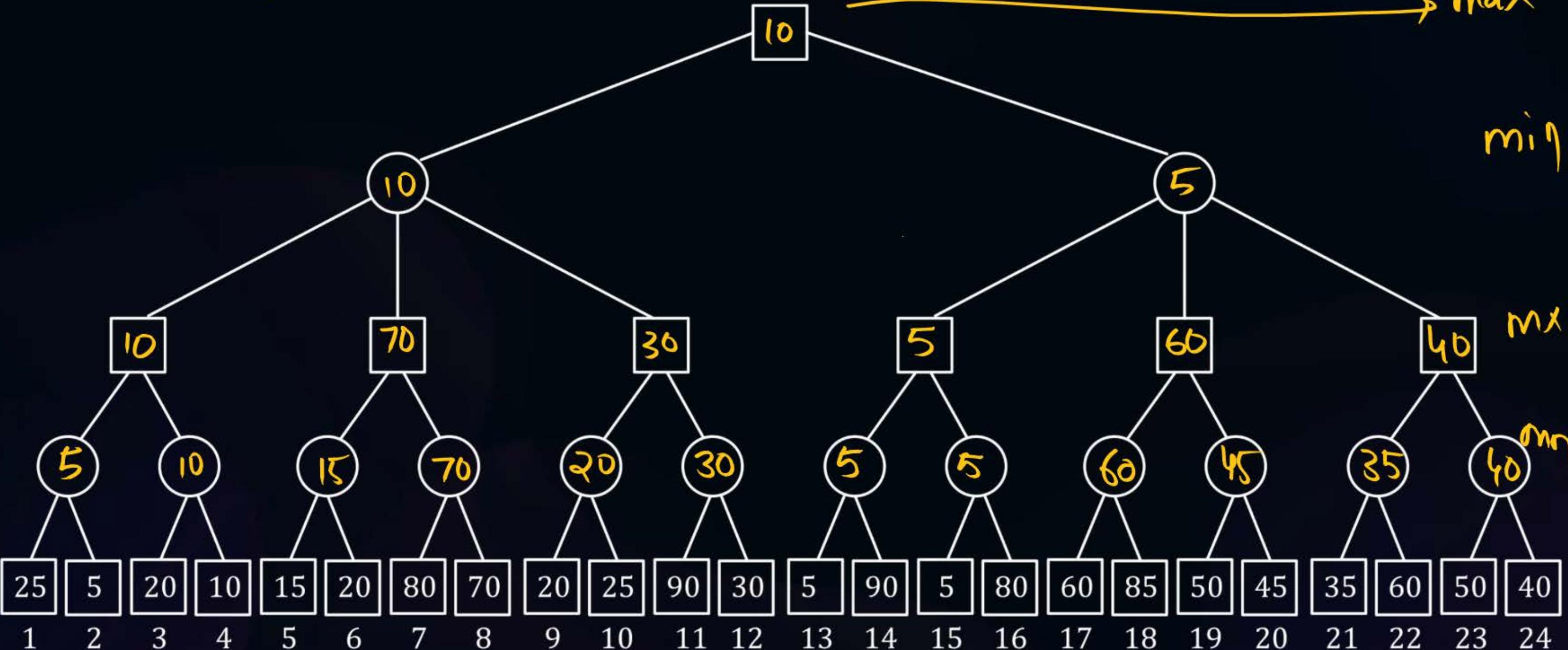
Pruned  
nodes

=  $\gamma, j, o, p$



# Topic: Alpha-Beta Pruning

minimax



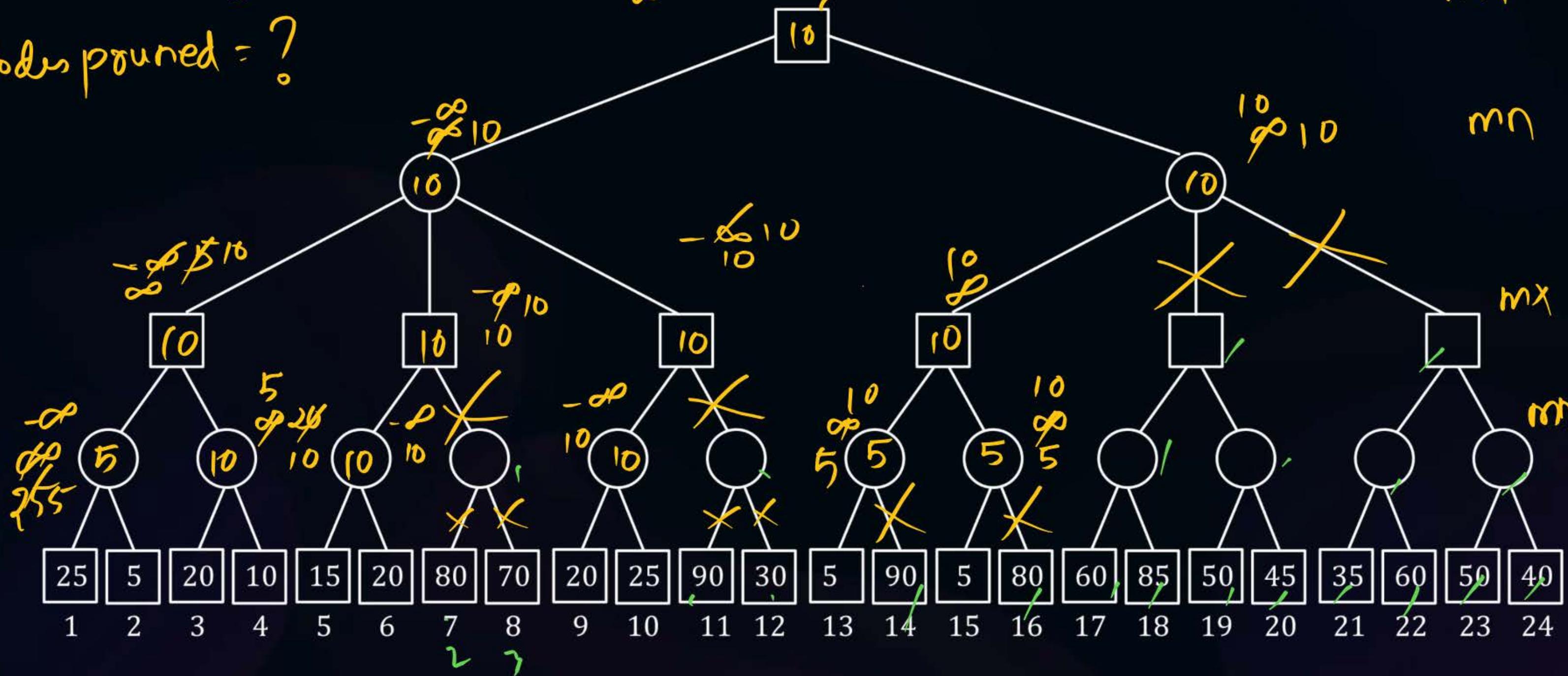


# Topic: Alpha-Beta Pruning



$\alpha$ - $\beta$  Pruning

Nodes pruned = ?





# THANK - YOU

Happy Diwali!





# DS & AI ENGINEERING



## Artificial Intelligence

### Adversarial Search

Lecture No.- 03

By- Aditya sir



# Recap of Previous Lecture



Topic

Topic

Topic

Topic

Minimax

$\alpha$ - $\beta$  Pruning



# Topics to be Covered



Topic

Topic

Topic

Questions

Expecti - Minimax

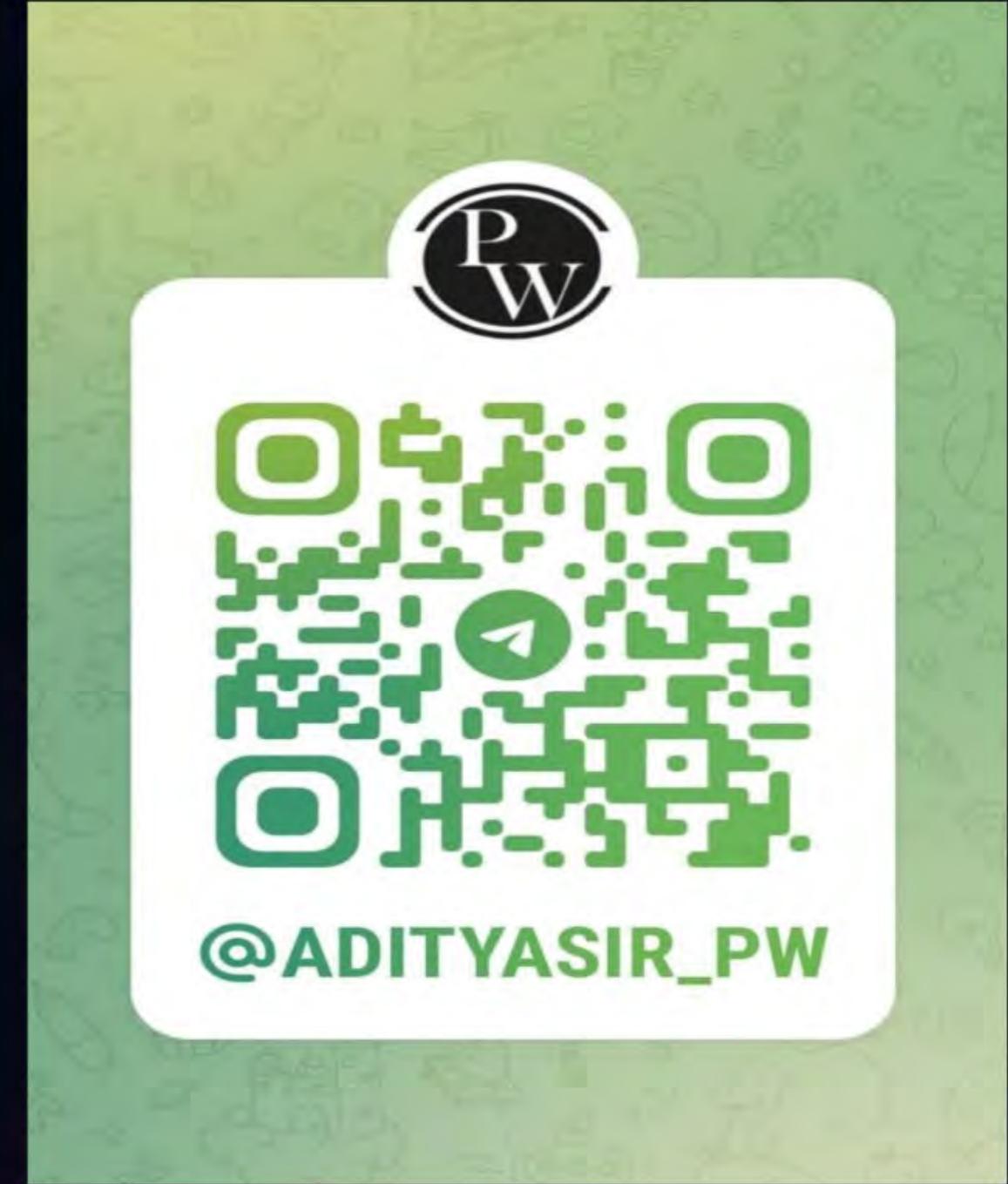
$\alpha$ -cut,  $\beta$ -cut



## About Aditya Jain sir

1. Appeared for GATE during BTech and secured AIR 60 in GATE in very first attempt - City topper
2. Represented college as the first Google DSC Ambassador.
3. The only student from the batch to secure an internship at Amazon. (9+ CGPA)
4. Had offer from IIT Bombay and IISc Bangalore to join the Masters program
5. Joined IIT Bombay for my 2 year Masters program, specialization in Data Science
6. Published multiple research papers in well known conferences along with the team
7. Received the prestigious excellence in Research award from IIT Bombay for my Masters thesis in ML
8. Completed my Masters with an overall GPA of 9.36/10
9. Joined Dream11 as a Data Scientist
10. Have mentored 15,000+ students & working professionals in field of Data Science and Analytics
11. Have been mentoring & teaching GATE aspirants to secure a great rank in limited time
12. Have got around 27.5K followers on Linkedin where I share my insights and guide students and professionals.

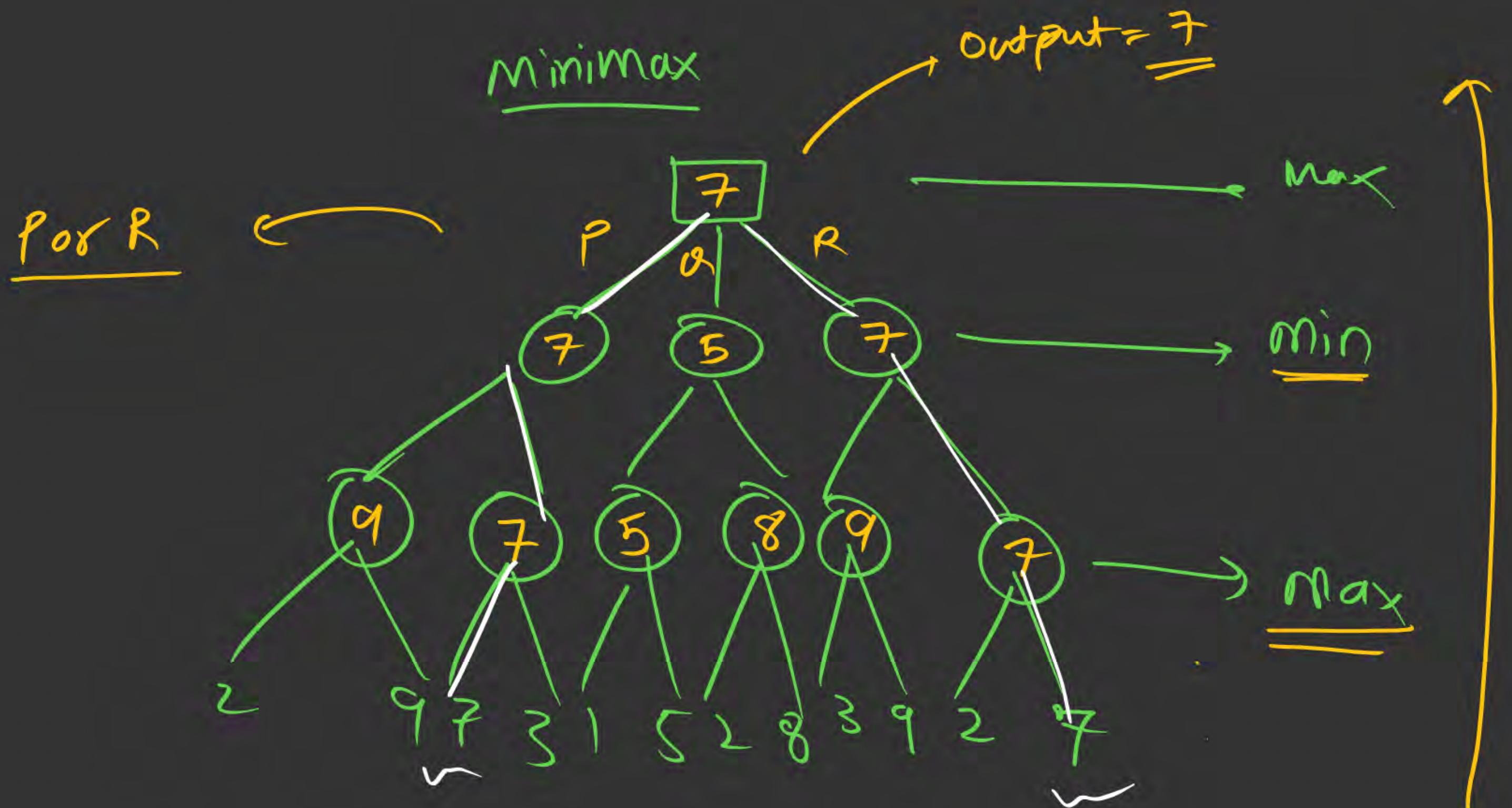




Telegram

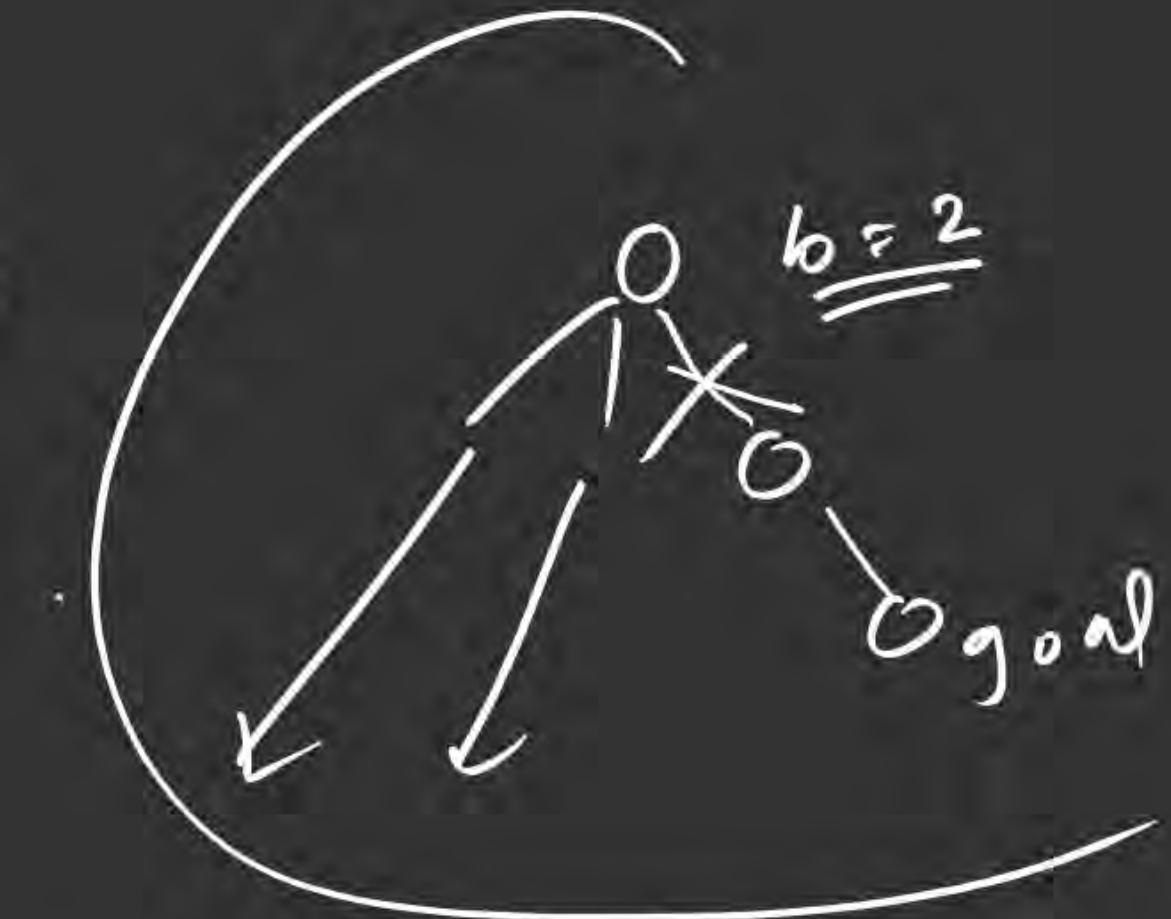
**Telegram Link for Aditya Jain sir:**

**[https://t.me/AdityaSir\\_PW](https://t.me/AdityaSir_PW)**



## ★ Optimization over Mini Max Alg.

$\alpha$ - $\beta$  Pruning



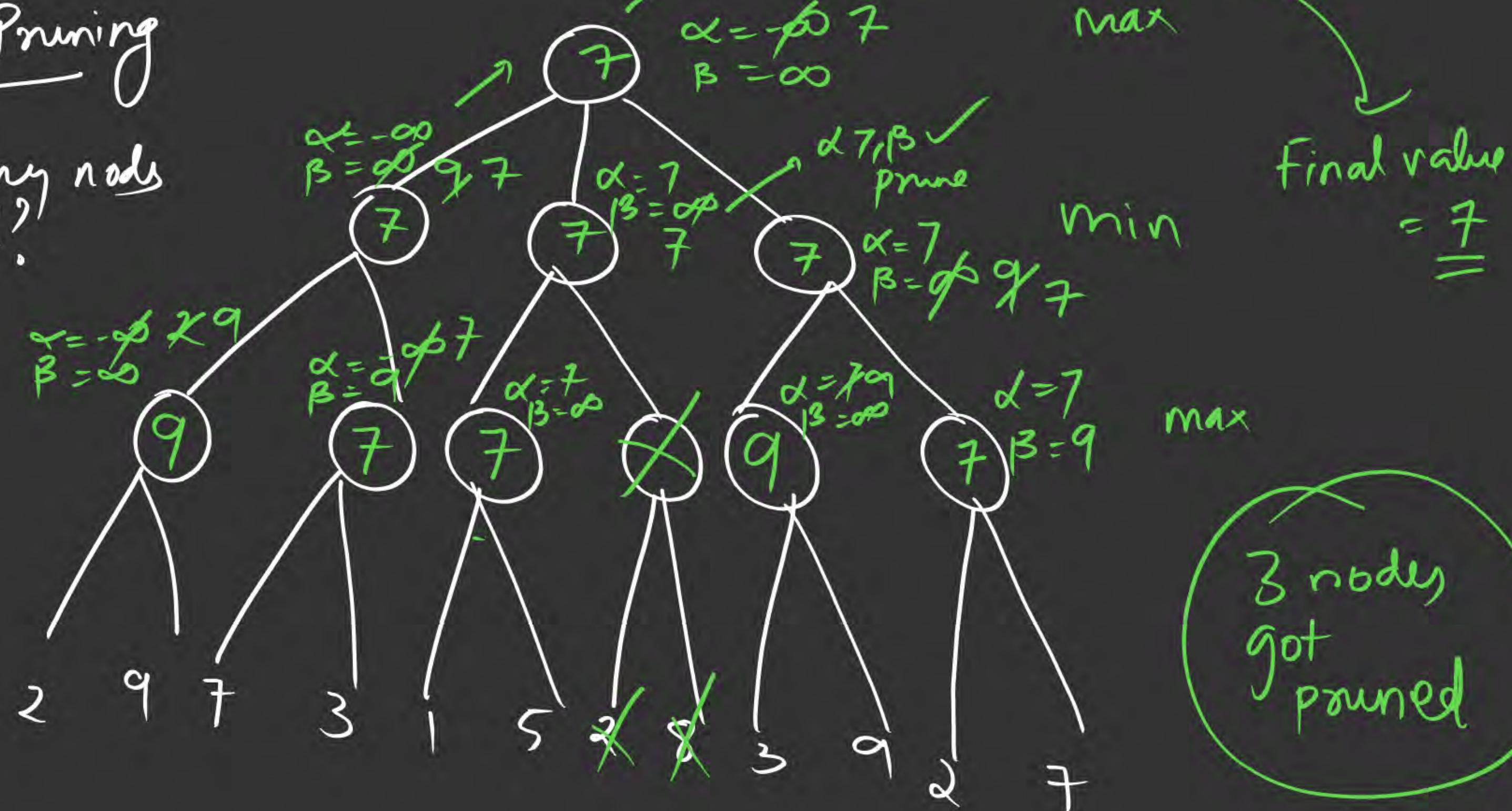
Both  $\rightarrow$  Mini Max  
&  $\alpha$ - $\beta$  Pruning

} always give same  
Final answer

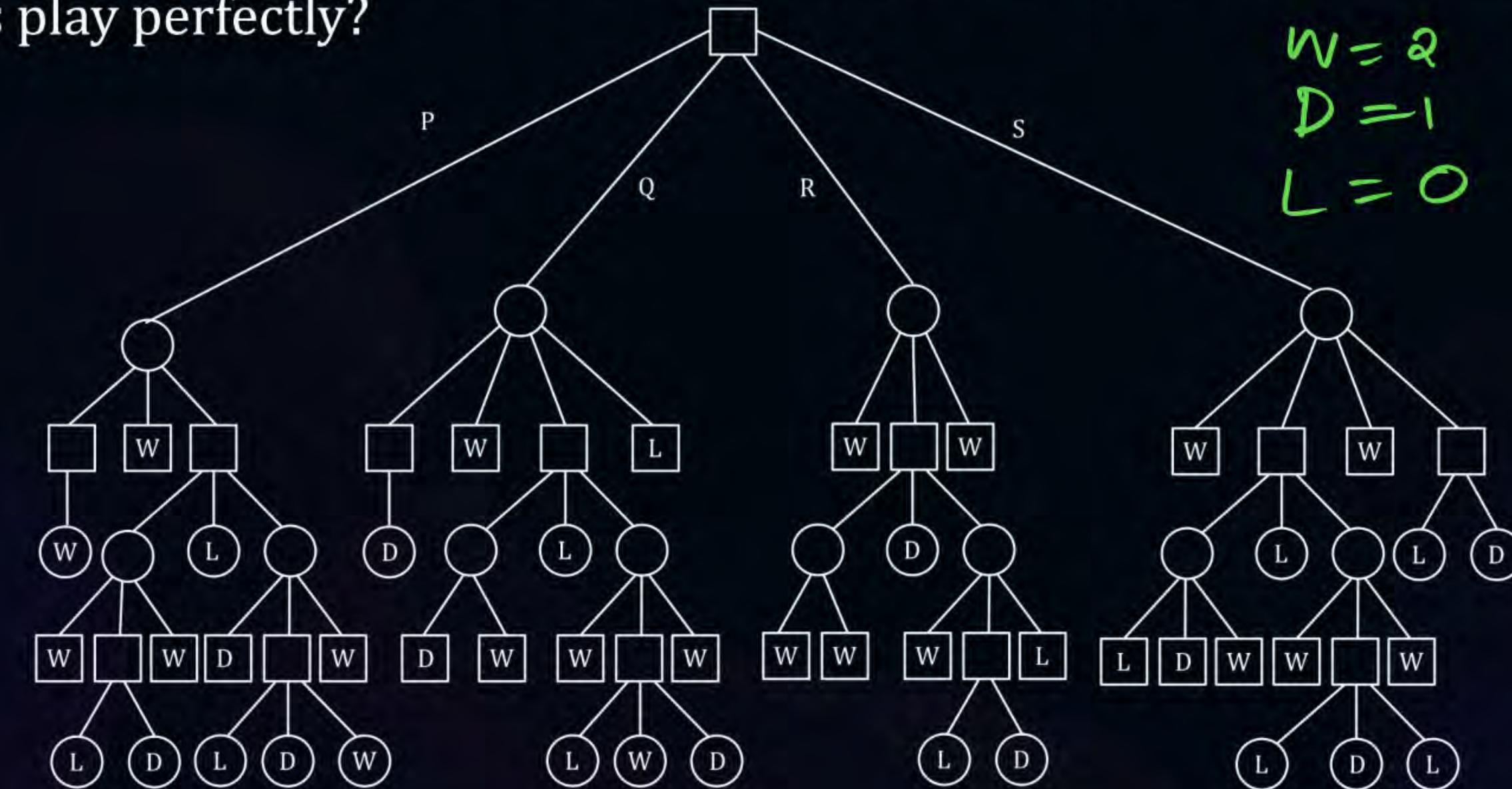
Prune only when  
 $\underline{\alpha} > \underline{\beta}$

## $\alpha$ - $\beta$ Pruning

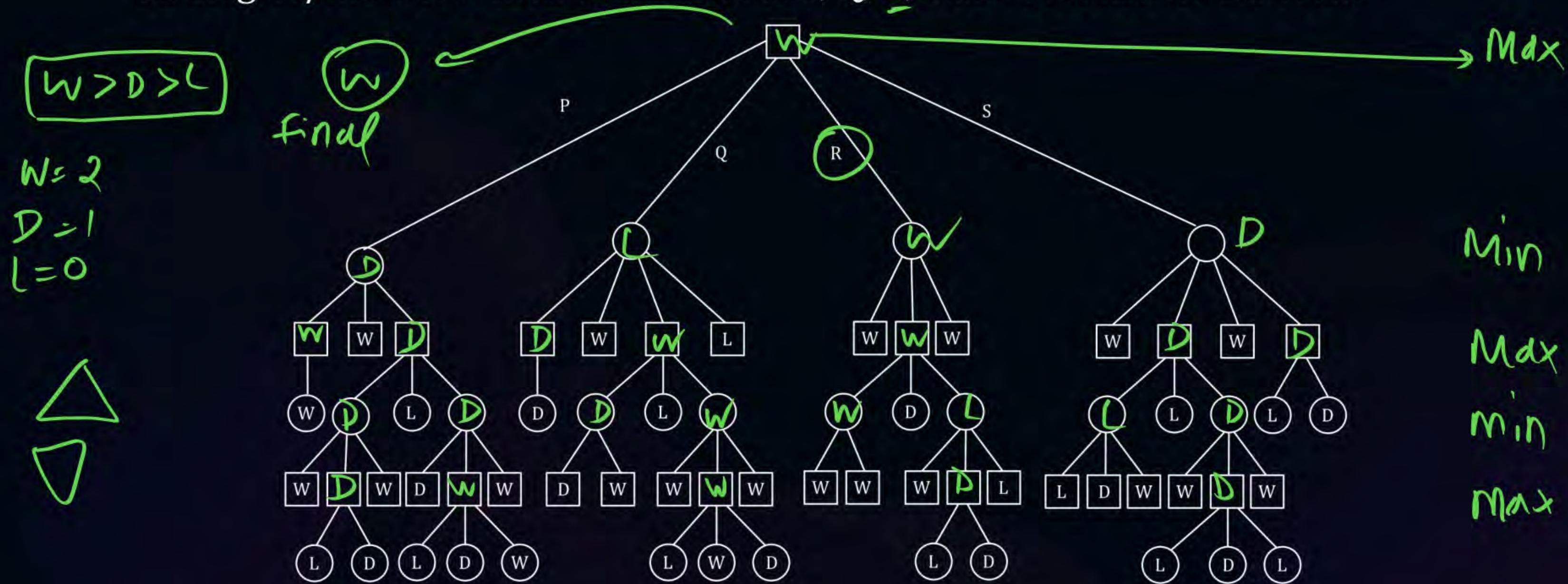
How many nodes pruned?



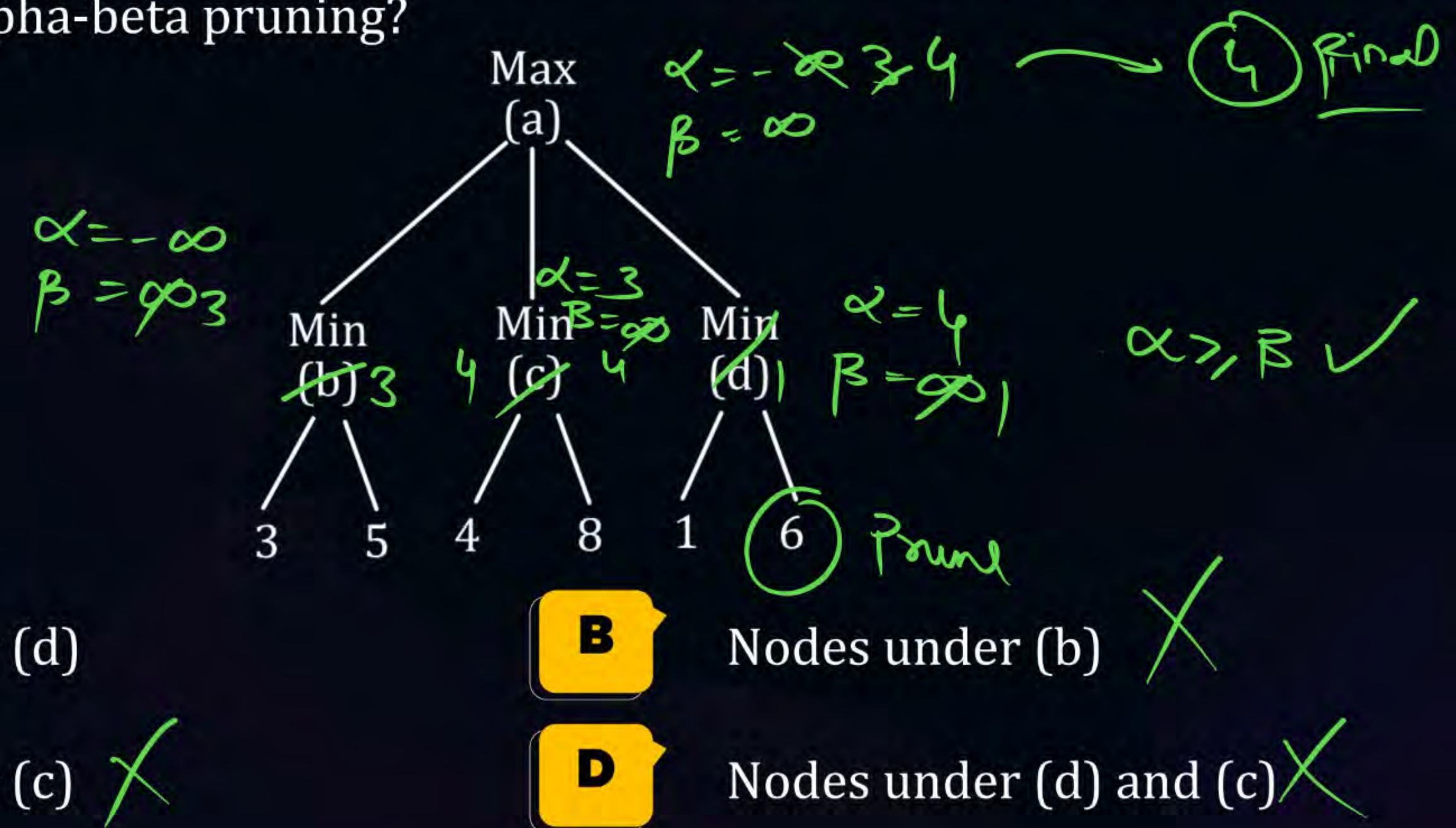
#Q. The figure shows a game tree with evaluations W (win), L (loss) and D (draw) from Max's perspective. In this game tree the labels P, Q, R, S indicate strategies/moves. What is the outcome (W, L or D) of the game when both players play perfectly?



#Q. The figure shows a game tree with evaluations W (win), L (loss) and D (draw) from Max's perspective. In this game tree the labels P, Q, R, S indicate strategies/moves. Which of the moves P, Q, R, S are best moves for Max?



#Q. Consider the following game tree. The value below each node is the output of the utility function. The subtrees rooted at which of these nodes will be pruned because of alpha-beta pruning?



Imp Terms

$\alpha = \max$  Lower Bound for Max Player ↑

$\beta = \min$  Upper Bound for Min Player ↓



## Topic : Alpha Cut

Alpha Cut:

- Happens at a MIN node.
- **Meaning:** If a MIN node finds a value less than or equal to an already known best ( $\alpha$ ) value for its MAX ancestor, then MIN node stops exploring further.
- **Because:** The MAX ancestor will never allow this bad (lower) value, so no need to continue.



**At MIN node:** If value  $\leq \alpha$ , then alpha cut (prune remaining branches).

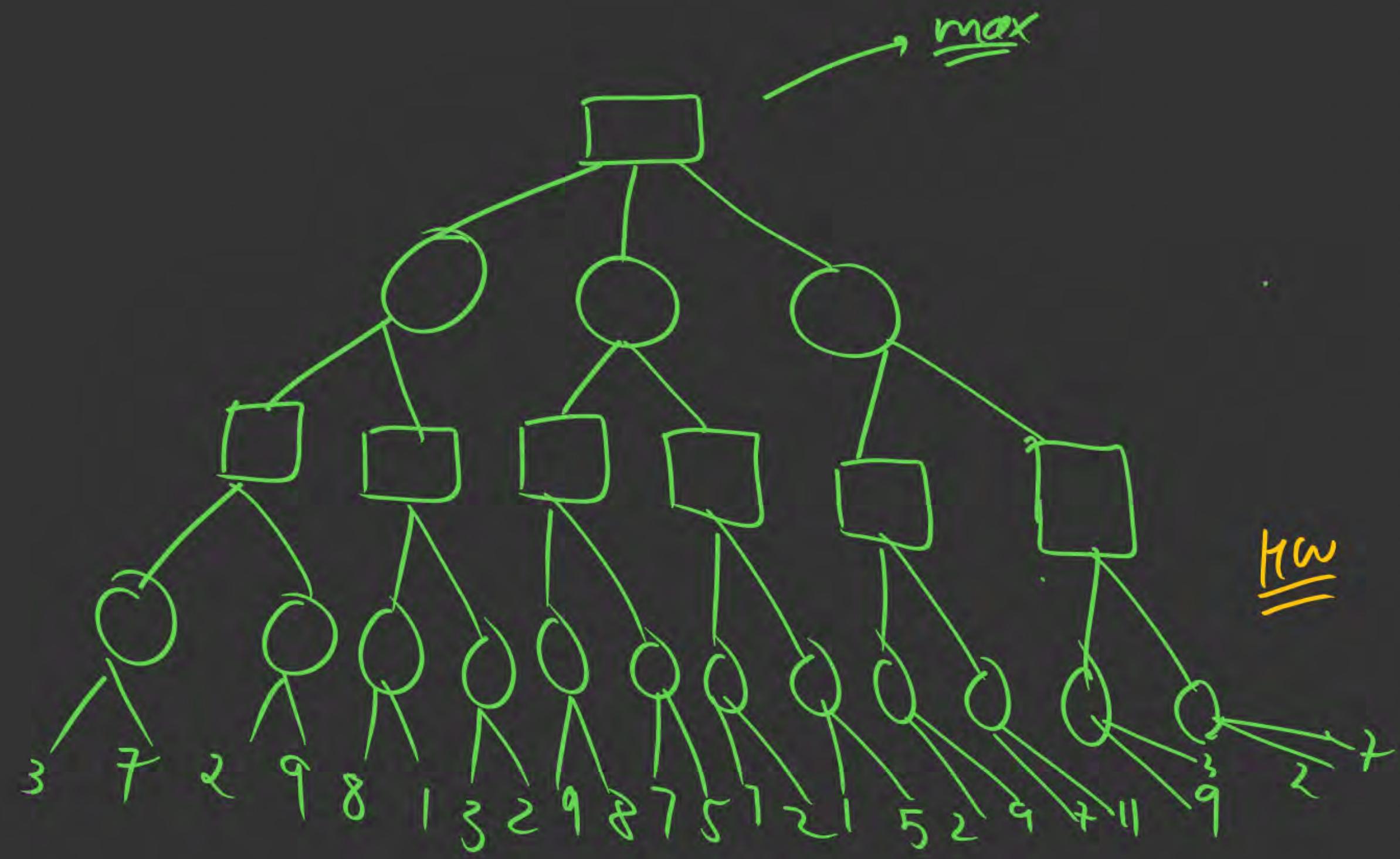


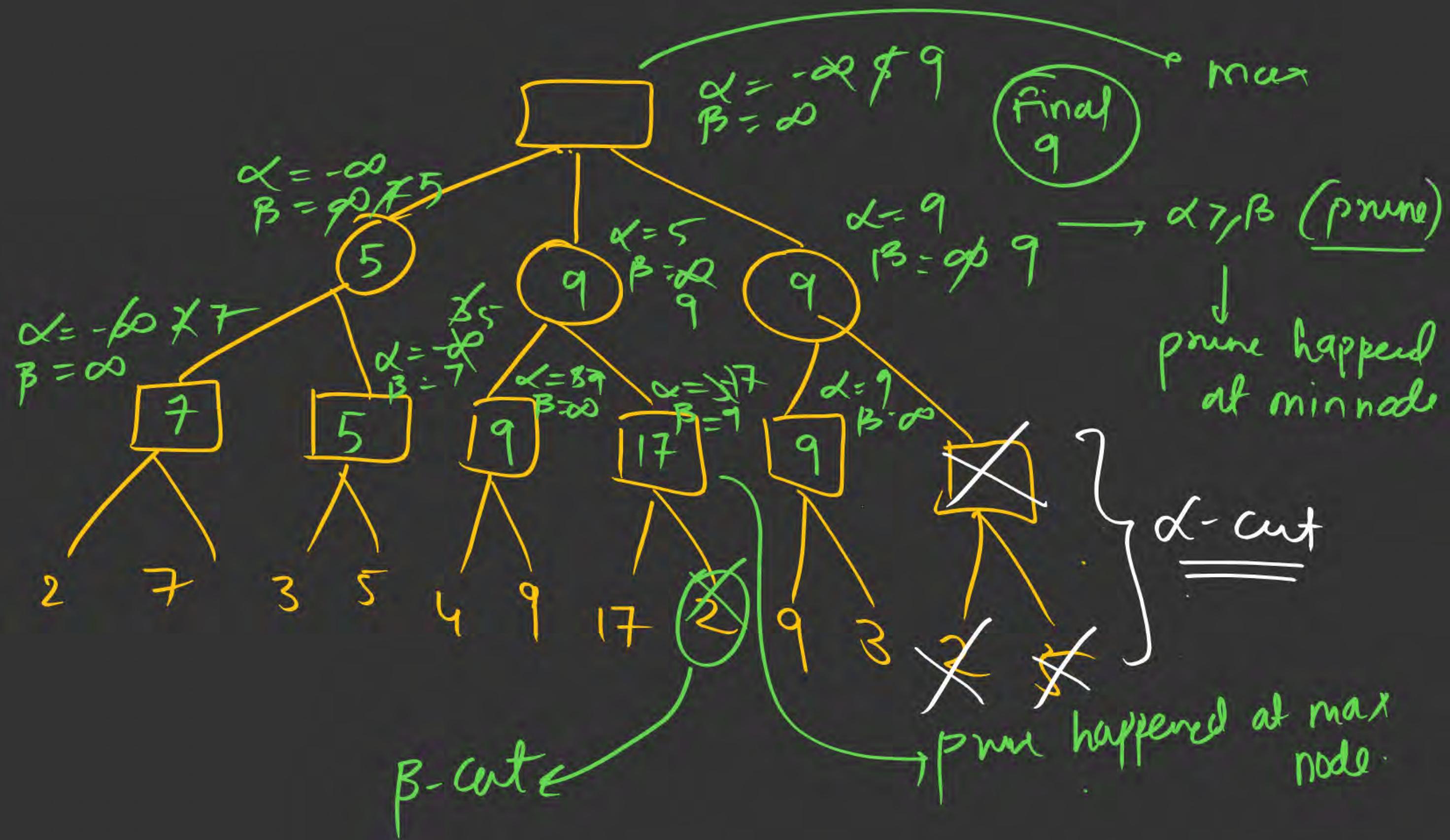
## Topic : Alpha Cut

### Beta Cut:

- Happens at a MAX node.
- Meaning: If a MAX node finds a value greater than or equal to an already known best ( $\beta$ ) value for its MIN ancestor, then MAX node stops exploring further.
- Because: The MIN ancestor will never allow this higher (worse for MIN) value, so no need to continue.

At MAX node: If  $\text{value} \geq \beta$ , then beta cut (prune remaining branches).







# Topic : Expectimax Algorithm in Game Theory



## What is Expectimax?

The **Expectimax algorithm** is an extension of the **Minimax algorithm** used in decision-making problems involving both adversarial and chance events. It's used when outcomes involve:

- **Maximizing player** (tries to get highest utility).
- **Minimizing player** (tries to minimize opponent's utility).
- **Chance nodes** (introduce random outcomes, like dice rolls or shuffled cards).



# Topic : Expectimax Algorithm in Game Theory



Each node in the game tree is one of three types:

| Node Type   | Role                                                     | Example                |
|-------------|----------------------------------------------------------|------------------------|
| Max Node    | Choose the move with max value                           | Player's turn          |
| Min Node    | Choose the move with min value                           | Opponent's turn        |
| Chance Node | Computes expected value-based on probability of outcomes | Dice roll or card draw |



# Topic : Expectimax Algorithm in Game Theory



## How it Works?

### Step-by-step Traversal:

1. Traverse the game tree recursively from leaf to root.
2. At leaf nodes: Use an evaluation function or utility value.
3. At Max node: Select the child node with the maximum value.
4. At Min node: Select the child node with the minimum value.
5. At Chance node: Compute the expected value using:

$$\text{Expected Value} = \sum (\text{Probability}_i \times \text{Value}_i)$$



# Topic : Expectimax Algorithm in Game Theory

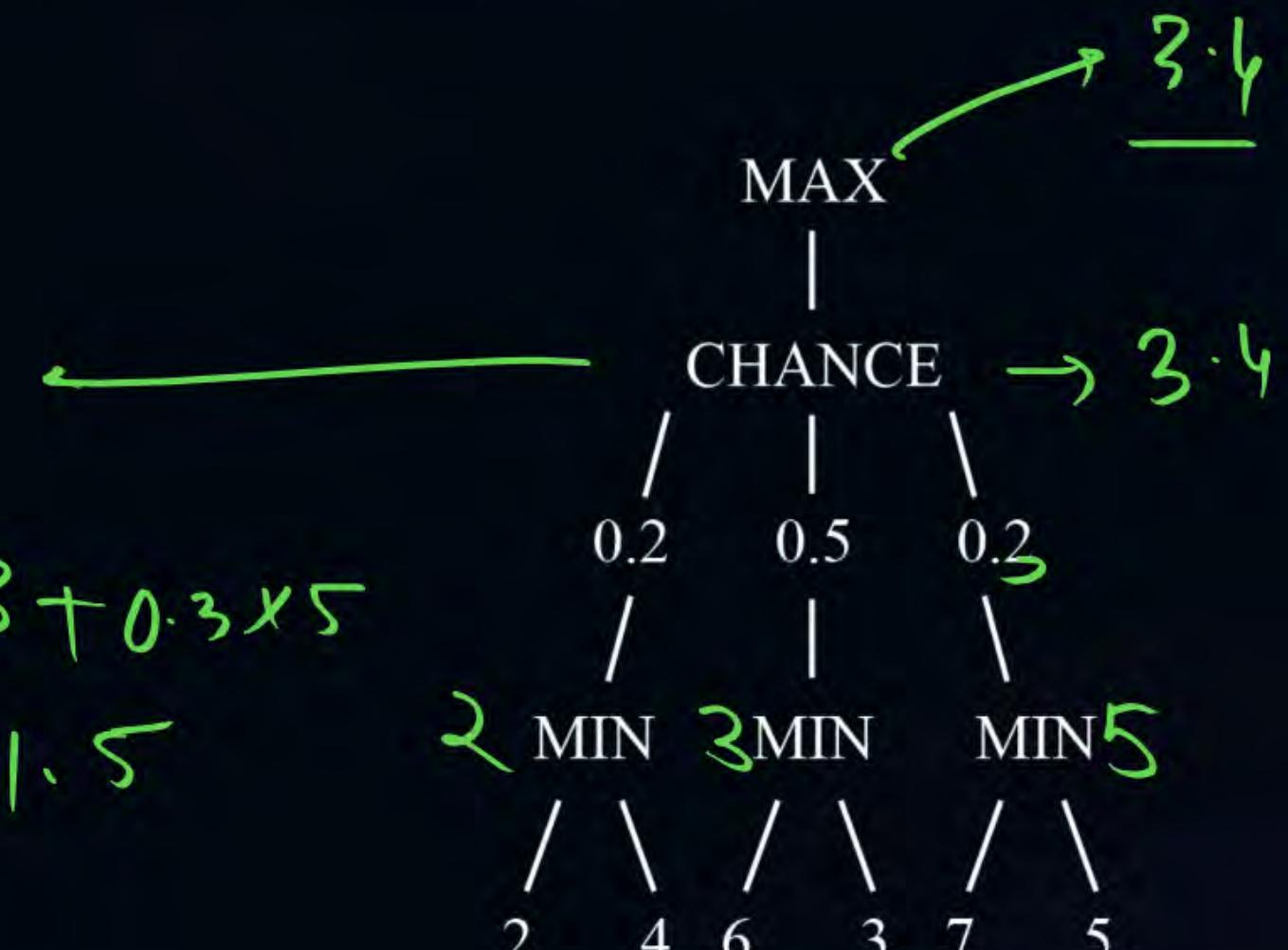


$$\sum p_i v_i$$

$$= 0.2 \times 2 + 0.5 \times 3 + 0.3 \times 5$$

$$= 0.4 + 1.5 + 1.5$$

3.4





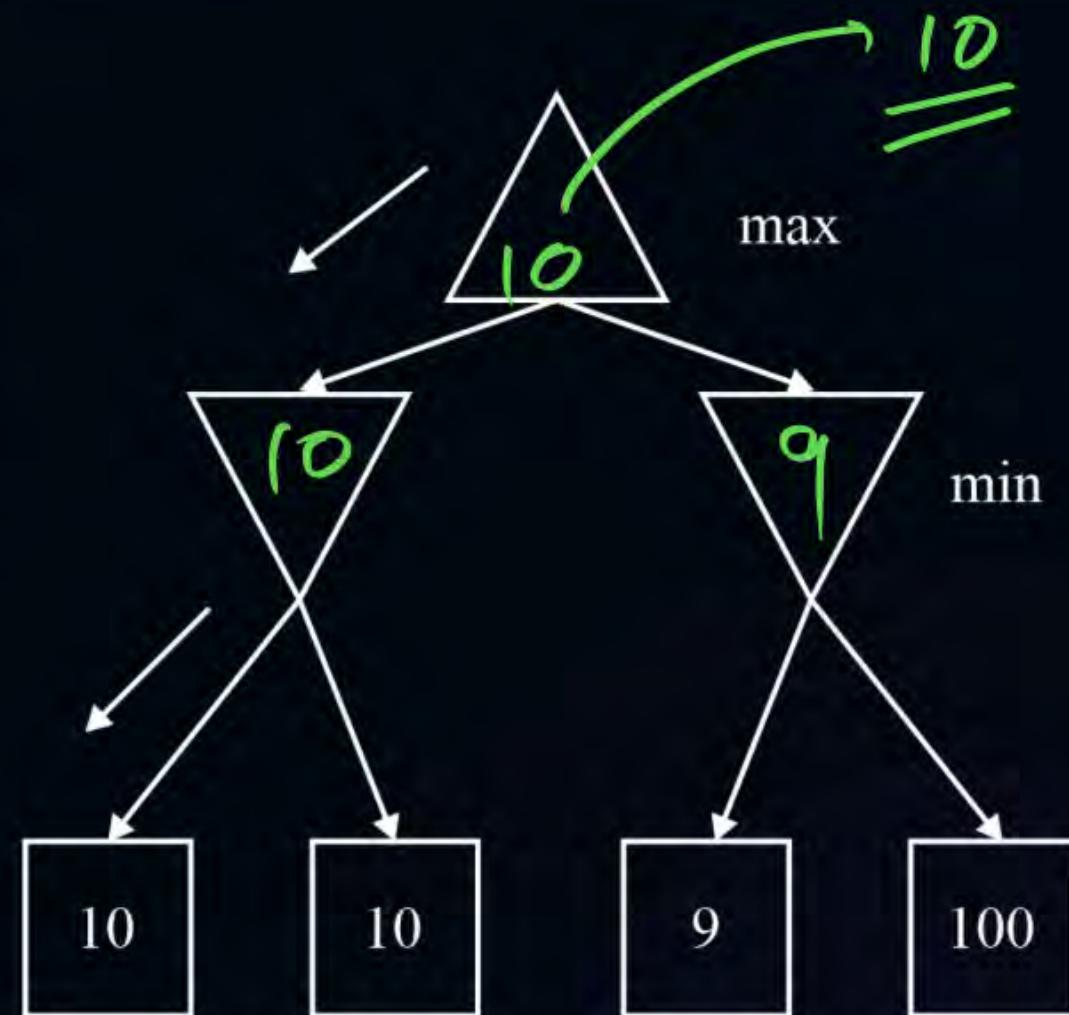
## Topic : Expectimax Algorithm in Game Theory



The Expectimax search algorithm is a game theory algorithm used to maximize the expected utility. It is a variation of the Minimax algorithm. While Minimax assumes that the adversary(the minimizer) plays optimally, the Expectimax doesn't. This is useful for modelling environments where adversary agents are not optimal, or their actions are based on chance.

## Topic : Expectimax Minimax

As we know that the adversary agent(minimizer) plays optimally, it makes sense to go to the left. But what if there is a possibility of the minimizer making a mistake(or not playing optimally). Therefore going right might sound more appealing or may result in a better solution.

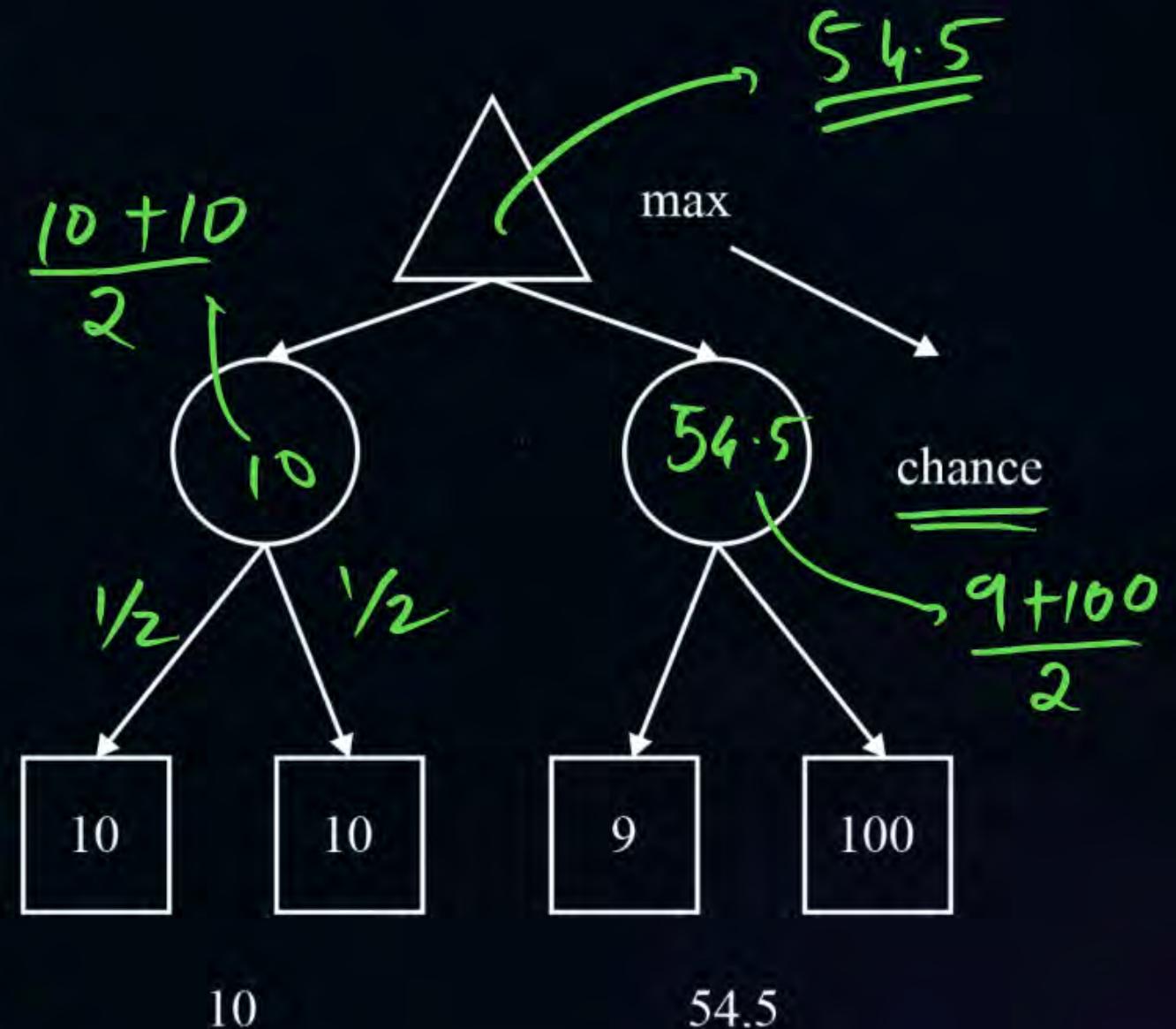




## Topic : Expectimax Minimax

In the below Expectimax tree, we have replaced minimizer nodes by chance nodes. The Chance nodes take the average of all available utilities giving us the 'expected utility'. Thus the expected utilities for left and right sub-trees are  $(10+10)/2=10$  and  $(100+9)/2=54.5$ . The maximizer node chooses the right sub-tree to maximize the expected utilities.

default → consider equi-probable  
mean → take avg of all branches





## Topic : Expectimax Minimax



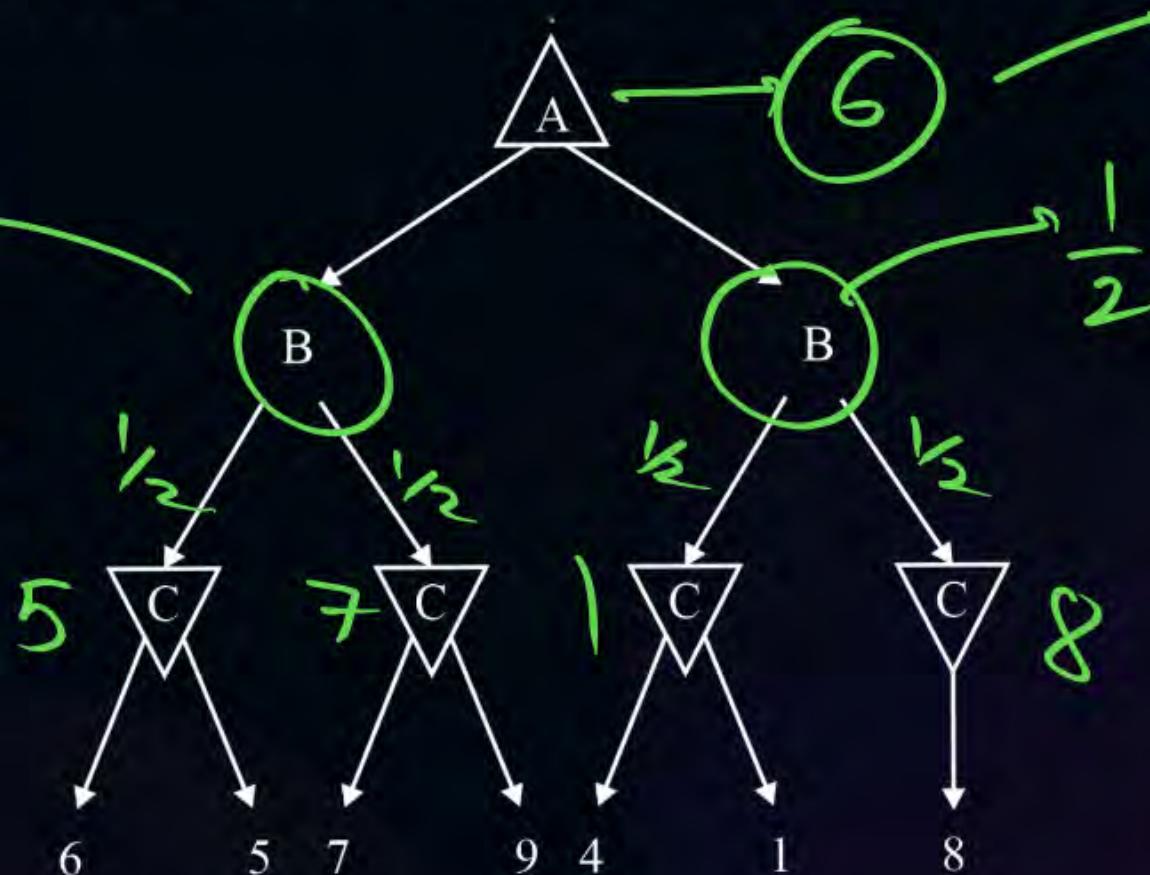
Recall that a chance node has the expected value of its children, and let each child have an equal probability of being chosen.

~~Perform pruning on the following game tree and fill in the values at letter nodes.~~ A is a maximizer, B is a chance node, and C is a minimizer. Assume that values can only be in the range 0-9 (inclusive).

A → max  
B → chance  
C → min

Ans: 6

$$\frac{5}{2} + \frac{7}{2} = \underline{\underline{6}}$$



$$\begin{aligned} & \max(6, 6.5) \\ &= 6.5 \\ &= \frac{8}{2} = \frac{9}{2} \\ &= 4.5 \end{aligned}$$



**THANK - YOU**