DS & AI ENGINEERING

Artificial Intelligence

Un-Informed search

Lecture No.- 04

By- Aditya sir

Recap of Previous Lecture

- Topic
- Topic — BFS
- Topic — DFS
- Topic

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 professions 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 (Al), DFS is often used for exploring possible states in search problems, finding solutions in game playing, planning, and more.

Topic : Uninformed Search

Step-1 ⇒ Two list visited, stack

- Start with starting node and insert it into stack.

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

LIFO

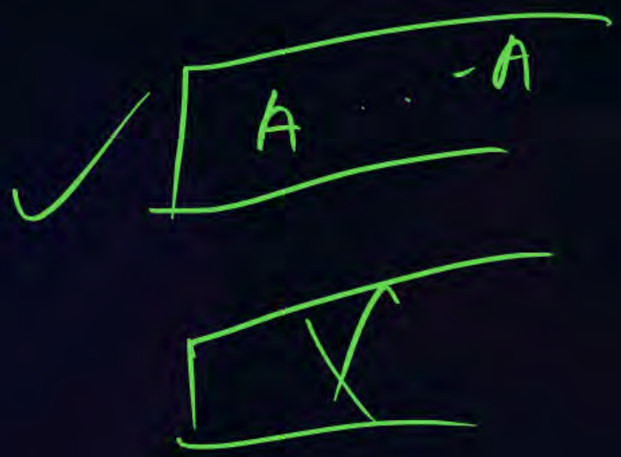
Step-3 ⇒

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

DFS

Stack → open

visited → closed

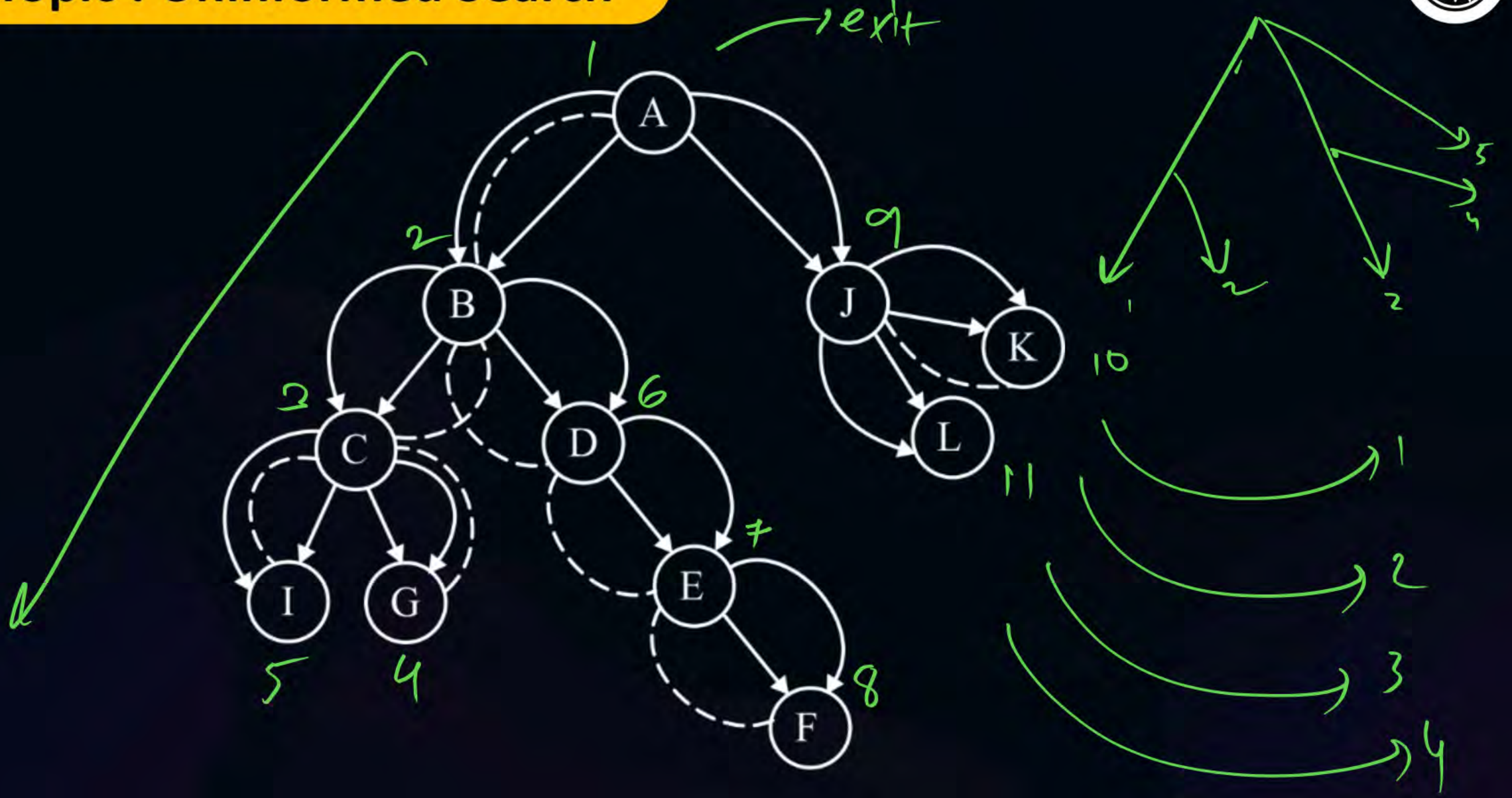


BFS

Queue → open

visited → 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.)

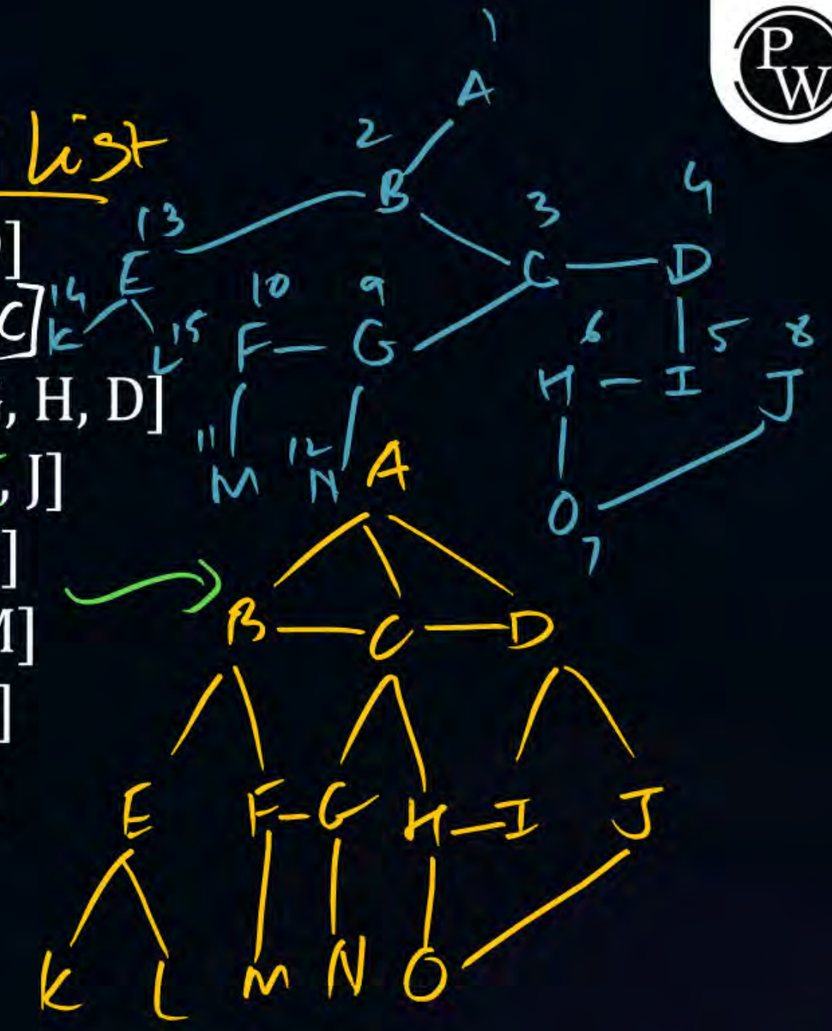
Not Complete

Topic : Uninformed Search

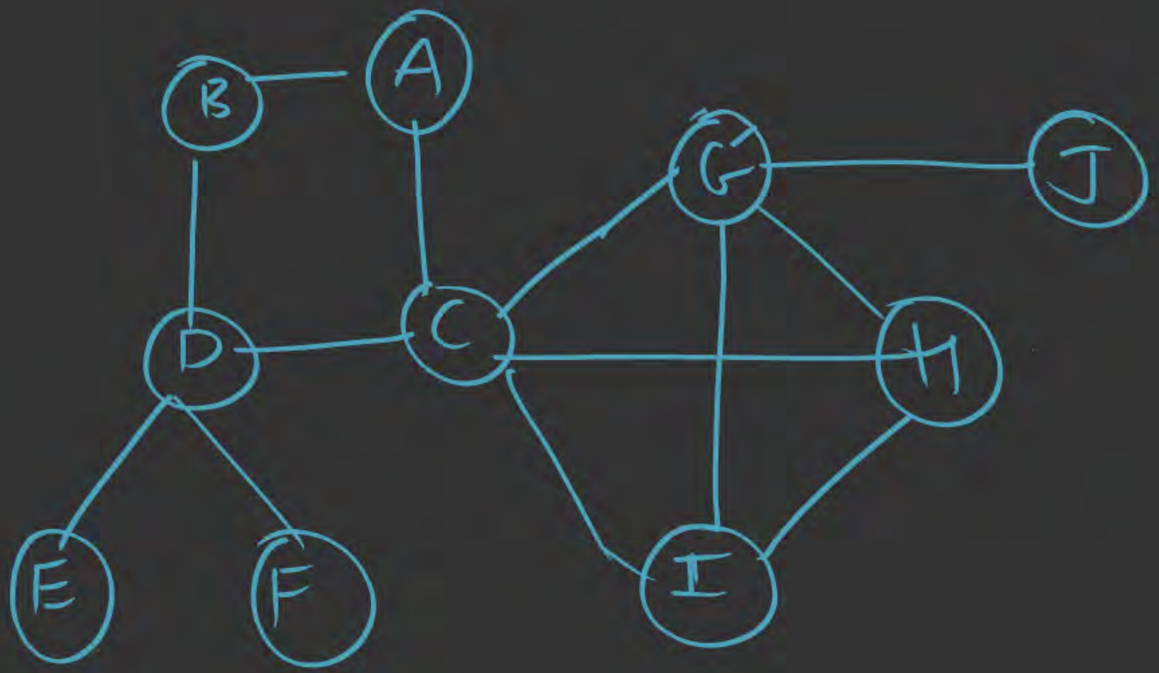
given

Graph: adj list

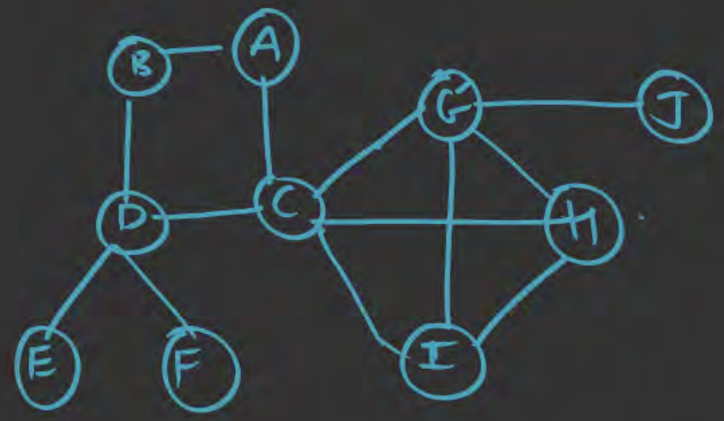
A → [B, C, D]
B → [A, E, F, C]
C → [A, B, G, H, D]
D → [A, C, I, J]
E → [B, K, L]
F → [B, G, M]
G → [C, F, N]
H → [C, I, O]
I → [D, H]
J → [D, O]
K → [E]
L → [E]
M → [F]
N → [G]
O → [H, J]



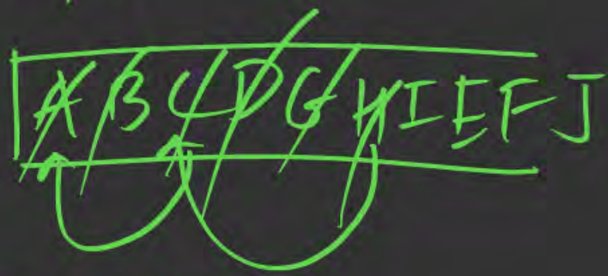
(9) BFS start at A.
Two Goal states: H & I. which will be reached First?



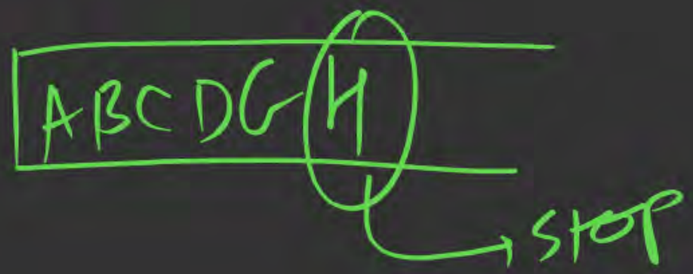
(Q) BFS start at A.
Two Goal states: H & I. which will be reached First? (alpha order)



BFS: open



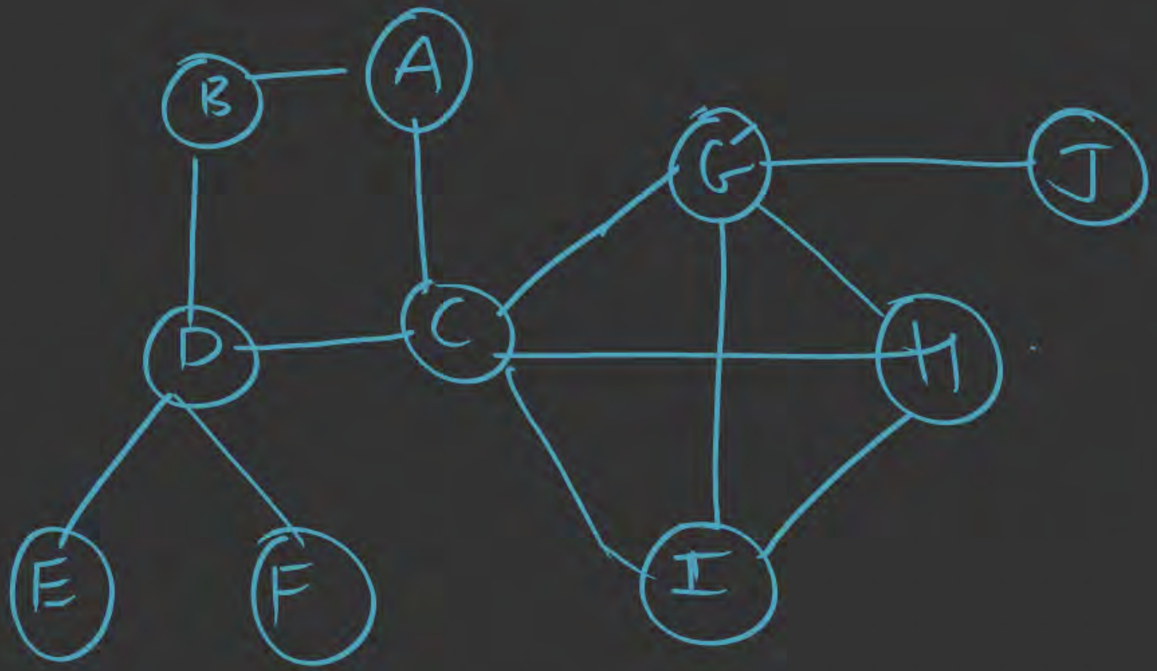
Closed

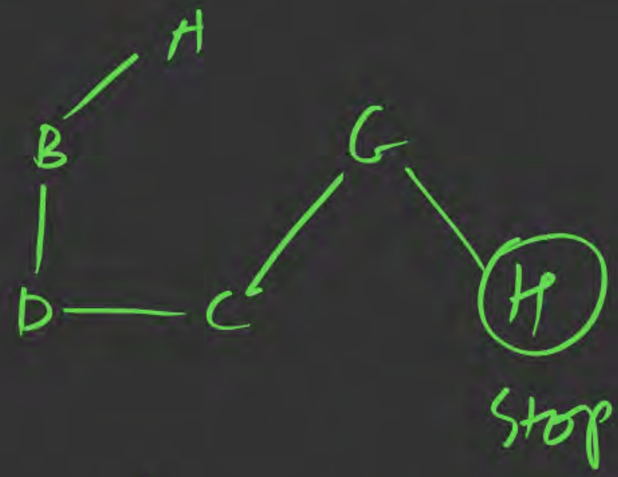


Ans: H

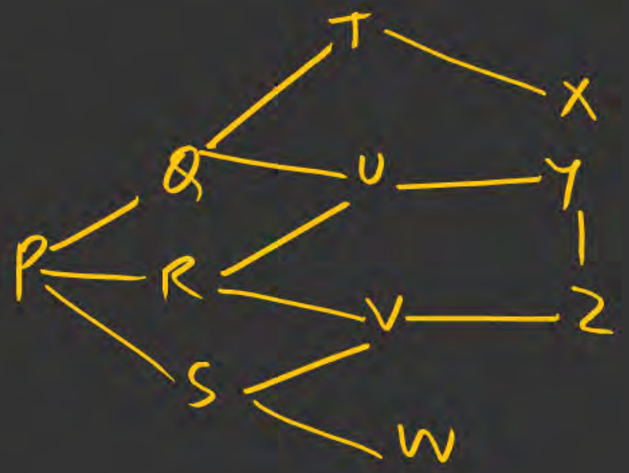
A → H.

(9) DFS start at A.
Two Goal states: H & I. which will be reached First?





Ans - H



Start P:

Goal X, Z.

Which First?

1) BFS

open

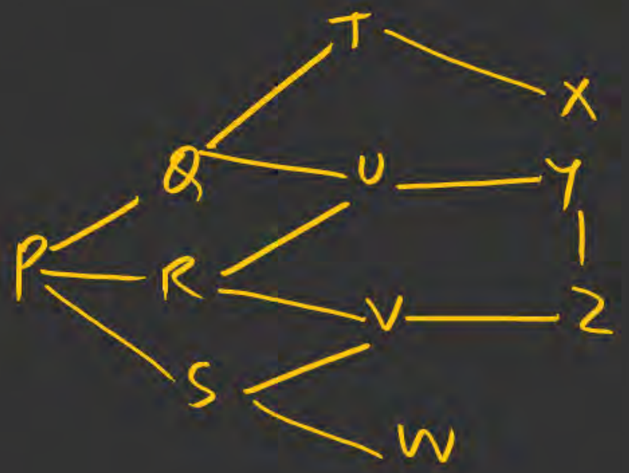
P Q R S T U V W X Y Z

Closed:

P Q R S T U V W X

goal

Ans - X

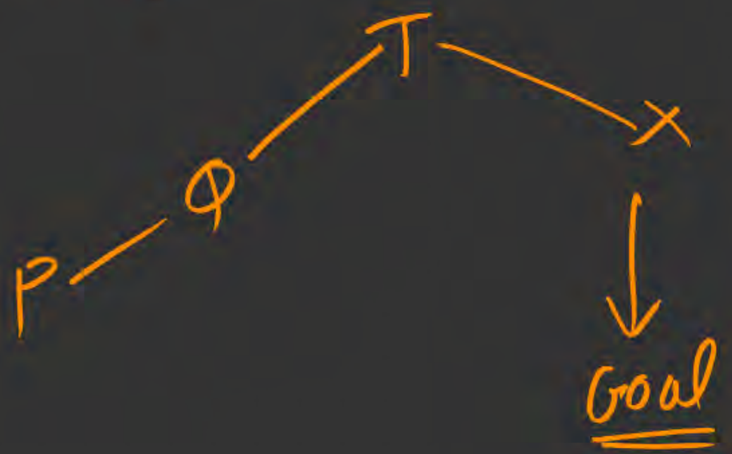


Start P:

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: → no. of nodes to be visited/(closed list) before finding goal.

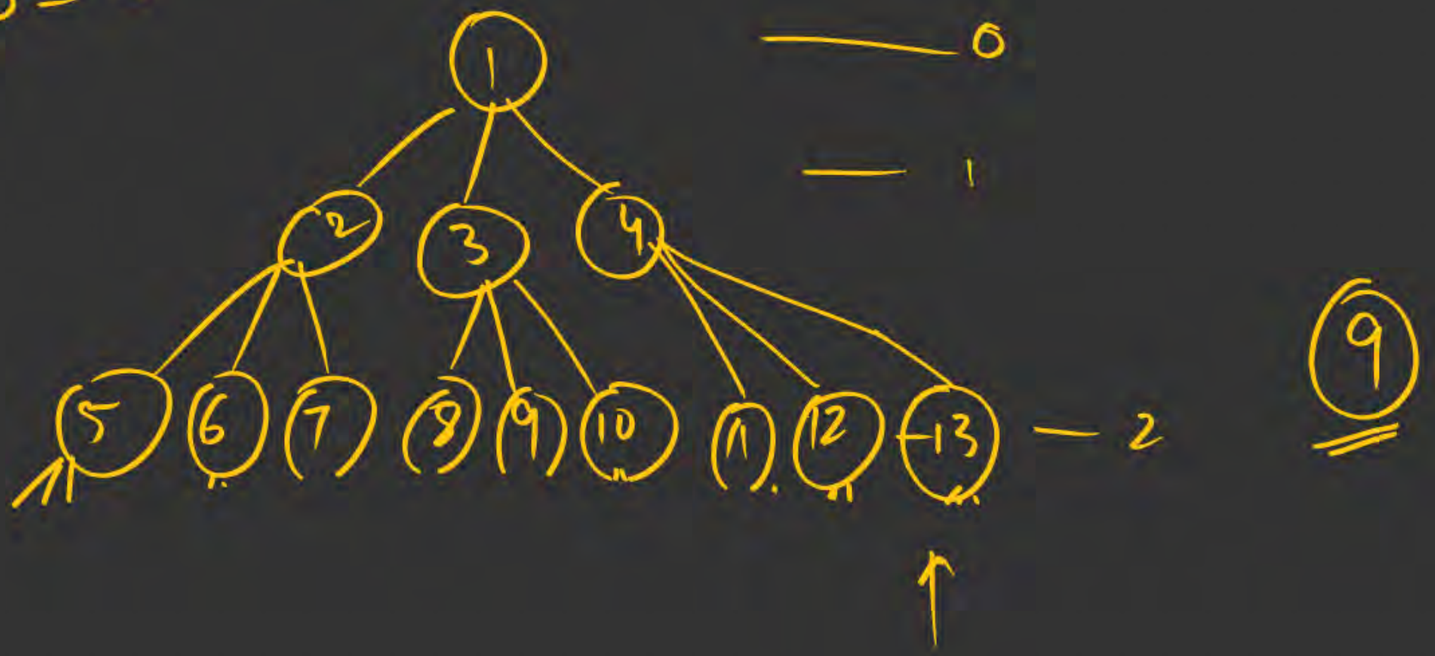
SC: No. of nodes stored in open list before finding goal

Tree: depth → d

branching factor → b

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

} Total nodes : $O(b^d)$



9

SC : $O(b^d)$

TC : $O(b^d)$

DFS

d → b

eg:- d = 2
b = 3

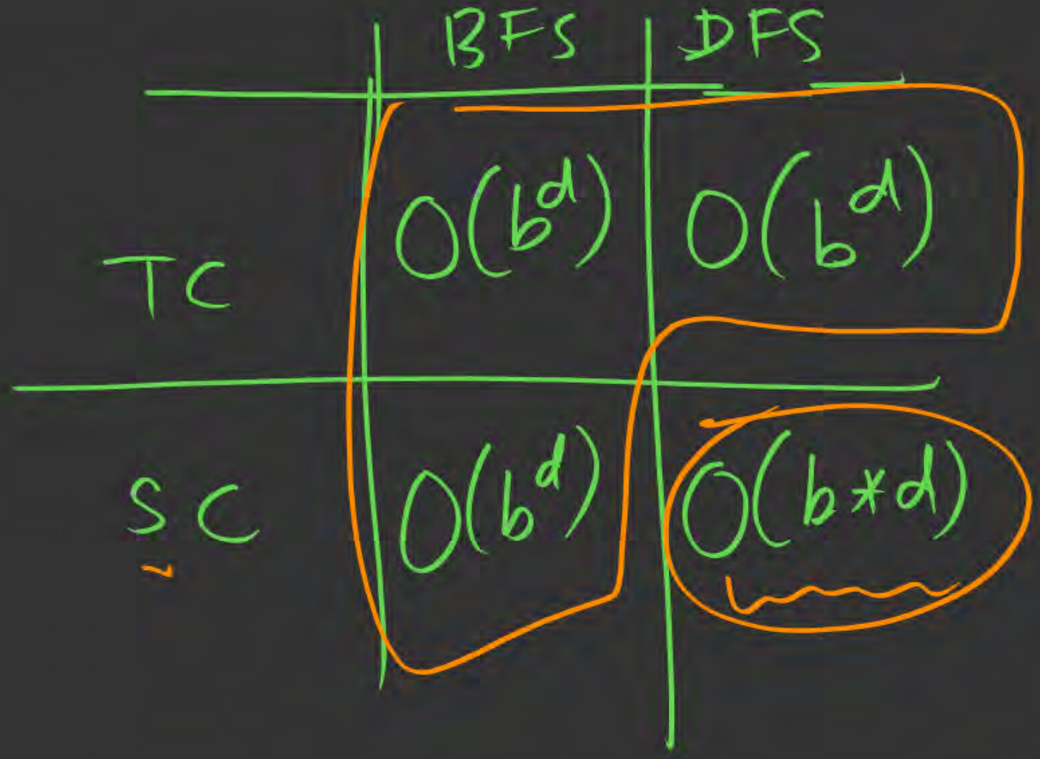
Total nodes : $O(b^d)$

43	765

SC : $O(b*d)$

TC : $O(b^d)$

Summary:



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:

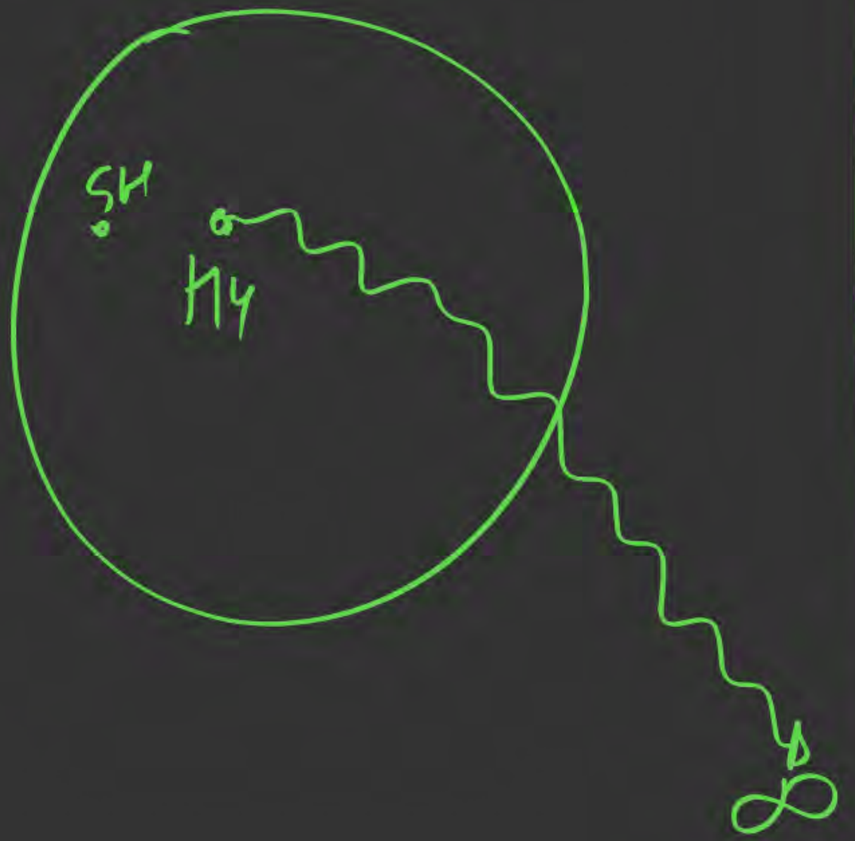
- There is the possibility that many states keep re-occurring, and there is no guarantee of finding the solution.
- DFS algorithm goes for deep down searching and sometime it may go to the infinite loop.

Topic : Uninformed Search

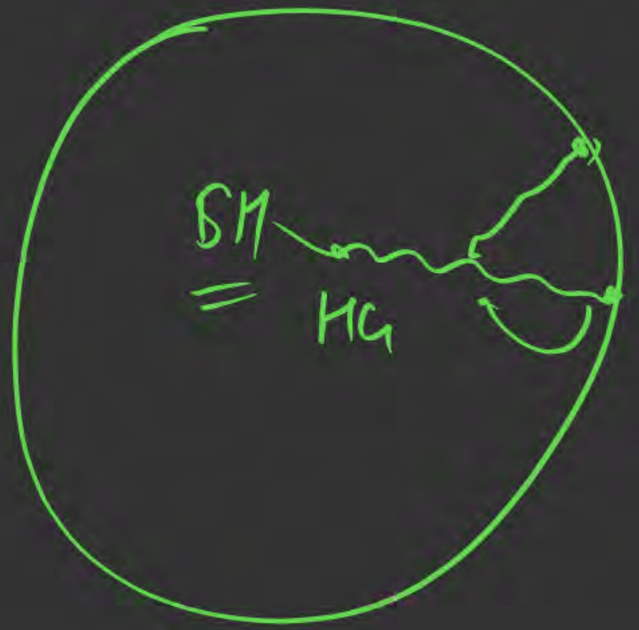
Depth Limited Search (DLS)

- 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.

DFS



DLS



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

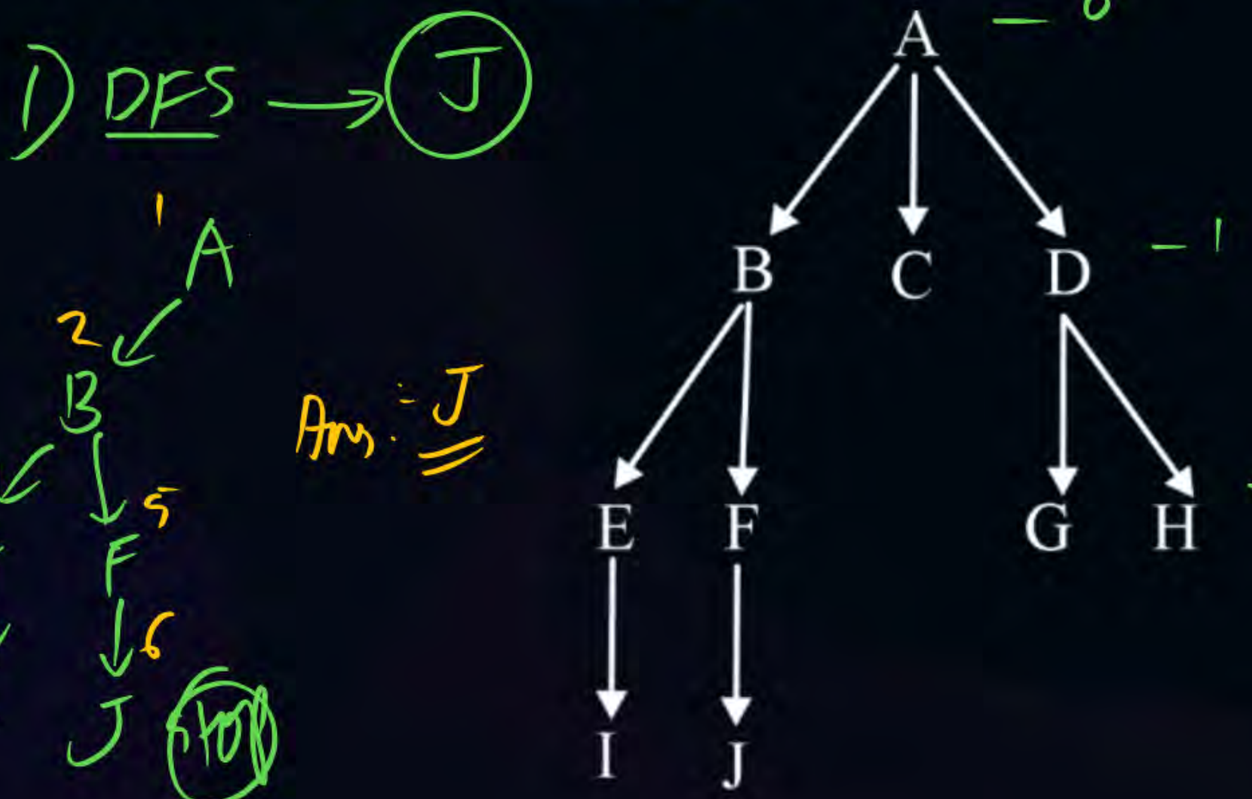
Depth Limited Search (DLS)

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

Apply DFS and DLS with depth limit 2.

(Root → d=0)

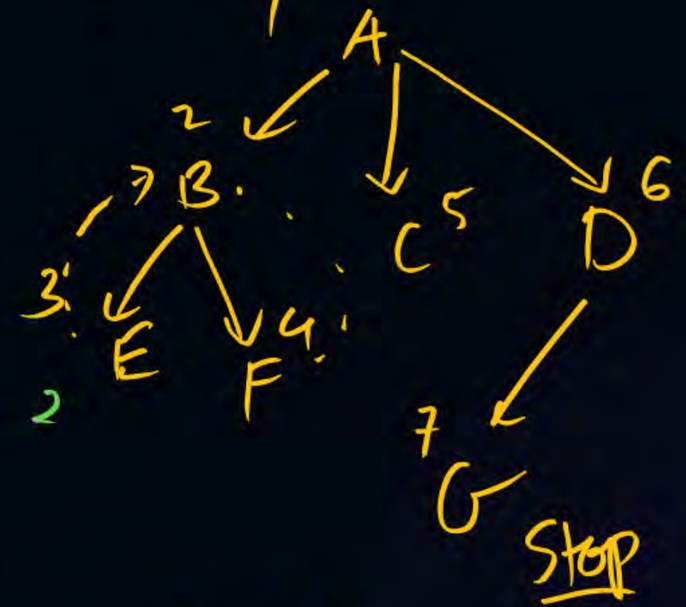
start at A



1) DFS → (J)

Ans: J

2) DLS



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 I > d
- 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