DS & AI ENGINEERING

Artificial Intelligence

Un-Informed search

Lecture No.- ०२

By- Aditya sir

Recap of Previous Lecture

- Topic
- Topic — Intro
- Topic — Uninformed basics
- Topic

Topics to be Covered

- Topic
- Topic — BFS ☆
- Topic — 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 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

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.

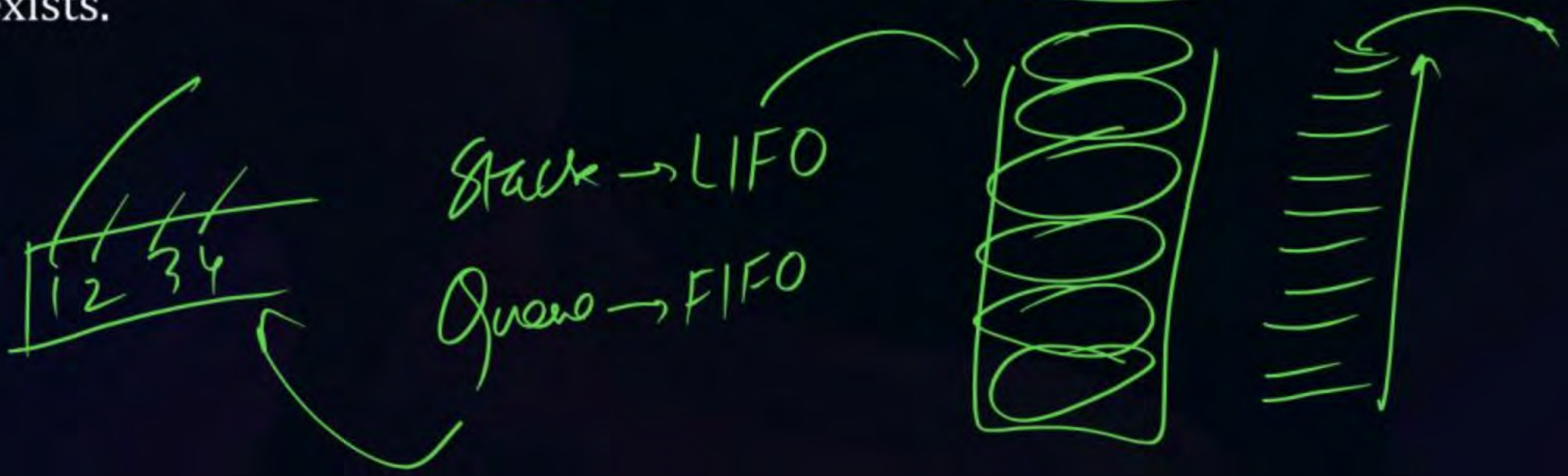
or uni-weighted

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.

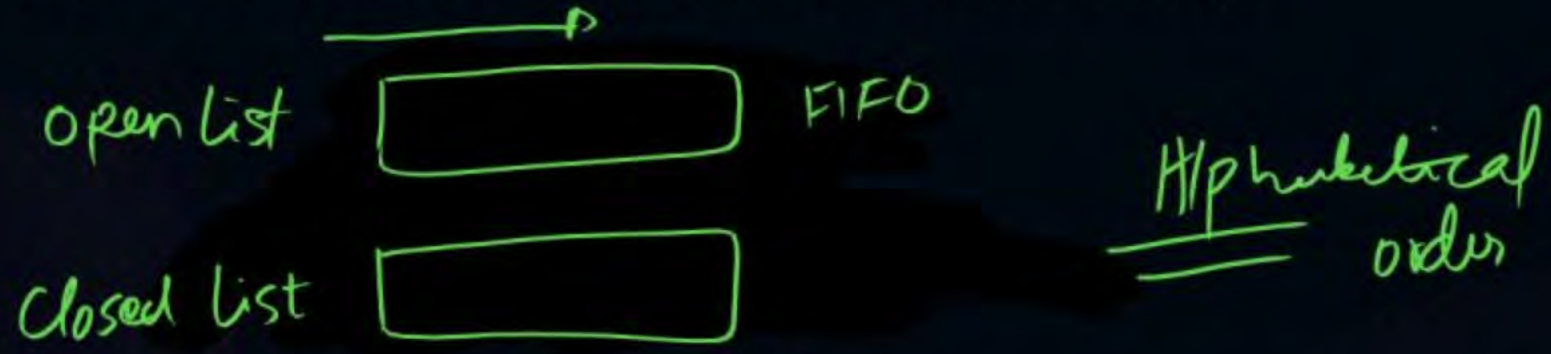


Topic : Uninformed Search

Breadth first search (Example)

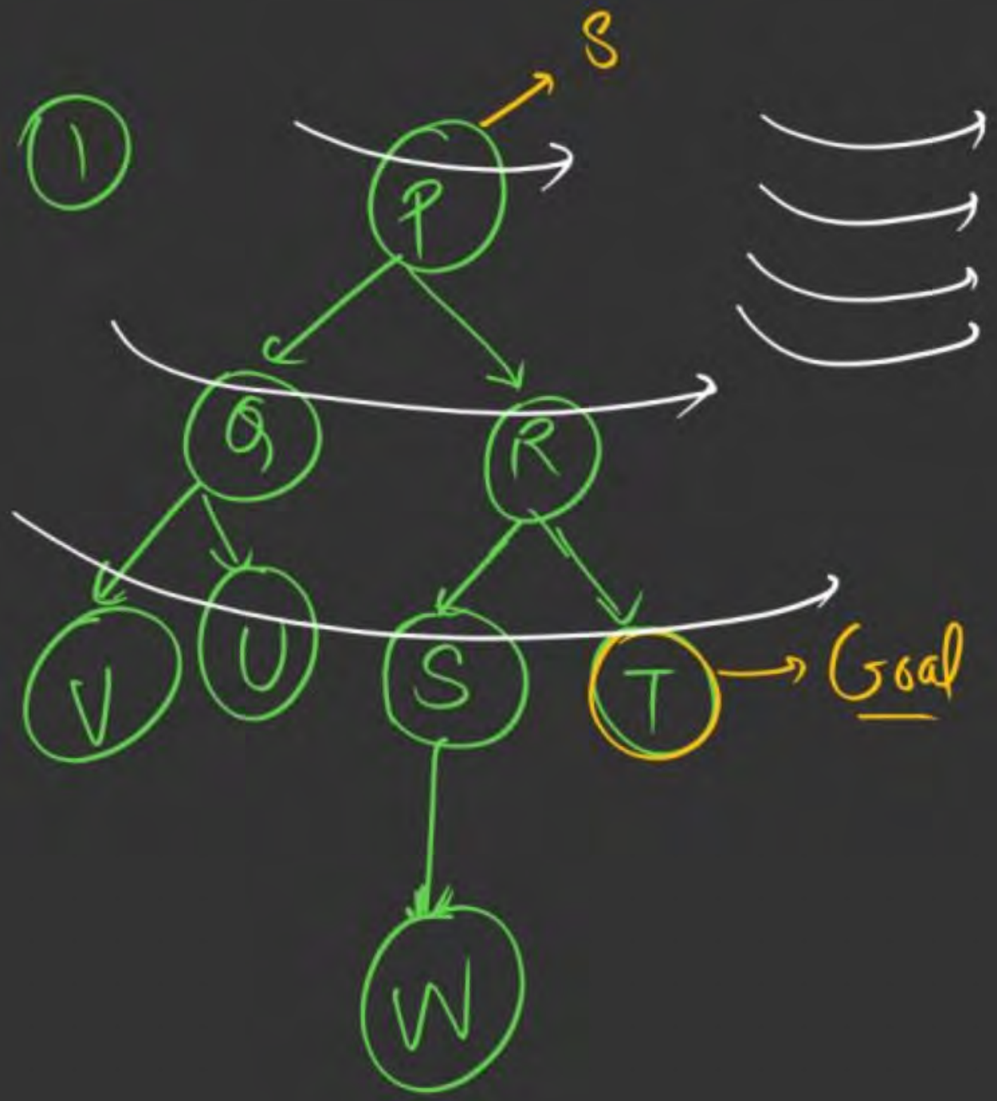
- BSF $\Rightarrow$ S $\longrightarrow$ G, We create 2 List

Step1: 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 Reached



Alphabetized

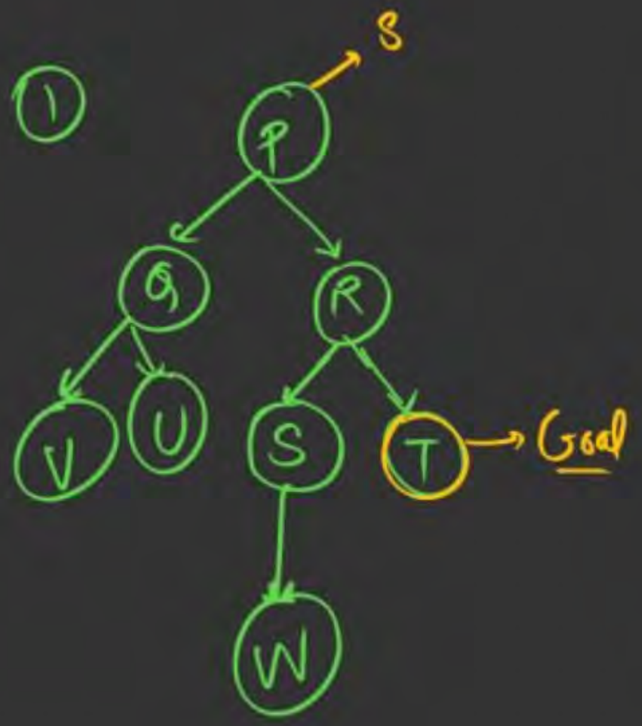
open (FIFO)

P ~~Q~~ ~~R~~ U ~~V~~ ~~S~~ ~~T~~ W

Closed

P Q R U V S T

Stop Algo



Alphabetized

open (FIFO)

P ~~Q R U V S T~~ W

Closed

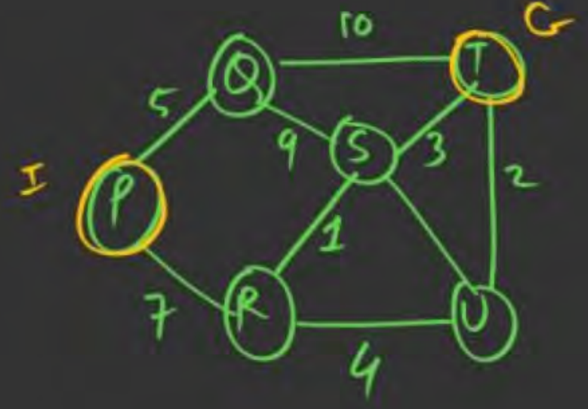
P Q R U V S T

Stop Algo

Shortest path: P → T

path: [P → R → T]

(2)



Open

P Q R S T U

Closed

P Q R S T

Stop

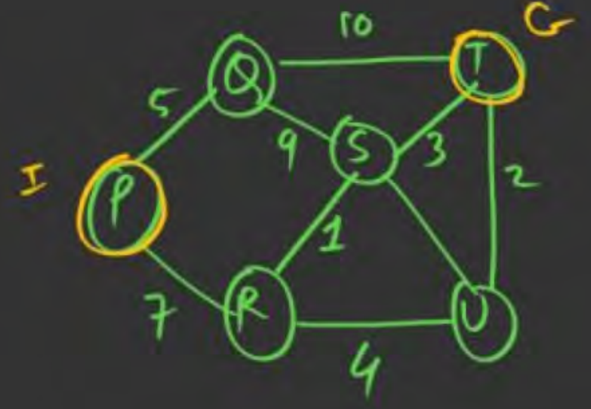
[P → Q → T]

Shortest path : P → T

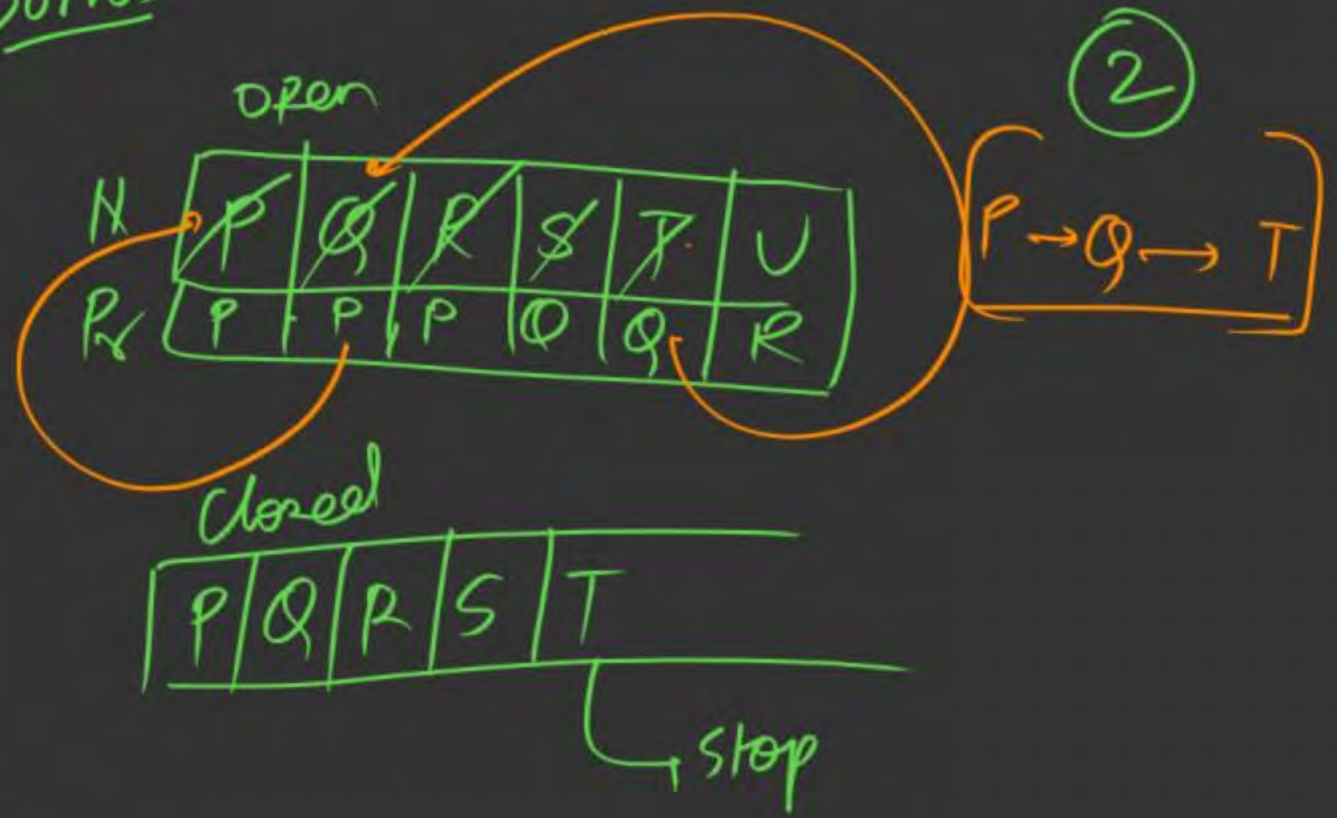
1) P R S T
2) P Q T
3) P R U T

Shortest path : path with min no. of edges

(2)



Bonus



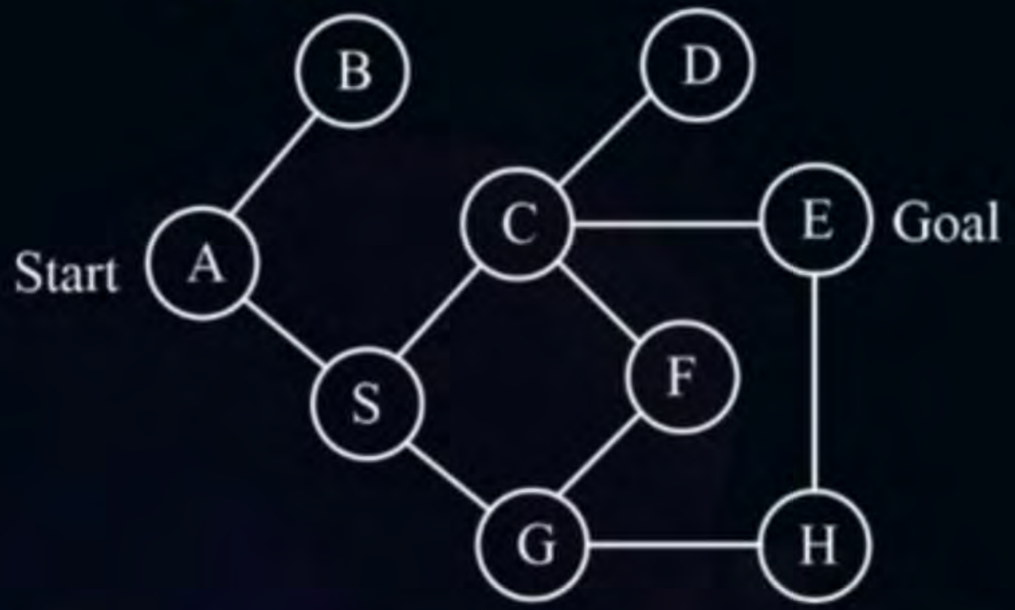
Shortest Path : P → T

1) PRST
2) PQT
3) PRUT

Topic : Uninformed Search

Let enter ~~ondes~~ nodes in alphabetical order

A → E. [A → S → C → E]



Visited (closed)

A B S C G D E |Stop.

path len = 3

path : A S C E

Queue

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

open

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

Topic : Uninformed Search

Bonus

Breadth first Search

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

Closed → open

- 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.

Topic : Uninformed Search

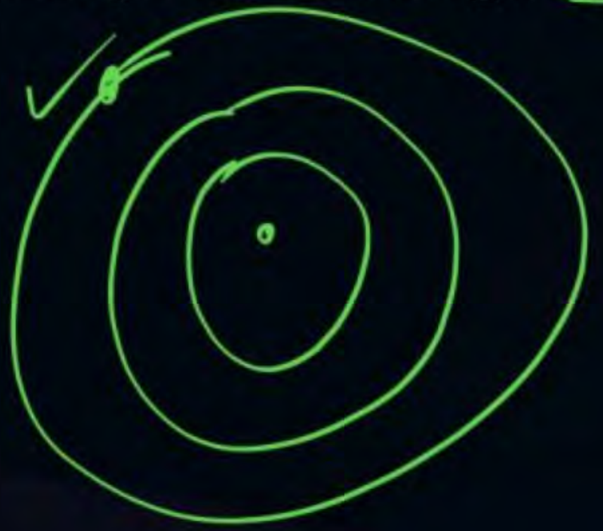
Breadth first Search

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

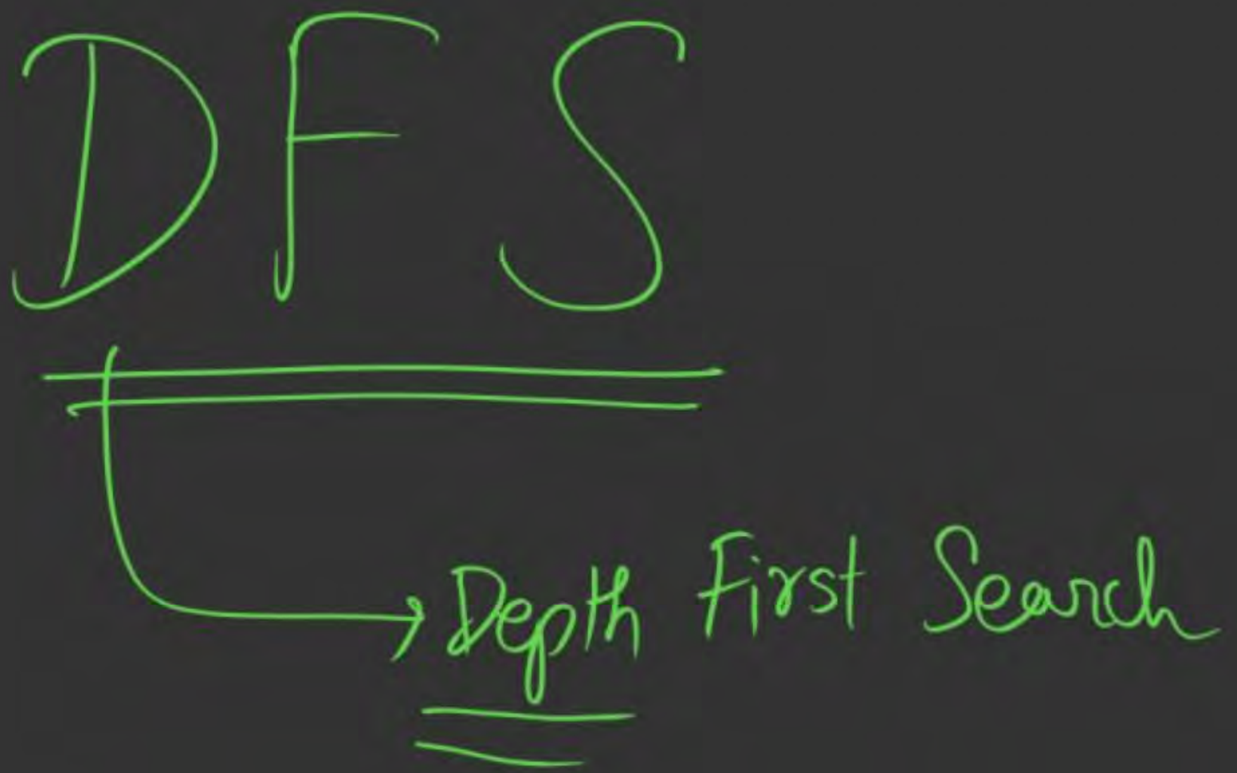
Breadth first Search

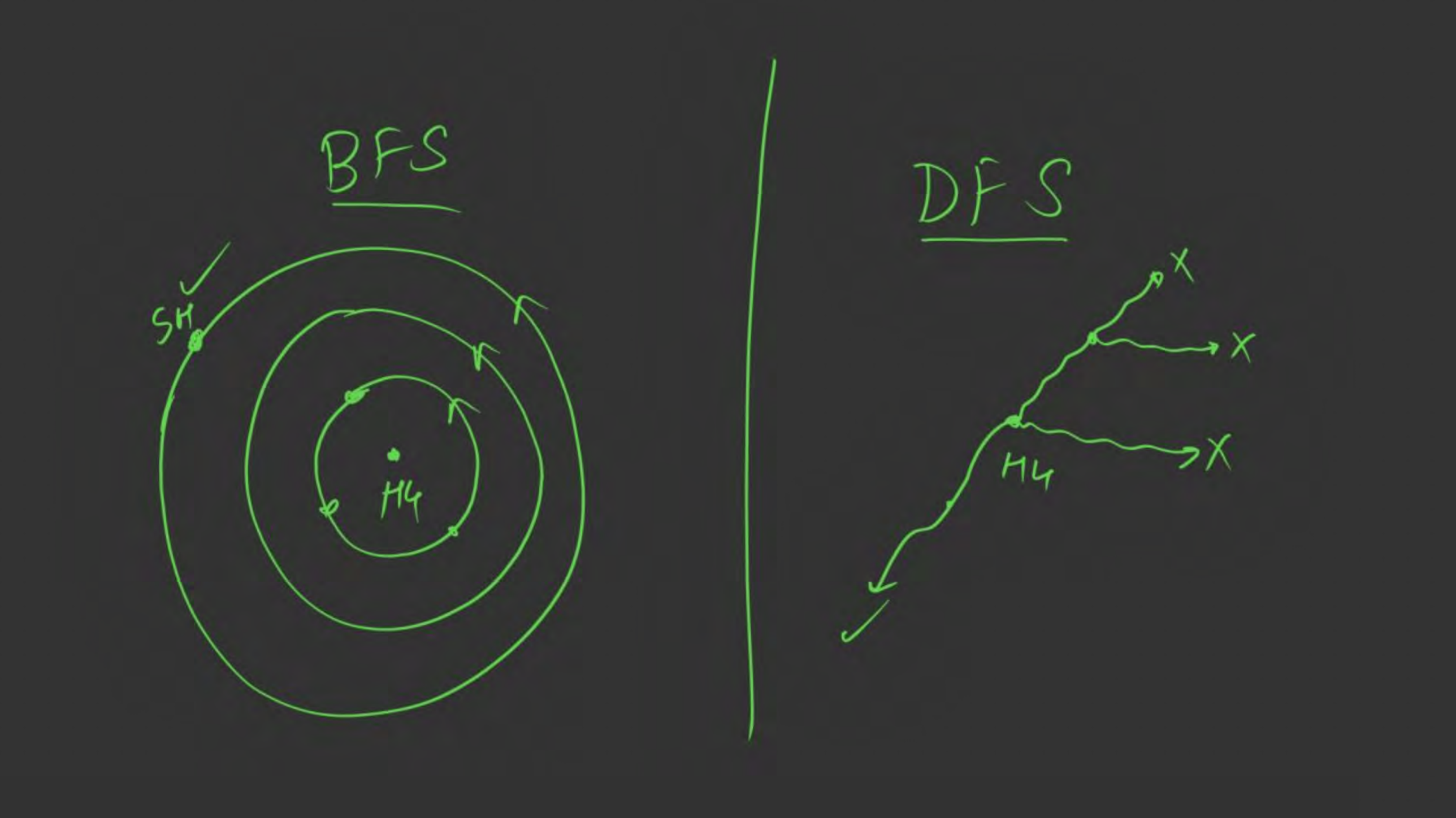
Disadvantages

- 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

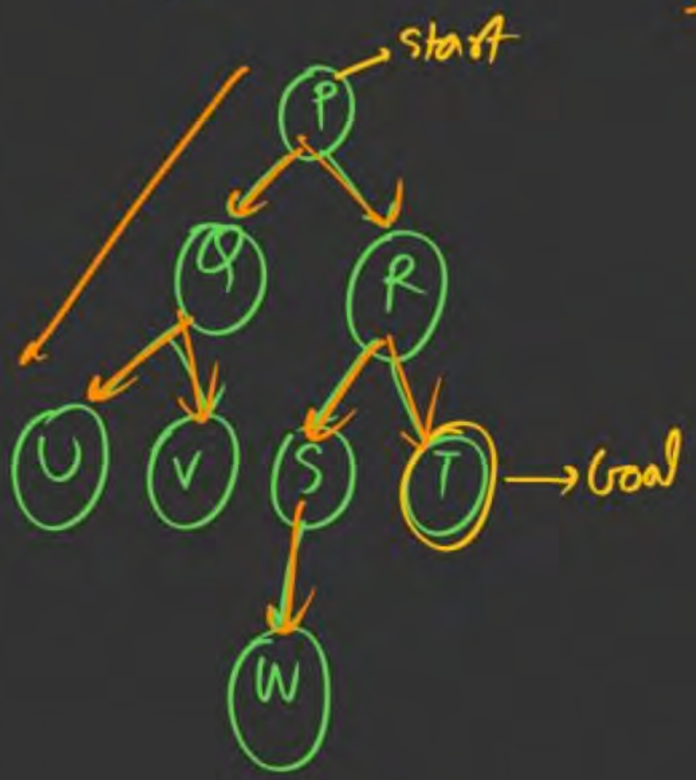
- This algorithm always give a result (if it exists)
- Because before removing the node form the queue, we check it with goal node
- And from root to the desired node, we can easily follow the parent – parent and get the path from root to the desired node.
- Path Reconstruction: To find the path from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


DFS
Depth First Search


BFS
SH
HG
DFS
X
X
X
HG

DFS

Alphabetical order



Open (LIFO)

P	~~P~~	~~Q~~	~~V~~	~~U~~	~~T~~	~~S~~	~~W~~	T

closed

P	Q	U	V	R	S	W	T

stop

THANK - YOU