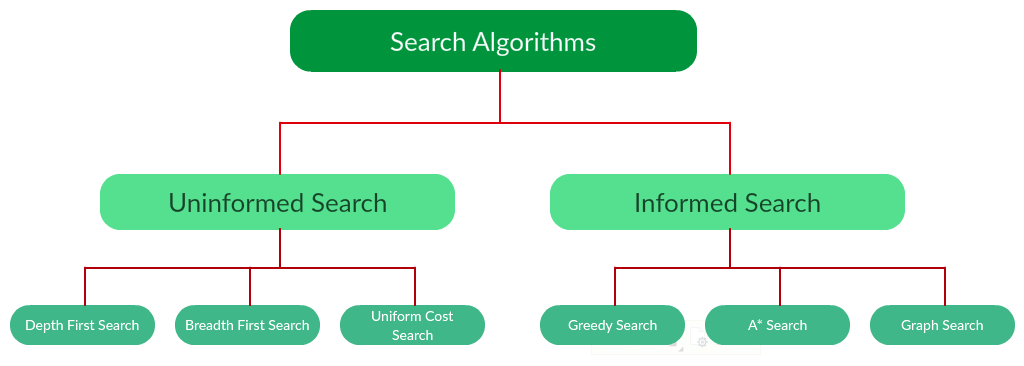
**Experiment No: 3**

**Aim :** To implement any uninformed search technique

**Theory :**

Search algorithms in AI are techniques used to explore and find solutions to problems by systematically searching through possible states or actions. They are foundational in fields like problem-solving, pathfinding, and decision-making. Here's an overview of some common search algorithms in AI



* Uninformed Search Algorithms:

Uninformed (or Blind) Search Algorithms explore the problem space without using any additional information or heuristics about the goal. They systematically visit and expand nodes to find a solution, typically used when no prior knowledge is available about the problem. Examples include Breadth-First Search (BFS), Depth-First Search (DFS), and Uniform Cost Search (UCS).

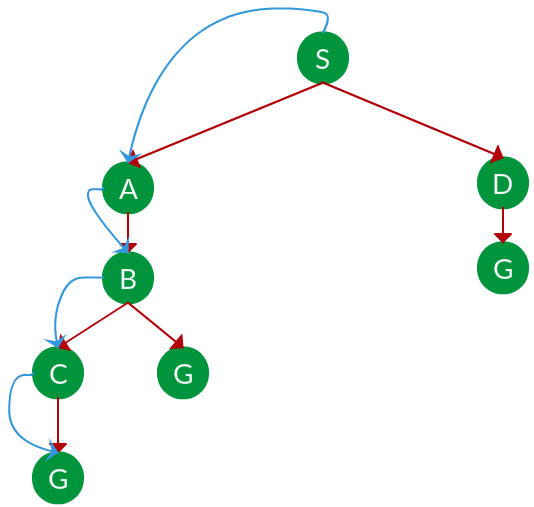
1. **Depth First Search:**

DFS explores as deep as possible along one path before backtracking to explore other branches. Essentially, it goes down a path until it hits a dead end (or the goal), and then it backtracks and explores the next available path.

DFS uses a stack (either explicitly or through recursion) to remember which nodes to return to after reaching a dead end.

Example:

The equivalent search tree for the above graph is as follows. As DFS traverses the tree “deepest node first”, it would always pick the deeper branch until it reaches the solution (or it runs out of nodes, and goes to the next branch). The traversal is shown in blue arrows.



Path:   S -> A -> B -> C -> G

1. **Breadth First Search:**

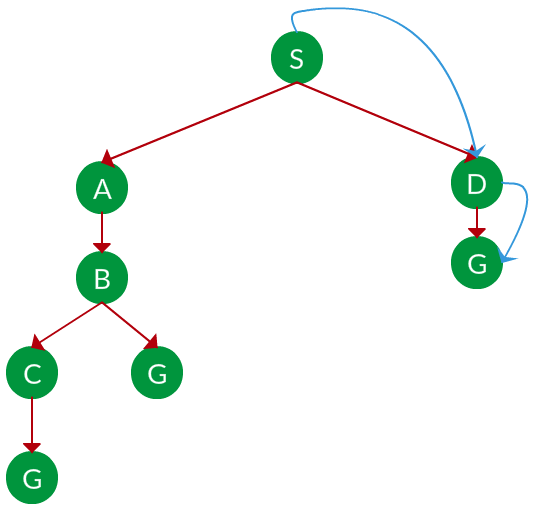
BFS explores all nodes at the present depth level before moving on to the next level. It starts from the initial node, explores all its neighbors, then explores the neighbors of those nodes, and so on.

BFS effectively expands all nodes at depth 0 first, then depth 1, depth 2, etc. in layers.

It uses a queue to keep track of nodes to be explored next, ensuring that it explores level by level.

Example:

The equivalent search tree for the above graph is as follows. As BFS traverses the tree “shallowest node first”, it would always pick the shallower branch until it reaches the solution (or it runs out of nodes, and goes to the next branch). The traversal is shown in blue arrows.



Path: S -> D -> G

1. **Uniform Cost Search:**

Uniform Cost Search (UCS) is an uninformed search algorithm that is a variant of BFS, but instead of expanding nodes level by level, UCS expands nodes based on the lowest cumulative cost to reach that node from the starting point.

UCS uses a priority queue (often implemented as a min-heap) to always explore the node with the lowest path cost first. The cost can vary between nodes, and UCS is designed to guarantee the shortest path in graphs where edges may have different weights or costs.

Program :

from collections import deque

def bfs(graph, start, goal):

queue = deque([start])

visited = set([start])

parent = {start: None}

while queue:

node = queue.popleft()

if node == goal:

path = []

while node is not None:

path.append(node)

node = parent[node]

return path[::-1]

for neighbor in graph[node]:

if neighbor not in visited:

visited.add(neighbor)

queue.append(neighbor)

parent[neighbor] = node

return None

graph = {

'A': ['B', 'C'],

'B': ['A', 'D', 'E'],

'C': ['A', 'F'],

'D': ['B'],

'E': ['B', 'F'],

'F': ['C', 'E']

}

start = 'A'

goal = 'F'

path = bfs(graph, start, goal)

print(graph)

print("Start from :",start)

print("Goal is :",goal)

print("Path to goal:", path)

Output :

