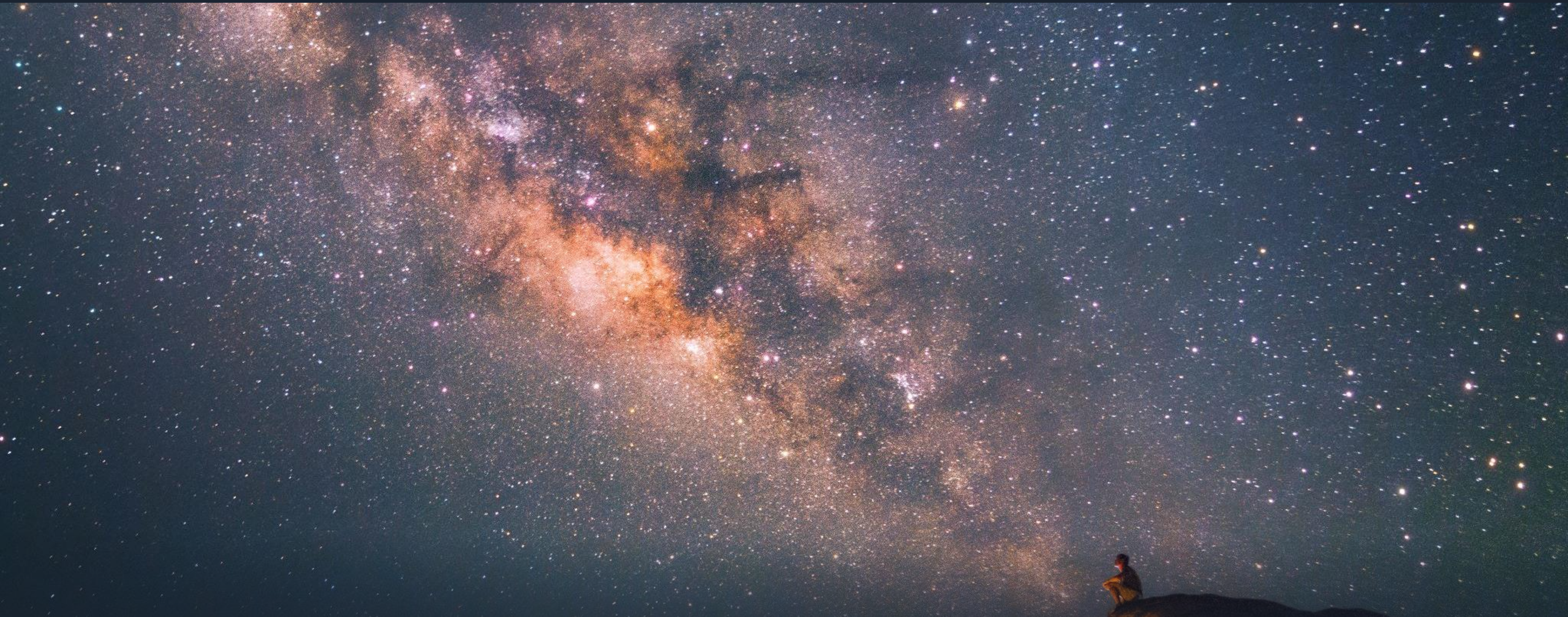
Module 1 Chapter 2

Artificial Intelligence

VBDS1402

Uninformed Search



Types of Search

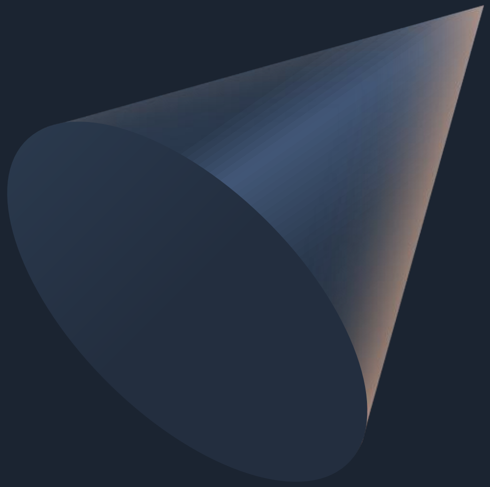
Uninformed Search

Informed Search

In this session you will learn:

Uninformed Search Methods

- Depth First Search
- Uniform Cost search



Uninformed Search (Blind Search)

The term means that the strategies have no additional information about states beyond that provided in the problem definition.

All they can do is generate successors and distinguish a goal state from a non-goal state.

Types of Uninformed Search

- Breadth First
- Depth – first
- Uniform Cost
- Depth Limited
- Iterative deepening DFS
- Bidirectional

Depth First Search (Uninformed/Blind Search)

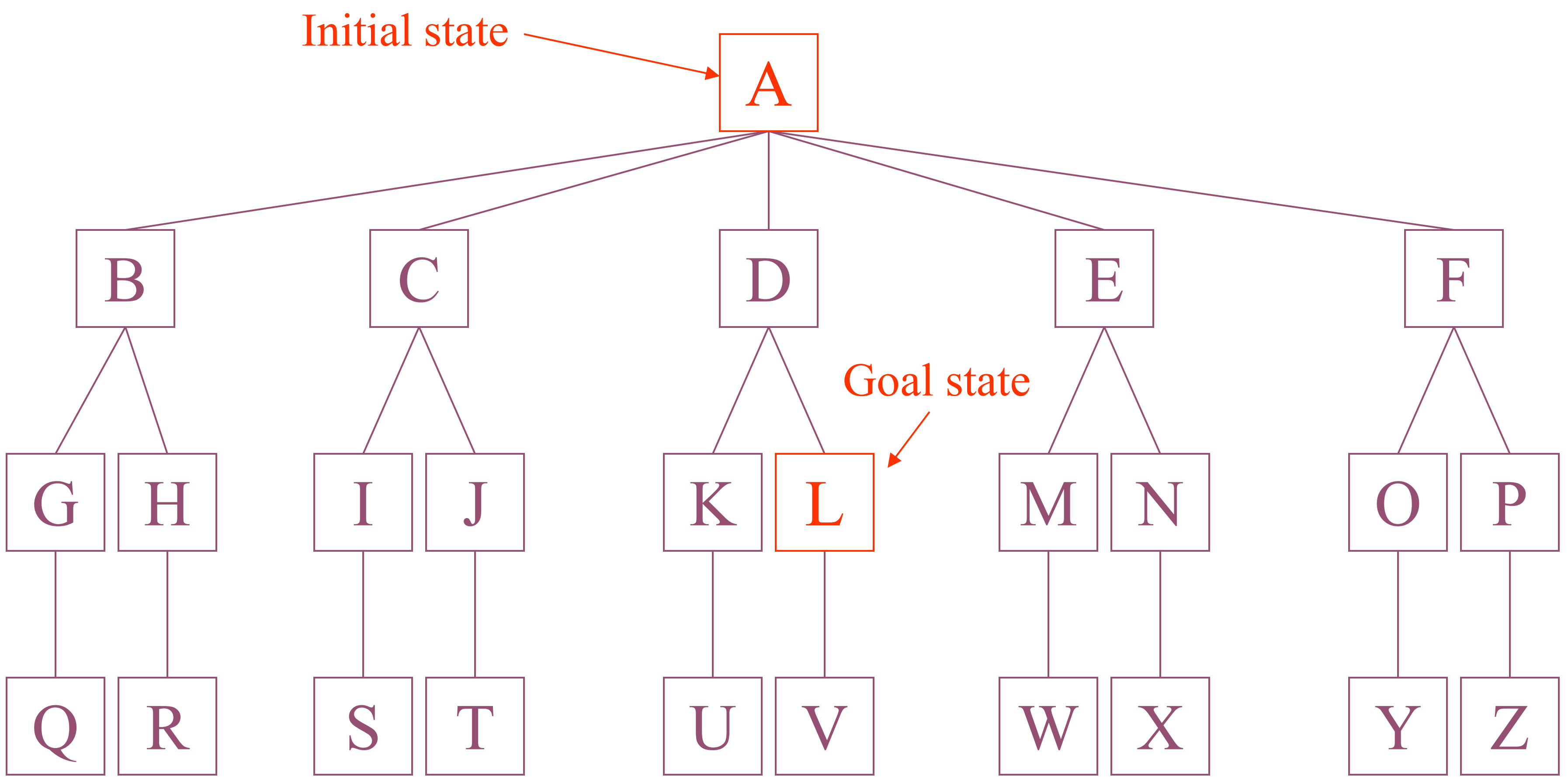
Depth-first search always expands the deepest node in the current frontier of the search tree.

The search proceeds immediately to the deepest level of the search tree, where the nodes have no successors.

The depth-first search algorithm is an instance of the graph-search algorithm and uses a LIFO queue.

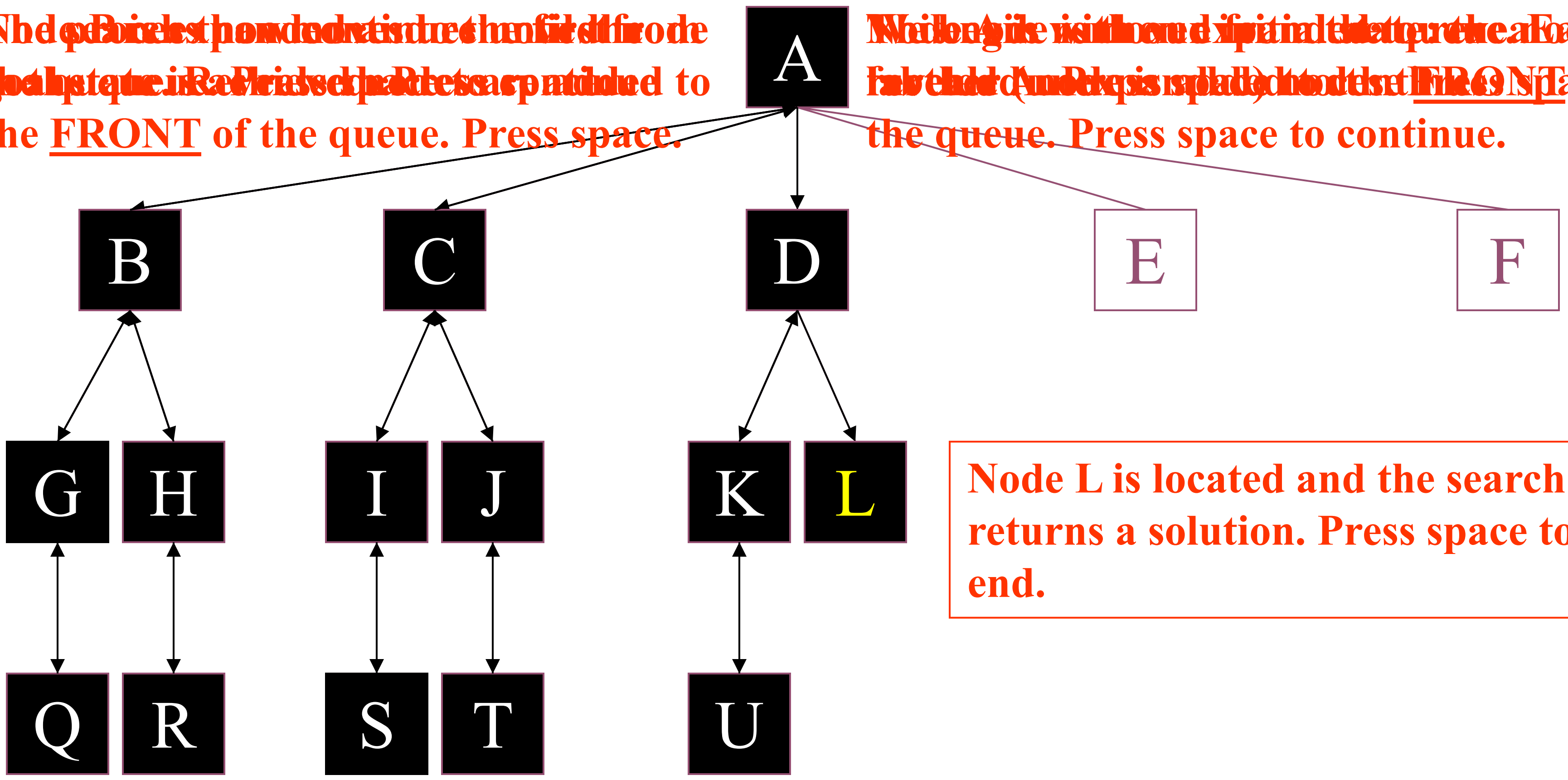
A LIFO queue means that the most recently generated node is chosen for expansion. This must be the deepest unexpanded node.

The example node set



Press space to see a DFS of the example node set [illegible] the FRONT of the queue. Press space.

[illegible] the queue. Press space to continue.



Node L is located and the search returns a solution. Press space to end.

Press space to [illegible]

Size of Queue: 0	Queue: Empty

Nodes expanded: 14	FINISHED SEARCH	Current level: 2

DEPTH-FIRST SEARCH PATTERN

Depth First Search (Uninformed/Blind Search)

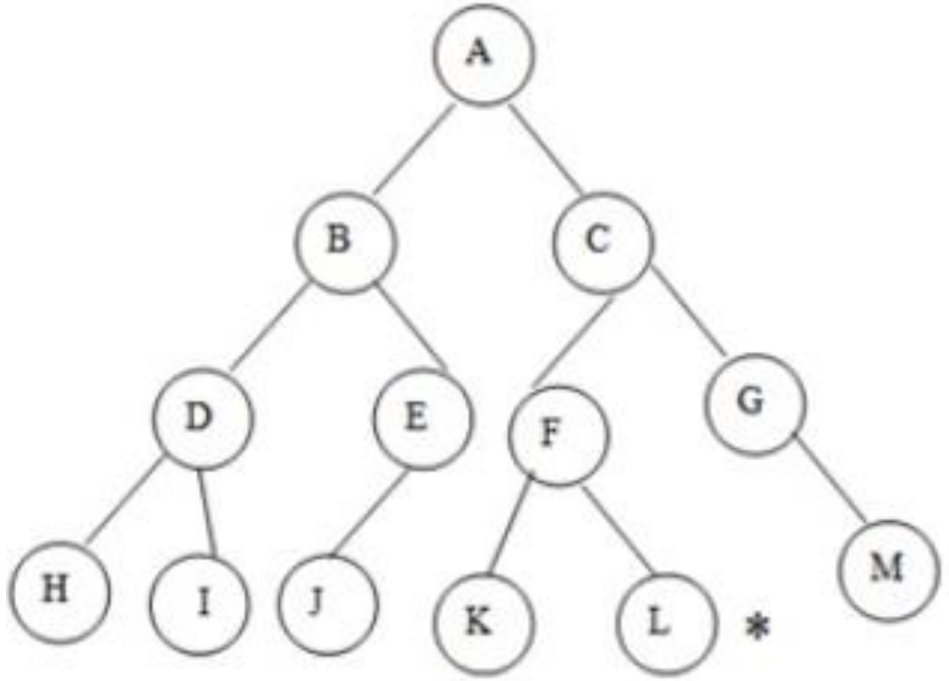


Figure Examples of DFS

- 1. open = [A]; closed = []
- 2. open = [B,C]; closed = [A]
- 3. open = [D,E,C]; closed = [A,B]
- 4. open = [H,I,E,C]; closed = [A,B,D]
- 5. open = [I,E,C]; closed = [A,B,D,H]
- 6. open = [E,C]; closed = [A,B,D,H,I]
- 7. open = [J,C]; closed = [A,B,D,H,I,E]
- 8. open = [C] closed = [A,B,D,H,I,E,J]
- 9. open = [F,G] closed = [A,B,D,H,I,E,J,C]
- 10 open = [K,L,G] closed = [A,B,D,H,I,E,J,C,F]
- 11 open = [L,G] closed = [A,B,D,H,I,E,J,C,F,K]
- 12 Next is L, Goal Node Reached

Depth first search algorithm

Non-recursive implementation of DFS:

- Push the root node on the stack.
- While(Stack is not empty)
    - Pop a node from the stack;
    - If node is a goal node then return success;
    - Push all children of node onto the stack;
- Return failure.

Recursive implementation of DFS:

- If node is a goal, return success;
- For each child c is node
    - If DFS(c) is successful,
    - Return success.
- Return failure

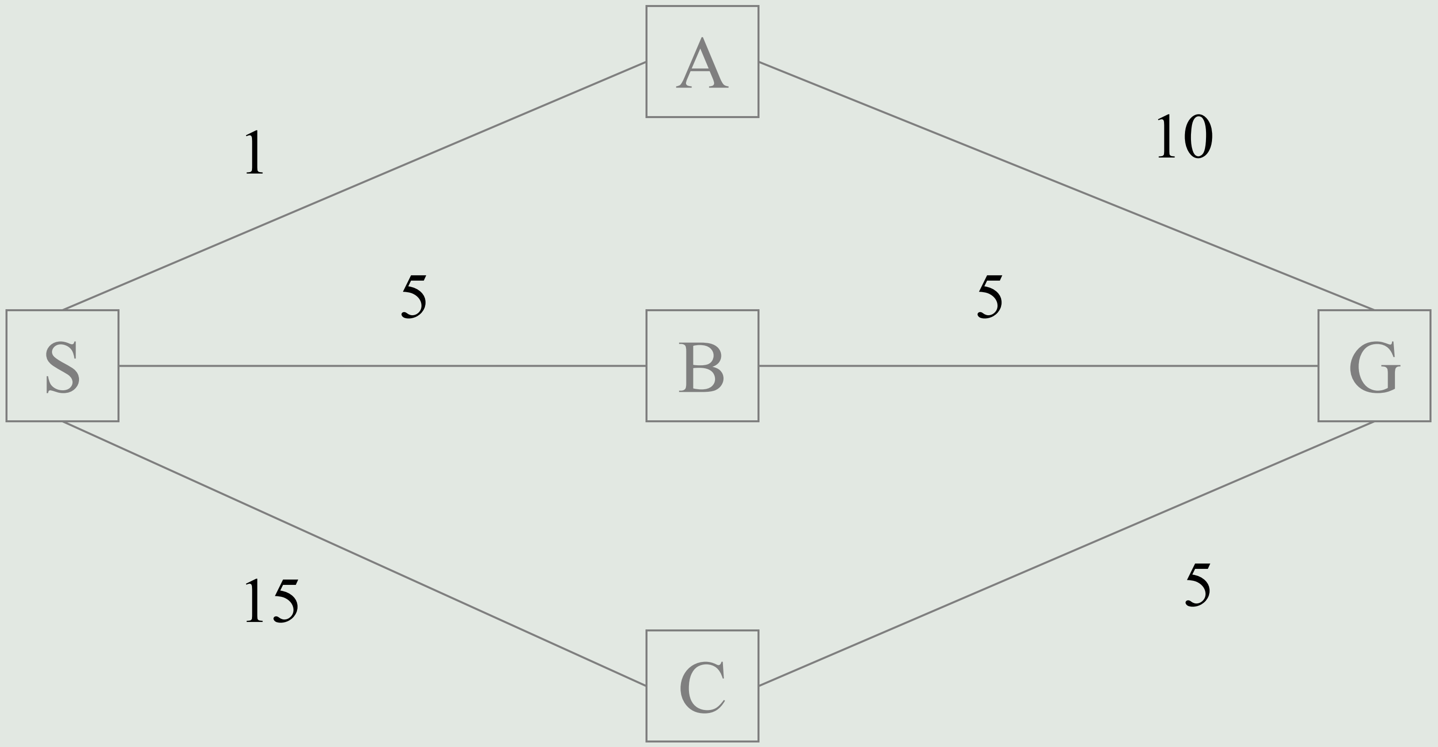
Applications of Depth-First Search Algorithm

- Detecting cycle in a graph
- Path Finding
- Topological Sorting
- To test if a graph is bipartite
- Finding Strongly Connected Components of a graph
- Solving puzzles with only one solution, such as mazes.

Uniform Cost (Uninformed/Blind Search)

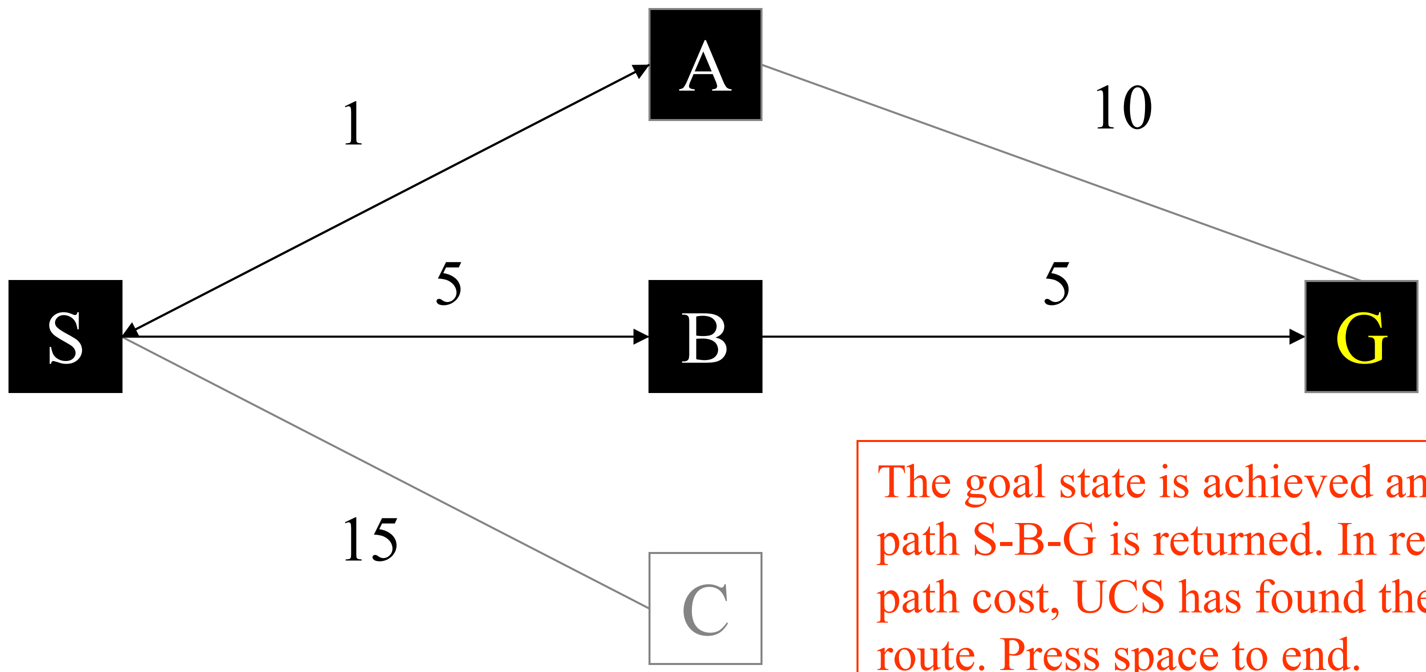
- Uniform-cost search is a searching algorithm used for traversing a weighted tree or graph.
- This algorithm comes into play when a different cost is available for each edge.
- The primary goal of the uniform-cost search is to find a path to the goal node which has the lowest cumulative cost.
- Uniform-cost search expands nodes according to their path costs form the root node.
- It can be used to solve any graph/tree where the optimal cost is in demand.
- A uniform-cost search algorithm is implemented by the priority queue. It gives maximum priority to the lowest cumulative cost.
- Uniform cost search is equivalent to BFS algorithm if the path cost of all edges is the same.

Consider the following problem…



We wish to find the shortest route from node S to node G; that is, node S is the initial state and node G is the goal state. In terms of path cost, we can clearly see that the route *SBG* is the cheapest route. However, if we let breadth-first search loose on the problem it will find the non-optimal path *SAG*, assuming that A is the first node to be expanded at level 1. Press space to see a UCS of the same node set…

[illegible]
proceed to goal state. Press space.



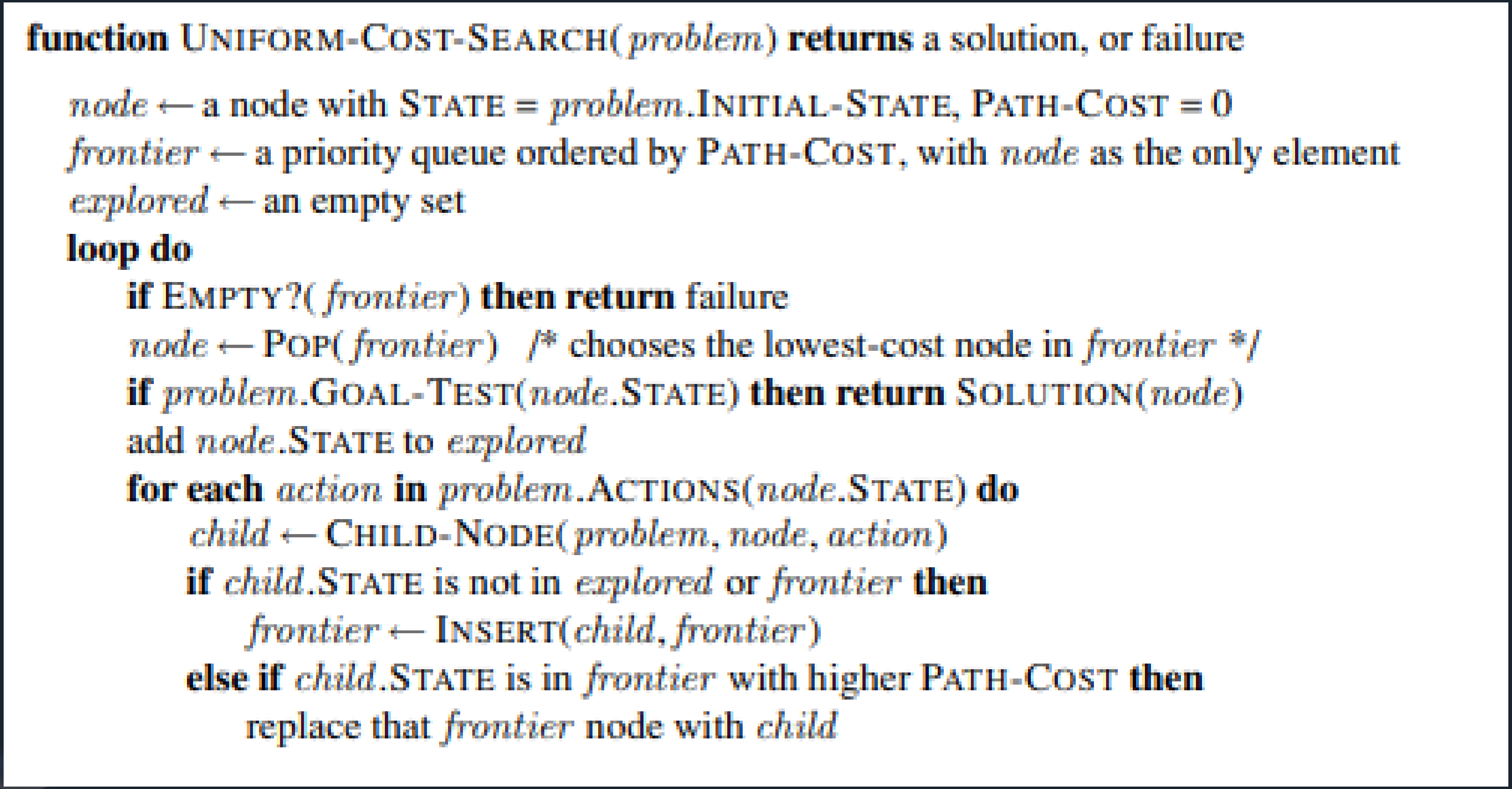
The goal state is achieved and the path S-B-G is returned. In relation to path cost, UCS has found the optimal route. Press space to end.

Press space to begin the search

Size of Queue: 0	**Queue: Empty**	
Nodes expanded: 3	**FINISHED SEARCH**	**Current level: 2**

UNIFORM COST SEARCH PATTERN

Algorithm



```
function UNIFORM-COST-SEARCH(problem) returns a solution, or failure
  node ← a node with STATE = problem.INITIAL-STATE, PATH-COST = 0
  frontier ← a priority queue ordered by PATH-COST, with node as the only element
  explored ← an empty set
  loop do
      if EMPTY?(frontier) then return failure
      node ← POP(frontier)   /* chooses the lowest-cost node in frontier */
      if problem.GOAL-TEST(node.STATE) then return SOLUTION(node)
      add node.STATE to explored
      for each action in problem.ACTIONS(node.STATE) do
          child ← CHILD-NODE(problem, node, action)
          if child.STATE is not in explored or frontier then
              frontier ← INSERT(child, frontier)
          else if child.STATE is in frontier with higher PATH-COST then
              replace that frontier node with child
```

Uniform Search

Advantages & Disadvantages

ADVANTAGES

It helps to find the path with the lowest cumulative cost inside a weighted graph having a different cost associated with each of its edge from the root node to the destination node.

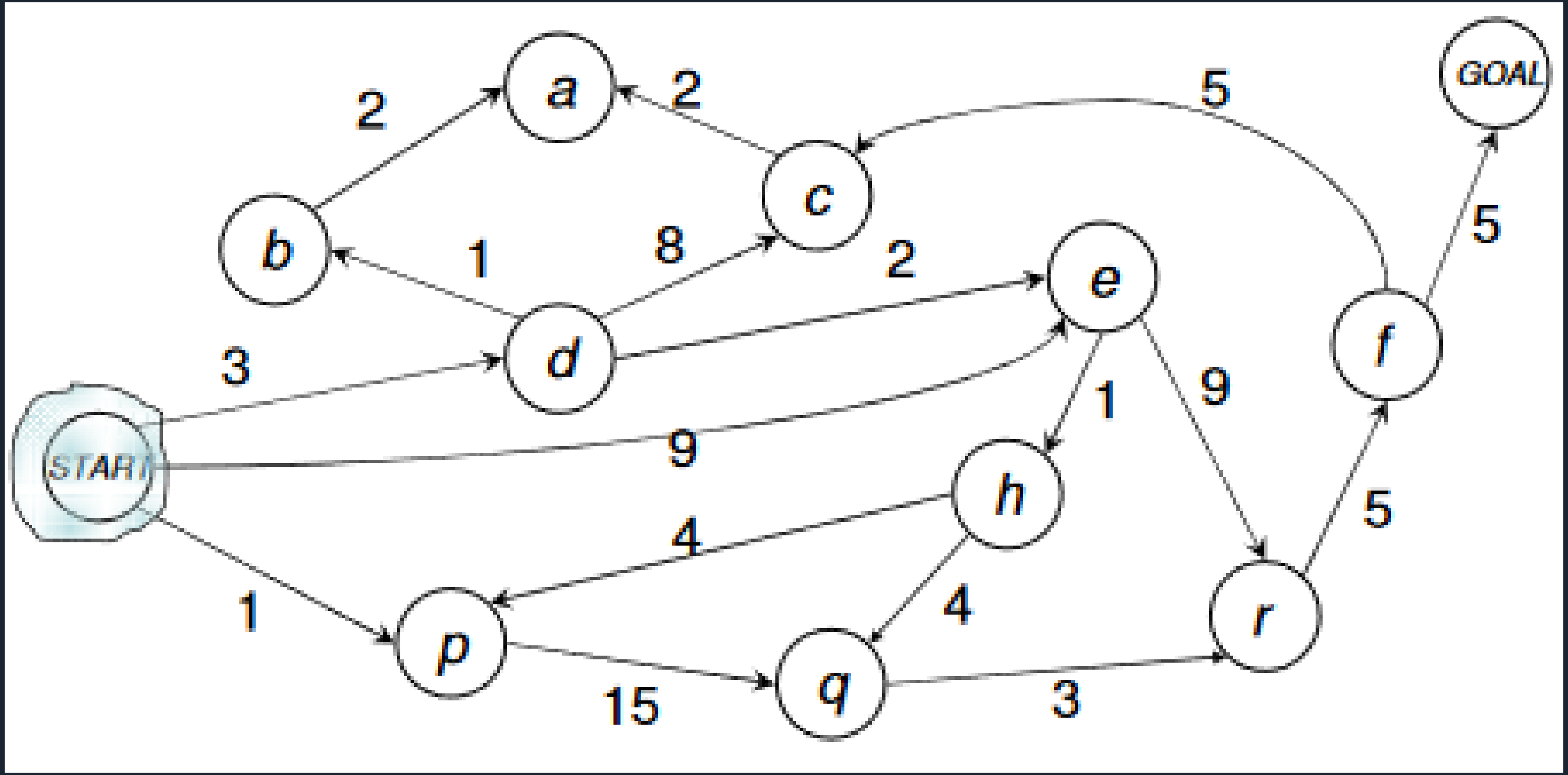
Uniform cost search is optimal because at every state the path with the least cost is chosen.

DISADVANTAGES

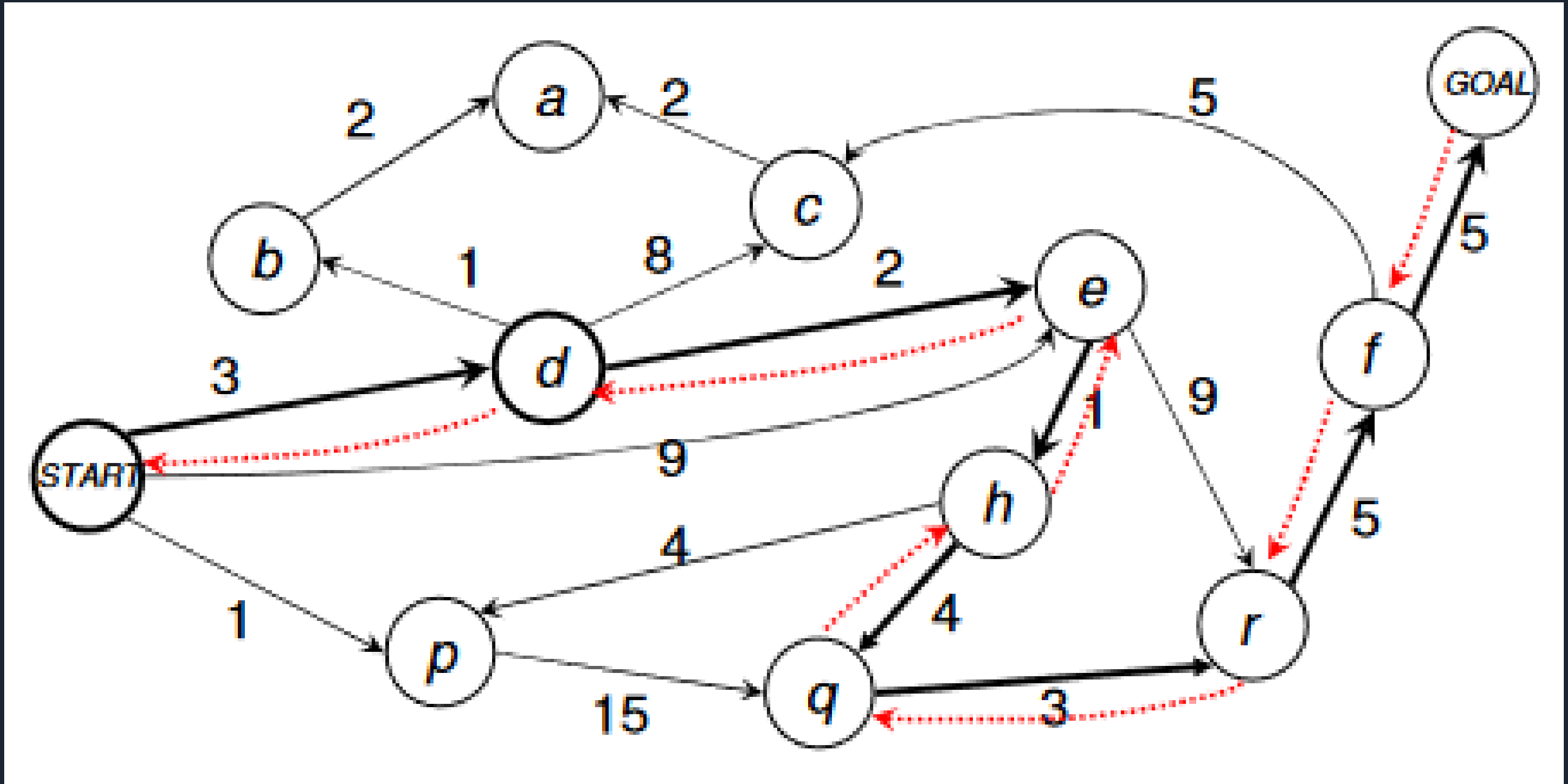
The open list is required to be kept sorted as priorities in priority queue needs to bemaintained.

The storage required is exponentially large.

It does not care about the number of steps involve in searching and only concerned about path cost. Due to which this algorithm may be stuck in an infinite loop.



EXERCISE



Final path: {*START, d, e, h, q, r, f, GOAL*}

ANSWER

Take care and Keep safe