Artificial Intelligence

Problem solving by searching: Uninformed Search

by
Abdul Sahli Fakharudin
Faculty Computer Systems & Software Engineering
sahli@ump.edu.my

Chapter Description

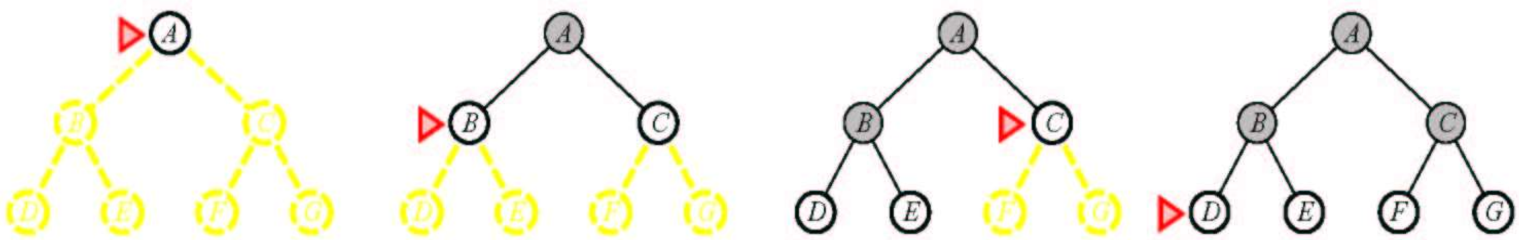
- Expected Outcomes
  - Student able to review the breadth firsts search, death first search, depth limited search, iterative deepening search and uniform cost search
  - Student able to analyse and apply the searches to solve a given problem

- References
  - ……..

Content #1

- What is artificial intelligence?
- History of artificial intelligence
- Example of artificial intelligence application

Breadth First Search (BFS)

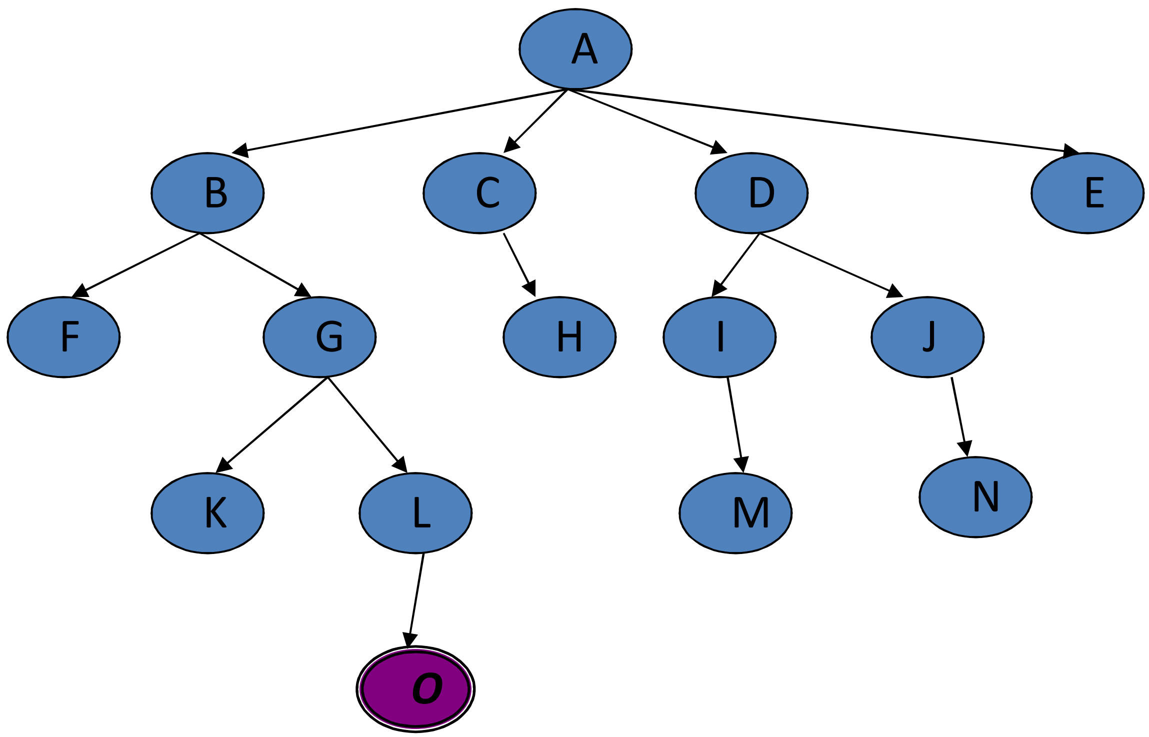


Main idea: Expand all nodes at depth (i) before expanding nodes at depth (i + 1) Level-order Traversal.

Implementation: Use of a First-In-First-Out queue (FIFO). Nodes visited first are expanded first. Enqueue nodes in FIFO (first-in, first-out) order.

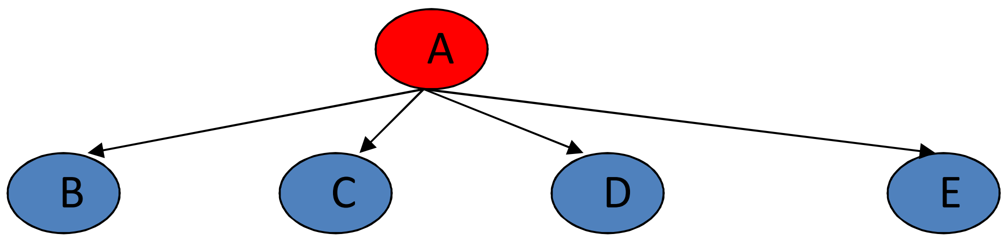
Breadth First Search

Given the following state space (tree search), give the sequence of visited nodes when using BFS (assume that the node ***O*** is the goal state):



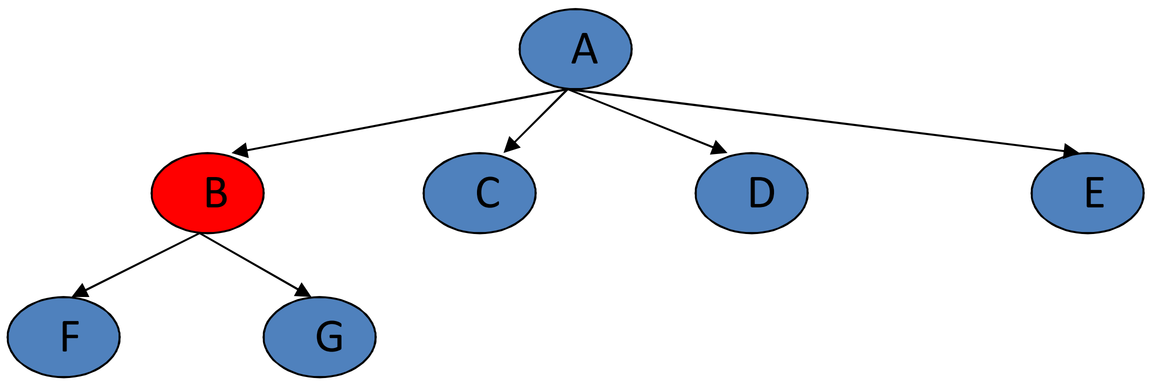
Breadth First Search

- A,



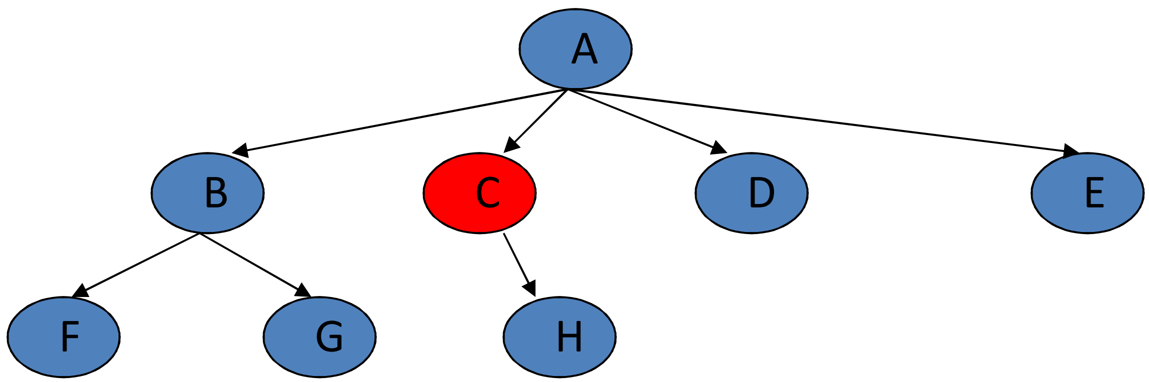
Breadth First Search

- A,
- B,



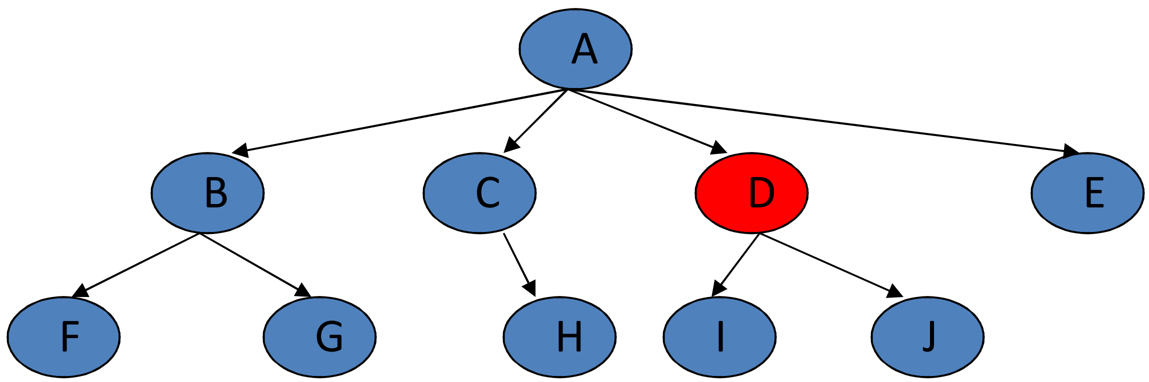
Breadth First Search

- A,
- B,C



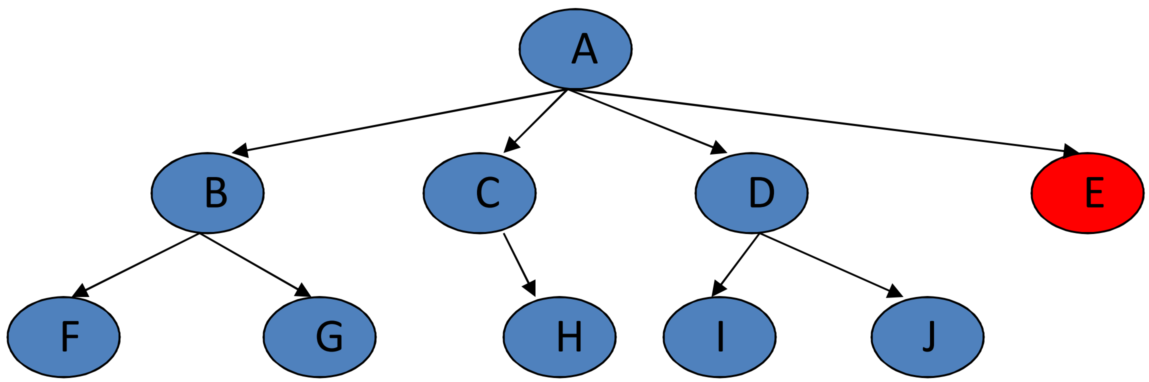
Breadth First Search

- A,
- B,C,D



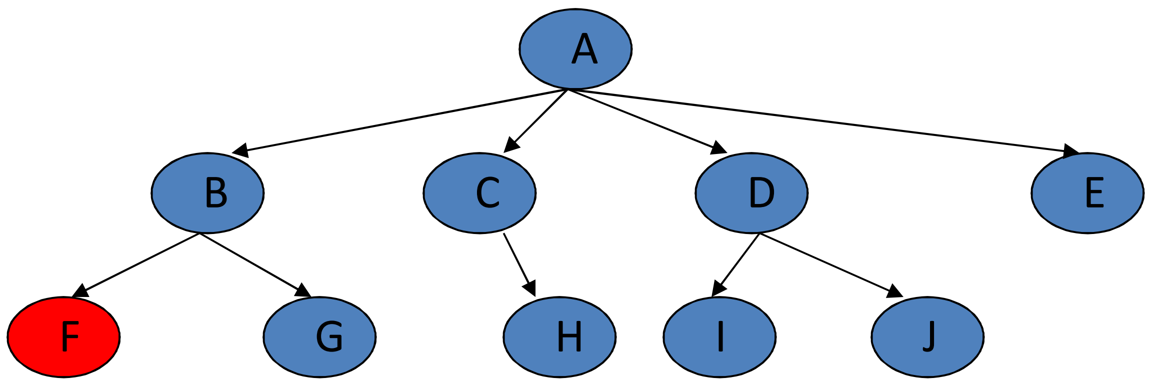
Breadth First Search

- A,
- B,C,D,E



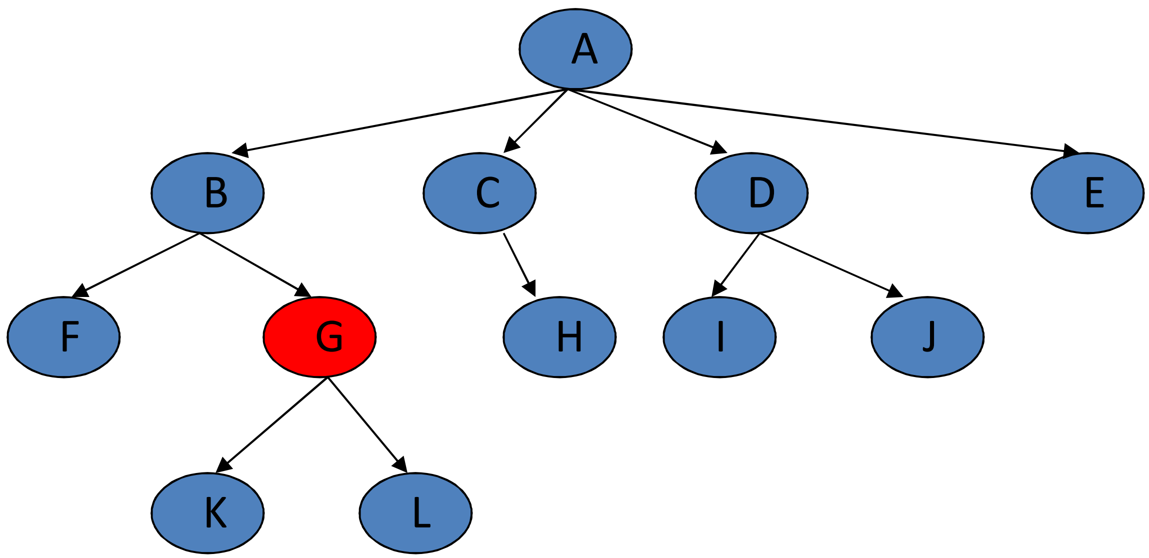
Breadth First Search

- A,
- B,C,D,E,
- F,



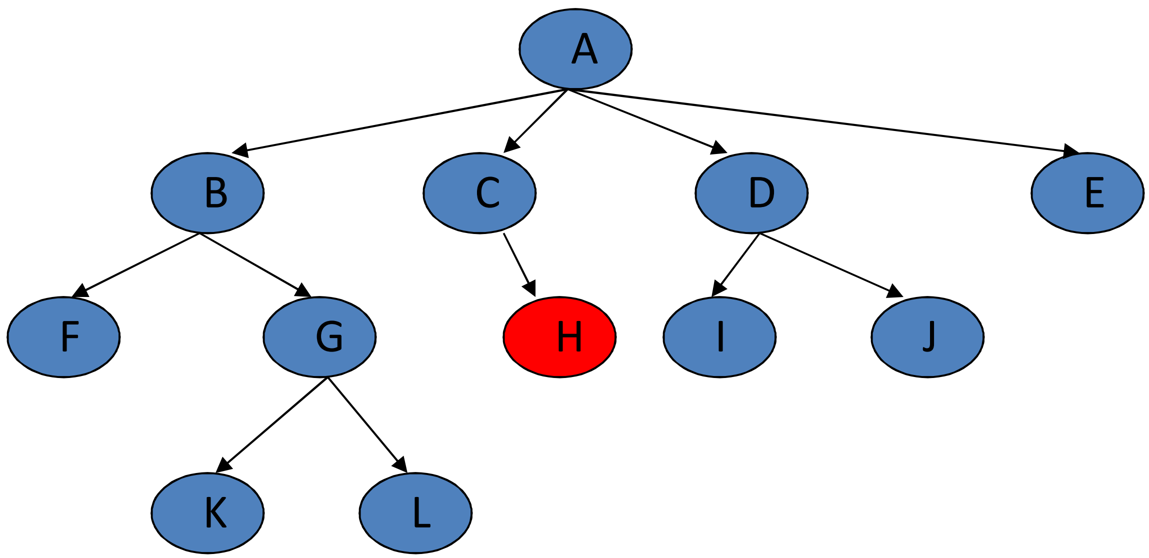
Breadth First Search

- A,
- B,C,D,E,
- F,G



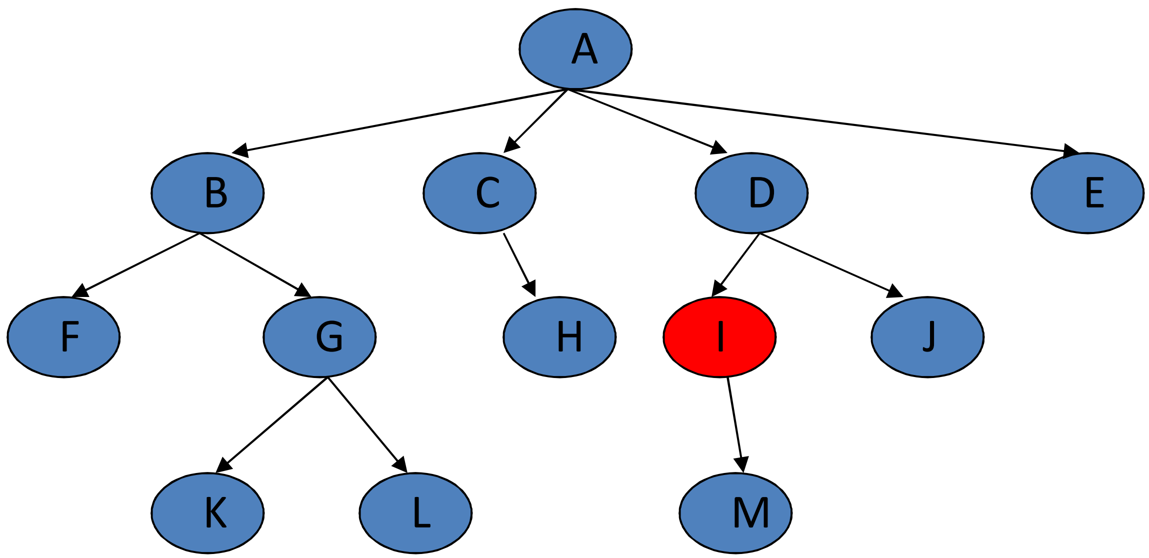
Breadth First Search

- A,
- B,C,D,E,
- F,G,H



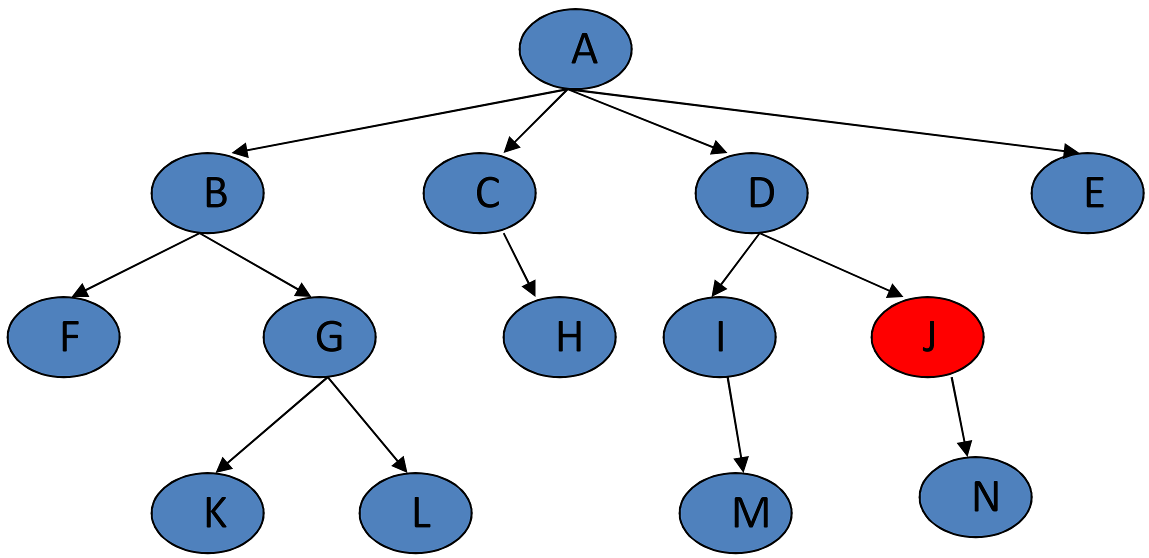
Breadth First Search

- A,
- B,C,D,E,
- F,G,H,I



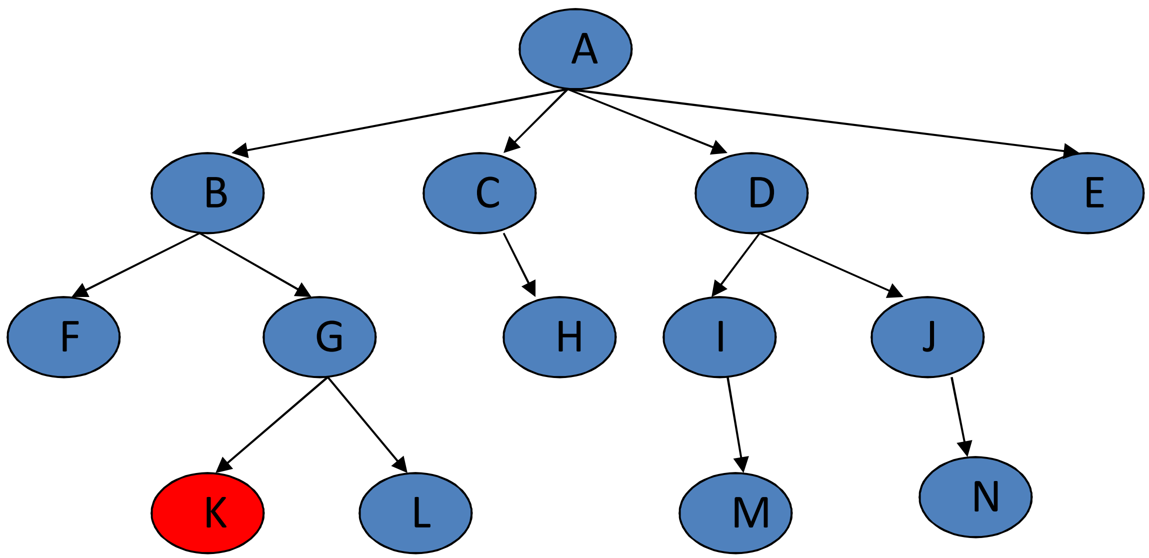
Breadth First Search

- A,
- B,C,D,E,
- F,G,H,I,J,



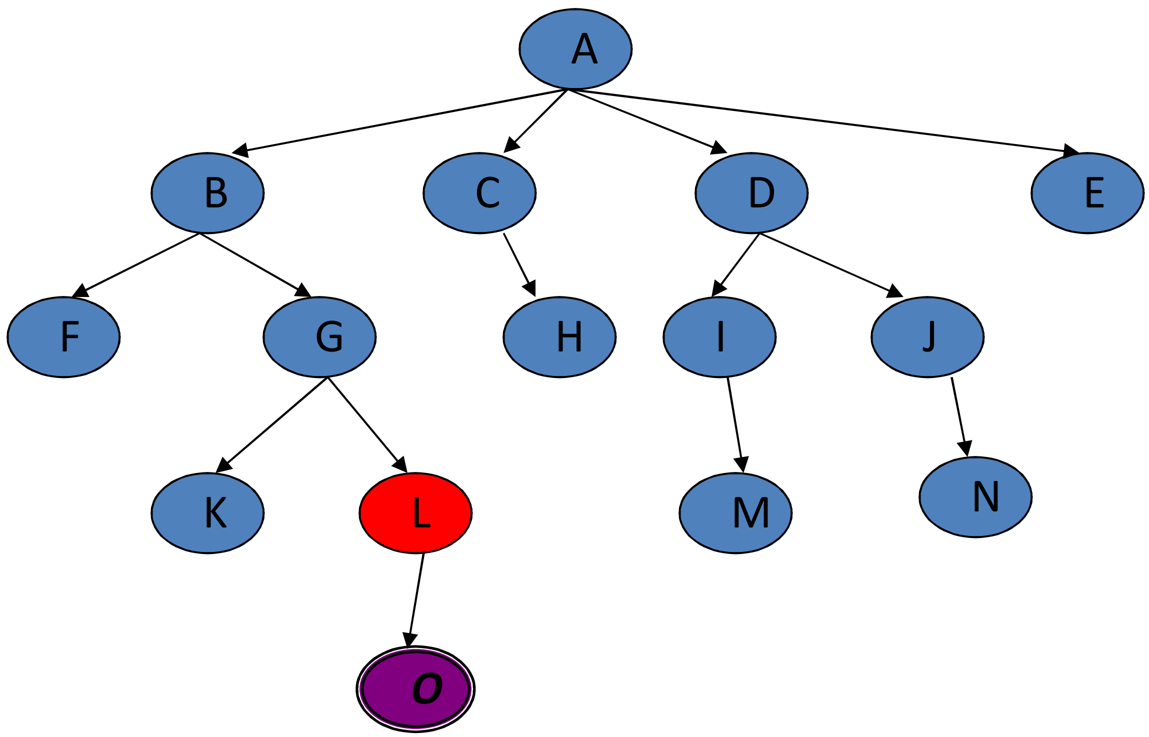
Breadth First Search

- A,
- B,C,D,E,
- F,G,H,I,J,
- K,



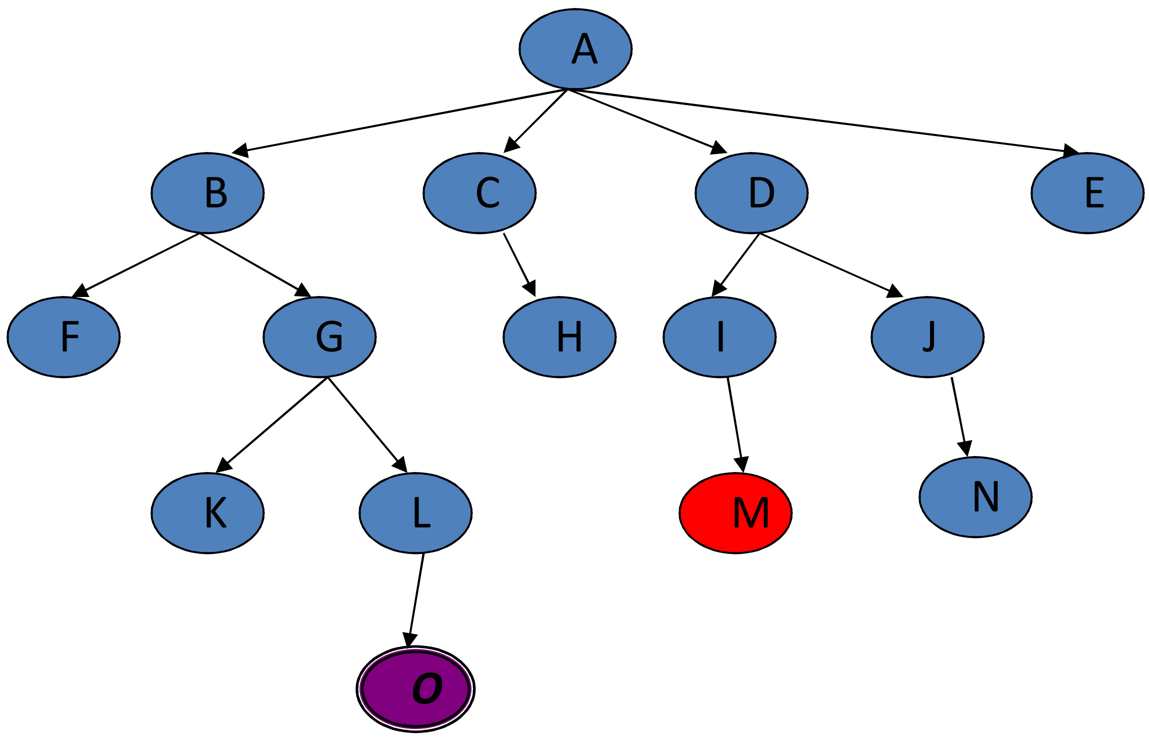
Breadth First Search

- A,
- B,C,D,E,
- F,G,H,I,J,
- K,L



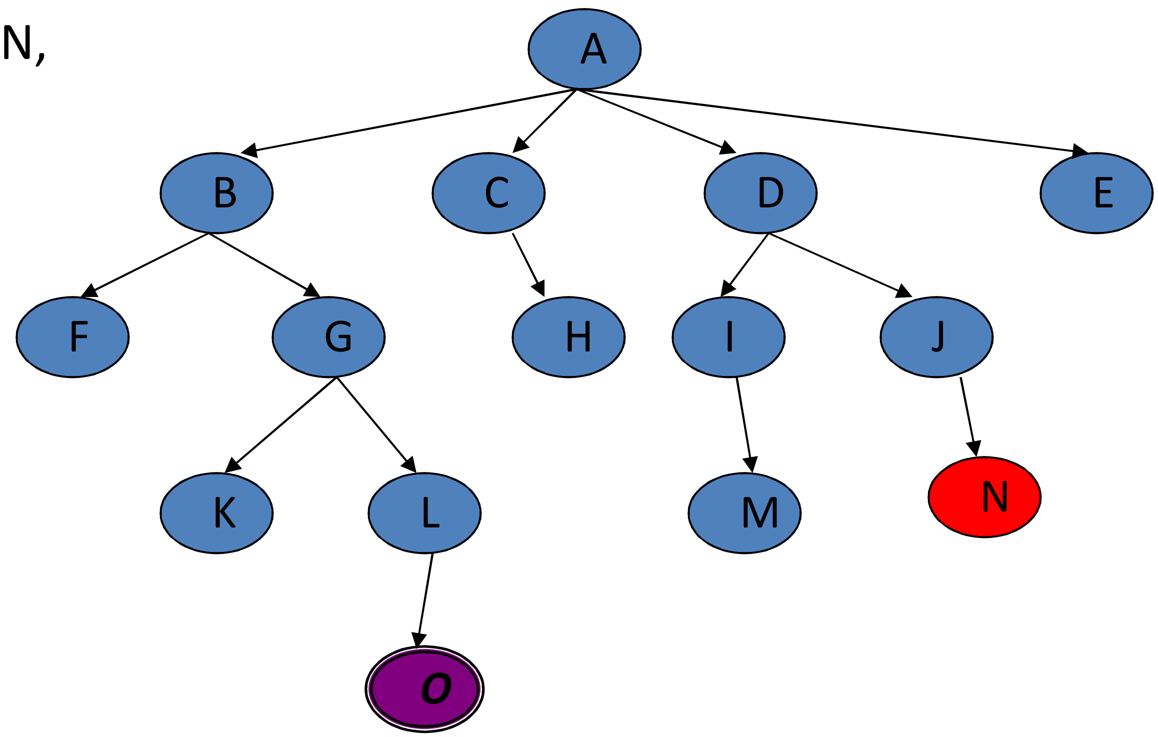
Breadth First Search

- A,
- B,C,D,E,
- F,G,H,I,J,
- K,L, M,



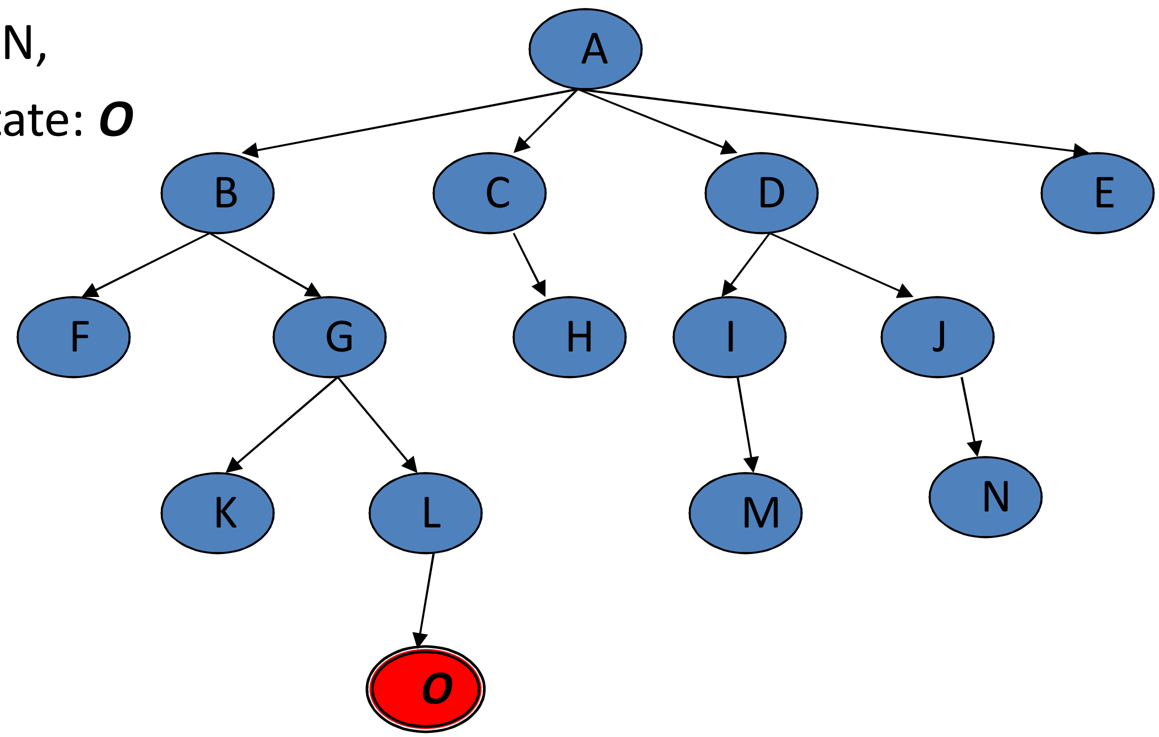
Breadth First Search

- A,
- B,C,D,E,
- F,G,H,I,J,
- K,L, M,N,



Breadth First Search

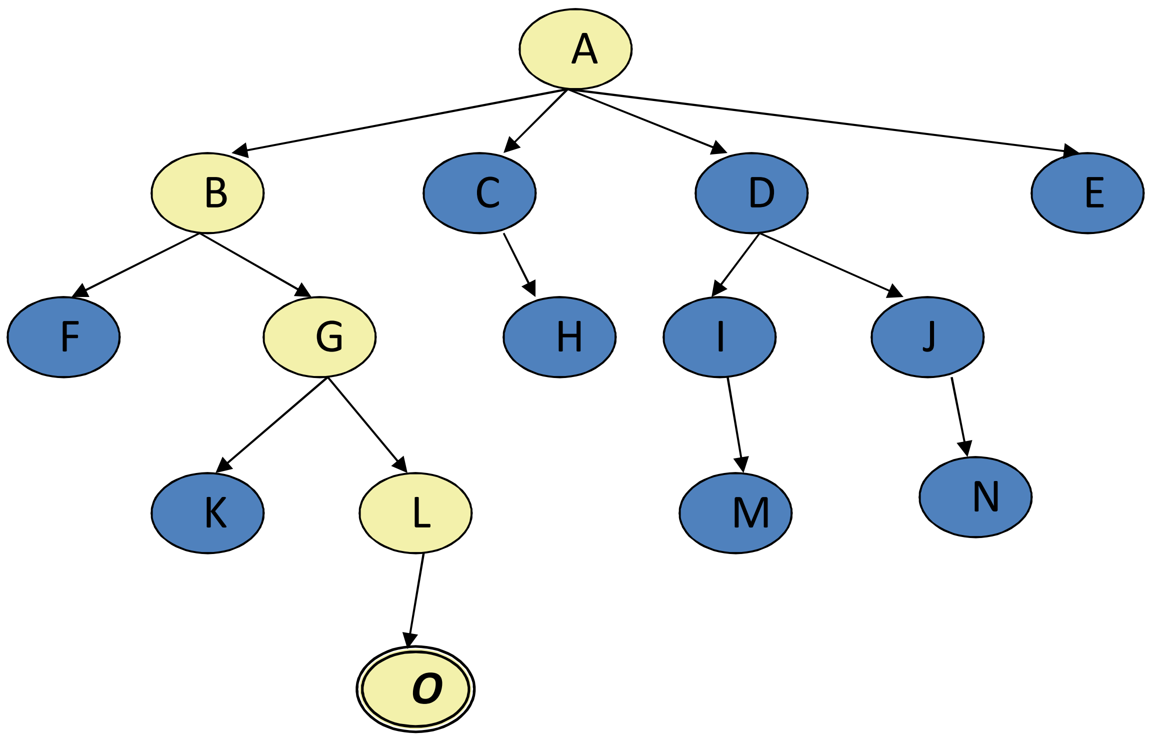
- A,
- B,C,D,E,
- F,G,H,I,J,
- K,L, M,N,
- Goal state: ***O***



Breadth First Search

- The returned solution is the sequence of operators in the path:

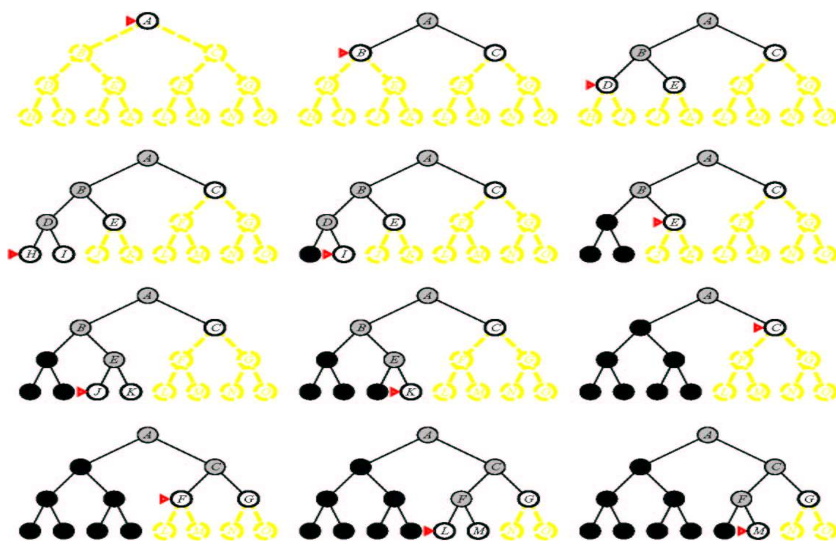
  A, B, G, L, O



Depth First Search

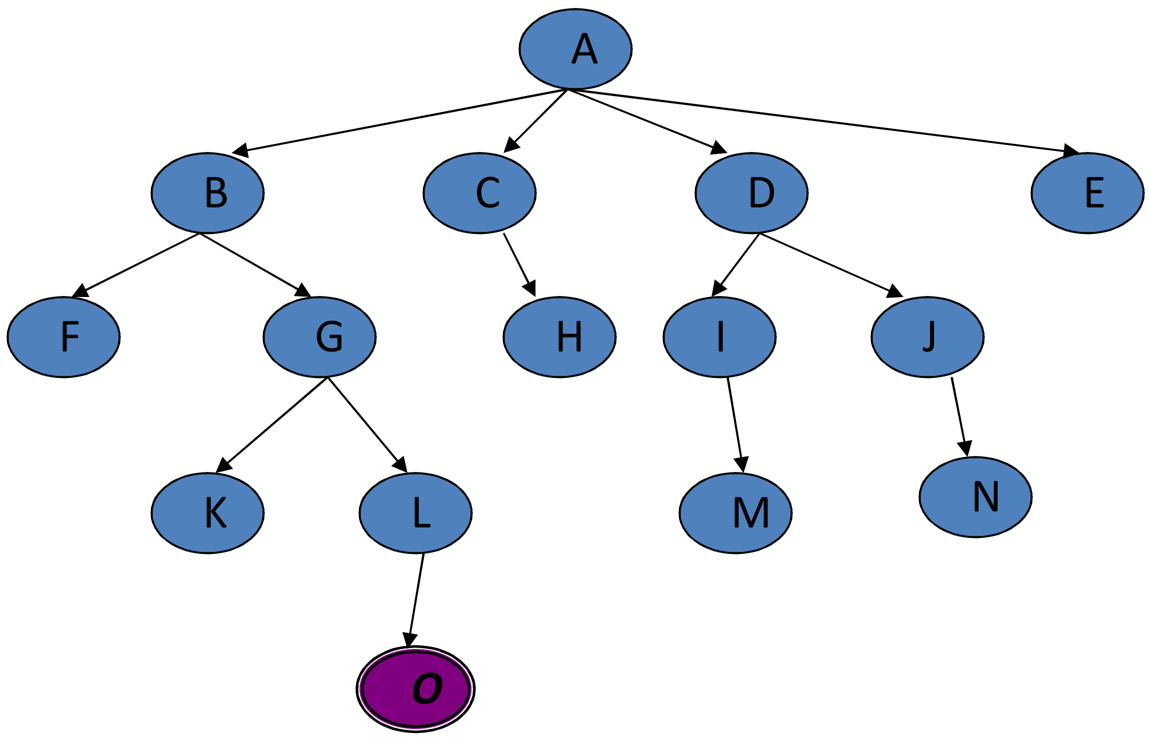
Main idea: Expand node at the deepest level (breaking ties left to right).

Implementation: use of a Last-In-First-Out(LIFO) or stack operation. Enqueue nodes in LIFO (last-in, first-out) order.



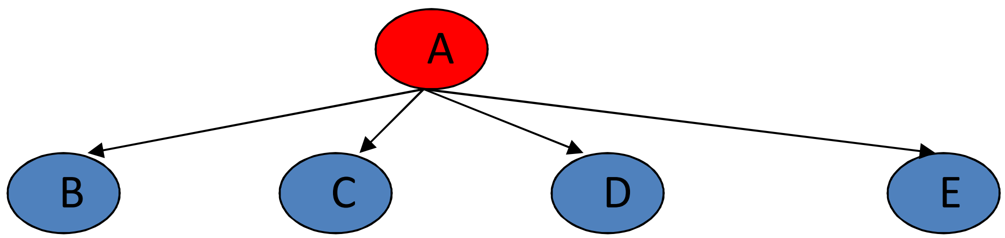
Depth First Search (DFS)

Given the following state space (tree search), give the sequence of visited nodes when using DFS (assume that the node***O*** is the goal state):



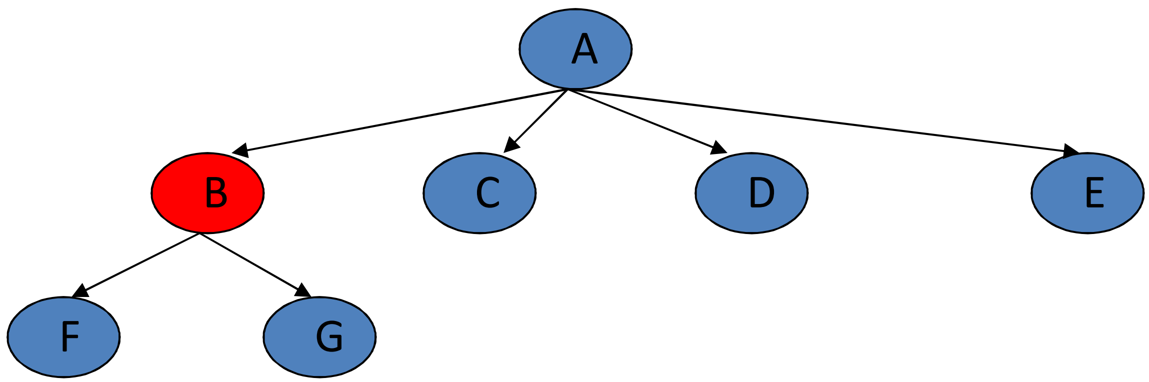
Depth First Search

- A,



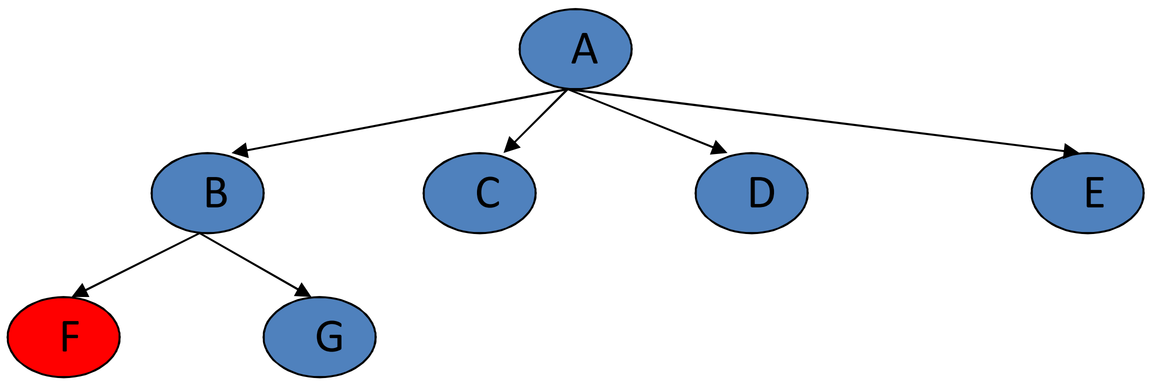
Depth First Search

- A,B,



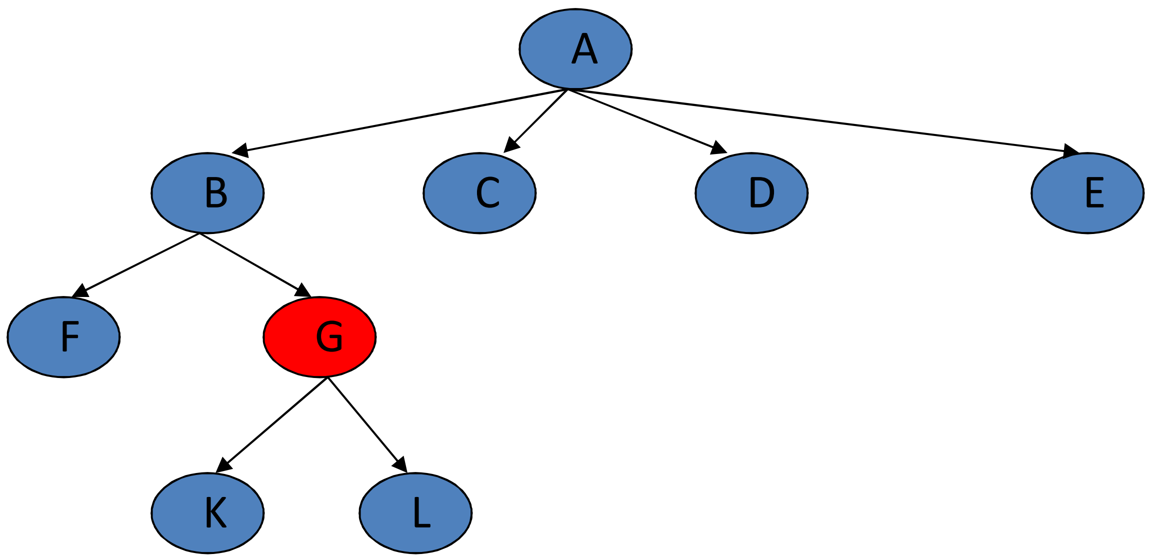
Depth First Search

- A,B,F,



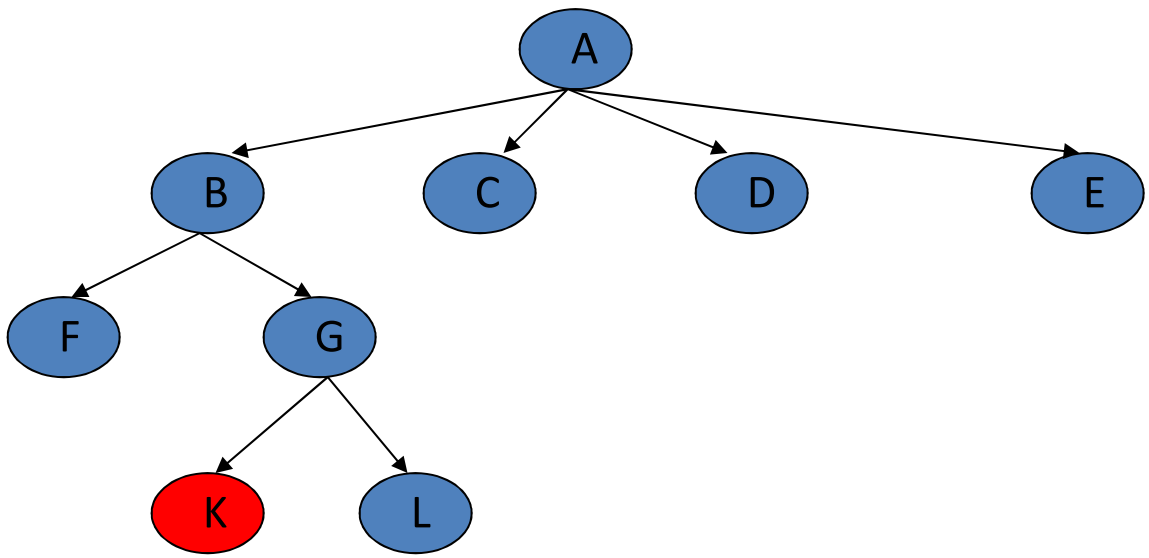
Depth First Search

- A,B,F,
- G,



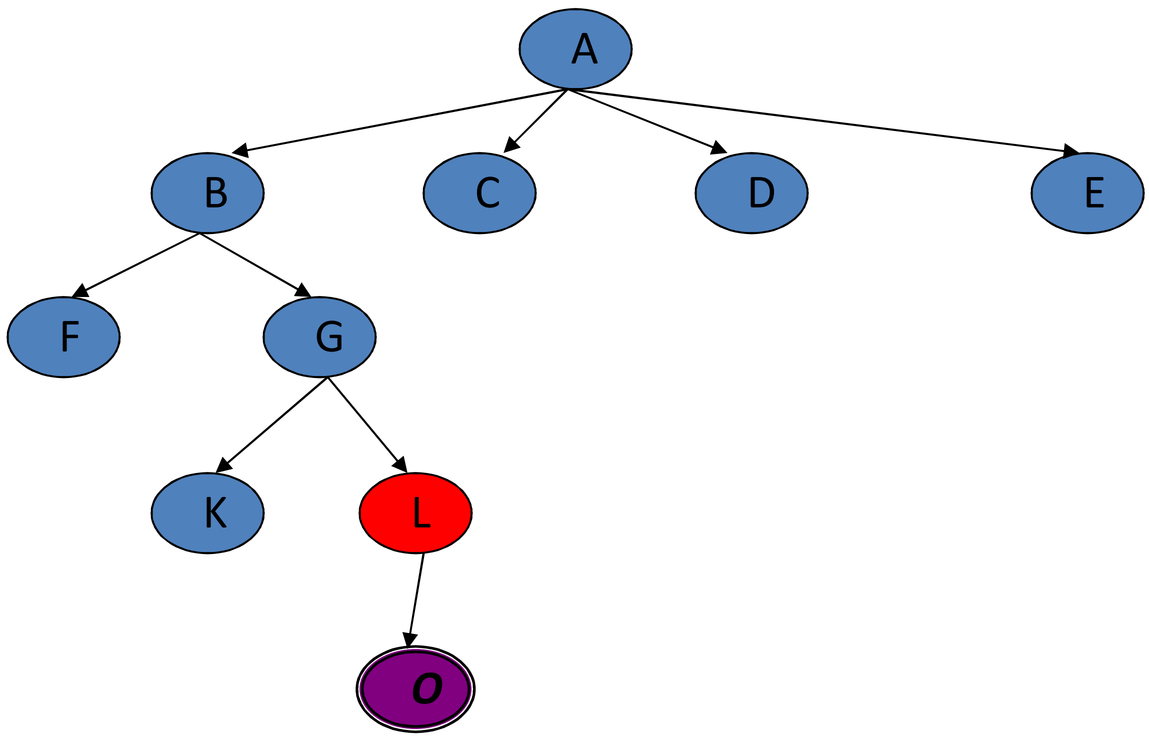
Depth First Search

- A,B,F,
- G,K,



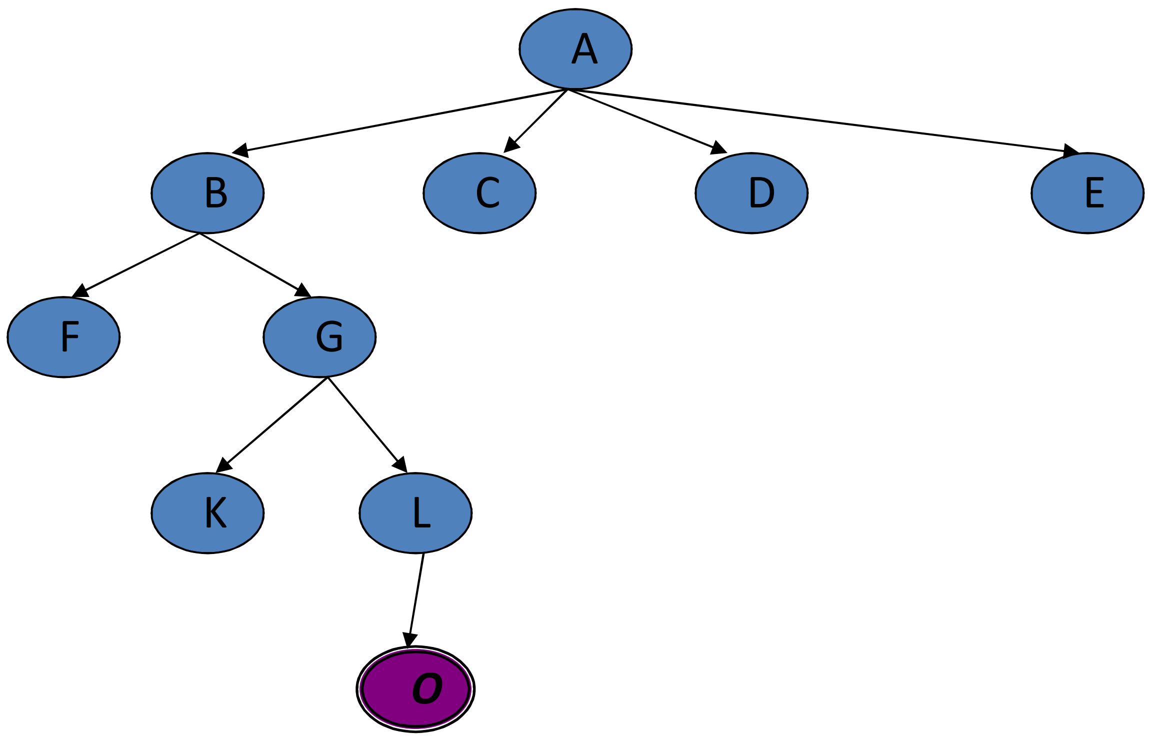
Depth First Search

- A,B,F,
- G,K,
- L,



Depth First Search

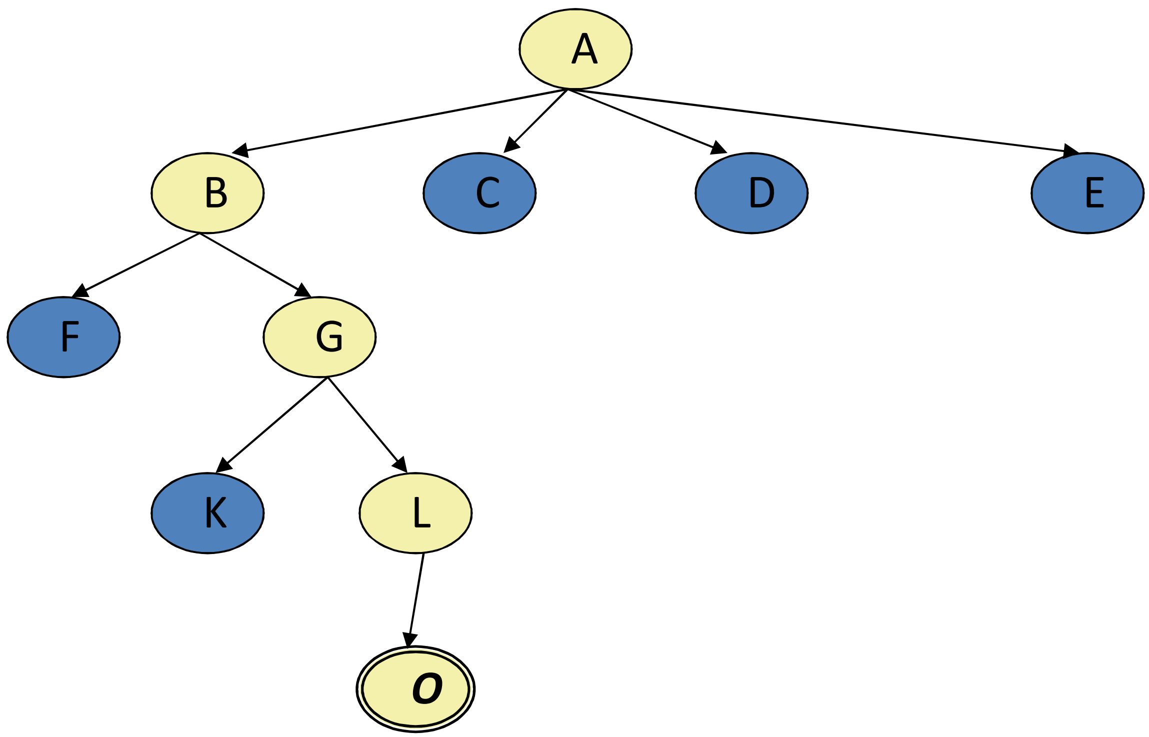
- A,B,F,
- G,K,
- L, *O: Goal State*



Depth First Search

The returned solution is the sequence of operators in the path:

A, B, G, L, O



Depth-Limited Search (DLS)

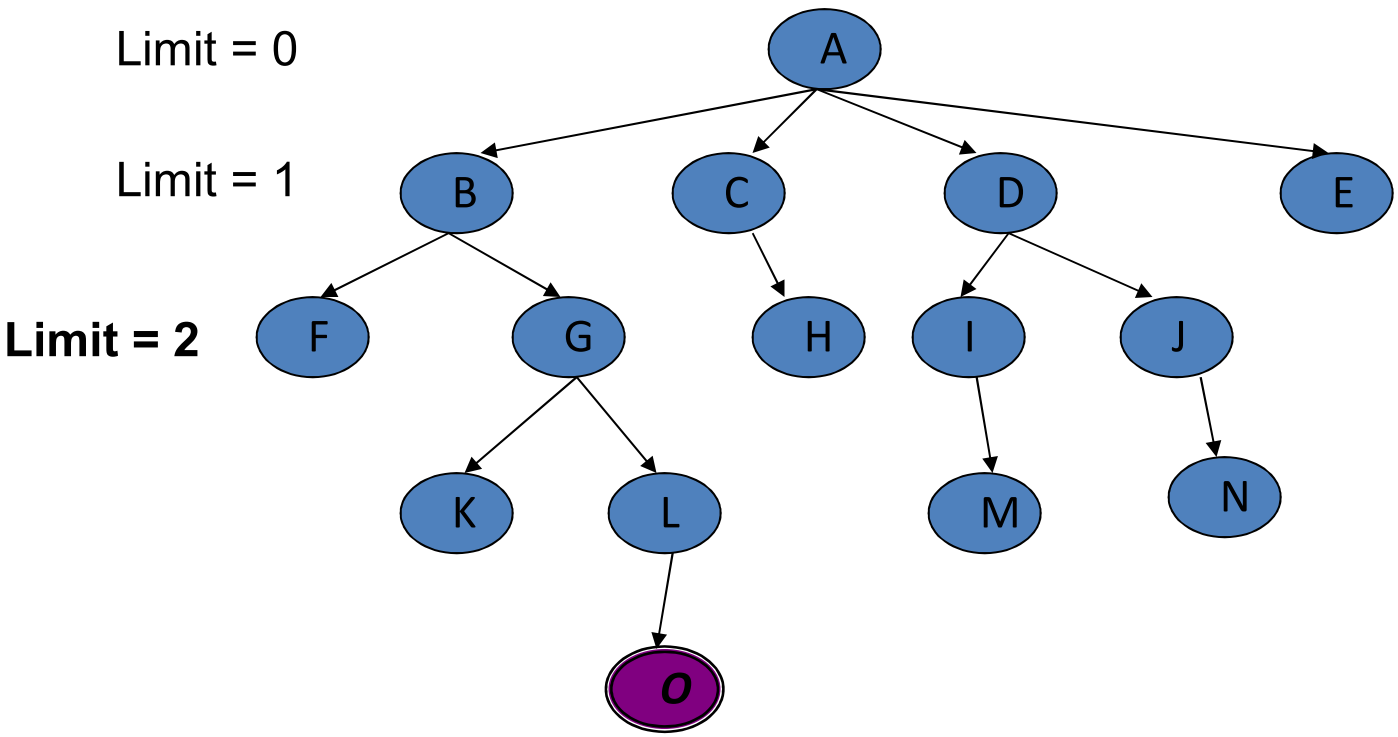
Main idea: *Expand node at the deepest level, but limit depth to L.*

Implementation:

Enqueue nodes in LIFO (last-in, first-out) order. But limit depth to L

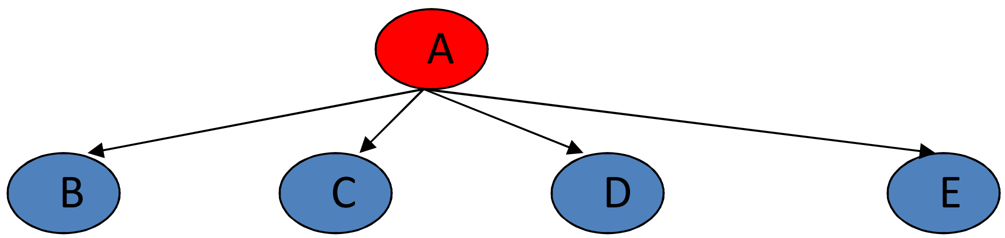
Depth-Limited Search (DLS)

Given the following state space (tree search), give the sequence of visited nodes when using DLS (Limit = 2):



Depth-Limited Search (DLS)

- A,

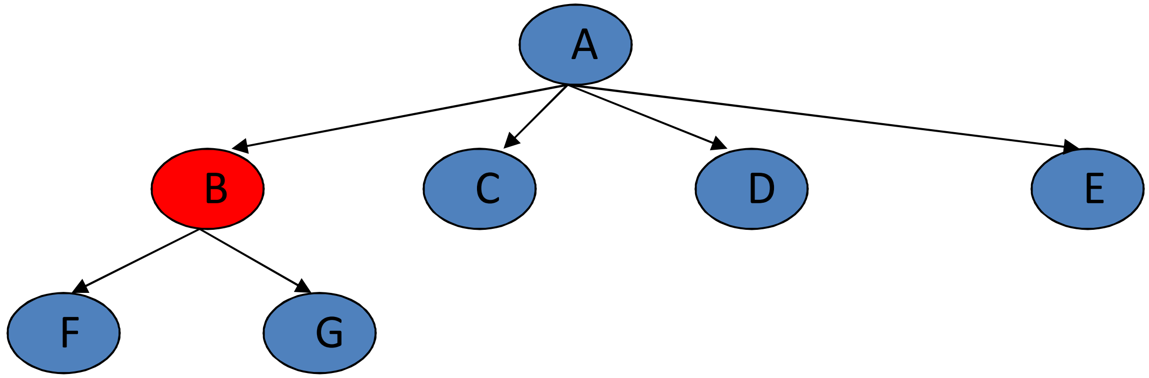


Limit = 2

Depth-Limited Search (DLS)

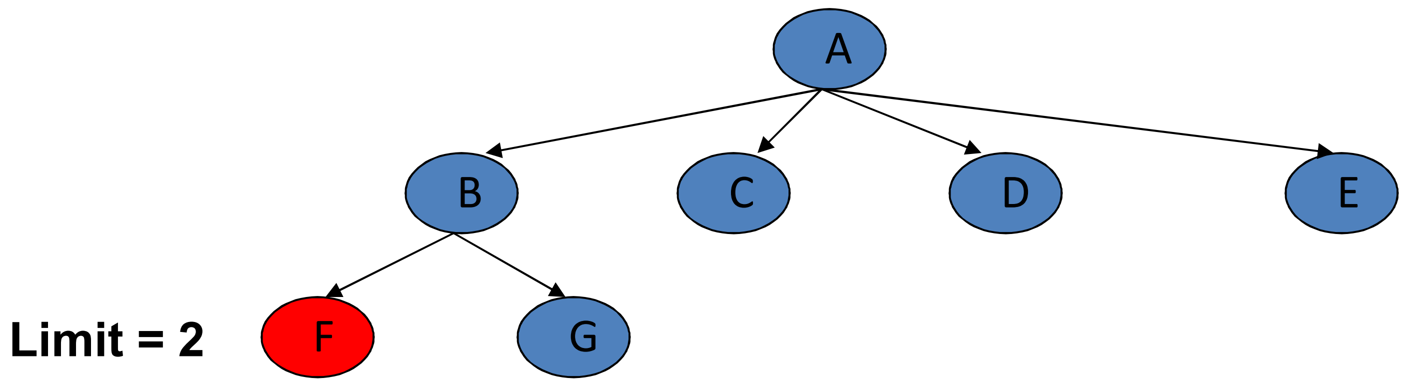
- A,B,

Limit = 2



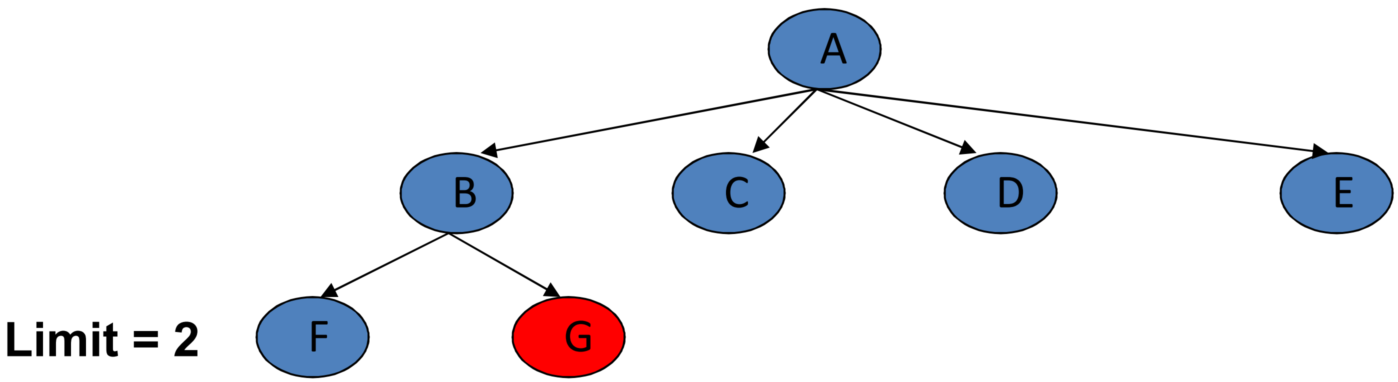
Depth-Limited Search (DLS)

- A,B,F,



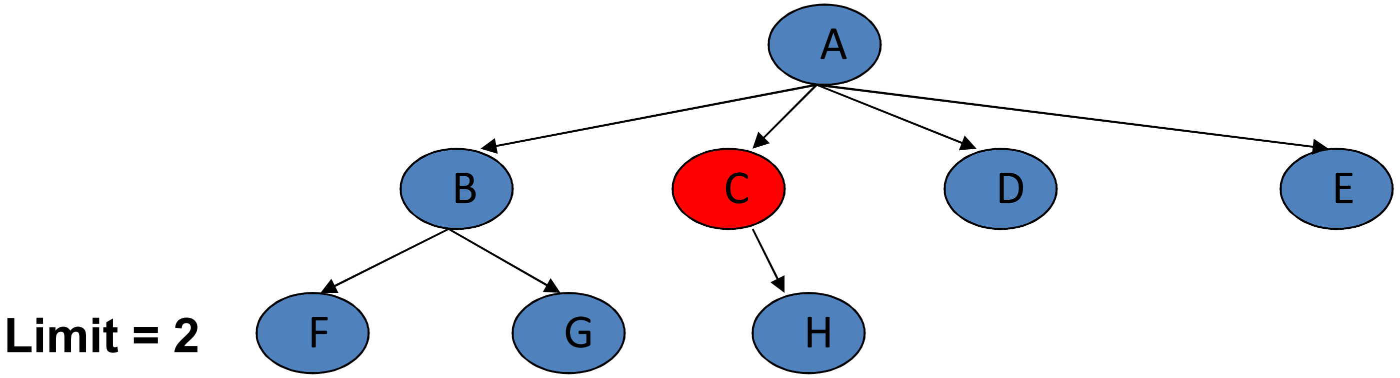
Depth-Limited Search (DLS)

- A,B,F,
- G,



Depth-Limited Search (DLS)

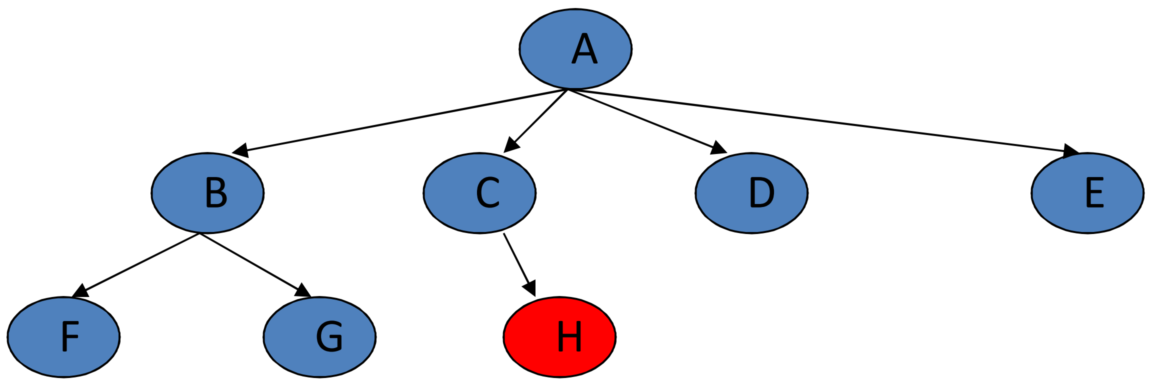
- A,B,F,
- G,
- C,



Depth-Limited Search (DLS)

- A,B,F,
- G,
- C,H,

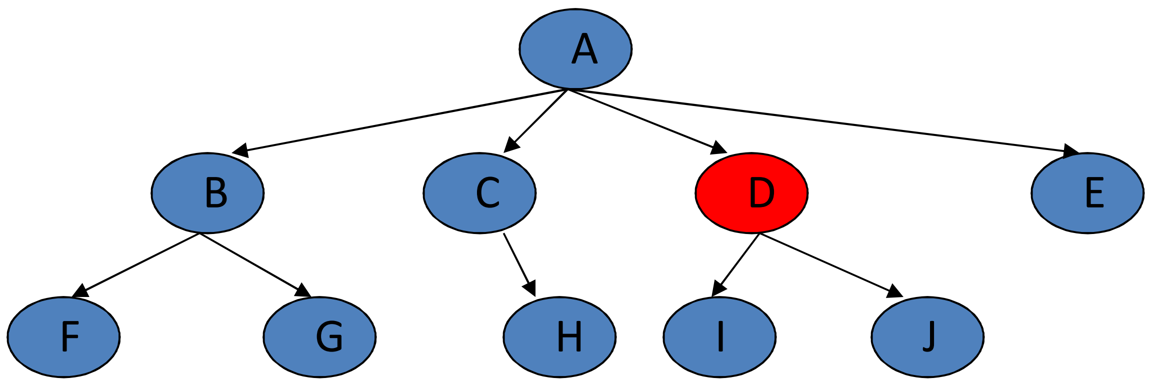
Limit = 2



Depth-Limited Search (DLS)

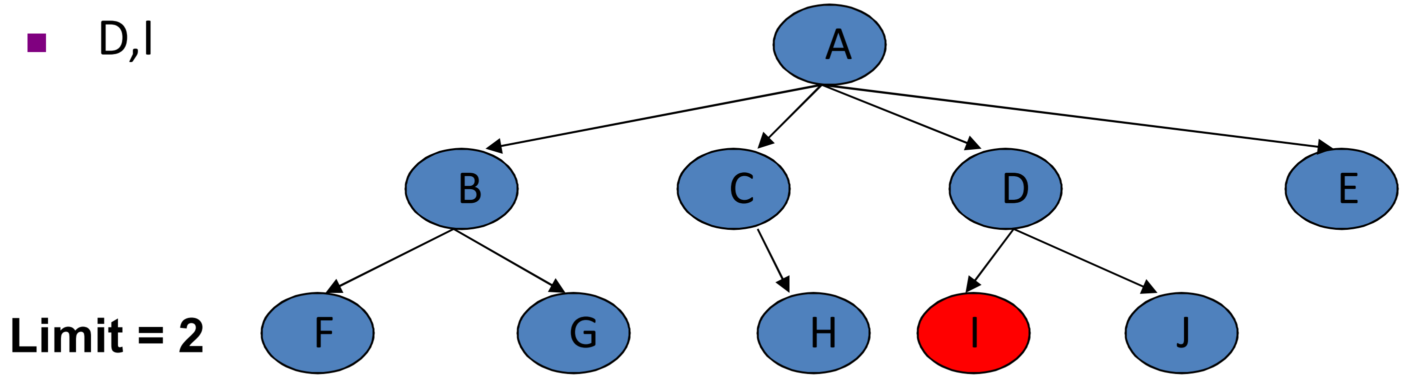
- A,B,F,
- G,
- C,H,
- D,

Limit = 2



Depth-Limited Search (DLS)

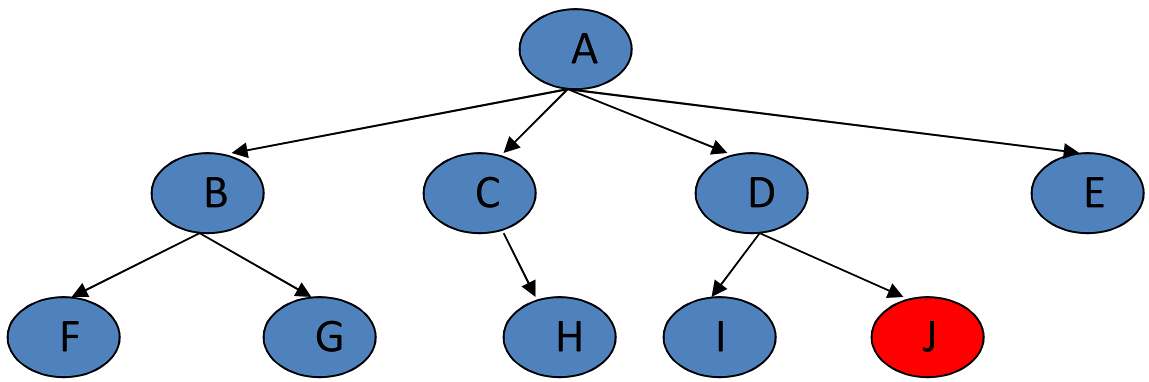
- A,B,F,
- G,
- C,H,
- D,I



Depth-Limited Search (DLS)

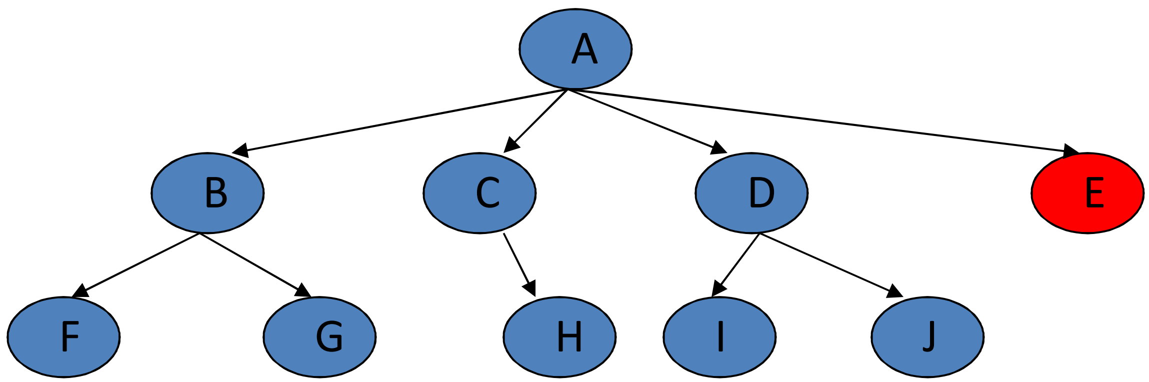
- A,B,F,
- G,
- C,H,
- D,I
- J,

Limit = 2



Depth-Limited Search (DLS)

- A,B,F,
- G,
- C,H,
- D,I
- J,
- E

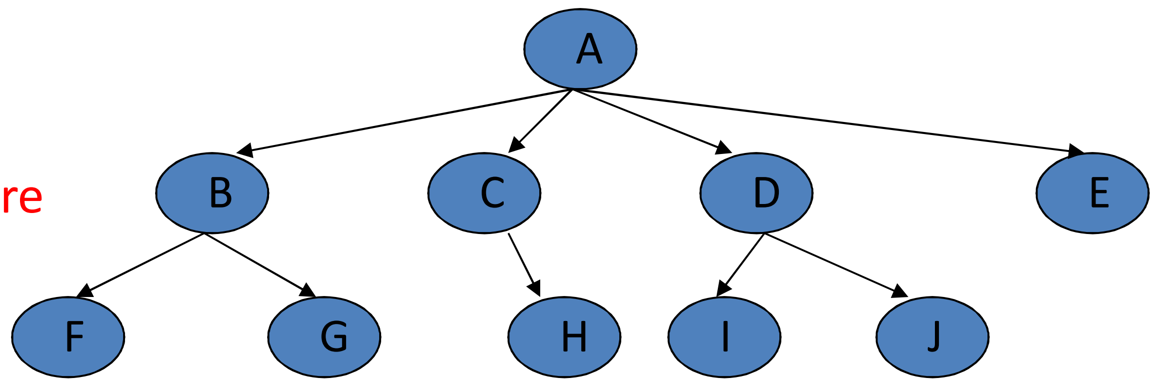


Limit = 2

Depth-Limited Search (DLS)

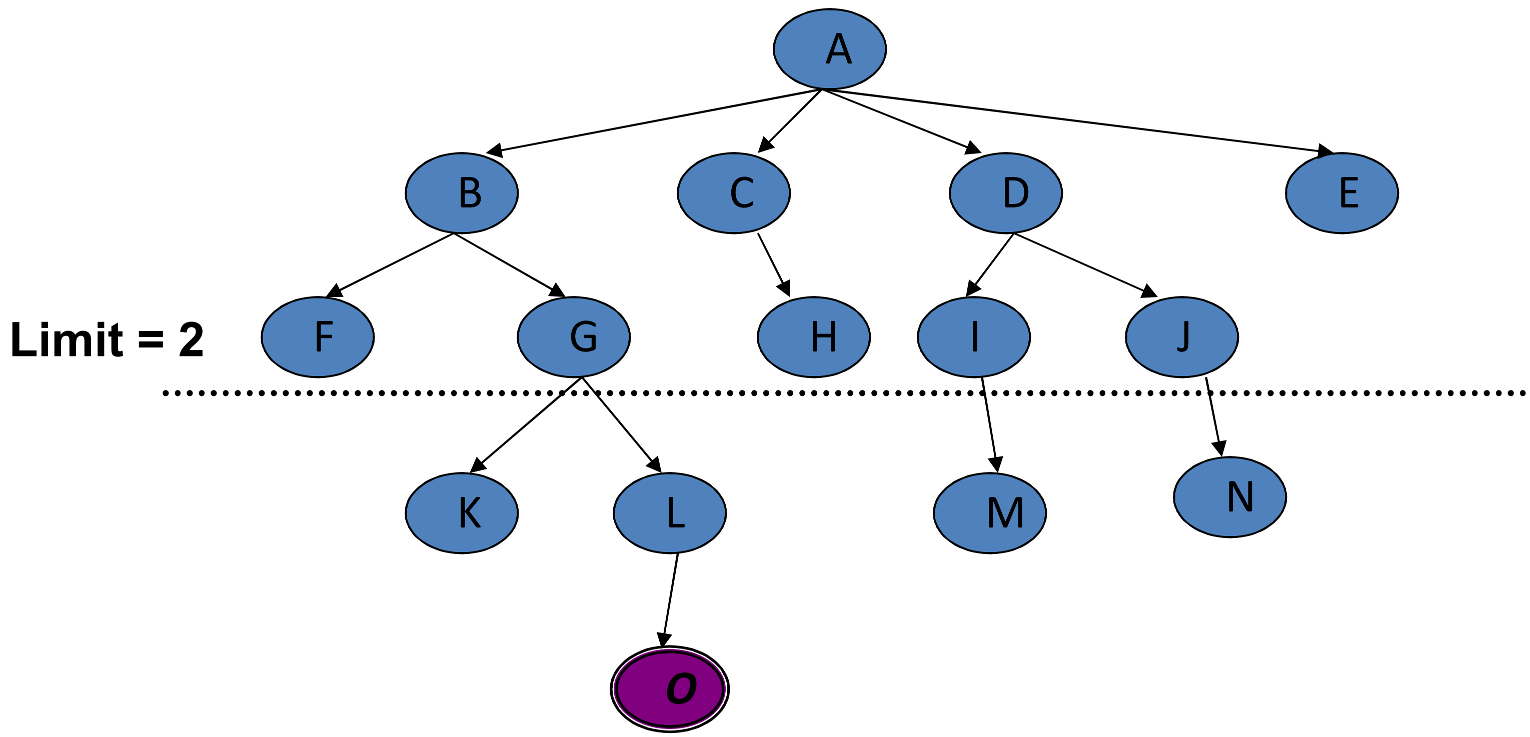
- A,B,F,
- G,
- C,H,
- D,I
- J,
- E, Failure

Limit = 2



Depth-Limited Search (DLS)

- DLS algorithm returns Failure (no solution)
- The reason is that the goal is beyond the limit (Limit =2): the goal depth is (d=4)



Basic Search Algorithms Uninformed Search

Iterative Deepening Search (IDS)

Iterative Deepening Search (IDS)

```
function ITERATIVE-DEEPENING-SEARCH():

   for depth = 0 to infinity do
      if DEPTH-LIMITED-SEARCH(depth) succeeds
     then return its result
   end
   return failure
```

Iterative Deepening Search (IDS)

- **Key idea**: Iterative deepening search (IDS) applies DLS repeatedly with increasing depth. It terminates when a solution is found or no solutions exists.

- IDS combines the benefits of BFS and DFS: Like DFS the memory requirements are very modest (O(*bd*)). Like BFS, it is complete when the branching factor is finite.

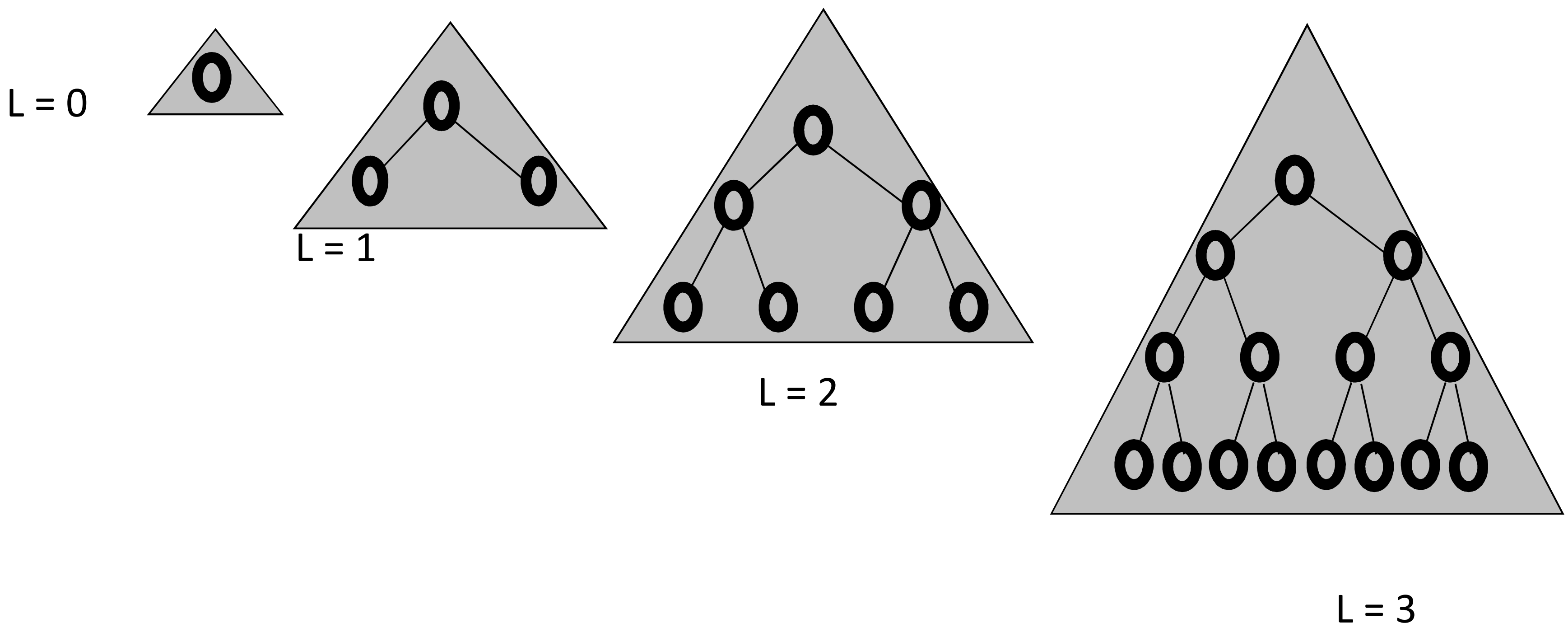
- The total number of generated nodes is :

$$N(\text{IDS})=(d)b + (d-1)\, b^2 + \ldots+(1)b^d$$

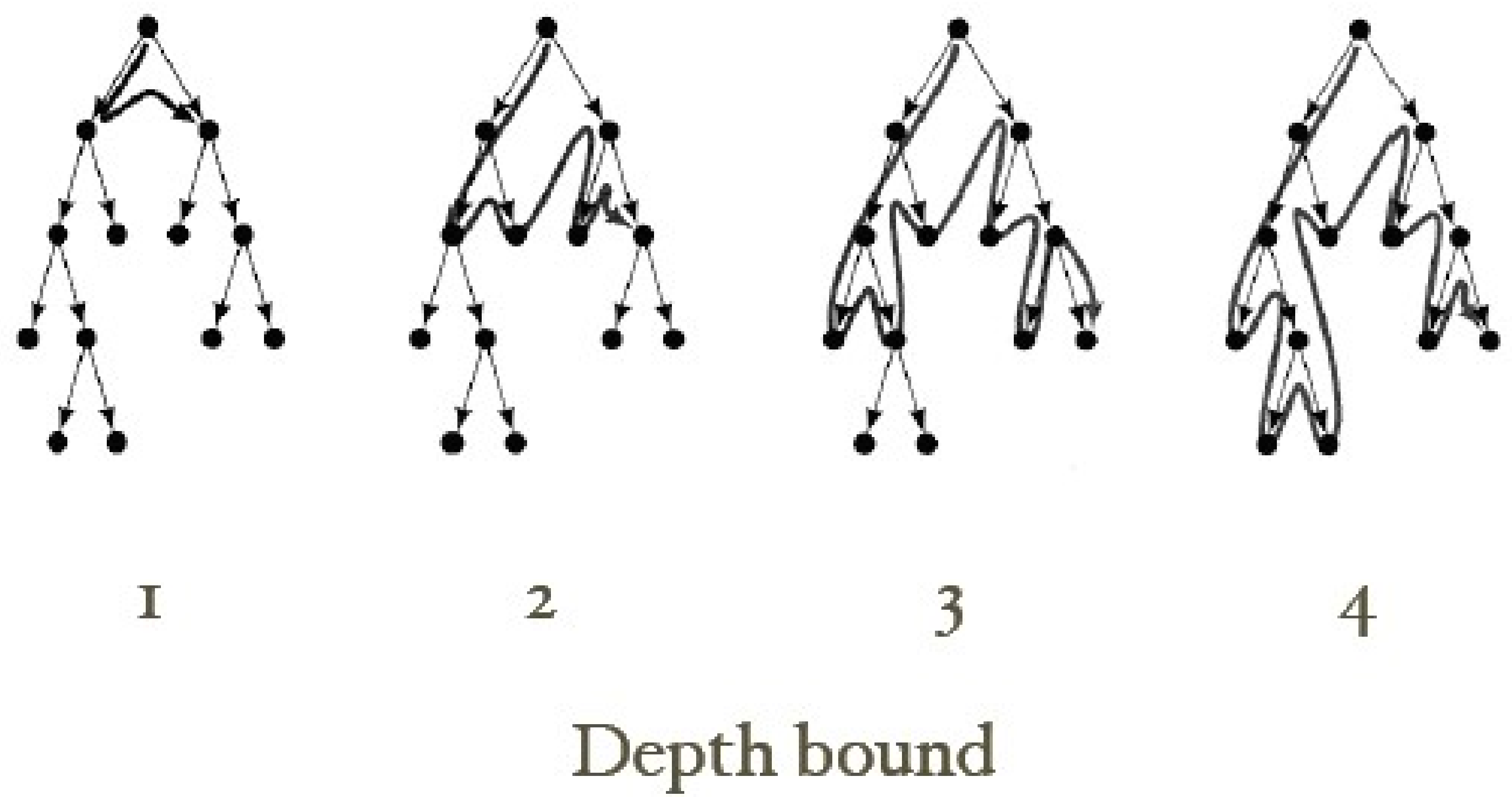
- In general, iterative deepening is the preferred uninformed search method when there is a large search space and the depth of the solution is not known.

Iterative Deepening Search (IDS)

- **Key idea**: Iterative deepening search (IDS) applies DLS repeatedly with increasing depth. It terminates when a solution is found or no solutions exists.

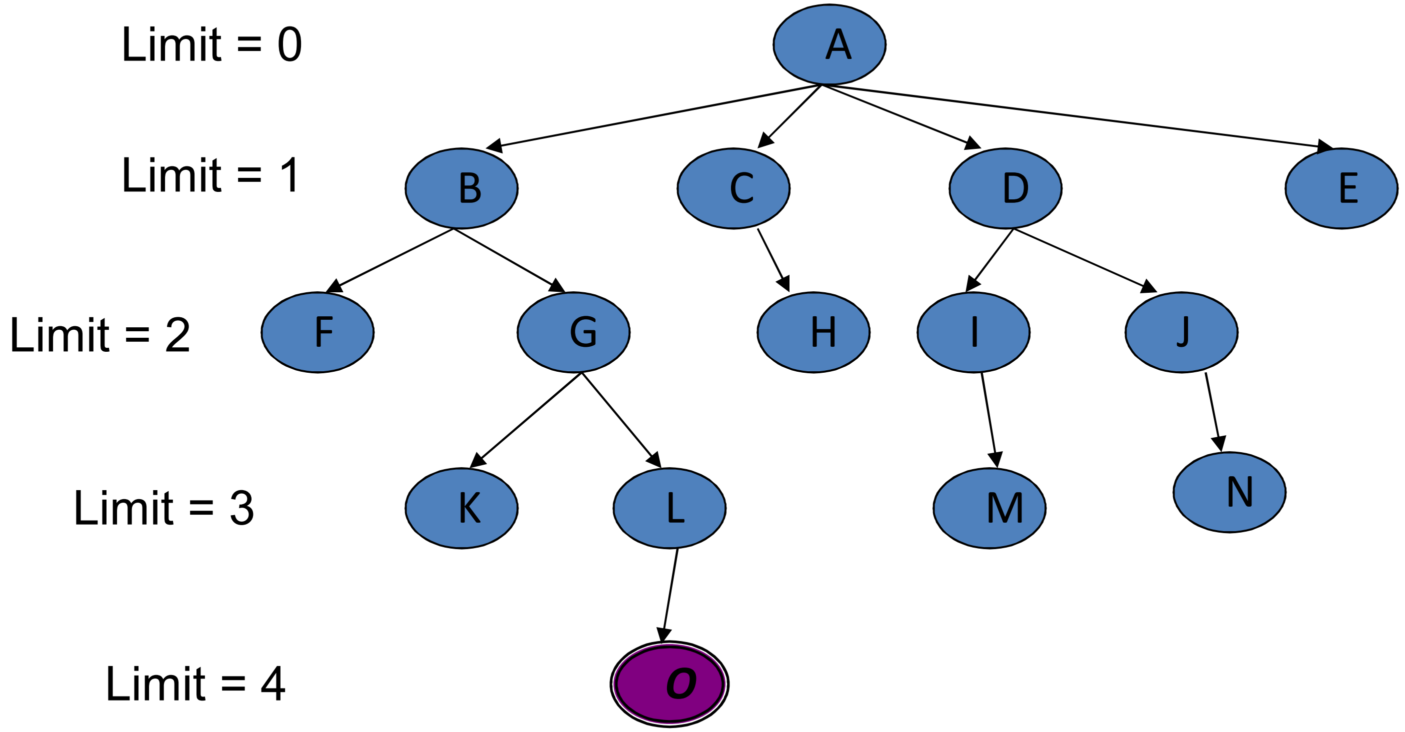


Iterative Deepening Search (IDS)



Iterative Deepening Search (IDS)

Given the following state space (tree search), give the sequence of visited nodes when using IDS:



Uniform Cost Search (UCS)

- **Main idea**: *Expand the cheapest node. Where the cost is the path cost g(n)*.

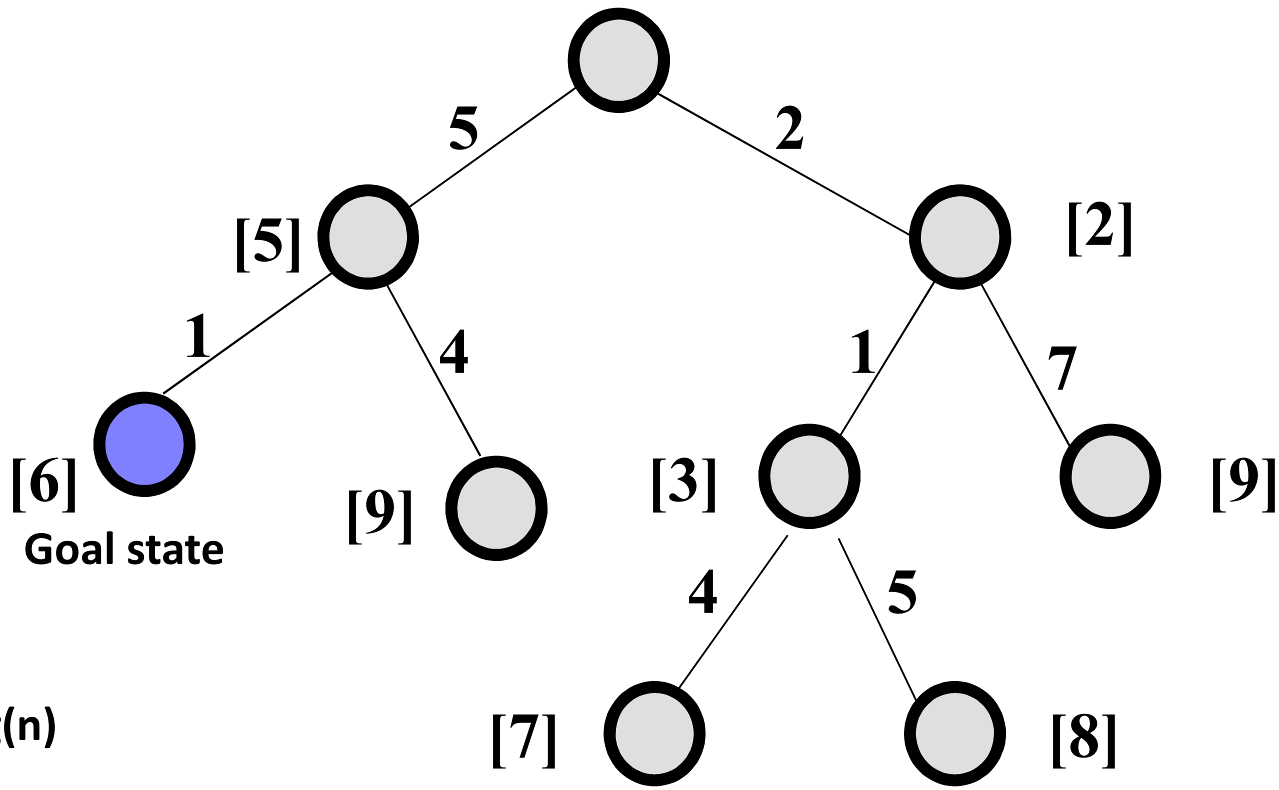
- **Implementation**:

  Enqueue nodes in order of cost g(n).

  QUEUING-FN:- insert in order of increasing path cost.

  Enqueue new node at the appropriate position in the queue so that we dequeue the cheapest node.

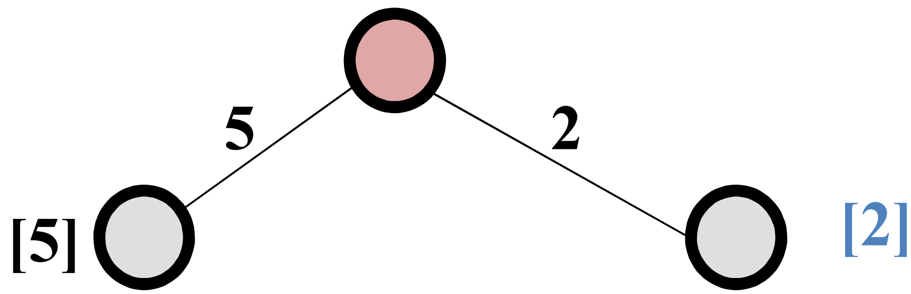
Uniform Cost Search (UCS)



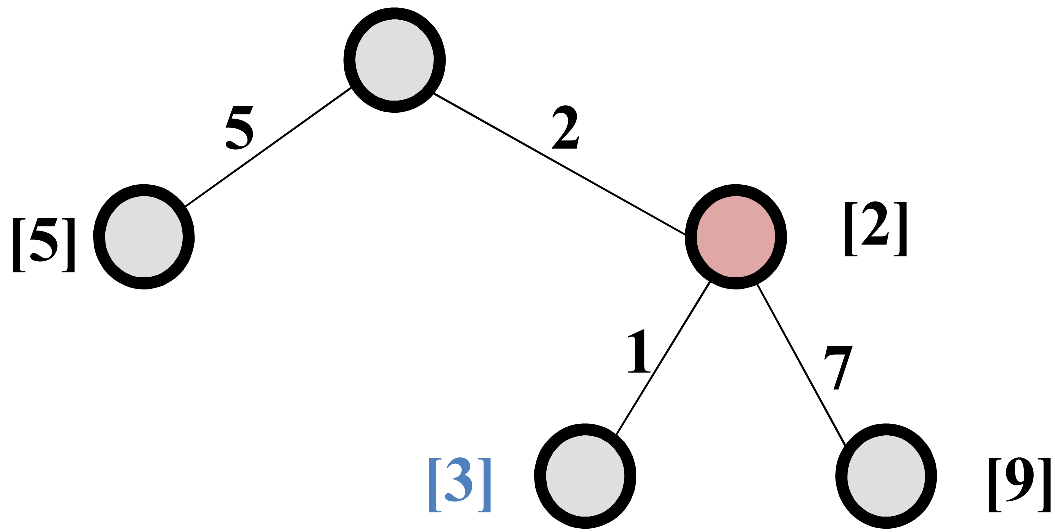
[x] = g(n)

path cost of node n

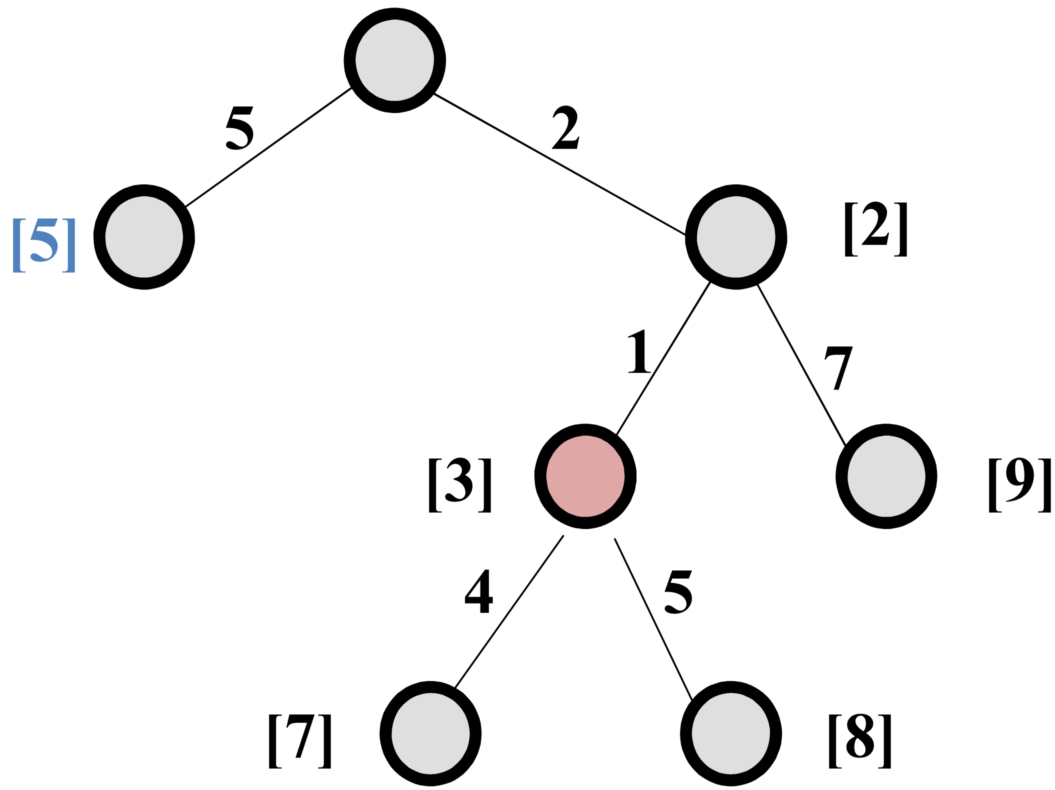
Uniform Cost Search (UCS)



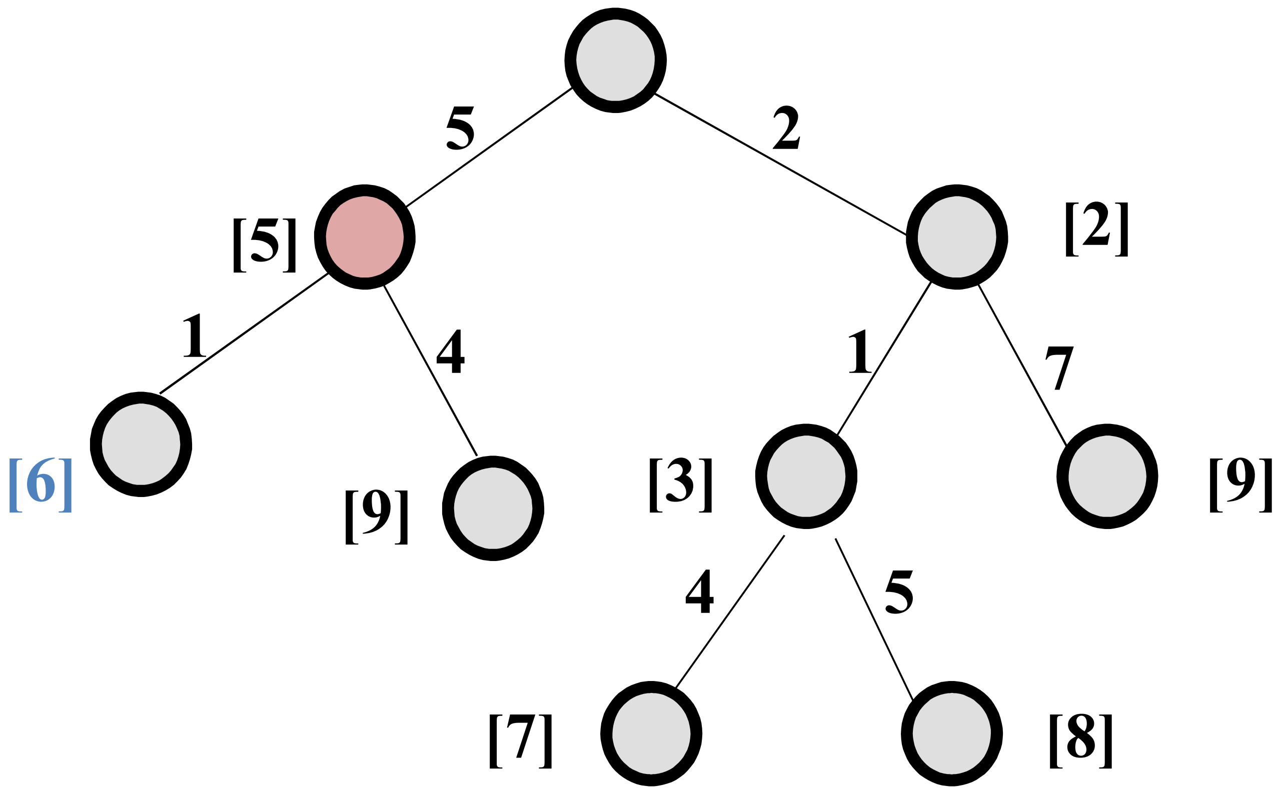
Uniform Cost Search (UCS)



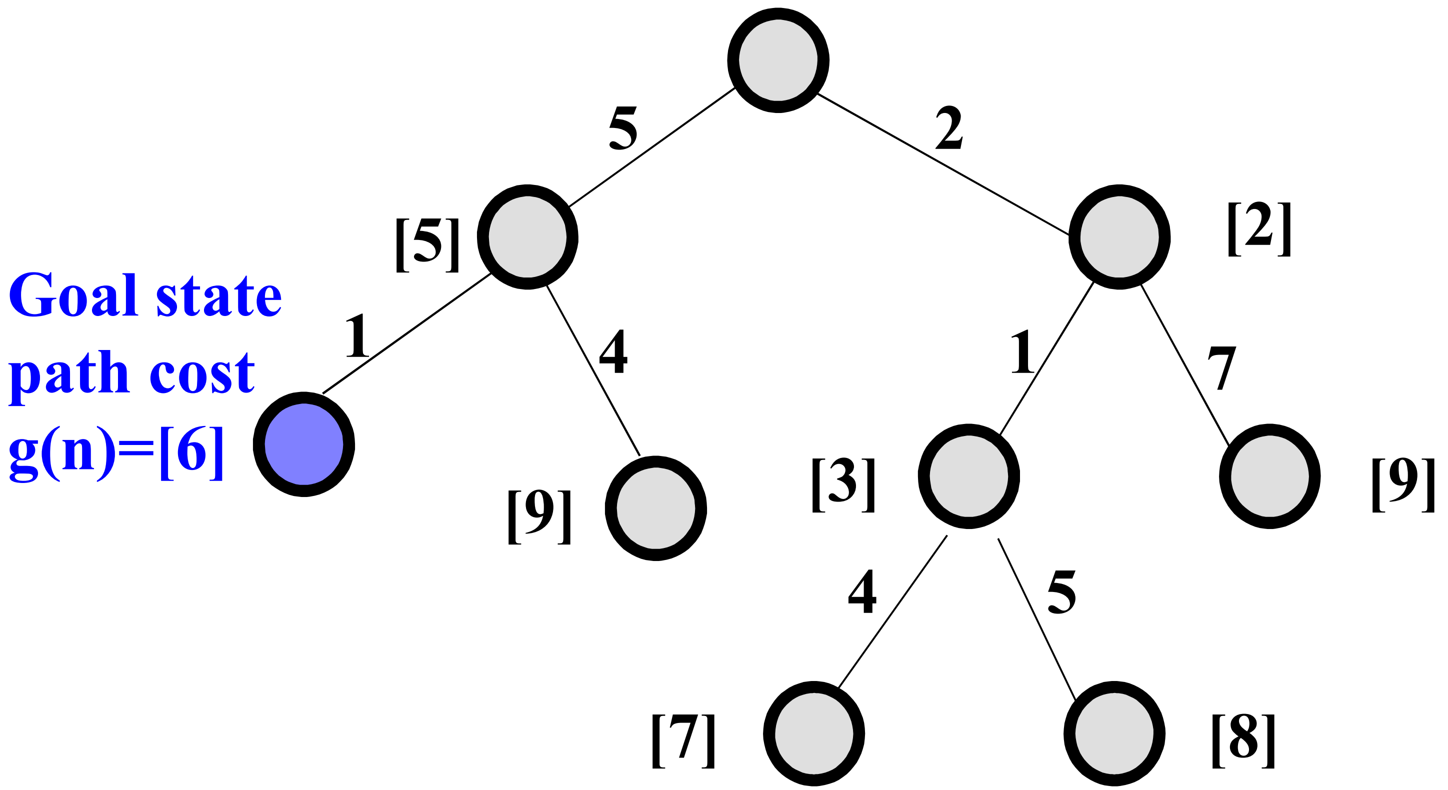
Uniform Cost Search (UCS)



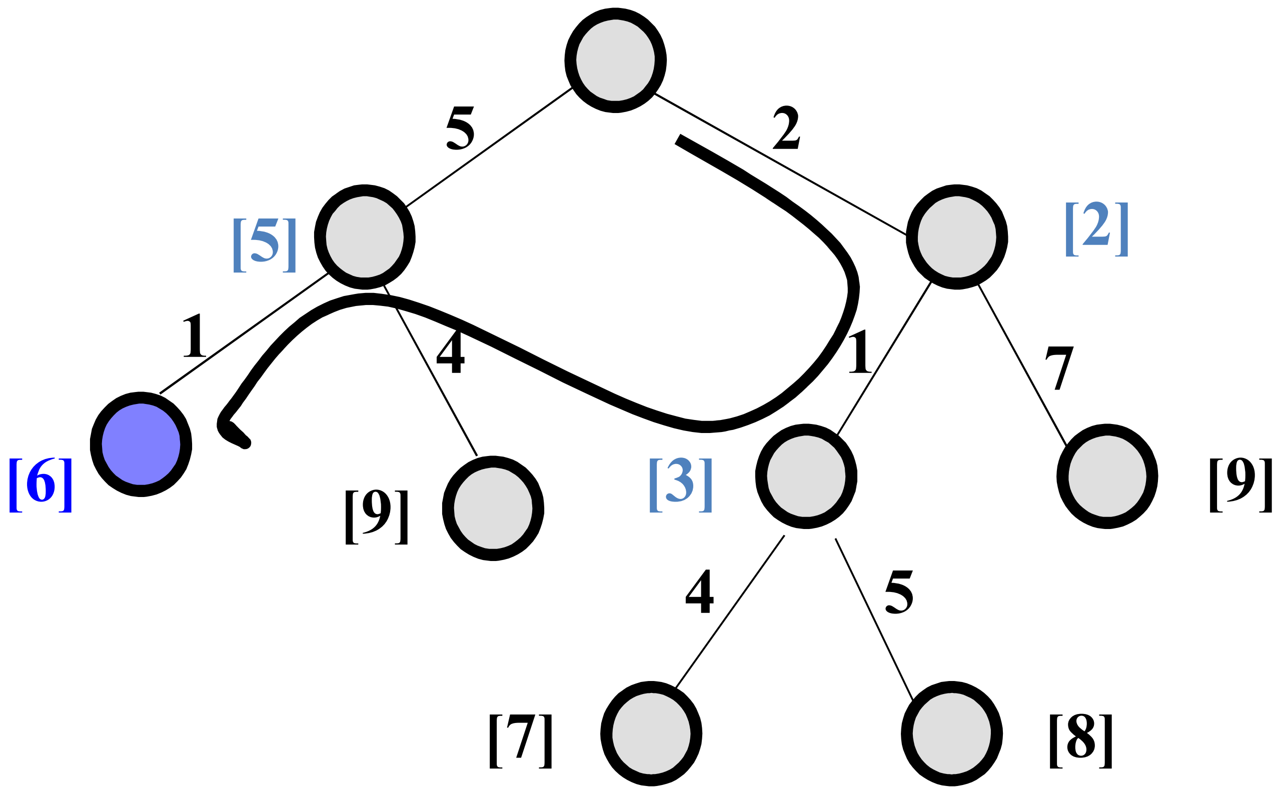
Uniform Cost Search (UCS)



Uniform Cost Search (UCS)



Uniform Cost Search (UCS)



Conclusion of The Chapter

- Conclusion #1
  - BFS implement FIFO (queue) for the search operation
- Conclusion #2
  - DFS implement LIFO (stack) for the search operation
- Conclusion #3
  - DLS implement DFS with level limitation
- Conclusion #4
  - IDS implement DLS with level increment
- Conclusion #5
  - UCS use cheapest cost node to expand