University of Science and Technology
Faculty of Computer Science and Information Technology

Artificial Intelligence (AI)

4th Year B.Sc : Information Technology

Academic Year : 2017-2018

Instructor : Diaa Eldin Mustafa Ahmed

Problem Solving
(Searching Strategies)-1/2

You will be expected to know

- Problem formalization.
- Problem Space (state , state space ,search tree , search node , goal , action and successor function)
- Uninformed Search :Depth First Search (DFS), Breadth First Search (BFS), Depth First Iterative Deeping Search (DFIDS).
- Heuristic Search: Best First Search, Hill Climbing, Constraint Satisfaction

Problem solving

Using artificial intelligence
(using intelligence similar to human intelligence or using Intelligent Agents)

- We want:
  - To automatically solve a problem.

Formalization

Searching technique

- We need:
  - A representation of the problem.
  - Algorithms that use some strategy to solve the problem defined in that representation.

Problem Formulation

- Goal formulation
  - Based on the current situation and the agent's performance measure, is the first step in problem solving.
- Goal is a set of states:
  - The agent's task is to find out which sequence of actions will get it to a goal state.
- Problem formulation :
  - is the process of deciding what sorts of actions and states to consider, given a goal .

Search Techniques for Artificial Intelligence

- Searching strategies is a central topic in Artificial Intelligence.
- This part of the course will show why search is such an important topic, present a general approach to representing problems to do with search.
- Introduce several search algorithms, and demonstrate how to implement these algorithms in Prolog.
- Problem solving strategies
  - Representing problem solution.
  - Basic search strategies.
    - ✓ Informed search strategies.
    - ✓ Uninformed search strategies .
- Search in AI.
  - Automated reasoning
  - Theorem proving
  - Game playing
  - Navigation

Classification of AI searching Strategies

- **Searching Strategies in AI**
  - **Un-informed (Blind Search)** — بحث أعمي
    - Depth-first (DFS)
      - Depth-Limited (DLS)
        - Iterative Deepening
    - Breadth-First (BFS)
    - Cost-First (CFS)
  - **Informed (Heuristic Search)** — بحث إرشادي(المعلوم)
    - Best-First Search (BFS)
      - A* Search
    - Hill Climbing
    - Constraint Satisfaction

A very large number of AI problems are formulated as search problems.

معظم المشكلات في الذكاء الإصطناعي يتم صياغتها كمشكلات بحث

What is Search strategy ?

❑ Search

- ➢ Search is the systematic examination of states to find path from the start/root state to the goal state.
- ➢ Search usually results from a lack of knowledge.
- ➢ Search explores knowledge alternatives to arrive at the best answer.
- ➢ Search algorithm output is a solution, ie, a path from the initial state to a state that satisfies the goal test.

Defining a Search Problem

A well-defined problem can be described by

- State space **S** **:** all possible configurations(*situations*) of the domain of interest .
  - *All **states reachable** from **initial** by any sequence of **actions.***
  - *Is a **graph** whose **nodes** are the set of all **states**, and whose **links** are **actions** that transform one state into another.*
  - *Each state is an abstract representation of the environment.*
  - *The state space is **discrete**.*
- An initial (start) state: $s_0 \in \mathbf{S}$
  - *Usually the **current state .***
  - *Sometimes **one** or **several** hypothetical states ("what if …")*
- Goal states $\mathbf{G} \subset \mathbf{S}$ **:**
  - *The set of **end states** –Often defined by a **goal test** rather than enumerating a set of states.*

Defining a Search Problem

A well-defined problem can be described by

- **Operators (Action) A :**
  - *The **actions available**–often defined in terms of a mapping from a state to its **successor**.*
  - *Is something that the **agent** can **choose** to do.*
- **Search tree :**
  - *(A **graph** with **no undirected loops**) in which the **root node** is the **start state** and the set of **children** for each node consists of the states reachable by taking any action.*
- **Search node :**
  - *Is a **node** in the **search tree**.*
- **Goal :**
  - *Is a state that the **agent (solution)** is trying to reach.*

Defining a Search Problem

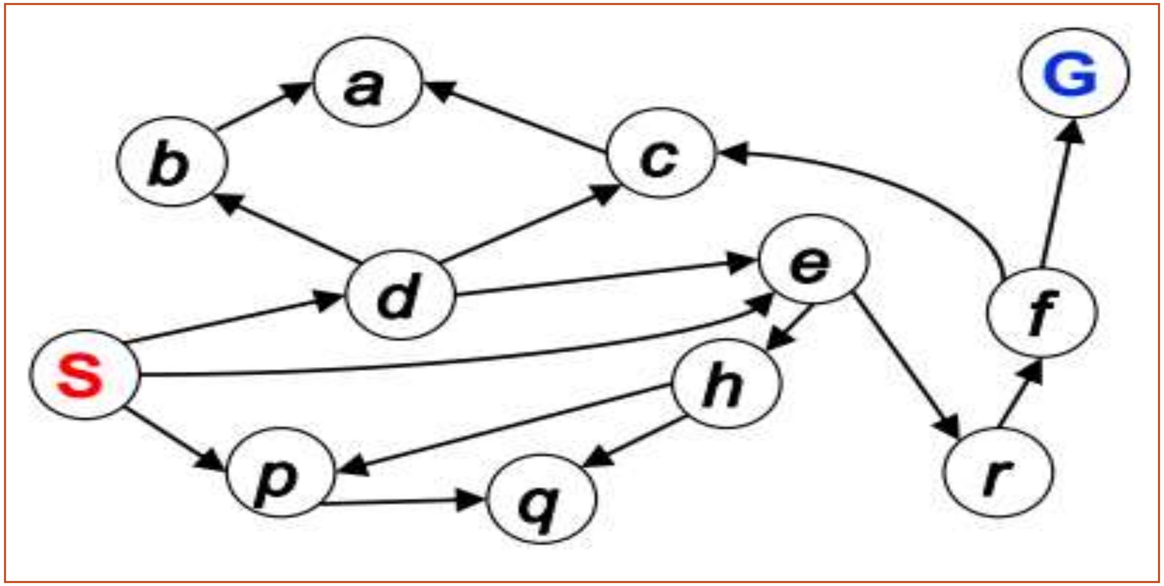
A well-defined problem can be described by

- **Path :**
  - *A sequence of states and operators.*
  - *Sequence through state space.*
- **Path cost:**
  - *A **number** associated with **any path**.*
  - *Measures the **quality of the path**.*
  - ***Sum of costs** of individual **actions** along the path.*
  - *Usually the **smaller**, the **better**.*
- **Solution of a search problem:**
  - *Is a **path from** S_0 **to some** $S_g \in G$.*
- **Optimal solution:**
  - *Any path with minimum cost.*
- **Goal test :**
  - *Test to determine if at **goal state.***

State Space Graph versus Search Trees

❑State Space Graph

- Graph of **states** with arrows pointing to successors.
- State graph shows the possible states of a system.
- A state is a node in which a system can be at any given time.
- May contain a loop.

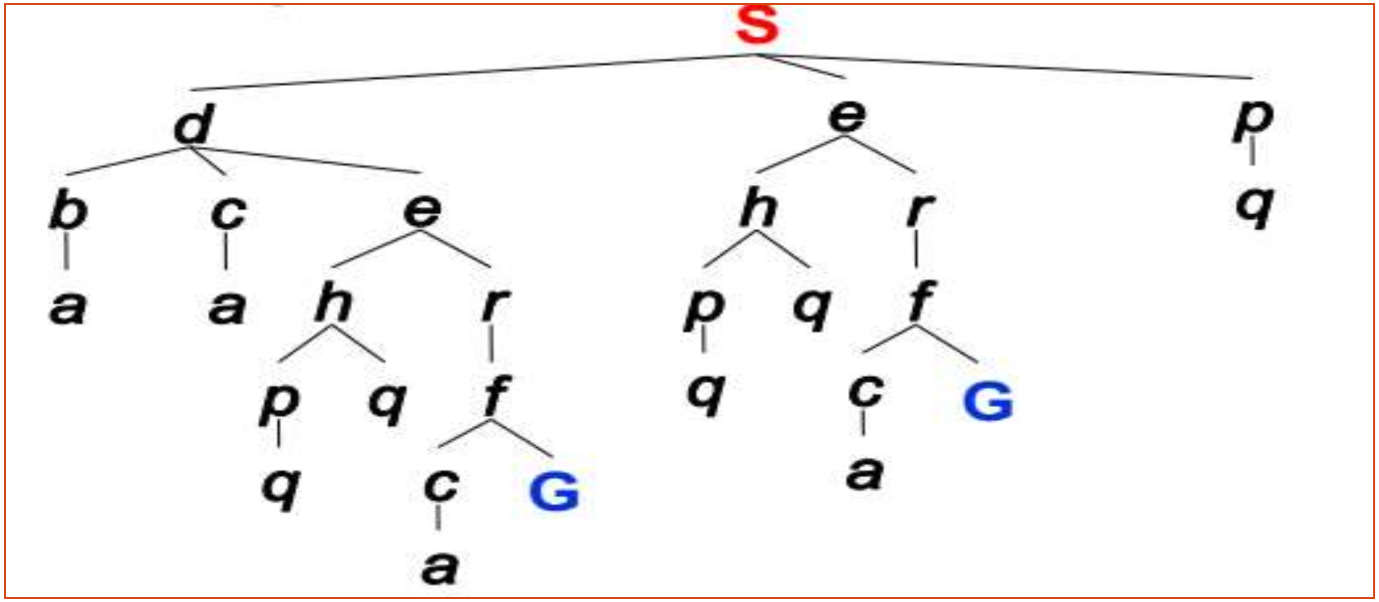


State Space Graph

Defining a Search Problem

❑State Space Tree

- Tree is a special case of a graph.
- The **topmost node** in a tree is called the **root node**; at root node all operations on the tree begin.
- Each **NODE** in in the search tree is an entire **PATH** in the problem graph.
- Represent a **plan** (sequence of actions) which results in the node's state .



State Space Tree

Defining a Search Problem

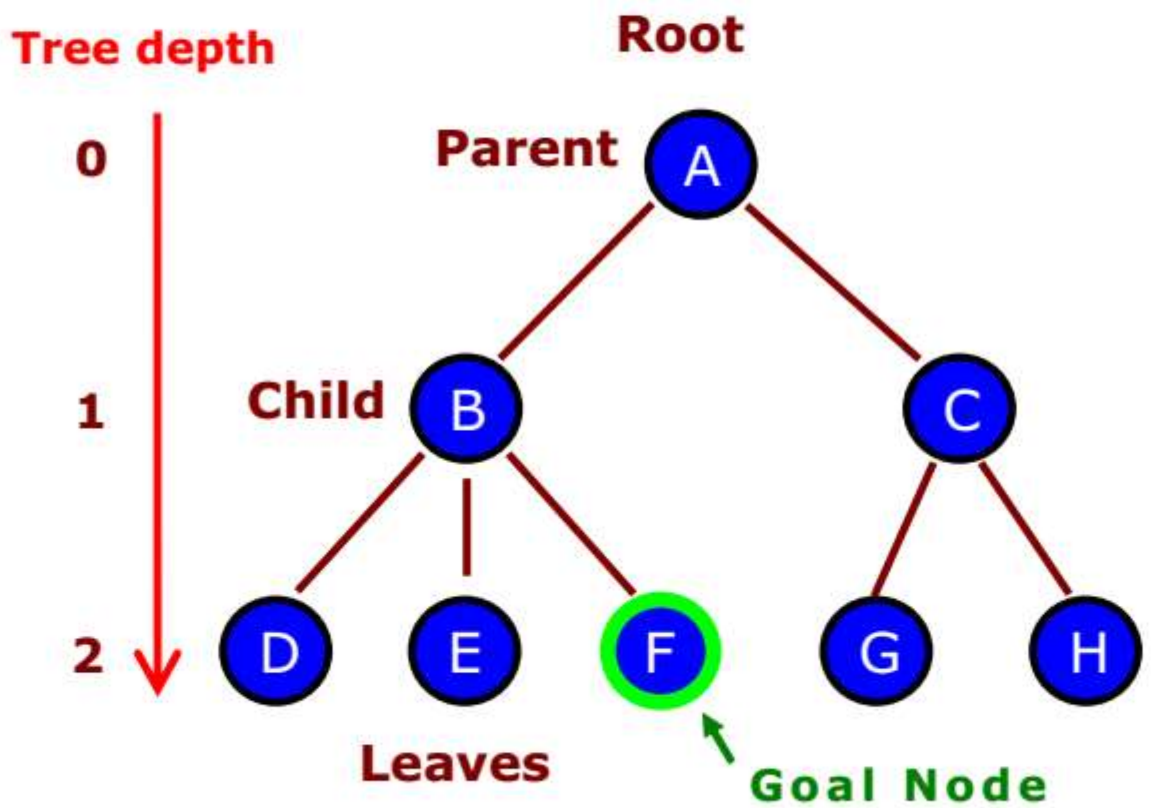
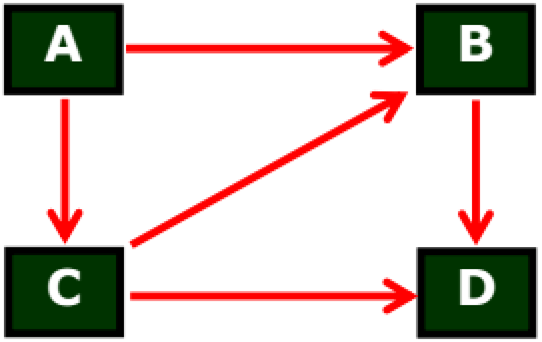


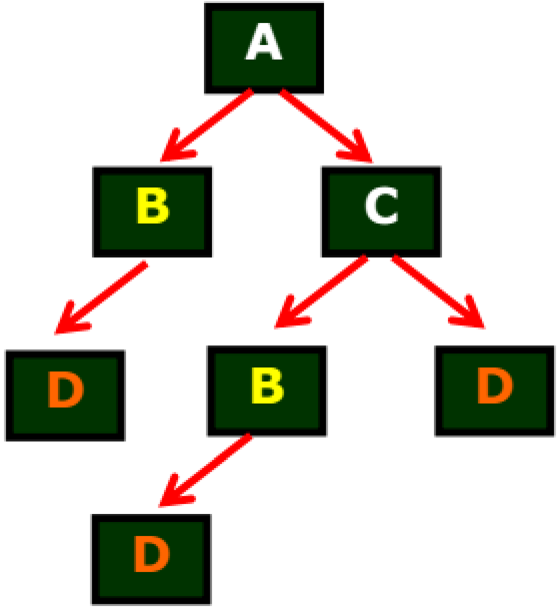
Fig. A simple example unordered tree

Graph vs. Tree

Graph

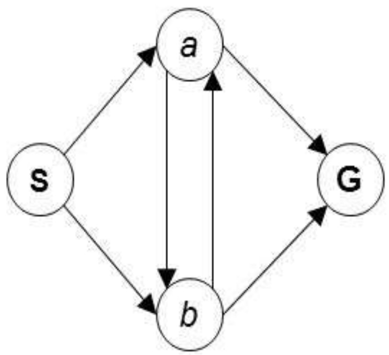


Trees

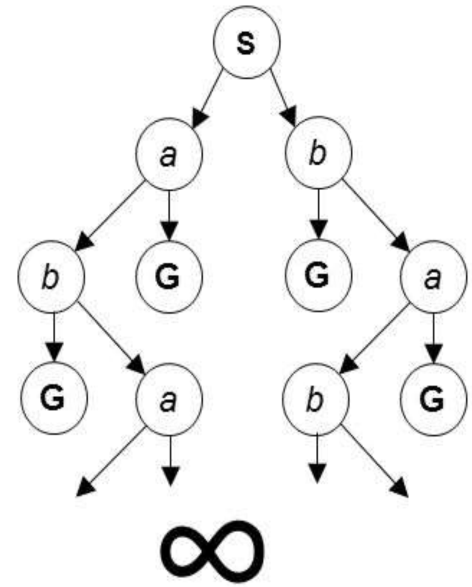


Quiz: State Space Graphs vs. Search Trees

Consider this 4-state graph:



How big is its search tree (from S)?

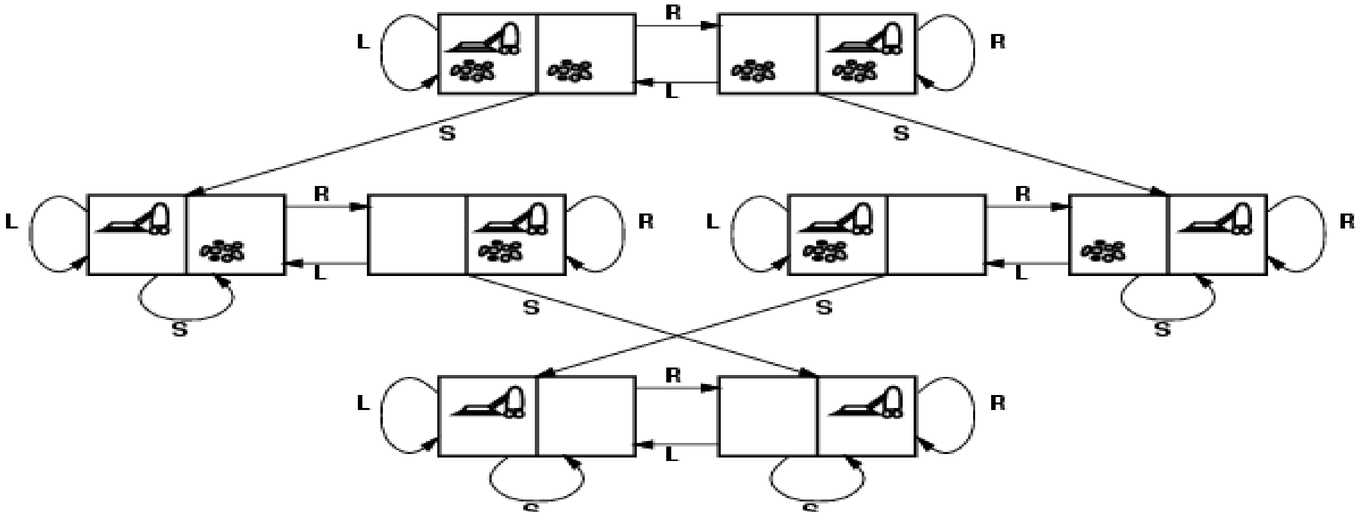


Important: Lots of repeated structure in the search tree!

Problem Solving by Search

- An important aspect of intelligence is *goal-based* problem solving.
- The *solution* of many *problems* can be described by finding a sequence of *actions* that lead to a desirable *goal*.
- Each action changes the *state* and the aim is to find the sequence of actions and states that lead from the *initial (start)* state to a *final (goal)* state.

Example (1) : Vacuum Cleaner world state space graph



States? Discrete: dirt and robot location

Initial state? Any

Actions? *Left*, *right*, *suck*

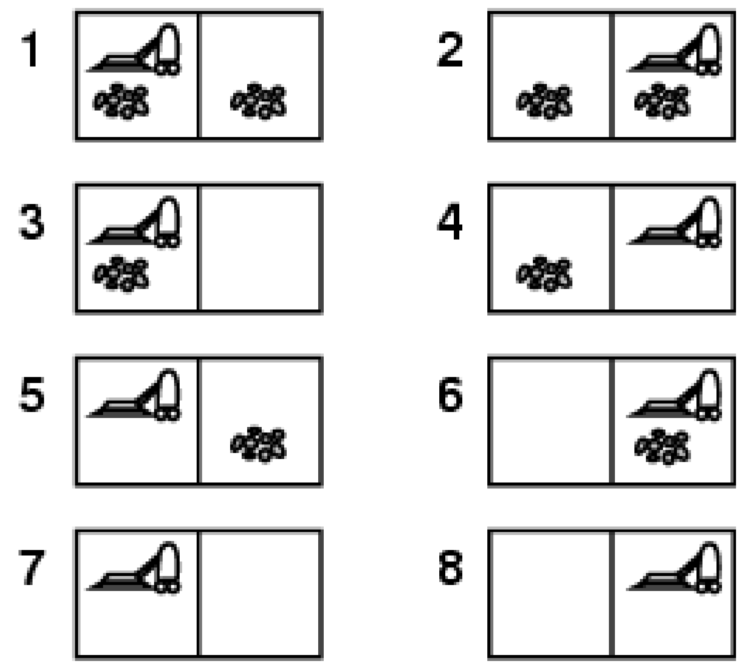
Goal test? No dirt at all locations

Path cost? 1 per action

Solution Optimal sequence of operations(actions)

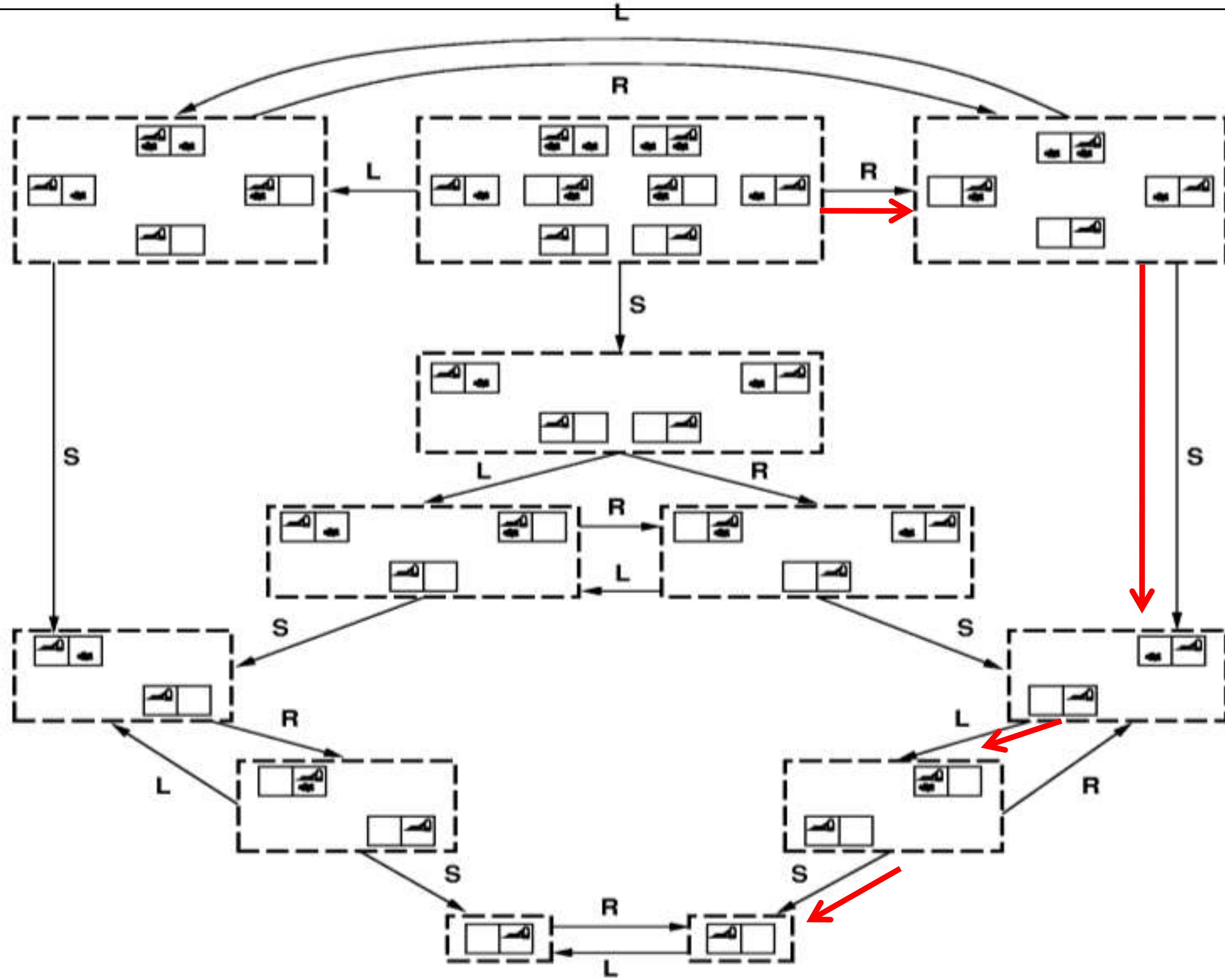
Example (1) : Vacuum Cleaner world state space

- **Observable**, start in #5.
  Solution? ***[Right , Suck]***
- **Observable**, start in #2.
  Solution? ***[Suck , Left , Suck]***
- **Observable**, start in #6.
  Solution? ***[Suck , Left]***
- **Observable**, start in #1.
  Solution? ***[Suck , Right , Suck]***

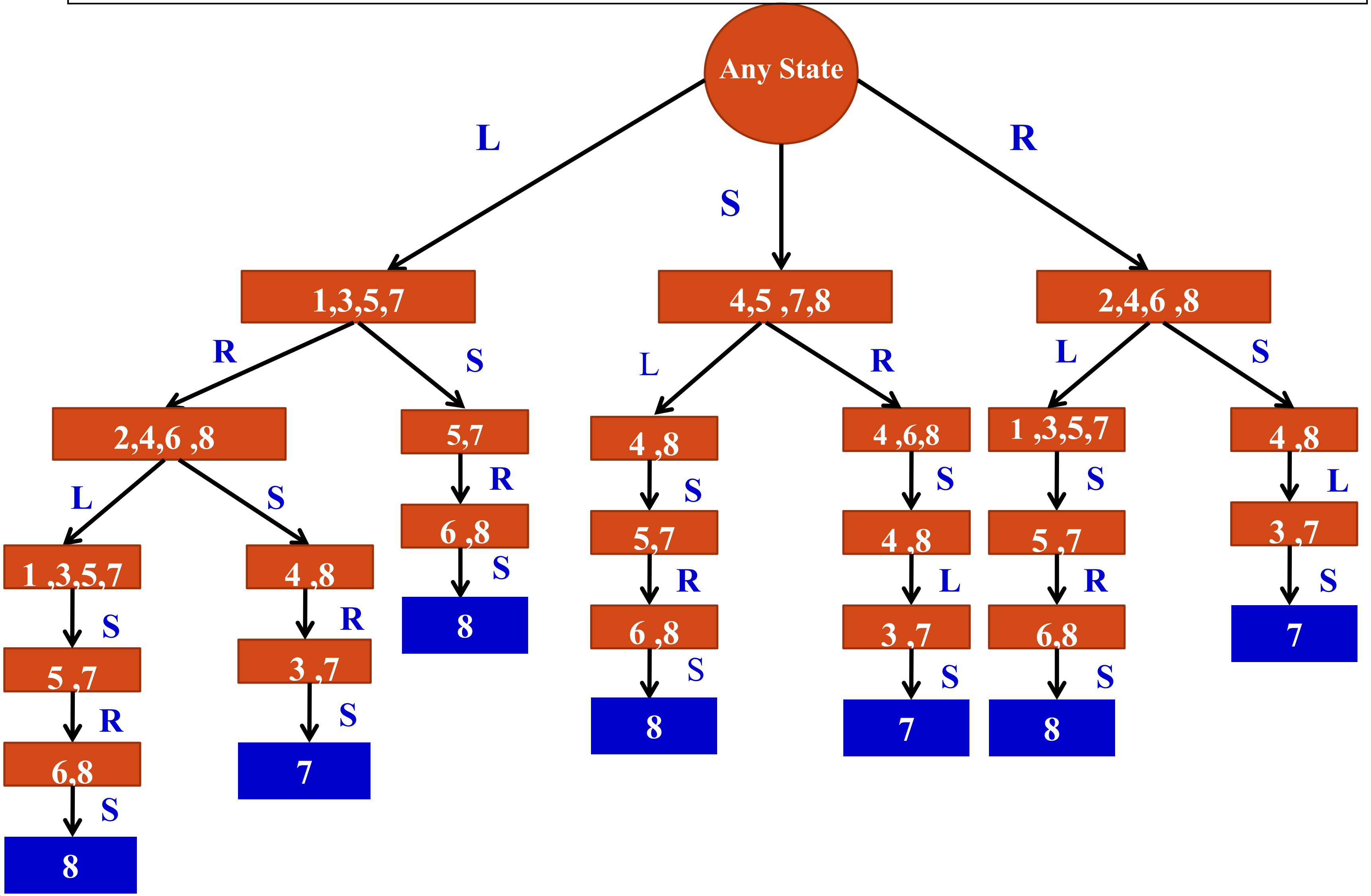


- **Unobservable**, start in {1,2,3,4,5,6,7,8}
  Solution? ***[Right , Suck , Left , Suck]***

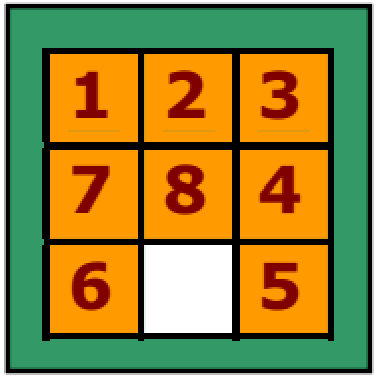
Example (1) : Vacuum Cleaner world state space graph



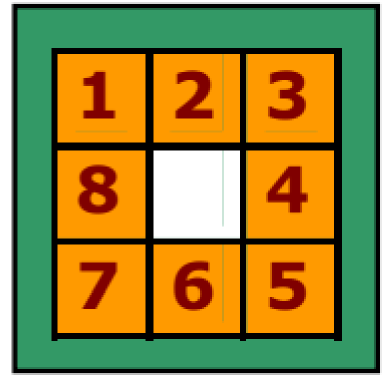
Example (1) : Vacuum Cleaner world state space tree



Example(2): The 8-puzzle Problem
state space

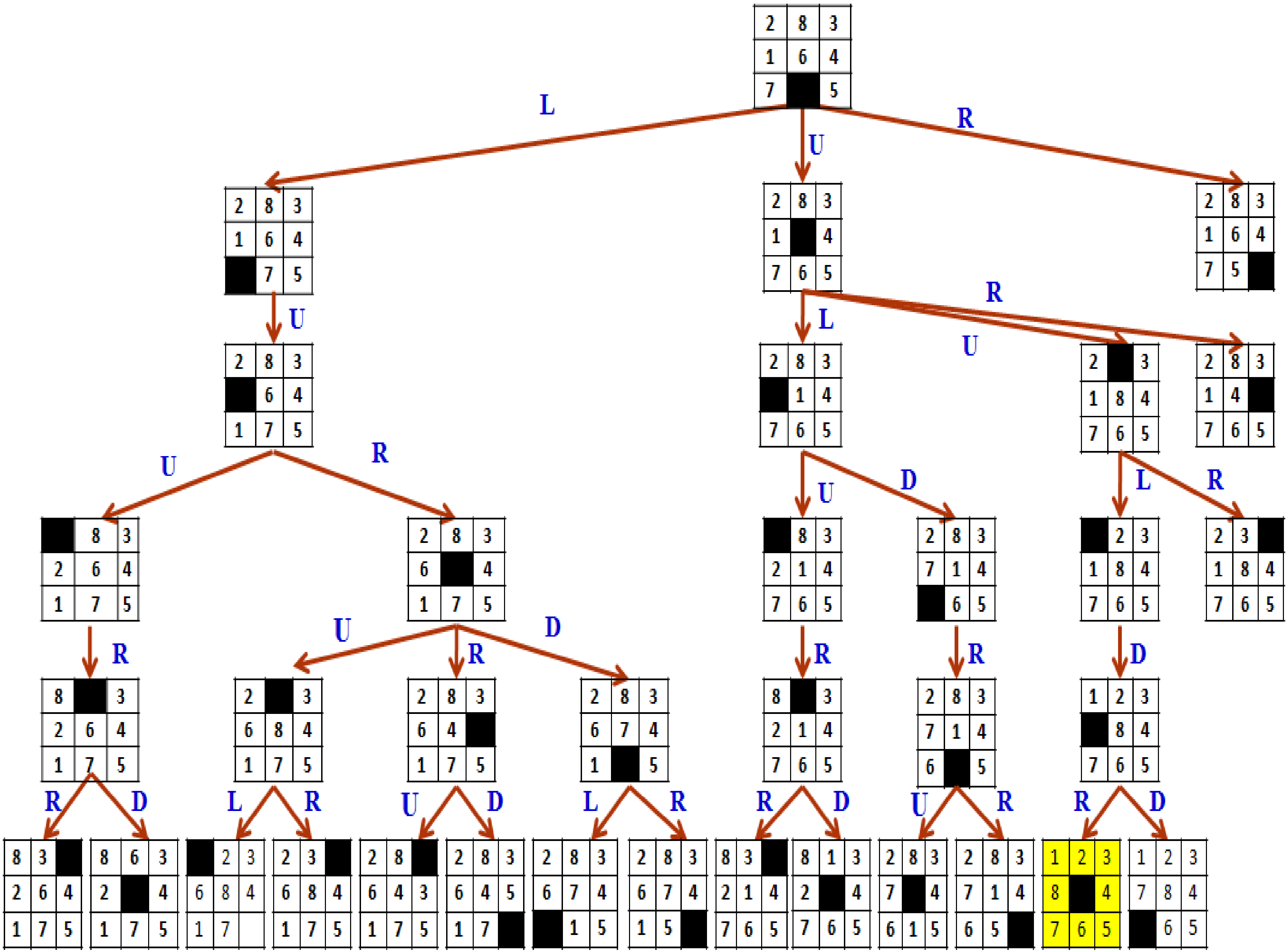


Initial State: any configuration

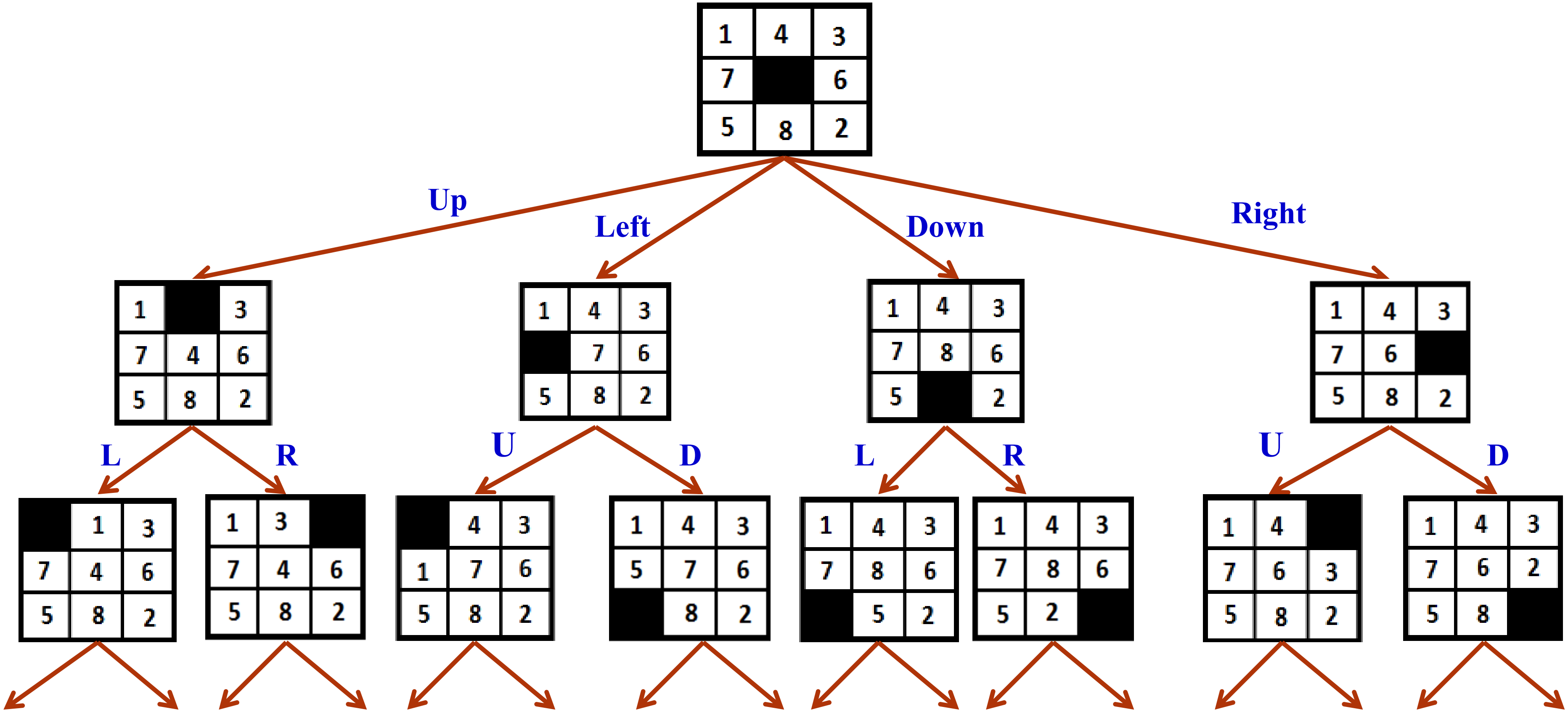


Goal State : tiles in a specific order

States?	Locations of tiles
Initial State?	Given
Actions?	Move blank left, right, up, down
Goal test?	Goal state (given)
Path cost?	1 per move
Solution:	Optimal sequence of operators

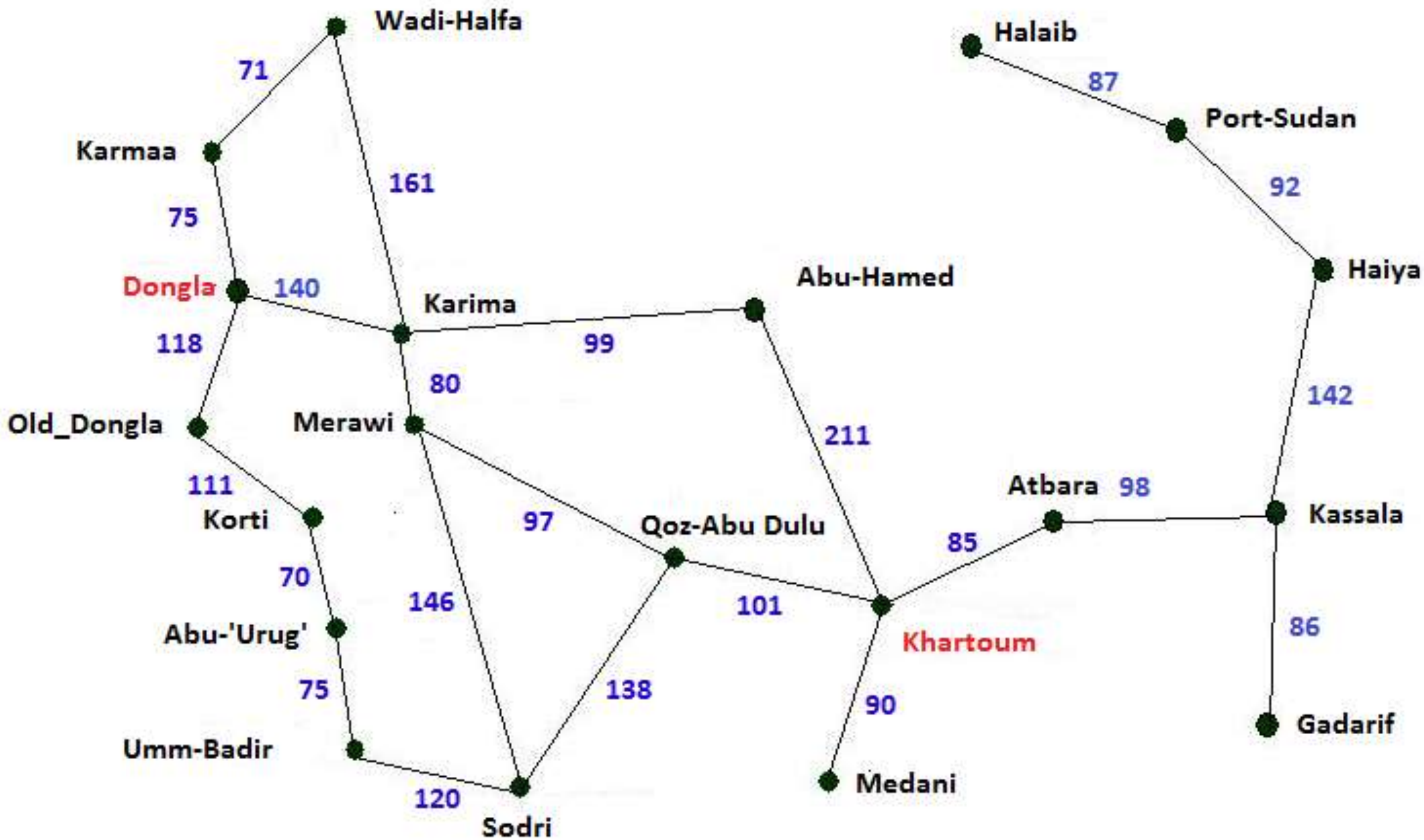

L
U
R
U
R
L
U
R
U
R
U
D
L
R
R
U
R
D
R
R
D
R
L
R
U
D
L
R
R
D
U
R
R
D

Example(2): The 8-puzzle Problem
State space tree



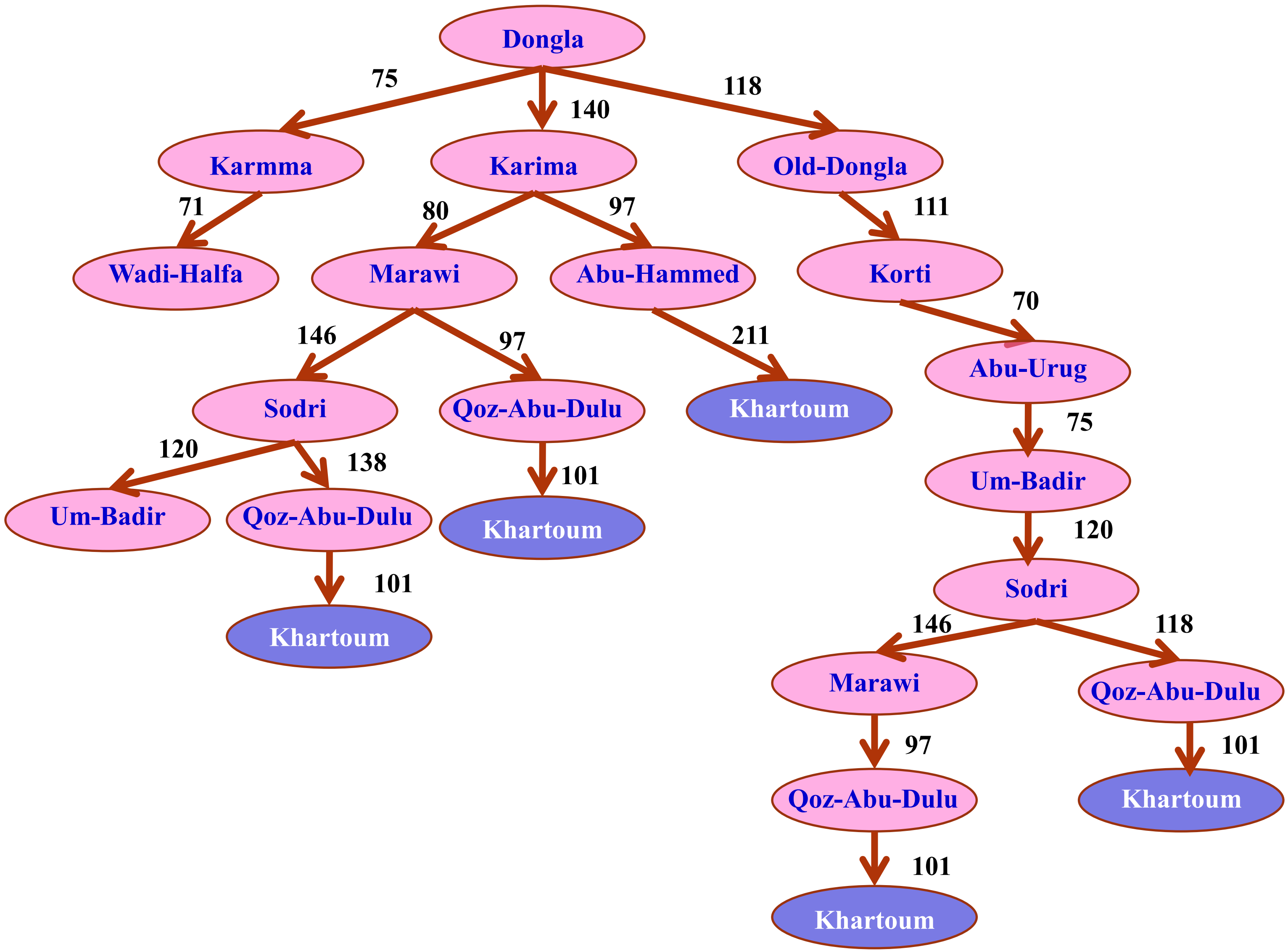
Example (3) :Route Planning: Dongla ⟶ Khartoum

- On holiday in Khartoum; currently in Dongla.

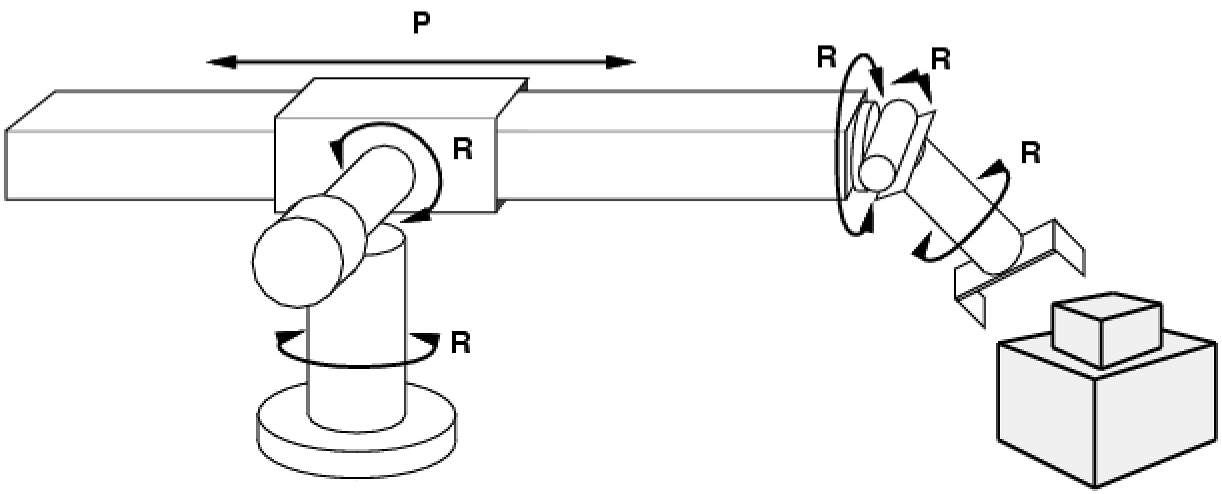


Example (3) :Route Planning: Dongla ⟶ Khartoum

States?	Various cities
Initial State?	Dongla
Actions?	Drive between cities or choose next city
Goal test?	Be in Khartoum
Path cost?	Distance in km
Solution:	Sequence of cities, e. g. Dongla, Karima , Abu-hammed, Khartoum.


Dongla
75
140
118
Karmma
Karima
Old-Dongla
71
80
97
111
Wadi-Halfa
Marawi
Abu-Hammed
Korti
70
146
97
211
Abu-Urug
Sodri
Qoz-Abu-Dulu
Khartoum
75
120
138
101
Um-Badir
Um-Badir
Qoz-Abu-Dulu
Khartoum
120
101
Khartoum
Sodri
146
118
Marawi
Qoz-Abu-Dulu
97
101
Qoz-Abu-Dulu
Khartoum
101
Khartoum

Example (4): Robotic assembly



- **states?**: real-valued coordinates of robot joint angles parts of the object to be assembled
- **initial state?:** rest configuration
- **actions?**: continuous motions of robot joints
- **goal test?**: complete assembly
- **path cost?**: time to execute

Uninformed search

- ***Uninformed Search*** (also called ***blind*** search) is a search that has no information about its domain.
  - Use only the information available in the problem definition.
  - Generates the search tree without using any domain specific knowledge.
  - The only thing that a blind search can do is distinguish a non-goal state from a goal state.
  - Brute-force algorithms search, through the search space, all possible candidates for the solution checking whether each candidate satisfies the problem's statement.

Uninformed search

- **Breadth-first search (BFS)**
- **Depth-first search (DFS)**
- **Uniform-cost search**
- **Depth-limited search**
- **Iterative deepening search**

Informed Search

- ***Informed Search*** algorithms use heuristic functions, that are specific to the problem, apply them to guide the search through the search space to try to reduce the amount of time spent in searching.
- .

Breadth-first search (BFS)

- A Search strategy, in which the highest layer of a decision tree is searched completely before proceeding to the next layer is called Breadth-first search (BFS).
  - When a state is examined, all of its siblings are examined before any of its children.
  - The space is searched level-by-level, proceeding all the way across one level before doing down to the next level.

    In this strategy, no viable solution is omitted and therefore guarantee that optimal solution is found.
  - This strategy is often not feasible when the search space is large.
  - **BFS** explores nodes nearest to the root before exploring nodes that are father or further away.

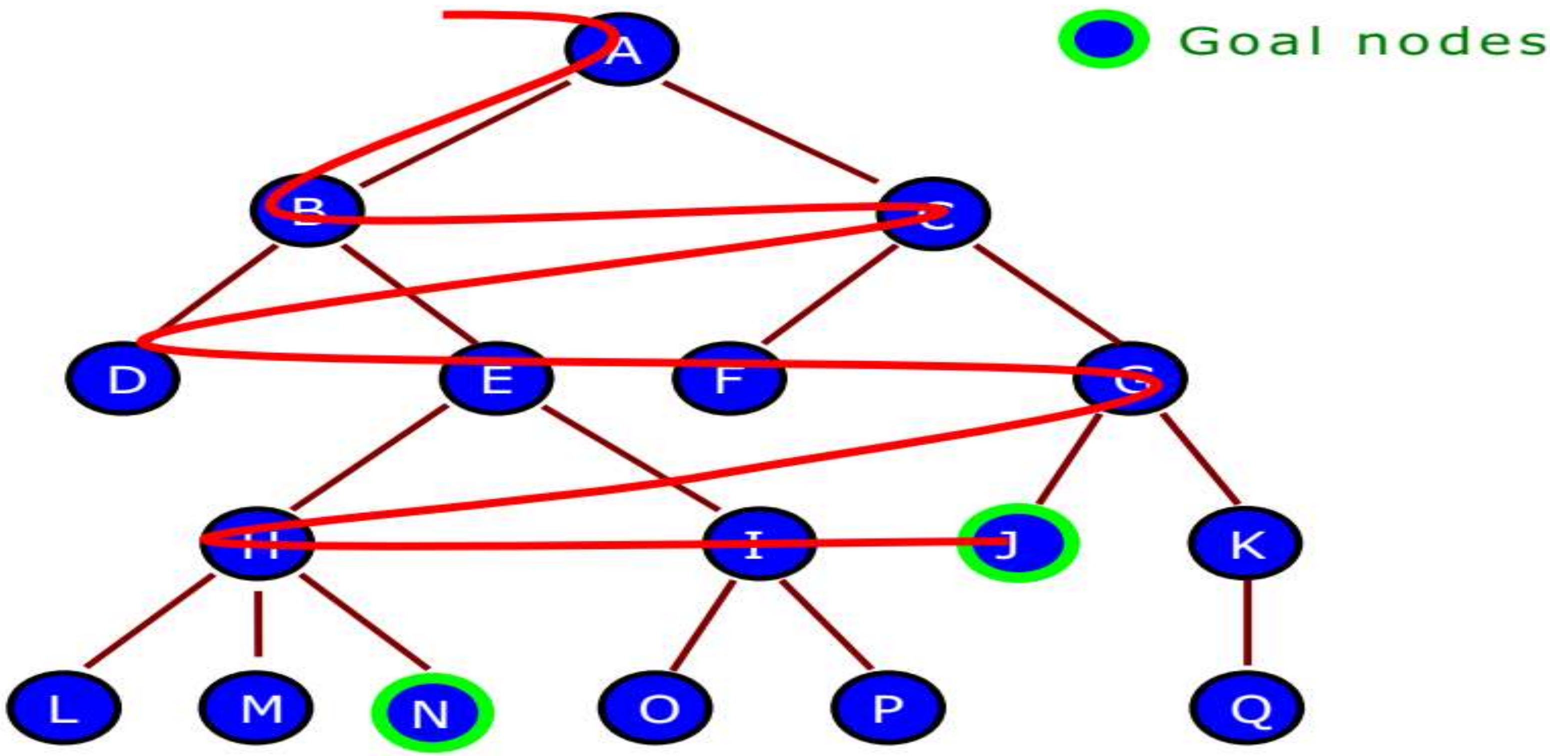
Breadth-first search (BFS)

- **Algorithm -** Breadth-first search

  - Put the root node on a queue;

```
while (queue is not empty)
  { remove a node from the queue;
    if (node is a goal node) return success;
    put all children of node onto the queue; }
return failure;
```

Breadth-first search (BFS)



Node are explored in the order :

A B C D E F G H I J K L M N O P Q

- After searching **A**, then **B**, then **C**, the search proceeds with **D,E, F, G**, ...
- The goal node **J** will be found before the goal node **N**.

Depth-first search (DFS)

- A search strategy that extends the current path as far as possible before backtracking to the last choice point and trying the next alternative path is called Depth-first search (DFS).
  - When a state is examined, all of its children and their descendants are examined before any of its siblings.
  - DFS goes deeper in to the search space when ever this is possible only when no further descendants of a state can found.
  - This strategy does not guarantee that the optimal solution has been found.
  - In this strategy, search reaches a satisfactory solution more rapidly than breadth first, an advantage when the search space is large.

Depth-first search (DFS)

The Breadth-first search (BFS) and depth-first search (DFS) are the foundation for all other search techniques.

- **Algorithm -** Depth-first search
  - Put the root node on a stack;

    while (stack is not empty)

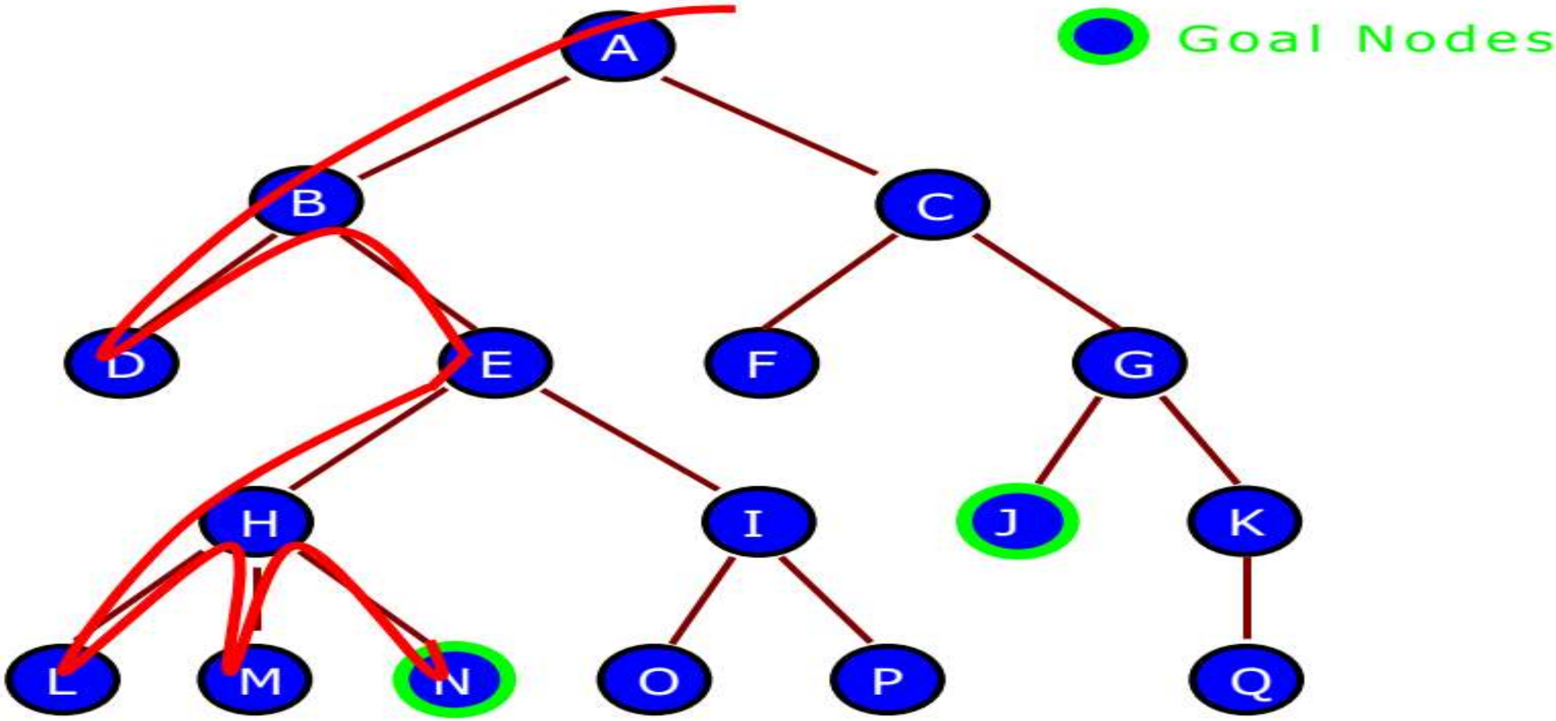
    { remove a node from the stack;

    if (node is a goal node) return success;

    put all children of node onto the stack; }

    return failure;

Depth-first search (DFS)



Node are explored in the order :

A B D E H L M N I O P C F G J K Q

- After searching node **A**, then **B**, then **D**, the search backtracks and tries another path from node **B** .
- The goal node **N** will be found before the goal node **J**.

DFS vs BFS Algorithm

Depth-first search	Breadth-first search

- **Compare Algorithms**

Depth-first search

```
Put the root node on a stack;
while (stack is not empty)
{
  remove a node from the stack;

   if (node is a goal node)
      return  success;

   put all children of node
   onto the stack;
}
return failure;
```

Breadth-first search

```
Put the root node on a queue;
while (queue is not empty)
{
  remove a node from the  queue;

   if (node is a goal node)
      return  success;

   put all children of node
   onto the queue;
}
return failure;
```

Thank You

End