Artificial Intelligence

Introduction

Search Algorithms in AI

- Search techniques are ubiquitous in AI.

A search problem have 3 main elements

- 1. Search Space : A set of possible solutions.
- 2. Start State : A state where agent begins the search.
- 3. Goal State : Looks at the current state and checks, whether the goal is achieved or not.

Search Algorithms in AI

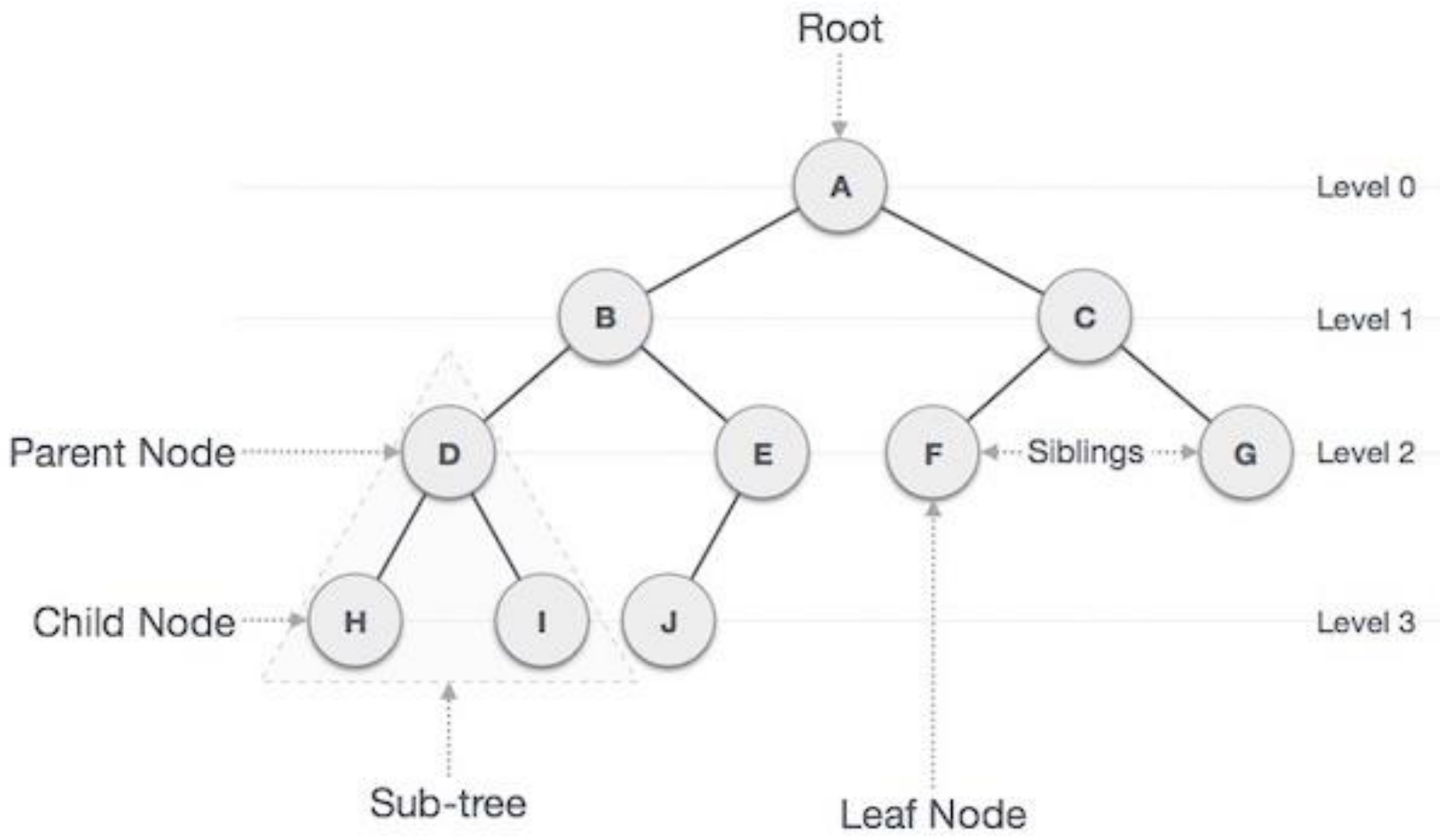
Important properties of Algorithm:

1. **Completeness** : For an random input, if the algorithms reaches to some particular solution. Then the algorithm is said to complete.
2. **Optimality** : If the solution is guaranteed to be the best solution.
3. **Time Complexity** : Measure of time the algorithm takes to come with a solution.
4. **Space Complexity** : The storage space required by an algorithm during execution.

Search Algorithms in AI

What is Tree and Graph data-structure ?

Tree Data Structure



Tree Data Structure

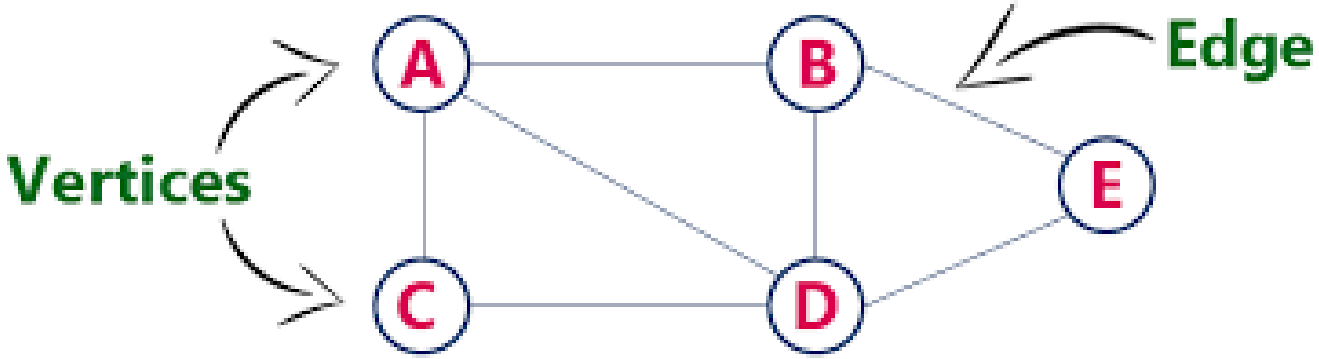
Search Algorithms in AI

Tree Data Structure

- Non-linear data structure and follows hierarchical model.
- Elements are arranged in levels
- It has a unique node known as parent/root node.
- It does not create any loop or cycle.
- Always have directed edges
- Other than Search, operations can be insert, delete, searching.

Search Algorithms in AI

Graph Data Structure



Graph Data Structure

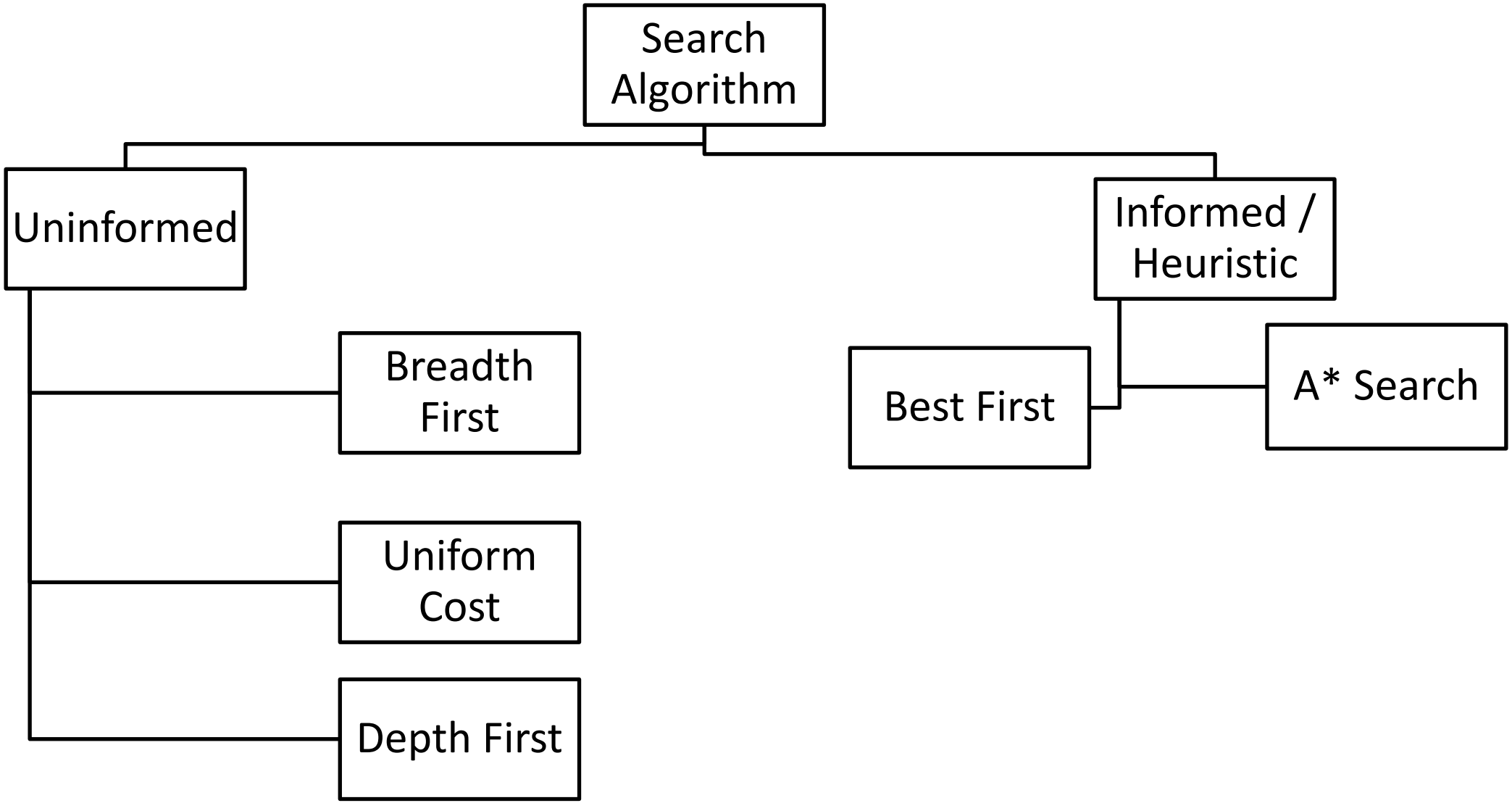
Search Algorithms in AI

Graph Data Structure

- Non-linear data structure and follows Network model.
- It has a NO unique node.
- It has loop or cycle.
- Can have directed or undirected edges.
- Used for searching shortest path in a network.

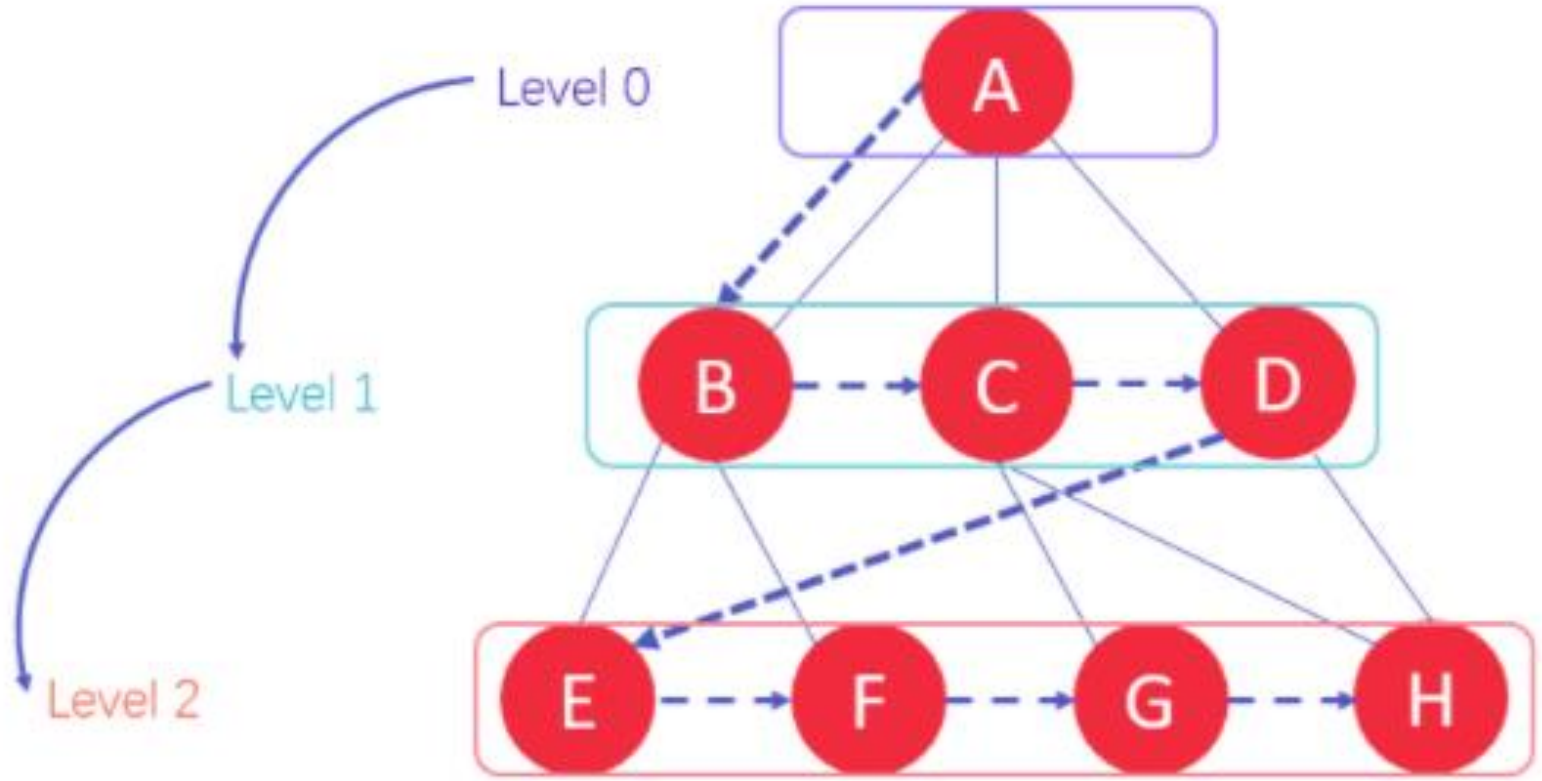
Search Algorithms in AI

Types of Search



Search Algorithms in AI

Breadth First Search :

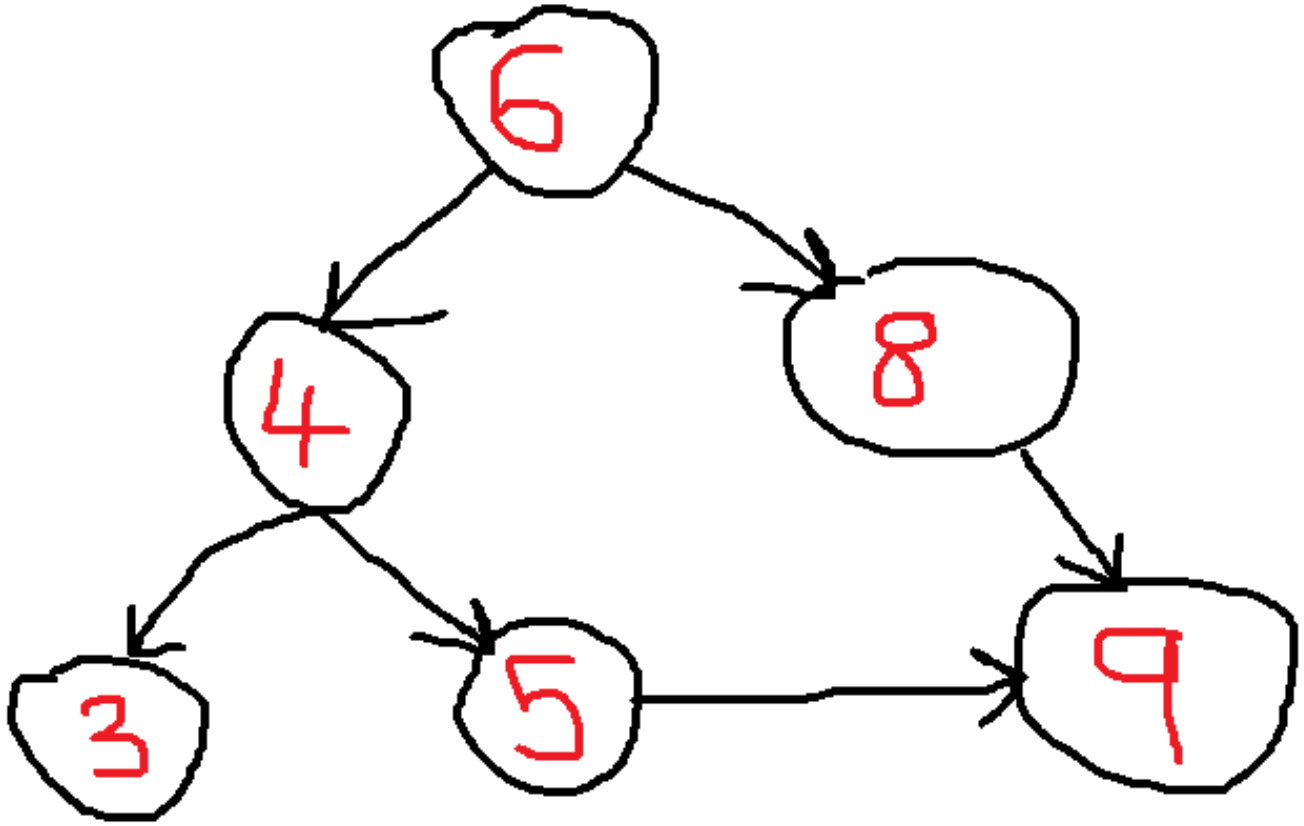


3 raised to 2 = 9. Hence it can find the node in 9 searches. 3 is max children for a node, 2 is no of levels

Ref : www.guru99.com

Search Algorithms in AI

Breadth First Search :



Search Algorithms in AI

Breadth First Search :

- It is uninformed / Brute-Force search technique.
- It uses Queue (FIFO) data structure while searching.
- Searches for nodes at same level first then moves to next level (Shallowest node preference). Hence also called Level order searching.
- It always searches. Hence the algorithm is complete.
- Time Complexity is O(V+E) or $O(b^d)$ b=branch factor (max number of children a node have, d=depth of a tree/graph [no. of levels])
- It requires more memory than DFS.

Search Algorithms in AI

BFS : The Algorithm

1. Enter starting nodes on a Queue.
2. If Queue is empty, then return stop.
3. If first element on Queue is Goal node then return success and stop,
   Else
4. Remove and expand first element from Queue and place children at end of queue.
5. Go to step 2.

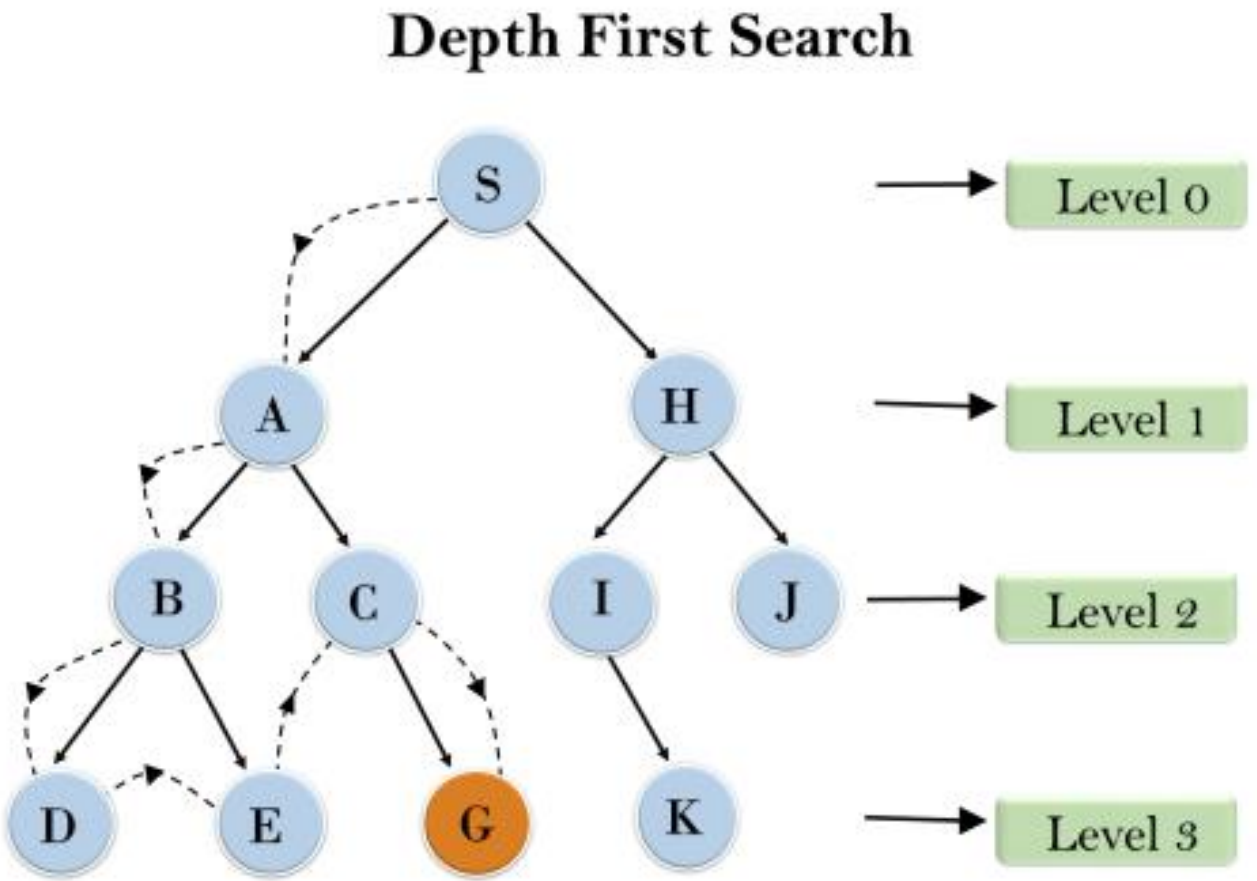
Search Algorithms in AI

Features Breadth First Search :

- The traversal starts from root node and go level by level.
- The visited nodes are added and removed from queue putting its next level node.
- Useful in finding shortest path in a tree/graph.
- BFS is slower compared to DFS
- space complexity of the BFS algorithm is O(V)
- Time Complexity is O(V+E) or $O(b^d)$

Search Algorithms in AI

Depth First Search:



2 raised to 3 = 8. Hence it can find the node in 8 searches. 2 is max children for a node, 3 is no of levels

Ref : www.guru99.com

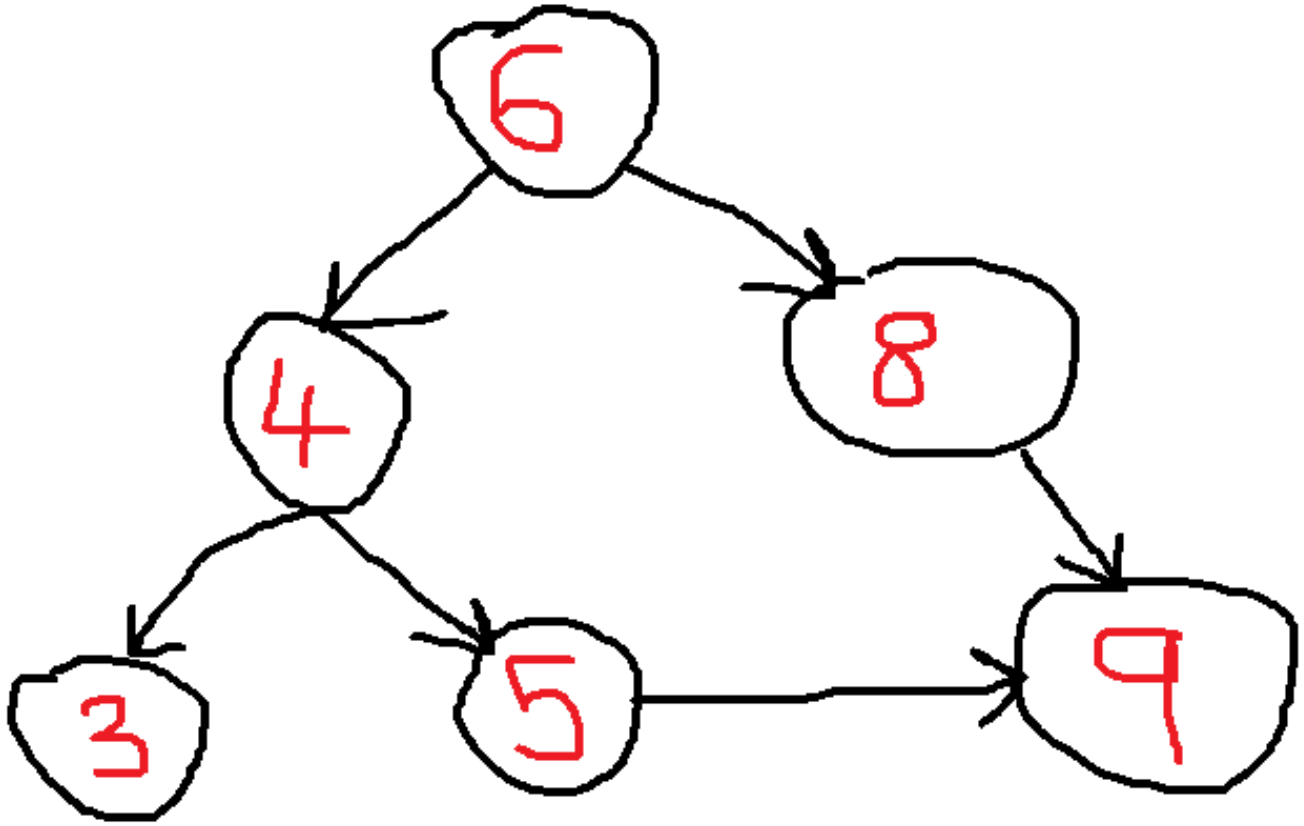
Search Algorithms in AI

Depth First Search :

- It is uninformed / Brute-Force search technique.
- It uses Stack (LIFO) data structure while searching.
- Searches for nodes in depth and than moves to next path.
- It may go endless in depth in infinite loop. Hence it is incomplete.
- Time Complexity is O(V+E) or $O(b^d)$ b=branch factor (max number of children a node have, d=depth of a tree/graph [no. of levels])
- space complexity of the BFS algorithm is O(V)

Search Algorithms in AI

Depth First Search : The Algorithm



Search Algorithms in AI

Informed Search Algorithms :

- Contains knowledge about reaching the goal node.
- This knowledge helps to find goal node by decreasing search spaces.
- It is also called Heuristic search.

Heuristic Function:

- This function takes current state as input and finds out how far the goal is.
- The value of the heuristic function is always positive.
- It expands nodes based on their heuristic value h(n).
- It maintains two lists, OPEN(Queue/Pq) and CLOSED/Visited list.
- In the CLOSED list, it places those nodes which have already expanded.
- In the OPEN list, it places nodes which have yet not been expanded.
- On each iteration, each node n with the lowest heuristic value is expanded and generates all its successors and n is placed to the closed list.
- The algorithm continues unit a goal state is found.

Search Algorithms in AI

About Best First Search:

- It is a **greedy search**
- Selects the best path
- It is **combination of DFS and BFS**
- It **uses Heuristic** function to search for the best path.
- It is **implemented using Priority Queue**.
- **Not optimal and Not complete** but can give Best path result.

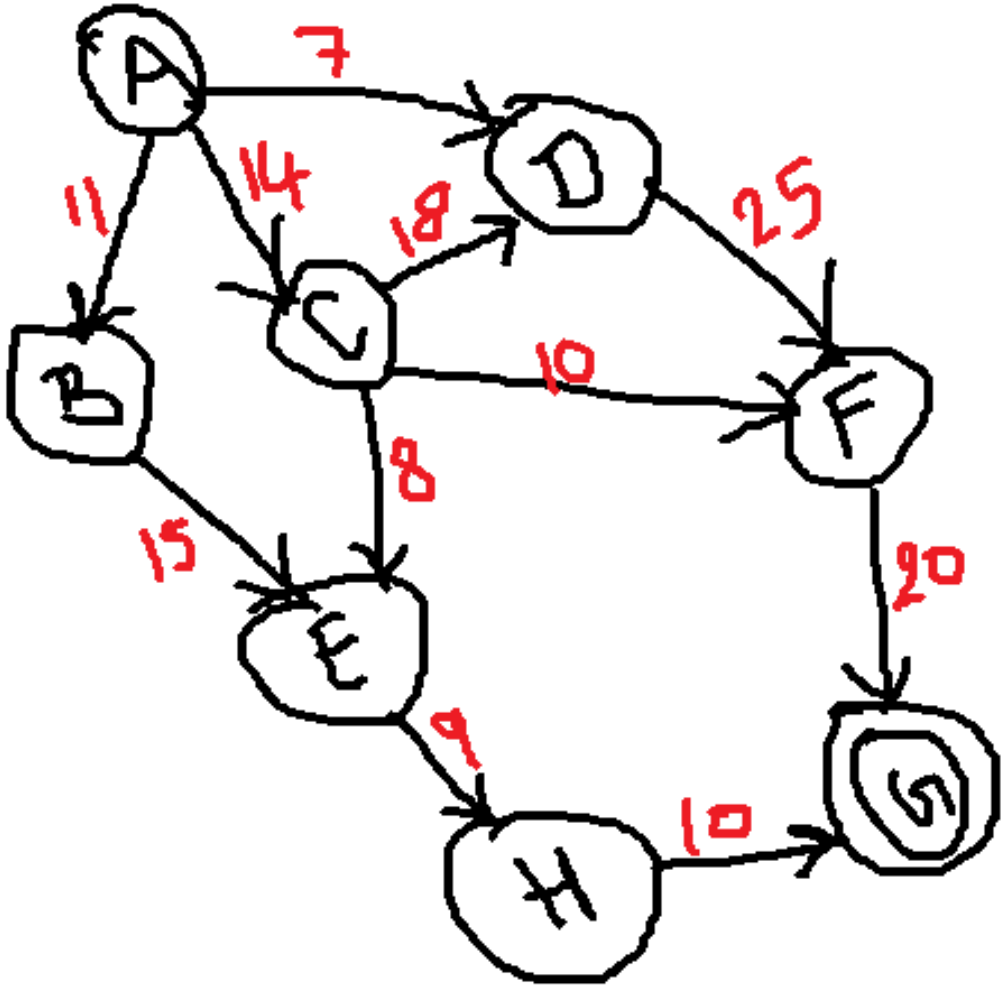
Search Algorithms in AI

Best First Search Algorithm:

- **Step 1:** Place the starting node into the OPEN list.
- **Step 2:** If the OPEN list is empty, Stop and return failure.
- **Step 3:** Remove the node n, from the OPEN list which has the lowest value of h(n), and places it in the CLOSED list.
- **Step 4:** Expand the node n, and generate the successors of node n.
- **Step 5:** Check each successor of node n, and find whether any node is a goal node or not. If any successor node is goal node, then return success and terminate the search, else proceed to Step 6.
- **Step 6:** For each successor node, algorithm checks for evaluation function f(n), and then check if the node has been in either OPEN or CLOSED list. If the node has not been in both list, then add it to the OPEN list.
- **Step 7:** Return to Step 2.

Search Algorithms in AI

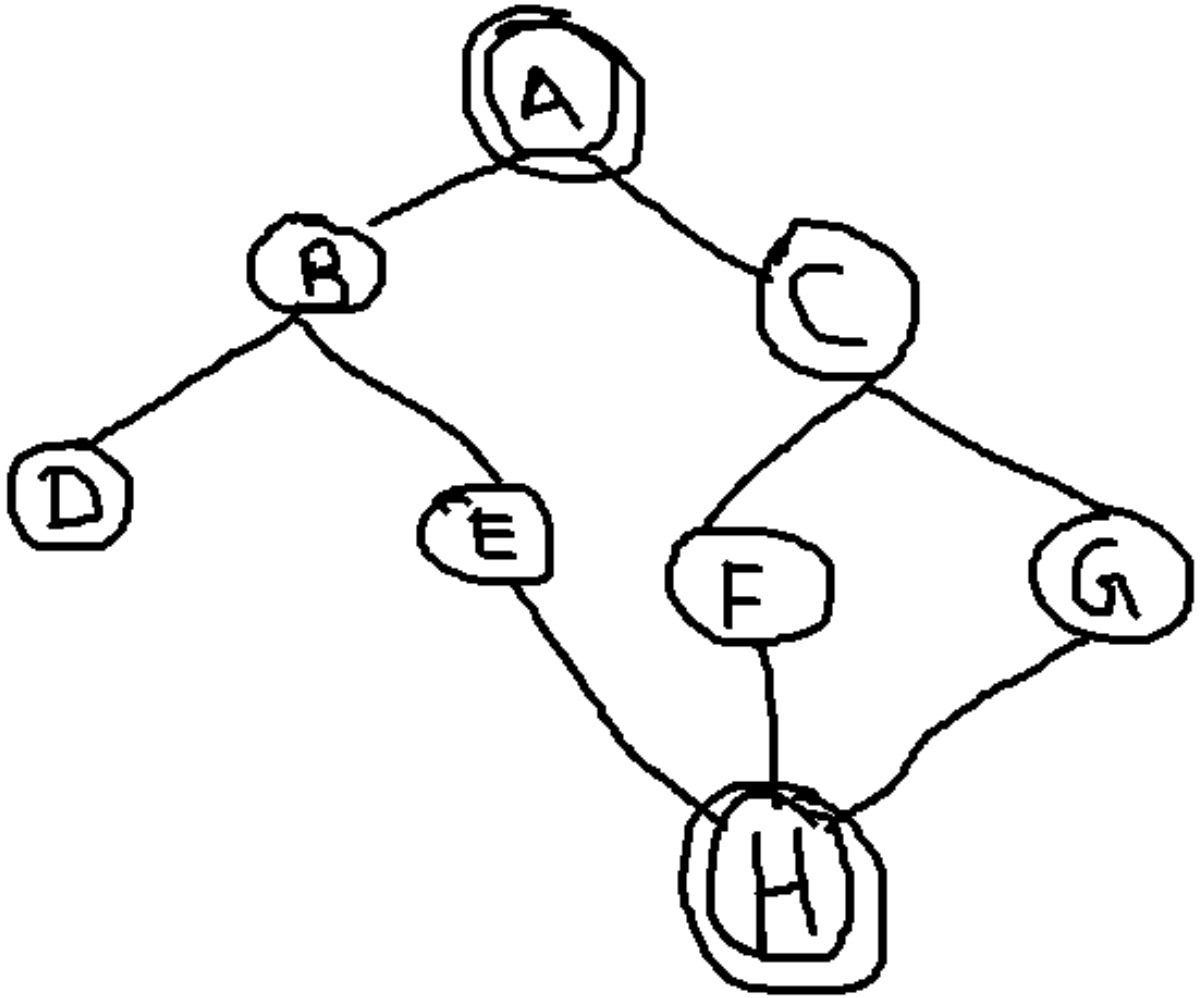
Best First Search Algorithm:



A→G = 40
B→G = 32
C→G = 25
D→G = 35
E→G = 19
F→G = 17
G→G = 0
H→G = 10

Search Algorithms in AI

Best First Search Algorithm:



A-H = 13
B-H = 12
C-H = 4
D-H = 7
E-H = 3
F-H = 8
G-H = 2
H-H = 0

Search Algorithms in AI

About A* Search: //(Dijikstra)

- It is **another form of Best First Search**. It is **optimal and complete**. Memory requirement is high.
- Selects the best path by using heuristic function and cost function.
- Hence combination of Uniform Cost Search and Best First Search.
- It uses Heuristic function to search for the best path.
- A* algorithm is similar to UCS except that it uses g(n)+h(n) instead of g(n).
- we can combine both costs as following, and this sum is called as a **fitness number**.
- F(n) = G(n) + H(n) //F(n) is a fitness number
- F(n) – The actual cost path from the **start node to the goal node**.
- G(n) – The actual cost path from the **start node to the current node**.
- H(n) – The actual cost/heuristic path from the **current node to goal node**.

Search Algorithms in AI

A*Search Algorithm:

Step 1 : Add Starting Node in OPEN list.

Step 2 : if OPEN list is empty return FAIL

Step 3 : Select node from OPEN list which has smallest value (g(cost)+h(heuristic))

if node= GOAL Node return Success

Step 4 : Expand node N and generate all successors

Then compute (g+h) for each successor node

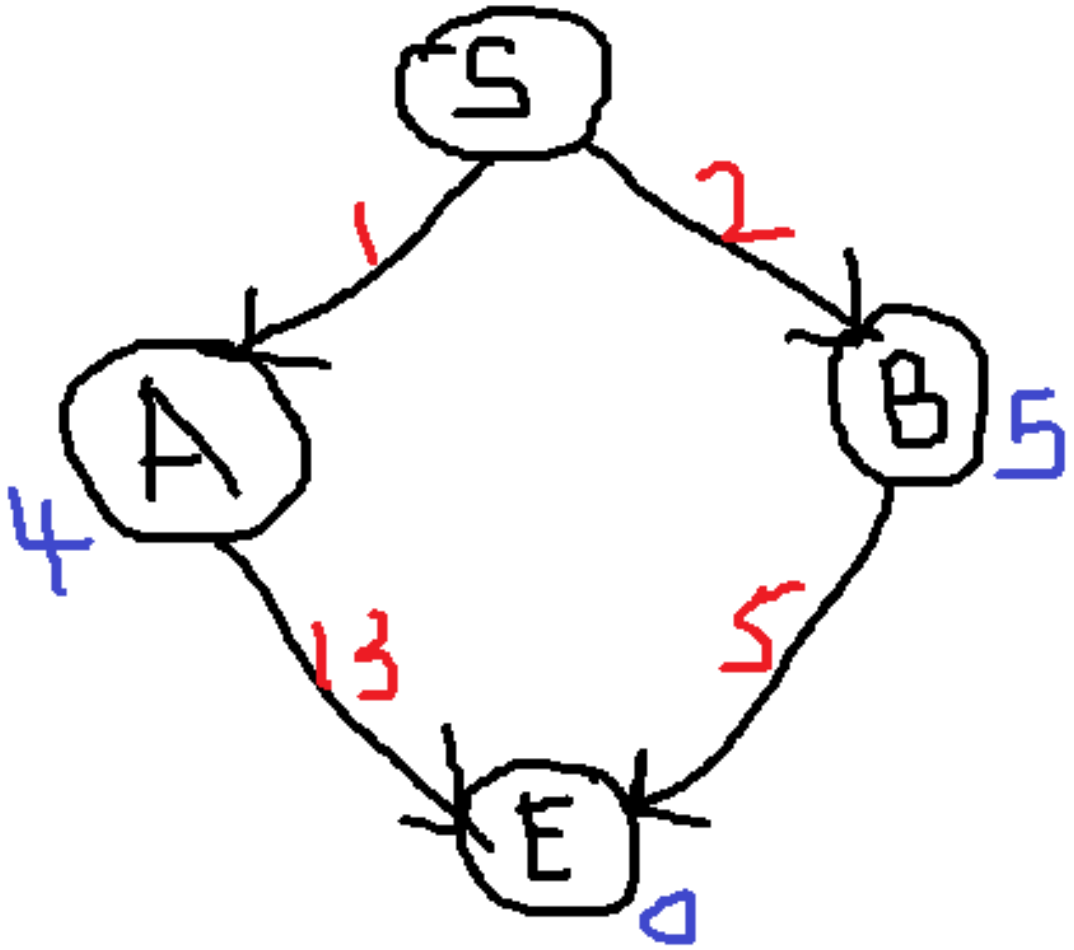
Step 5 : if node N is already in OPEN /CLOSED, attach to backpointer

Step 6 : Goto step 3

Step 7 : Exit

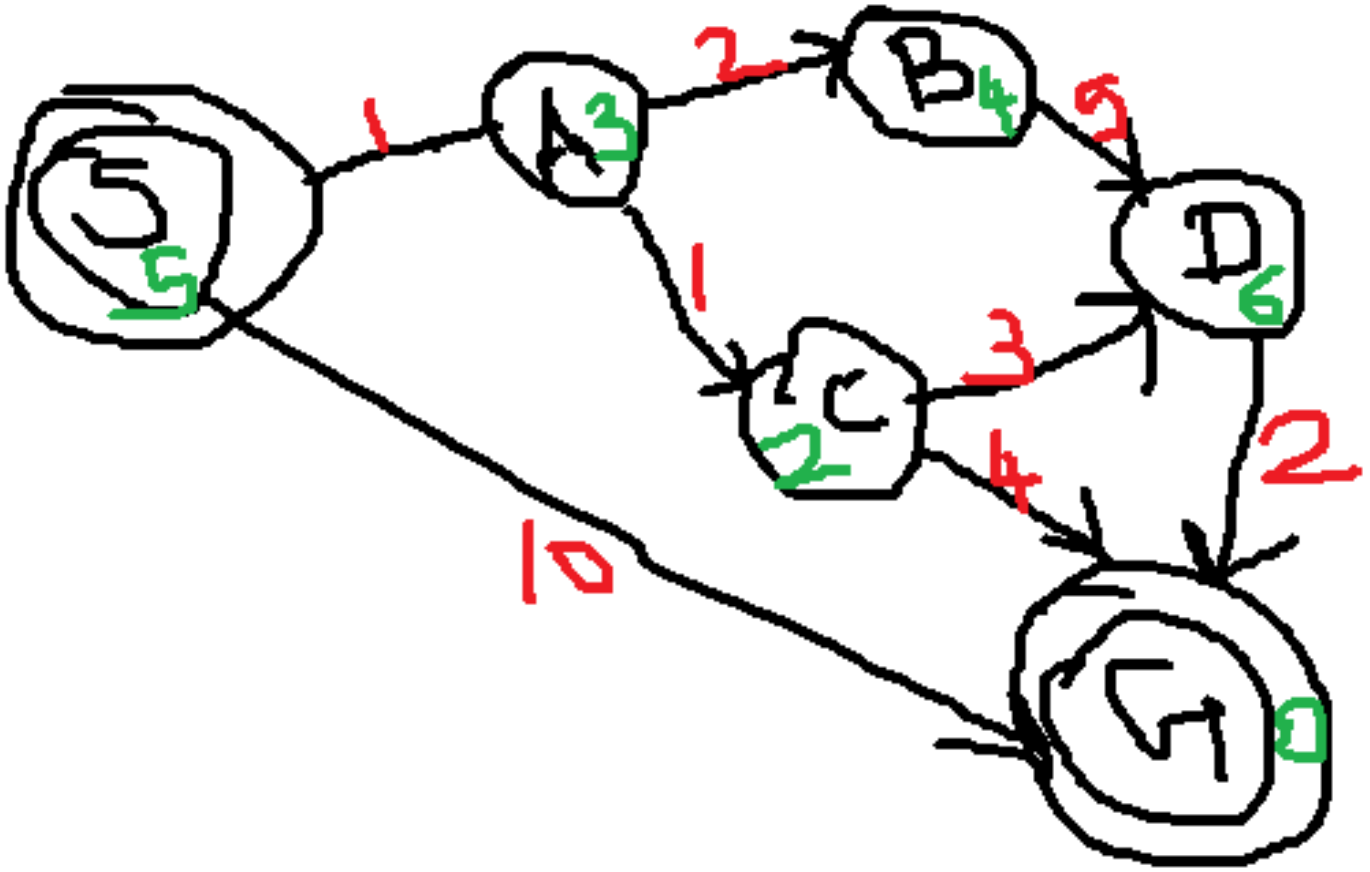
Search Algorithms in AI

A* Search Algorithm:



Search Algorithms in AI

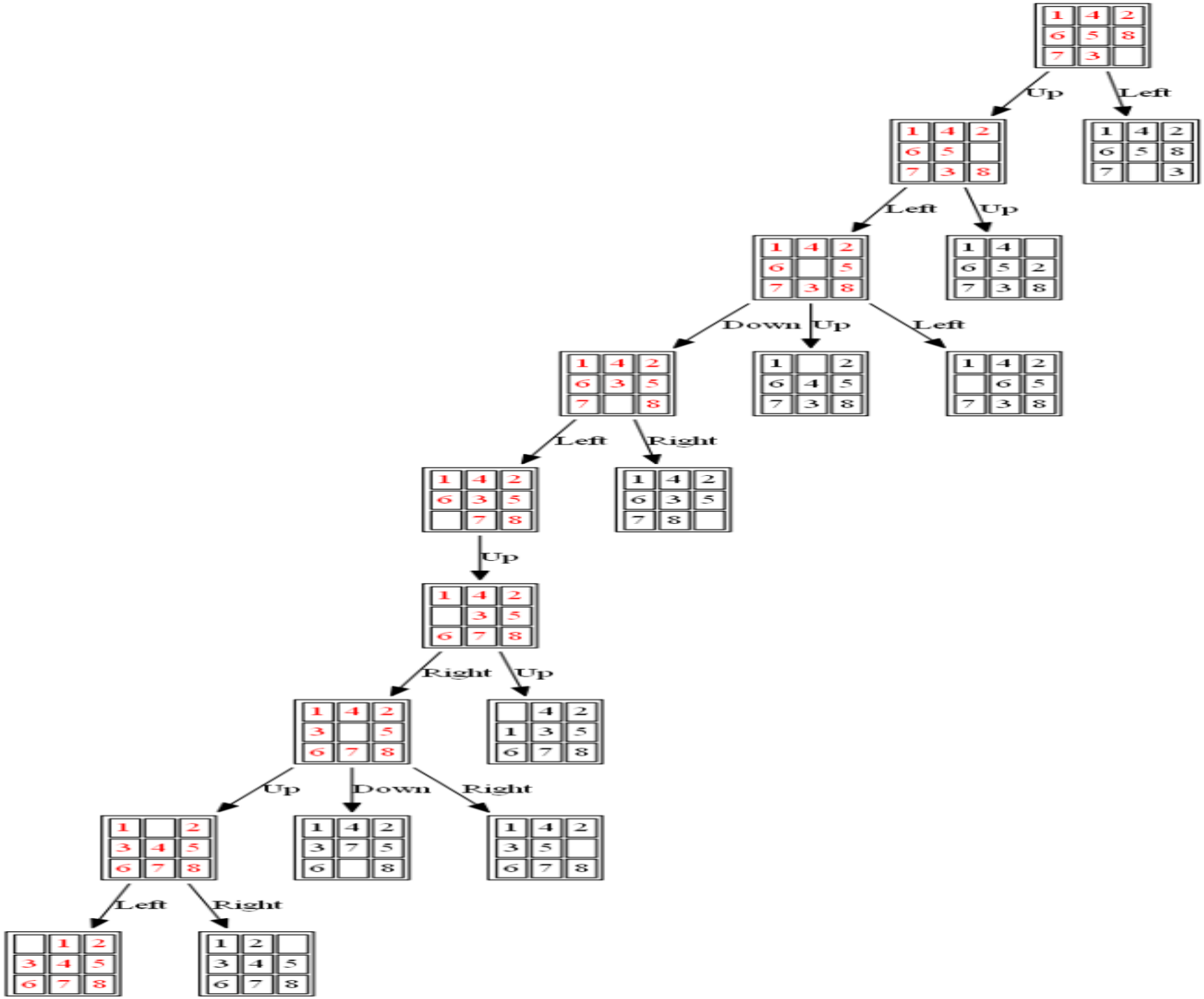
A* Search Algorithm:



Search Algorithms in AI

A*Search Algorithm:

8 Puzzle Problem

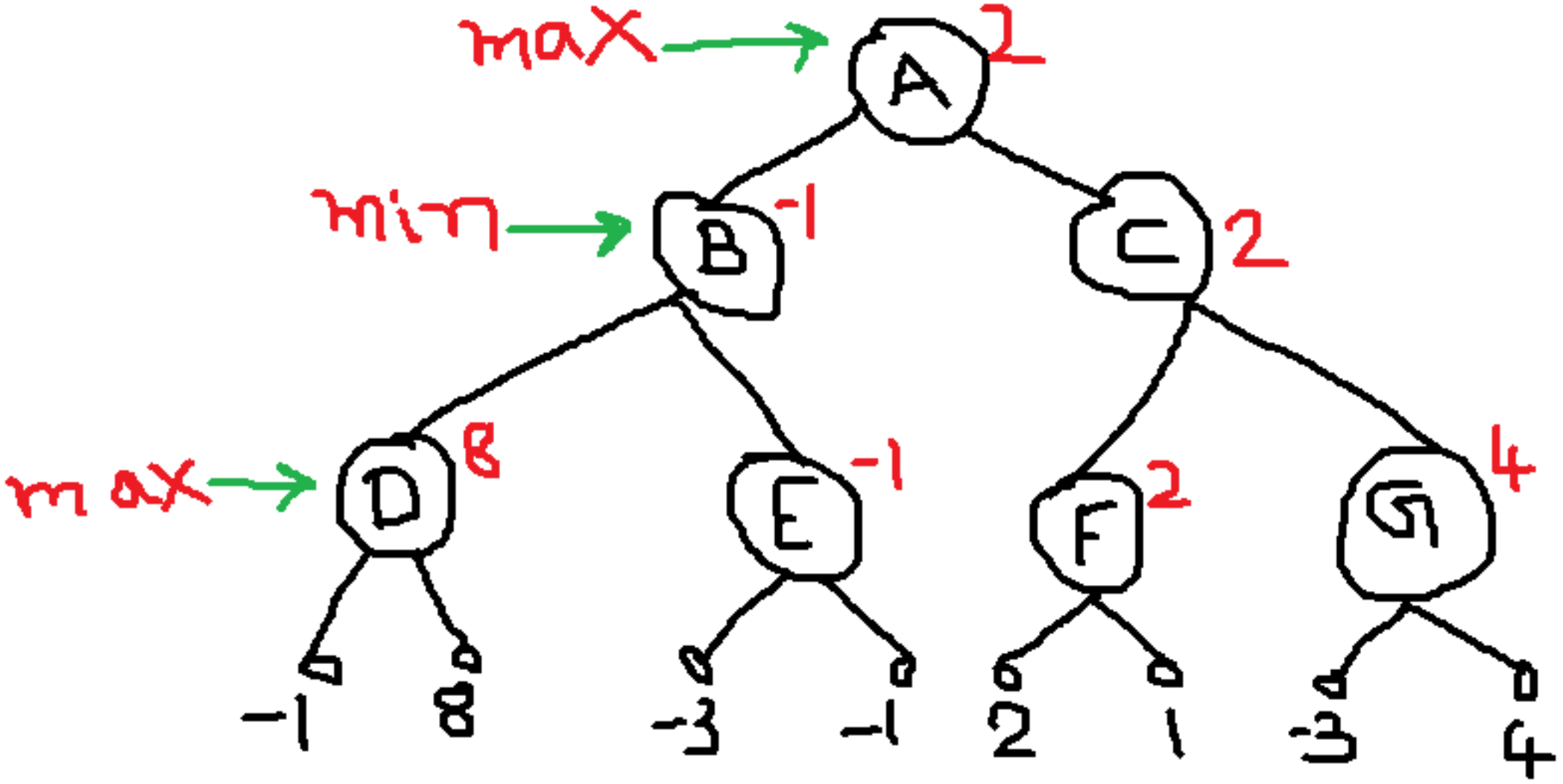


Search Algorithms in AI

Game Search (Mini-Max Algorithm):

- It is a type of Adversarial/Game Search. In adversarial search the problem to be solved is dynamic where one candidate try to plan against the strategy of other. Both (multiagent environment) the agents are working in the same environment finding out the best solution against each other.
- It is a **recursive/backtracking** algorithm.
- It is **complete** hence will find the solution.
- Is optimal if both the opponents are playing optimally.
- Used in decision making and **Game theory**.
- It **uses recursion** to search through the game tree.
- Used **for two player game**. Eg Chess, Tic-Tac-Toe.
- Here two player play the game, wherein **one is Min** and **other is Max**. ie one player tries to maximize the value for win and other tries to minimize the value for the opponent (mostly an AI agent).
- **Max will select the maximum value** and **Min will select the minimum value**.
- It **uses Depth-First-Search** to explore complete **game tree**.

Search Algorithms in AI



Search Algorithms in AI

Game Search (Mini-Max Algorithm):

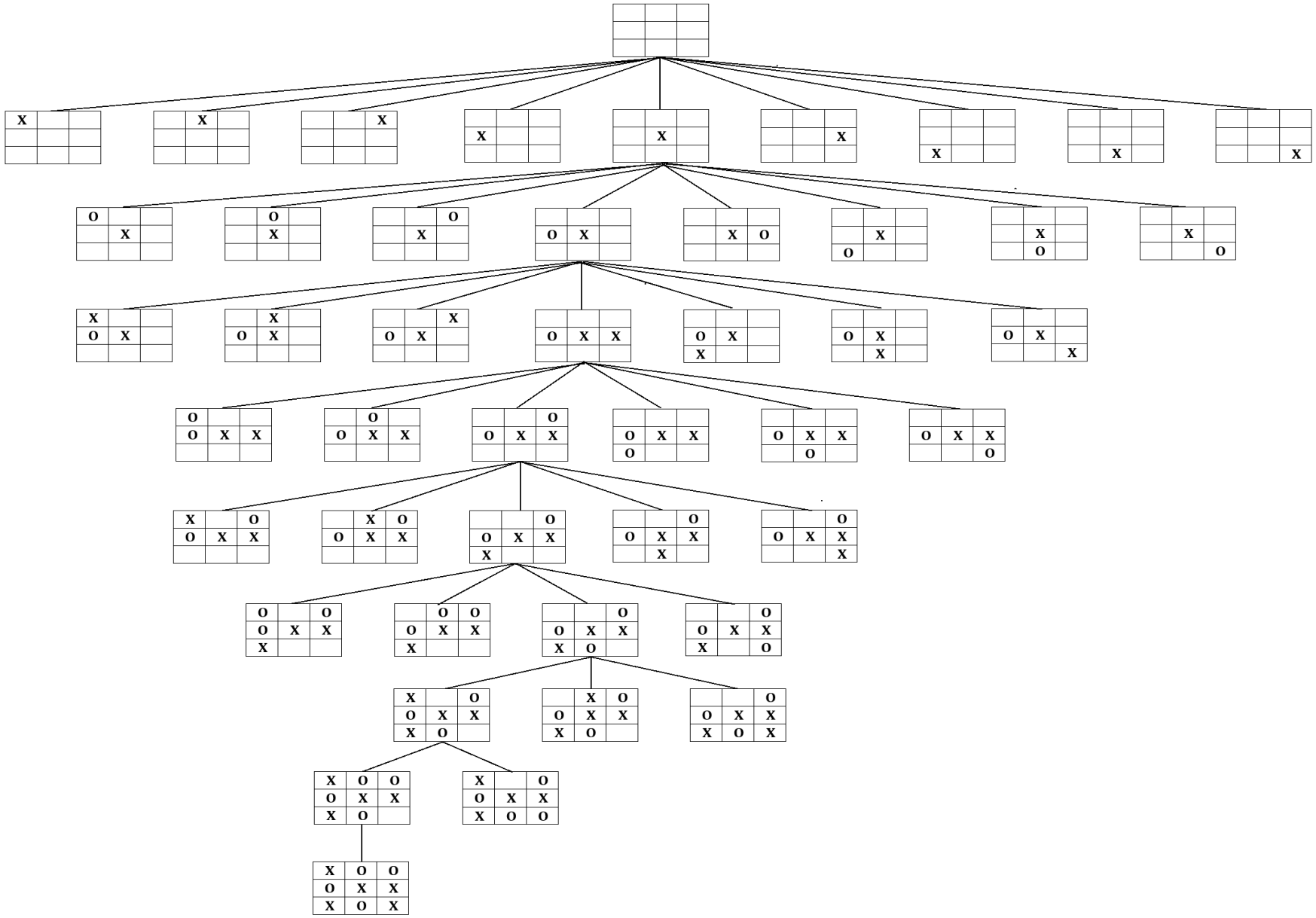
```
function minimax(node, depth, maximizingPlayer) is
   if depth ==0 or node is a terminal node then
       return static value of node

if MaximizingPlayer then      // for Max Player
   maxEva= -infinity
   for each child of node do
       eva= minimax(child, depth-1, false)
        maxEva= max(maxEva,eva)       //gives Maximum of the values
   return maxEva

else                          // for Min player
   minEva= +infinity
   for each child of node do
      eva= minimax(child, depth-1, true)
      minEva= min(minEva, eva)        //gives minimum of the values
   return minEva
```

Search Algorithms in AI

Game Search (Mini-Max Algorithm):

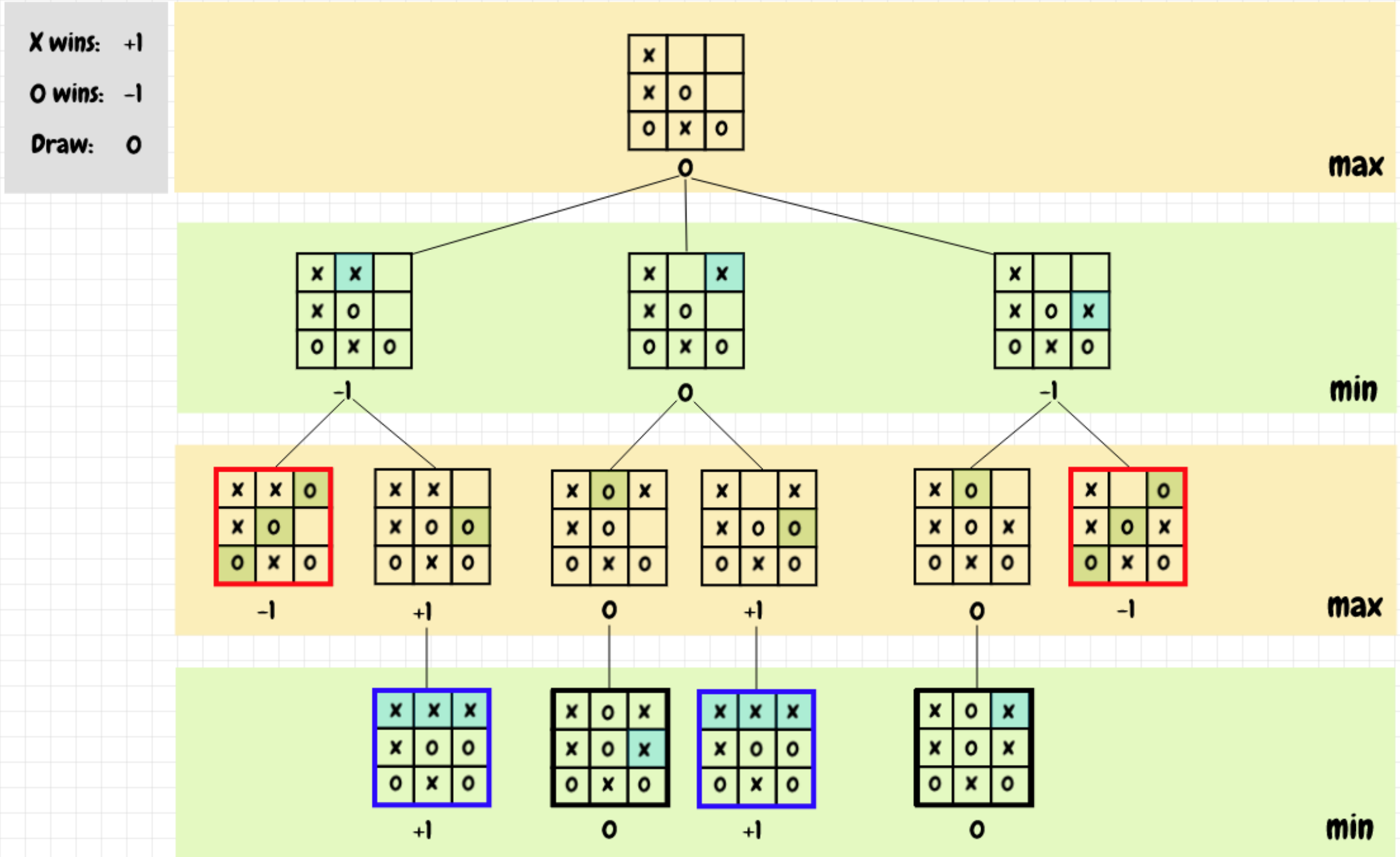
- The Minimax algorithm is Optimal and Complete.
- The time complexity is $O(b^d)$

Search Algorithms in AI



(Winner - X)

Search Algorithms in AI



References

1. Stuart Russel and Peter Norvig, “Artificial Intelligence – A Modern Approach”, 3rd edition, Pearson Education.