25-08-24

T. Vershith
2300030196
Sec-4
CSE

AIML ALM-01

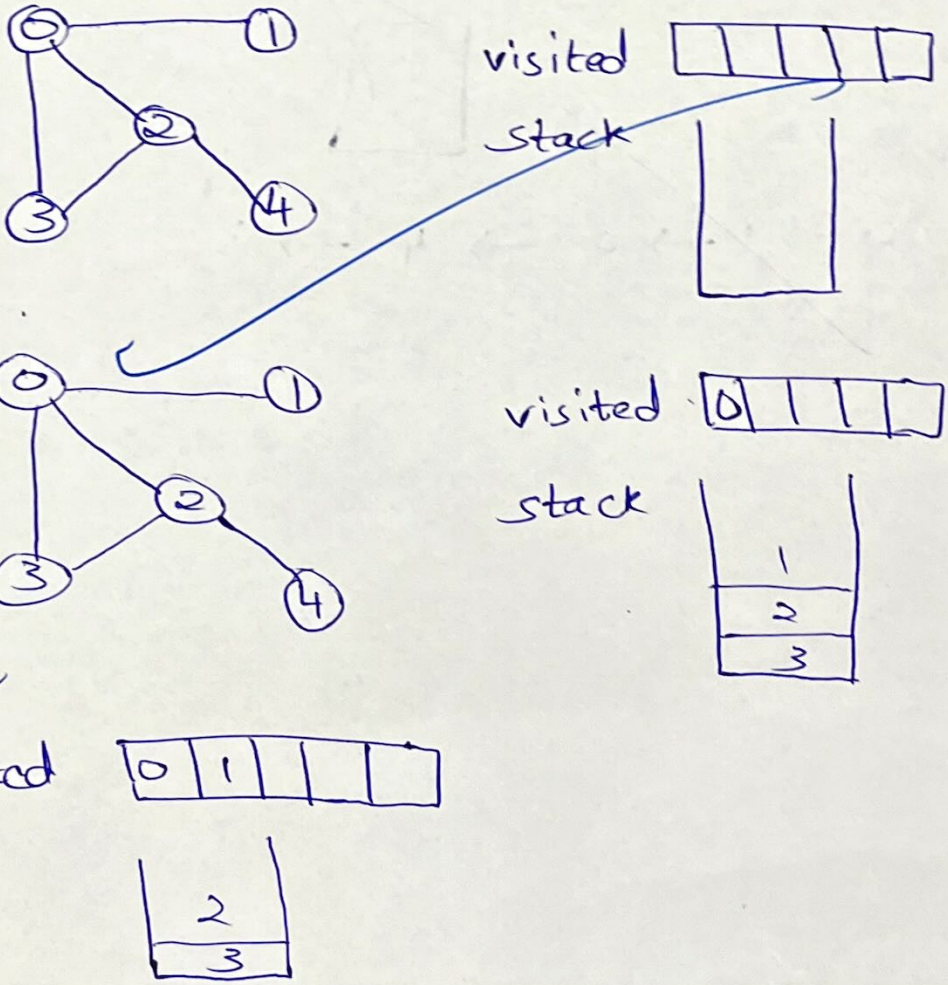
+ Uninformed Search :-

DFS (Depth First Search):-

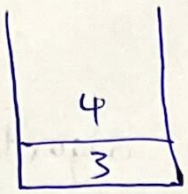
Depth first search is an algorithm used for travelling or searching tree or graph data structures. This algorithm starts at root (or any arbitary node in the case of a graph) & explores as far as possible along each branch before back tracking.

DFS is often implemented using a stack, either explicity or through the system's call stack in a recursive implementation.

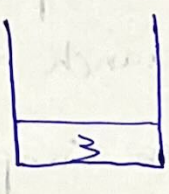
Ex':



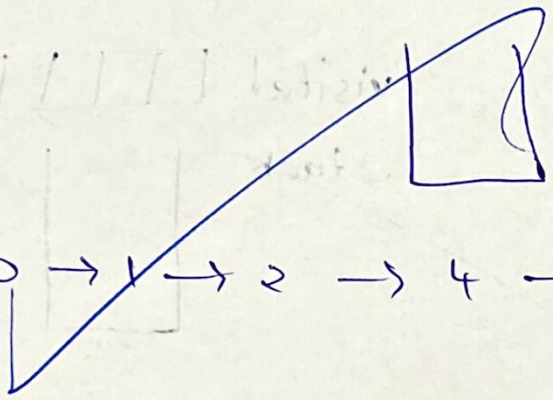
visited | 0 | 1 | 2 | | |



visited | 0 | 1 | 2 | 4 | |



visited | 0 | 1 | 2 | 4 | 3 |

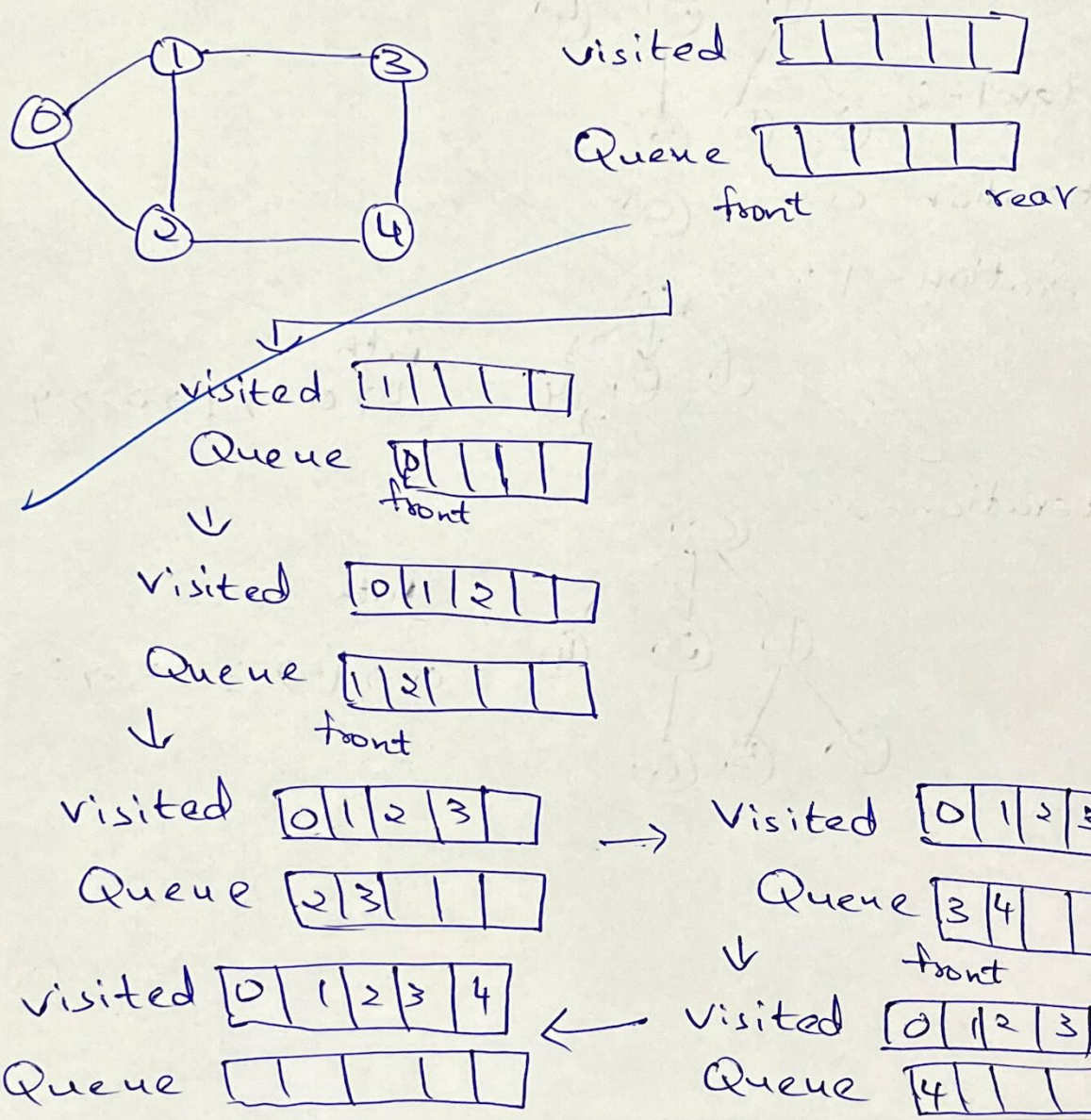


Path = 0 → 1 → 2 → 4 → 3

BFS (Breadth First Search) :-

Breadth first search is an algorithm used for travelling or searching tree or graph data structures. Unlike depth first search, BFS explores the neighbor nodes at the depth level.

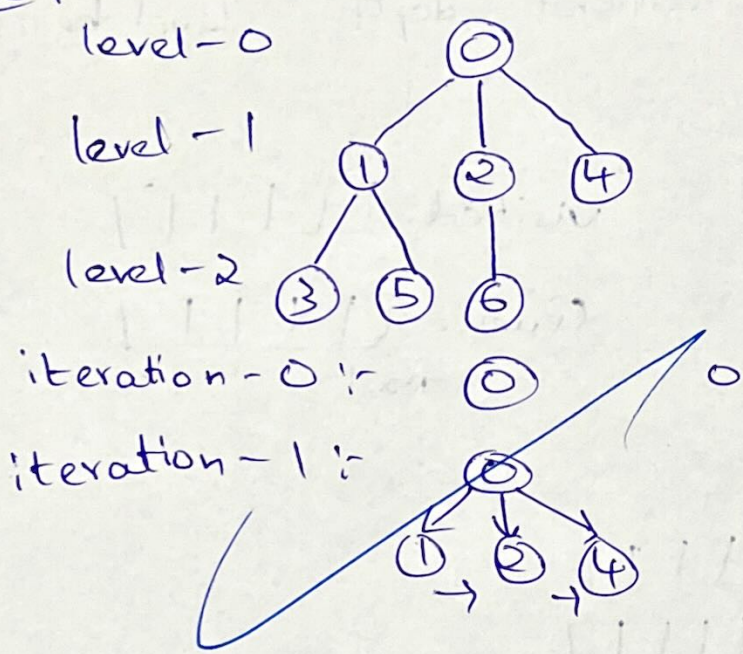
BFS is often used to find the shortest path in an unweighted graph, as it explores all nodes at the current depth level before moving deeper.



IDFS (Iterating deepening first search):

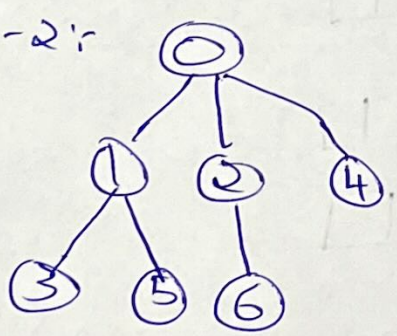
Iterative deepening depth first search is an algorithm that combines both DFS & BFS concept of exploring the search space level by level. I DFS is particularly useful in scenarios where the search space is large & the depth of solution is unknown.

Ex:



Path
↳ 0 → 1 → 2 → 4

iteration-2 :-



Path

0 → 1 → 3 → 5 → 2 → 6

Informed search :-

A* search Algorithm :-

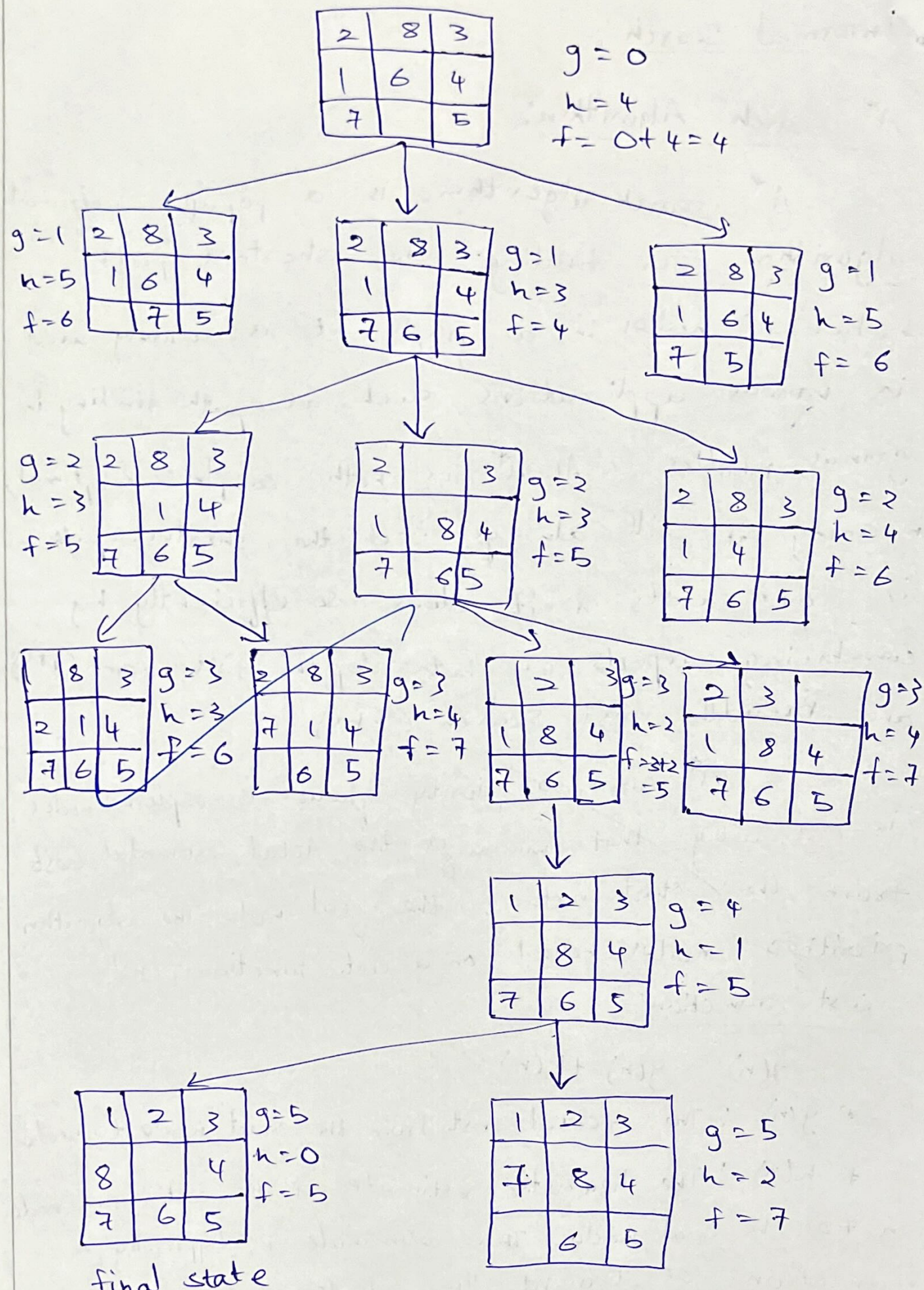
A* search algorithm is a popular & efficient algorithm for finding the shortest path between nodes in a graph. It is widely used in various applications, such as path finding in games, robotics & AI. A* is both complete & optimal, meaning it will always find the shortest path if one exists & it does so efficiently by combining aspects of both depth-first-search (DFS) and Breadth-first-search (BFS).

A* uses a priority queue to explore nodes in a way that minimizes the total estimated cost from the start node to the goal node. The algorithm prioritizes nodes based on a cost function 'f(n)'.

Cost function :-

$$f(n) = g(n) + h(n)$$

* $g(n)$: The actual cost from the start node to node n
* $h(n)$: The heuristic estimate of the cost from node n to the goal node. This eliminate is typically a function of straight-line distance or other domain-specific heuristics.


g = 0
h = 4
f = 0 + 4 = 4
g = 1
h = 5
f = 6
g = 1
h = 3
f = 4
g = 1
h = 5
f = 6
g = 2
h = 3
f = 5
g = 2
h = 3
f = 5
g = 2
h = 4
f = 6
g = 3
h = 3
f = 6
g = 3
h = 4
f = 7
g = 3
h = 2
f = 3+2 = 5
g = 3
h = 4
f = 7
g = 4
h = 1
f = 5
g = 5
h = 0
f = 5
final state
g = 5
h = 2
f = 7

Hill Climbing Algorithm:-

Hill climbing is a simple optimization algorithm used in AI to find the best possible solution for a given problem. It belongs to the family of local search algorithms & is often used in optimization problems where the goal is to find the best solution from a set of possible solutions

In hill climbing, the algorithm starts with an initial solution & then iteratively makes small changes to it in order to improve the solution. These changes are based on a heuristic function that evaluates the quality of the solution. The algorithm continuos to make these small changes until it reaches a local maximum, meaning that no further improvement can be made with the current set of moves.

Local maximum:- It is a state which is better than its neighbouring state however there exists a state which is better than it. This state is better here the value of the objective function is higher than its neighbours.

Global maximum:- It is a state which is better than its neighbouring state however there exists a state which is better than it. This state is better because here the value of the objective function is higher in value.

Plateau/flat local maximum:- It is a flat region of state space where neighbouring states have the same value.

Ridge:- It is a region that is higher than its neighbours but it self have a slope. It is special kind of local maximum.

Shoulder:- It is a plateau that has a uphill edge

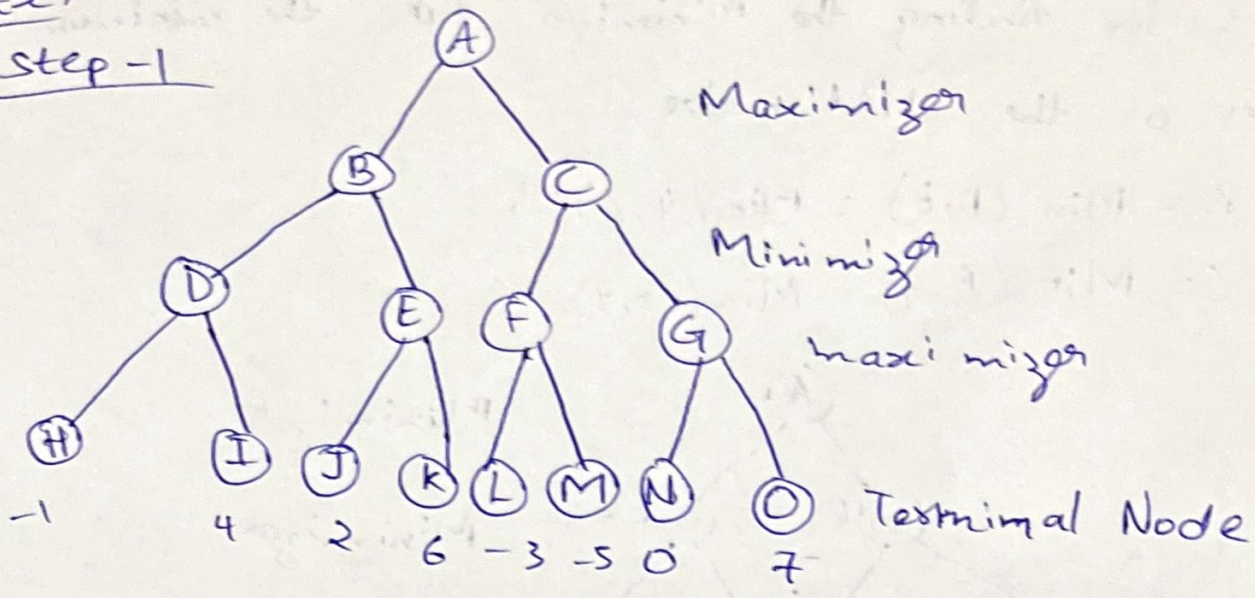
Min Max Algorithm:-

Min max algorithm is a recursive or back tracking algorithm which is used in decision making and game theory. It provides an optimal move for the player assuming that the opponent is also playing optimally.

MinMax algorithms uses recursion to search through the game tree. It is mostly used for game playing in AI. In this algorithm two players play the game one is Max & other is Min. Both players of the game are opponent with each other where Max will select the maximized value and Min will select the minimized value.

Ex:-

Step-1



Step-2

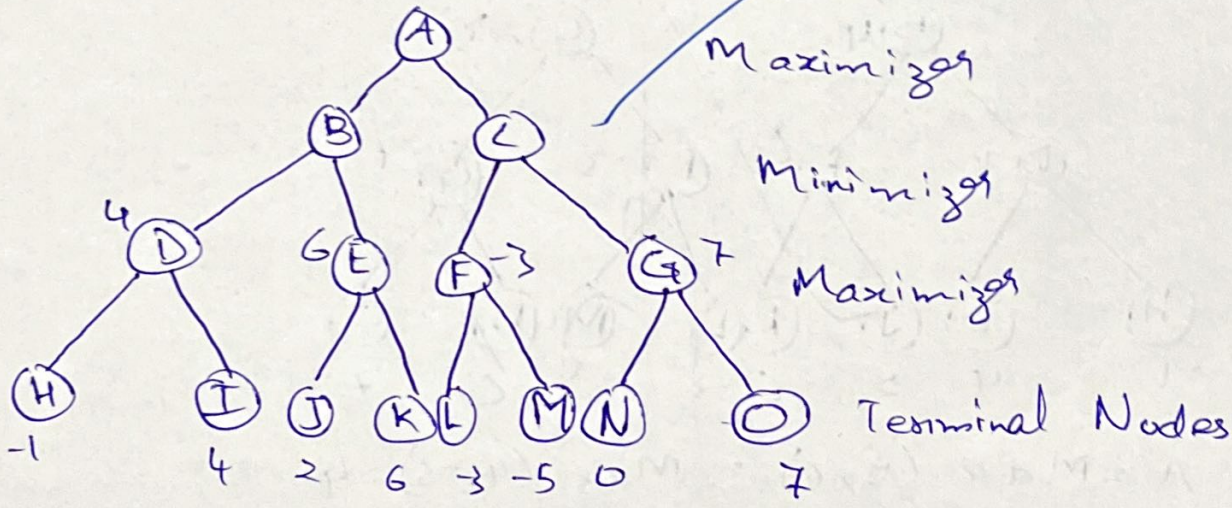
for the maximizes find the maximum values from the child nodes & fix it.

D = max (H, I) = max (-1, 4) = 4

E = max (J, K) = max (2, 6) = 6

F = max (L, M) = max (-3, -5) = -3

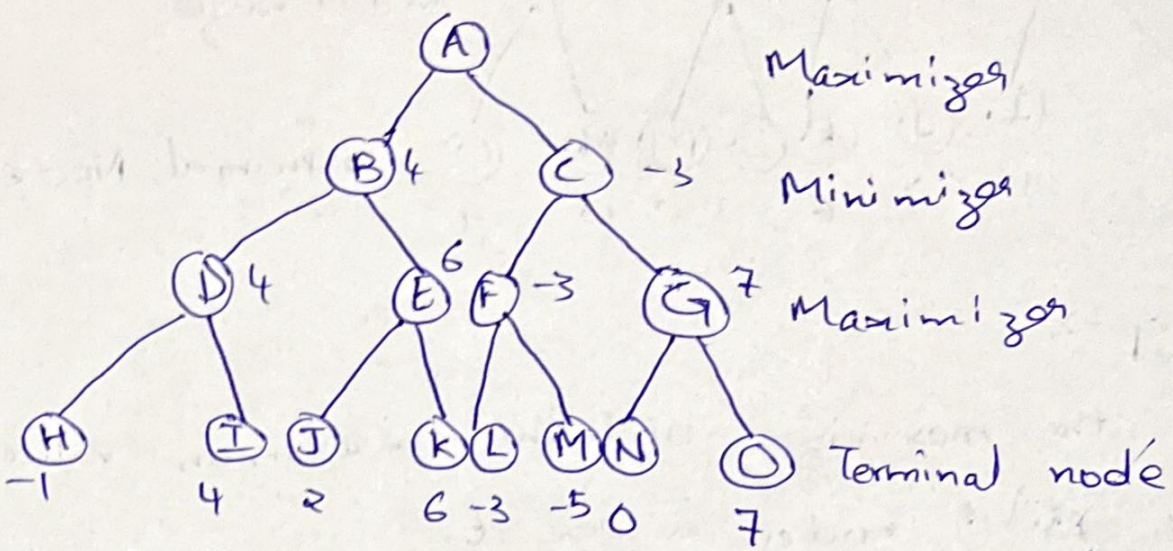
G = max (N, O) = max (0, 7) = 7



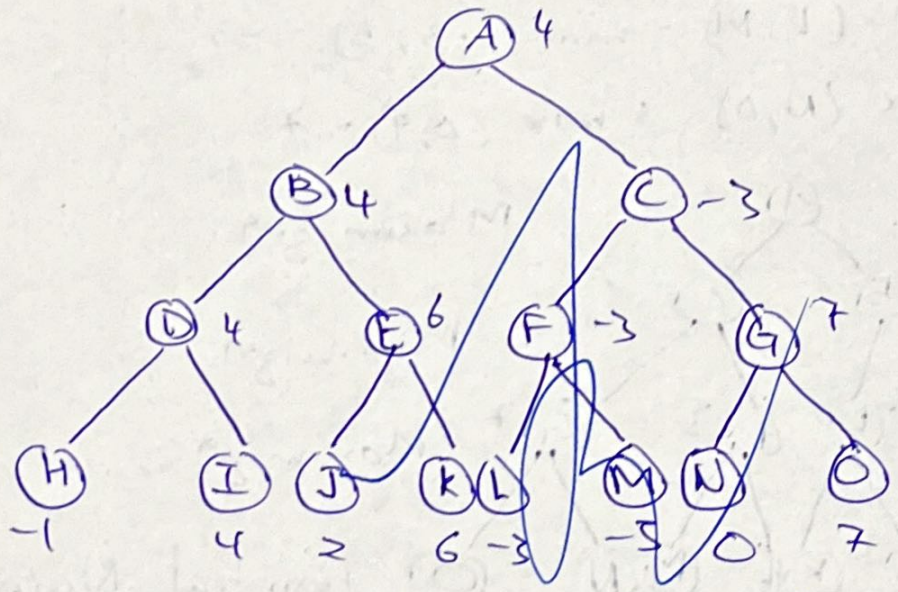
Step-3 for finding the Minimizer take the minimum values of the child nodes

$B = Min(D, E) = Min(4, 6) = 4$

$C = Min(F, G) = Min(-3, 7) = -3$



Step-4 for finding the maximizer take the maximum values from the child node.



$A = Max(B, C) = Max(4, -3) = 4$