Module - 2

Search Algorithms in AI

In AI, Search techniques are universal problem-solving methods. Rational agents or problem-solving agents in AI mostly used these Search Strategies or algorithms to solve a specific problem and provide the best result. Problem-solving agents are the goal based agents and use atomic representation

Search Algorithms Terminologies.

1. Search:

Searching is a step by step procedure to solve a search problem in a given Search space. A Search problem can have 3 main factors.

1. Search space

Search space represents a set of possible Solutions, which a System may have.

2. Start State

It is a State from where agent begins the Search

3. Goal Test

It is a function which observe the current State and Returns whether the goal State is achieved or not.

2. Search Tree

A tree representation of search problem is called search tree. The root of the search tree is the root node which is corresponding to the initial state.

3. Actions

It gives the description of all the available action to the agent.

4. Transition model

A description of what each action do, can be represented as transition model.

5. path cost

It is a function which assigns a numeric cost to each path.

6. Solution

It is an action sequence which leads from the start node to the goal node.

7. optimal solution

If a solution has the lowest cost among all solutions.

properties of search algorithms

The following are 4 essential properties of search algorithms to compare the efficiency of algorithms

1. Completeness

A search algorithm is said to be complete if it guarantees to return solution of atleast any solution exists for any random input.

2. optimality

If a solution found for an algorithm is guaranteed to be the best solution among all other solutions.

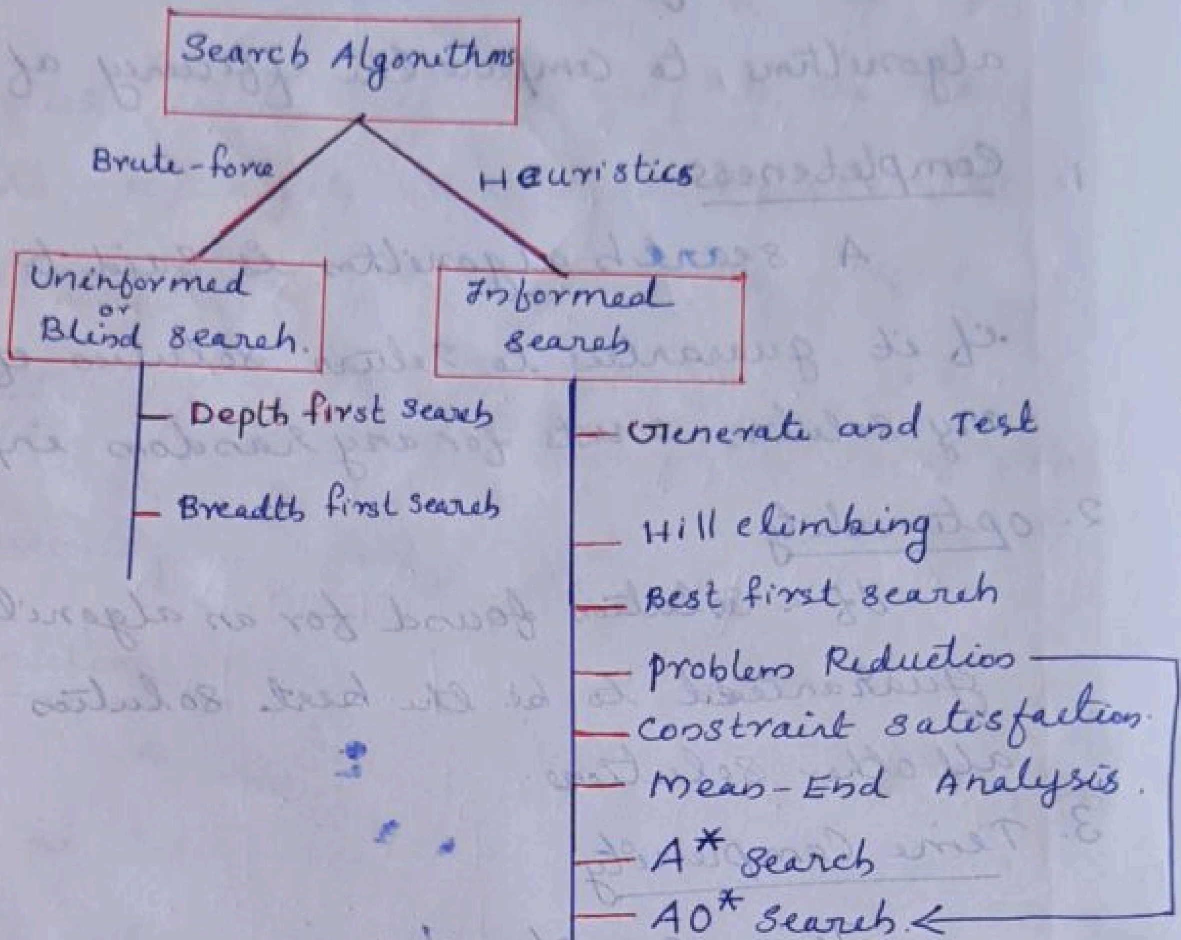
3. Time Complexity

Time complexity is the measure of time for an algorithm to complete its task.

4. Space Complexity

It is the maximum storage space required at any point during search, as the complexity of the problem.

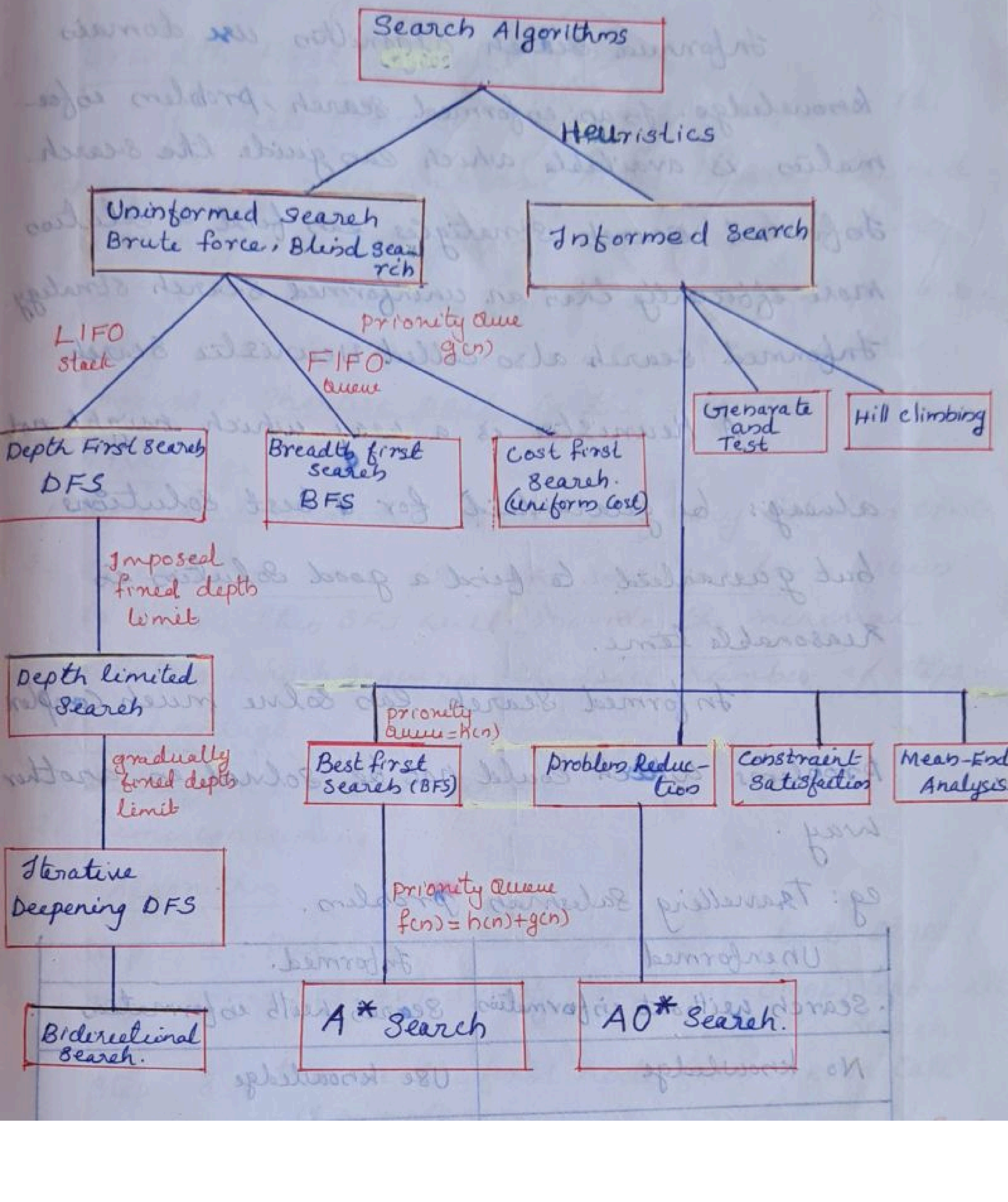
Types of Search Algorithms



1. Uninformed Search

The uninformed Search does not contain any domain knowledge such as closeness, the location of the goal. It operates in a brute-force way as it only includes information about how to traverse the tree & how to identify the leafs and goal nodes. Uninformed Search applies a way in which Search tree is Searched without any information about the search space like initial state operators and test for the goal, So it is also called blind Search. It examines each node of the tree untill it achieves the goal node.

Hierarchical Representation of Search algorithms



Informed Search

Informed Search algorithms use domain knowledge. In an informed search, problem information is available which can guide the search. Informed search strategies can find a solution more efficiently than an uninformed search strategy. Informed search also called Heuristic Search.

A Heuristic is a way which might not always be guaranteed for best solutions but guaranteed to find a good solution in reasonable time.

Informed Search can solve much complex problems which could not be solved in another way.

eg: Travelling Salesman problem.

Imp

Uninformed	Informed.
1. Search without information	Search with information
No knowledge	Use knowledge
Time consuming	Quick solution
More Complexity (Time, space)	Less Complexity (Time, space)
DFS, BFS etc	A*, AO*, Best First Search
Efficiency mediatory	High efficient
high cost	Low cost

Uninformed Search algorithms

1. Breadth First Search (BFS)

It starts from the root node, explores the neighbouring nodes first and moves towards the next level neighbours. It generate one tree at a time untill the solution is found. It can be implemented using FIFO Queue data structure. This method provides shortest path to the solution.

Advantage

1. BFS will provide a solution if any solution exists.
2. If there are more than one solution for a given problem, then BFS will provide the minimal solution which requires the least number of steps.

Disadvantage

1. It requires lots of memory
2. Time consuming

Algorithm

Step 1 : Put the initial node on a list START

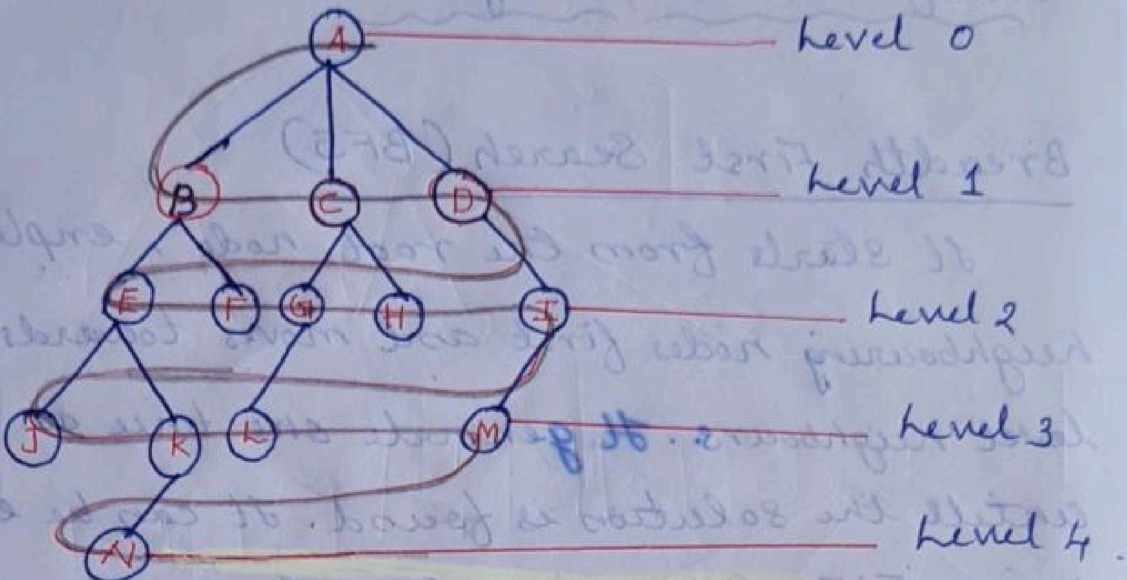
Step 2 : If (START is empty) or (START = GOAL) terminate search.

Step 3 : Remove the first node from START. Call this node a

Step 4 : If (a = Goal) terminate with success.

Step 5 : Else if node a has successors, generate all of them and add them at the tail of START

Step 6 : Goto Step 2



Queue

← ~~A~~ ←

~~B~~ C D

~~C~~ D E F

~~D~~ E F G H

~~E~~ F G H I

~~F~~ G H I J K

~~G~~ H I J K

~~H~~ I J K L

~~I~~ J K L

~~J~~ K L M

~~K~~ L M

~~L~~ M N

M N

~~N~~

branching factor → no of childrens.

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

Time complexity = $O(b^d)$, where 'b' is branch factor

space complexity = $O(b^d)$ 'd' is depth.

$d = 4$

$b = 4$

$b^d = 16$

imp

* Optimal algorithm * complete

* Shallowest node.

2. Depth First Search

It is implemented in Recursion with LIFO Stack data Structure. It creates the same set of nodes as BFS, only in the different manner.

As the nodes on the single path are stored in each iteration from root to leaf, the space requirement to store node is linear. With branching factor b and depth m, the storage space is bm

Advantage

1. Require very less memory.
2. Less time to reach the goal node

Disadvantage

1. No guarantee of finding solution
2. DFS goes for deep down searching of some time it may go to the infinite loop.

Algorithm

Step 1 : put the initial node on a list START

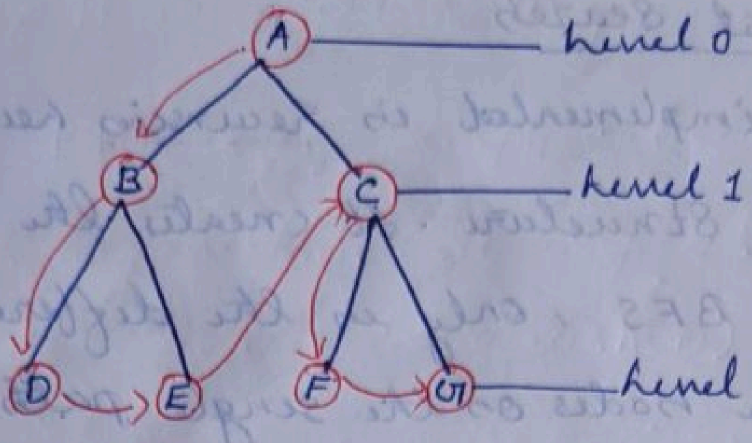
Step 2 : If (START is empty) or (START=GOAL) terminate Search

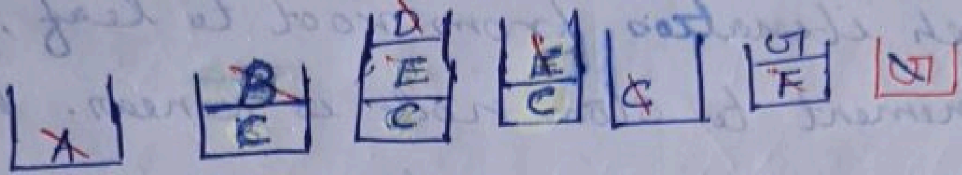
Step 3 : Remove the first node from START. Call this node a.

Step 4 : If (a=GOAL) terminate with success.

Step 5 : Else if node a has successors, generate all of them and add them at the beginning of START

Step 6 : Goto step 2.





A B D E C F G

Time complexity = $O(b^d)$ = $b=3$, $d=3=9$.

Space complexity = $O(b^d)$ = 9

* Deepest node.
* Incomplete
* Not optimal.

=> Its complexity depends on the numbers of path.
It cannot check duplicate node.

Informed Search Algorithms 16/10/2020

The informed search algorithm is more useful for large search space. Informed search algorithm uses the idea of Heuristic, so it is also called Heuristic Search.

Heuristic function

Heuristic is a function which is used in informed search, and it find the most promising path. It takes the current state of the agent as its input and produces the estimation of how close agent from the goal. The Heuristic method, however, might not always give the best solution, but it guaranteed to find a good solution in reasonable time. Heuristic function estimates how close a state is to the goal. It is represented by h(n) and it calculates the cost of an optimal path b/w the pair of states. The value of Heuristic function always positive.

h(n) $\boxed{h(n) <= h*(n)}$

Here h(n) is heuristic cost and h*(n) is the estimated cost. Hence Heuristic cost should be or less than or equal to the estimated cost

Pure Heuristic Search

PHS is the simplest form of Heuristic Search algorithms. It expands nodes based on their Heuristic value $h(n)$. It maintains two lists, OPEN and CLOSED list. In the CLOSED list, it places those nodes which have already expanded and in the OPEN list, it places nodes which have yet not been expanded.

On each iteration, each node [illegible] n with the lowest Heuristic value is expanded & generates all its successors and n is placed to the closed list. The algorithm continues until a goal state is found.

Heuristic in AI (Rule of thumb) (what, why, How)

⇒ It is a technique designed to solve a problem quickly.

1. Hill climbing

Hill climbing Algorithm is a local search algorithm which continuously moves in the direction of increasing elevation / value to find the peak of the mountain or best solution to the problem. It terminates when it reaches a peak value where no neighbour has a higher value.

Hill climbing algm is a technique which is used for optimizing the mathematical problem. One of widely used eg is Travelling Salesman problem, which we need to minimize the distance travelled by the salesman.

=> It is also called greedy local search.

=> HC is mostly used when good Heuristic is available

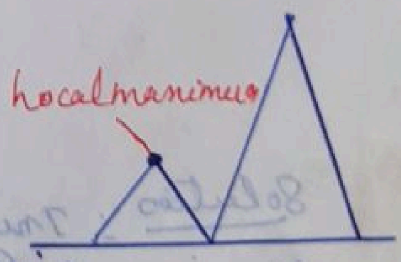
=> In this algroithm, we don't need to maintain & handle the search tree or graph as it only keeps a single current state.

Features of HC

1. Generate & Test variant
2. Greedy approach.
3. No back tracking

problem in HC Algorithm.

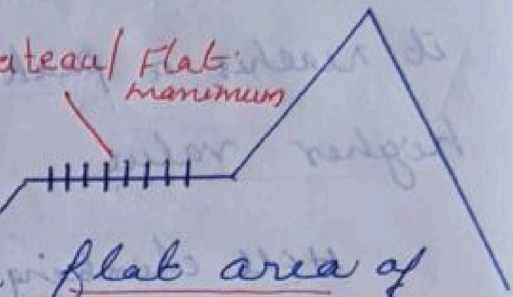
1. Local maximum.



A Local maximum is a peak state in the landscape which is better than each of its neighbouring states but, there is another state also present which is higher than local maximum.

Solution :- Back tracking technique can be a solution of the local maximum in state space landscape. Create a list of the promising path so that the algorithm can back track the search space and explore other path as well.

2. Plateau:

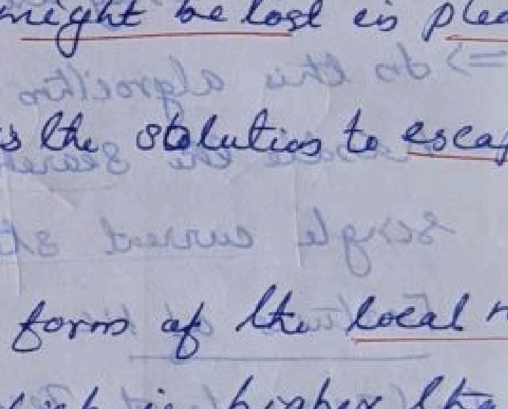


A plateau is the flat area of the search space in which all the neighbour search space in which all the neighbour states of the current state contains the same value, because of this algorithm does not find any best direction to move. A HC search might be lost in plateau.

Solution:- A big jump is the solution to escape from plateau.

3. Ridges

A ridge is a special form of the local maximum. It has an area which is higher than its surrounding area, but itself has a slope, and cannot be reached in a single move.



Solution: Trying different paths at the same time is the solution for circumventing ridges.

Algorithm

Step 1 : Put the initial node on a list START

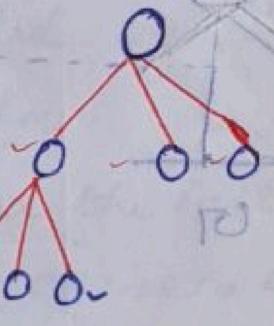
Step 2 : If (START = empty) or (START = GOAL) terminate Search.

Step 3 : Remove the first node from START. Call this node a.

Step 4 : If (a = GOAL) terminate search with success.

Step 5 : Else if node a has successors, generate all of them. Found out how far they are from the goal node. Sort them by the remaining distance from the goal and add them to the beginning of START

Step 6 : Goto Step 2.

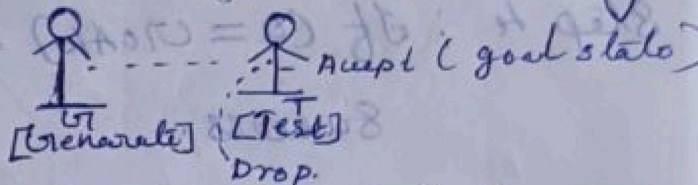


2. Generate and Test Algorithm 16/10/2020

Generate and Test algorithm is very simple algorithm that guarantees to find a Solution if done systematically and their exist a solution.

=> It is a Heuristic Techniques.
=> Depth First search with backtracking

Algorithm



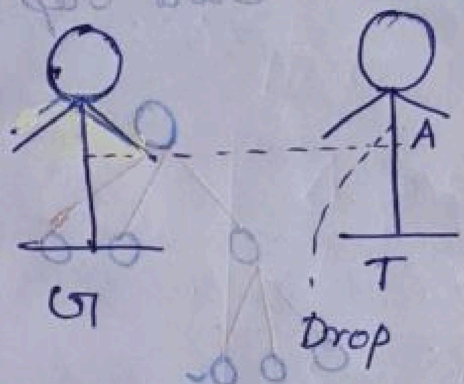
Step 1 : Generate a possible solution

Step 2 : Test to if this is a actual solution

Step 3 : If a solution is found, quit; otherwise Go to Step 1.

Properties of Good Generator

1. Complete
2. NonRedudant
3. Informed



3. Best First Search Algorithm (Greedy Search)

This algorithm always selects the path which appears best at that moment. It is the combination of DFS & BFS. It uses the heuristic function and search. Best first search allows us to take the advantages of both DFS & BFS. With the help of best-first search, at each step, we can choose the most promising node. In the best first search, we expand the node, which is closest to the goal node and the closest cost is estimated by Heuristic function, i.e.

$f(n) = g(n)$, where $h(n)$ = estimated cost from node n to the goal.

The greedy best first search algorithm is implemented by the priority queue.

Algorithm

Step 1 : put the initial node on a list START

Step 2 : If (START = empty) or (START = Goal) terminate search

Step 3 : Remove the 1st node from START, call this node a.

Step 4 : If (a = GOAL) terminate search with success.

Step 5 : Else if node a, has successors, generate all of them. Find out how far they are from the goal node. Sort all the children generated so far by the remaining distance from the goal.

Step 6 : Name this list as START 1

Step 7 : Replace START with START 1

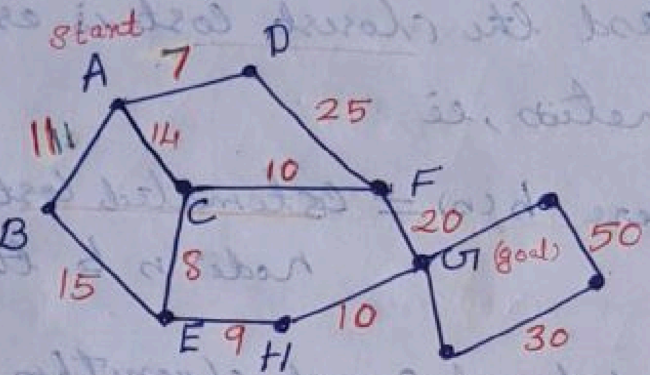
Step 8 : Go to step 2.

Advantages

1. BFS can switch b/w BFS & DFS, by gaining advantages of both the algorithms
2. This algorithm is more efficient than BFS & DFS

Disadvantage

1. It can behave as an unguided DFS in the worst case scenario.
2. It can get stuck in a loop as DFS.
3. Not optimal.



(Straight line distance)

Heuristic value

$A \to G = 40$

$B \to G = 32$

$C \to G = 25$

$D \to G = 35$

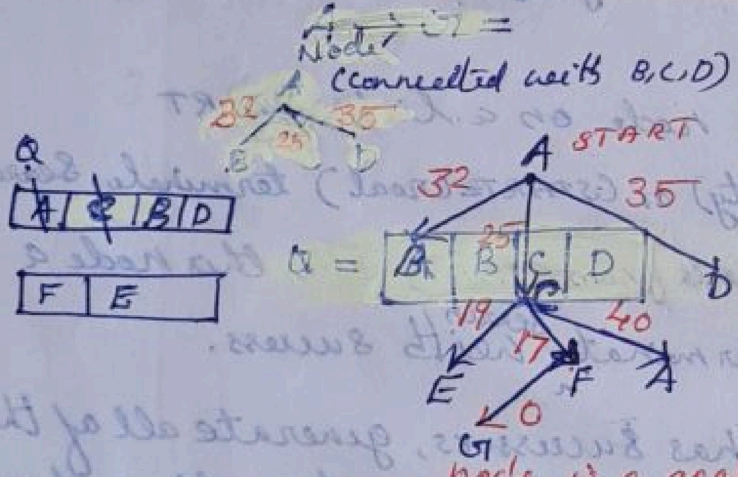
$E \to G = 19$

$F \to G = 17$

$H \to G = 10$

$G \to G = 0$

$A \to G =$



Final path $A \to C \to F \to G$

Space complexity $= O(b^m)$

$\therefore$ Time $\quad$ " $\quad = O(b^d)$

4 A* Algorithm

A* Search is the mostly common known form of Best first search. It uses heuristic function h(n), and cost to reach the node n from the start stage g(n). It has combined features of Uninformed cost search (UCS) & greedy best-first search, by which it solve problem efficiently. A* algorithm finds the shortest path through the search space using the heuristic function. This search algorithm expands less search tree & provides optimal result faster.

$$\boxed{f(n) = g(n) + h(n)}$$

Evaluation function + cost fn = fitness No

f(n): Estimated cost of the cheapest solution

g(n): cost to reach node n from start state

h(n): cost to reach from node n to goal node.

Advantages

1. A* alogorithm is the best algorithm other than other search algorithm.
2. A* algorithm is optimal & complete.
3. Solve very complex problem.

Disadvantages

1. Does not always produce shortest path as it mostly based on Heuristic & approximation.
2. A* has some complexity issues.
3. Memory requirements.

Algorithm

Step 1 : Put the initial node on a list START

Step 2 : If (START is empty) or (START = Goal) Terminate Search.

Step 3 : Remove the first node from START. Call this node a.

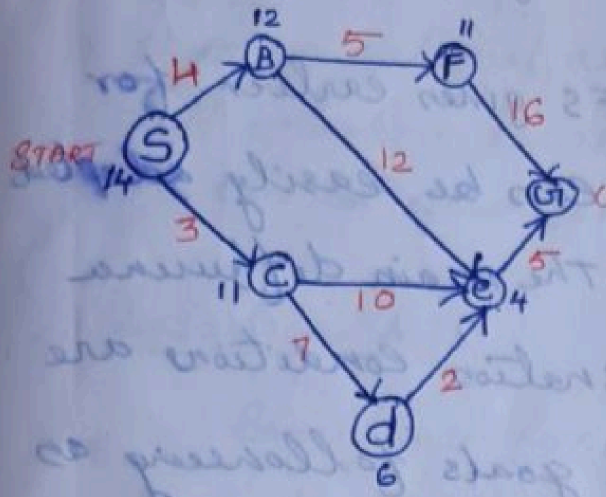
Step 4 : If (a = Goal) terminate search with success.

Step 5 : Else if node a has successors, generate all of them. Estimate the fitness number of the successors by totaling the evaluation function value and the cost function value. Sort the list by fitness number.

Step 6 : Name the new list as START 1.

Step 7 : Replace START with START 1.

Step 8 : Go to Step 2.



State	h(n)
S	14
B	12
C	11
D	6
e	4
F	11
G	0

(1) $f(S) = 0 + 14 = 14$

$f(S) = 14$

(2) $S \to B \quad S \to C$ }

$f(n)\ 4 + 12 = 16$

$S \to B = 16$

$S \to C = 3 + 11 = 14$

$\boxed{S \to B = 16 \quad S \to C = 14}$

(3) combine S & C

$SC \to e \quad , \quad S \to C \quad . \quad S \to e$

$S \to C \to e \implies 14 \qquad 3 + 10 + 4 = 17$

$\boxed{S \to C \to e = 17}$

$SC \to d \qquad S \to C \qquad S \to d$

(4) $S \to C \to d. \qquad 14 \qquad 3 + 7 + 6 = 16.$

$\boxed{S \to C \to d = 16 \quad S \to d = 16}$

(5) $S \to B \to F = 4 + 5 + 11 = 20$ $\boxed{S \to B \to F = 20}$

(6) $S \to B \to e = 4 + 12 + 4 = 20$ $\boxed{S \to B \to e = 20}$

(7) $S \to C \to d \to e = 3 + 7 + 2 + 4 = 16$ $\boxed{S \to C \to d \to e = 16}$

$TC = O(b^d)$

$SC = O(b^d)$

(8) $S \to C \to d \to e \to G = 3 + 7 + 2 + 5 + 0 = 17$ $\boxed{S \to C \to d \to e \to G = 17}$

(9) $S \to B \to F \to G = 4 + 5 + 16 + 0 = 25$ $\boxed{S \to B \to F \to G = 25}$

$\therefore$ Shortest path $= \underline{S \to C \to d \to e \to G = 17}$

5

AO* (AND-OR graph)

The DFS and BFS given earlier for OR trees or graphs can be easily adopted by AND-OR graph. The main difference lies in the way termination conditions are determined, since all goals following an AND nodes must be realized, whereas single goal node following an OR node will do. So for this purpose we are using AO* algorithm. Like ~~HF~~ A* algorithm here we will use two arrays & one HF.

Algorithm

Step 1 : Place the starting node into OPEN

Step 2 : Compute the most promising solution tree say ~~To~~ TO.

Step 3 : Select a node n that is both on OPEN and a member of TO. Remove it from OPEN and place it in CLOSE.

Step 4 : If n is the terminal goal node then levelled n a solved & levelled all the ancestors of n as solved. If the starting node is marked as solved then success and exit.

Step 5 : If n is not a solvable node, then mark n as unsolvable. If starting node is marked as unsolvable, then return failure and exit.

Step 6 : Expand n, find all its successors & find their heu. value, push them into OPEN

Step 7 : Return to Step 2.

Step 8 : Exit

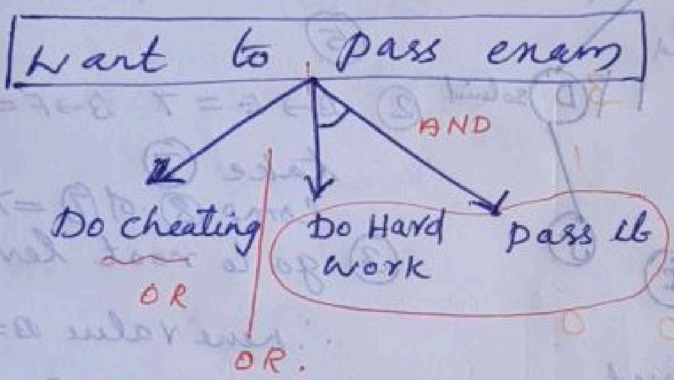
Advantage

1. It is an optimal algorithm.
   If traverse according to the ordering of nodes, It can be used for both OR and AND graph.

2. Disadvantage

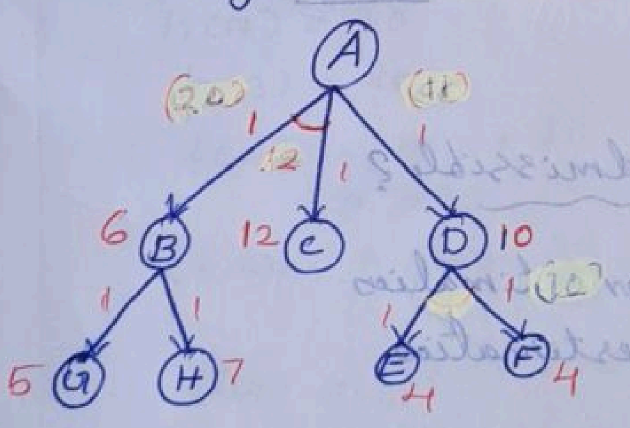
Sometimes for unsolvable nodes, it can't find the optimal path. It complexity is than other algorithms.

AO^* — problem decomposition.



$\Rightarrow$ AO^* does not explore all the solution path once it got a solution

A cost is 1



$A \to B \to C = 6 + 12 + \overset{1+1}{2} = 20$

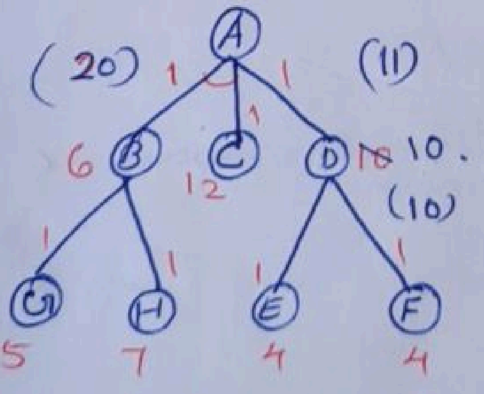
$A \to D = 10 + 1 = 11$

Compare $(20, 11)$ less is $A \to D$

So we select D

$D \to E$, $D \to F$

$D \to E \to F = 4 + 4 + 2 = 10$



ie $A \to D \to E$ ~~...~~ $= 5$

$A \to D \to F = \underline{\underline{5}}$

17/10/2020

Problem Reduction

We already know about the divide and conquer strategy. a solution to a problem can be obtained by decomposing it into smaller sub problems. Each of this sub-problem can then be solved to get its sub solution. These sub solution can then recombined to get a solution as a whole. This is called problem Reduction. This method generates arc which is called as AND arcs. One AND arc may point to any number of successor nodes, all of which must be solved inorder for an arc to point to a solution.

eg: AO*

Constraint Satisfaction problem [CSP]

→ It is a search procedure that operates in a space of constraint sets.

→ constraint satisfaction problems in AI have goal of discovering some problem state that satisfies a given set of constraints

eg: map coloring, sudoko, graph coloring etc.

→ CSP consists of 3 components V, D, C

→ V is a set of variables. $\{v_1, v_2 \ldots v_n\}$

→ D is a set of domains $\{D_1, D_2 \cdot\cdot D_3\}$ one for each variable.

→ C is a set of constraints that specify allowable combination of values. $\{c_1, c_2, c_3\}$

$$c_i = (scope, rel)$$

→ where scope is a set of variables that participate in the constraint.

→ Relation is relation that defines the values that variable can be take.

7. Means-Ends Analysis [MEA] JP

Means-End analysis is problem-solving techniques used in AI for limiting search in AI programs.

=> It is a mixture of Backward & forward search technique.

=> MEA analysis process centered on the evaluation of the difference b/w the current state & goal state.

How mean-End-Analysis works

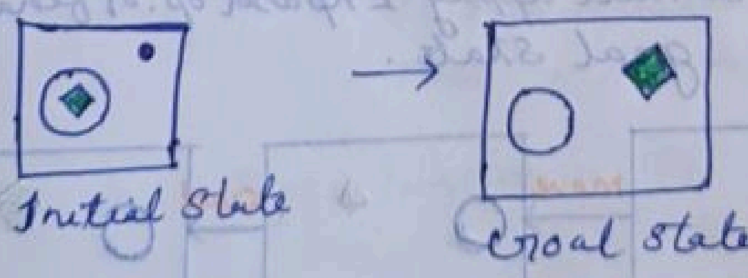
Step 1: First evaluate difference b/w initial state & final state.

Step 2: Select various operators which can be applied for each difference.

Step 3: Apply the operator at each difference, which reduces the difference b/w the current state and goal state.

Example of MEA

Ex



Solution

To solve the pbm, we will 1st find the difference b/w Initial state & goal state, and for each difference we will generate and new state & will apply the operators. The operator we have for this problem are

- Move
- Delete
- Expand.

1. Evaluating the initial state
   Compare diff of both states.

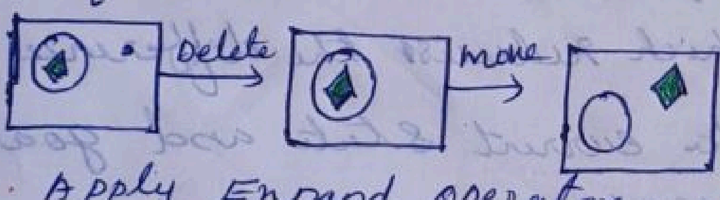
Initial state

2. Applying delete operator

   1st diffrence is no dot symbol in goal state. So apply delete operator to remove this dot.

3. Applying move operator

   After applying delete op, the new state occurs, again compare CTS. After compare, another diff is a square is outside the circle, so apply Move op.



4. Apply Expand operator

   compare GS, one diffrence is found that is the size of square. So will apply Expand op. & finally it will generate goal state.

22/10/2020.

Game playing

→ Minimax Algorithm

→ Alpha Beta (α, β) pruning

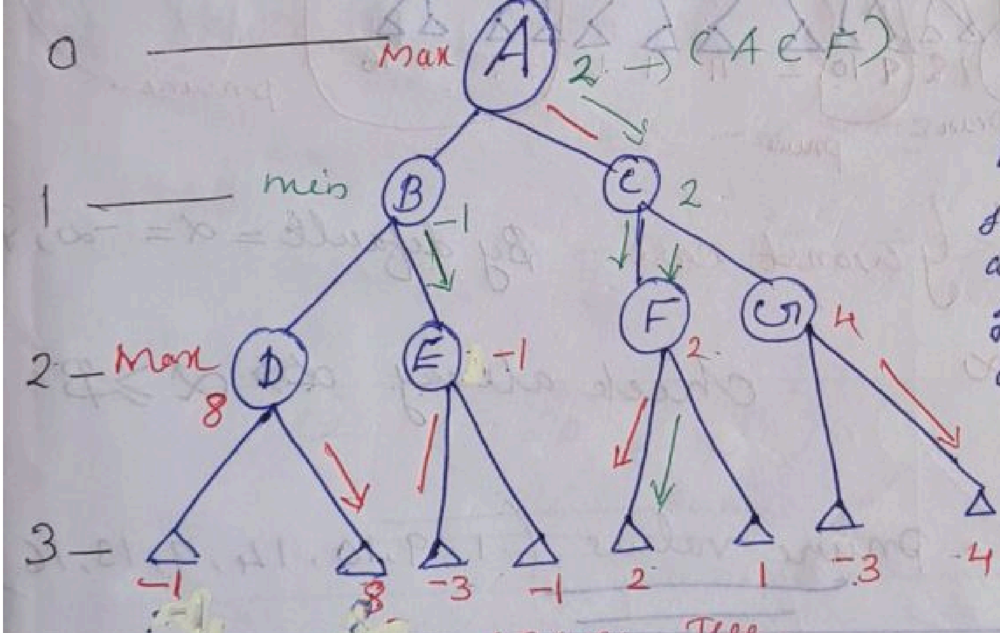
Minimax Algorithm

→ Backtracking Algorithm eg: Chess, Checker Tic-Tac-Toe etc

→ Best move Strategy is used

→ Max will try to maximize its utility. (Best move)

→ Min will try to minimize its utility (worst move)



Game Tree.

Branching factor
It is the no of children at each node th outdegree, If this value is not uniform, the average bF calculated

first Step : Take max value
Then min value
Then max value.

Time complexity $O(b^d)$

$= 2^3 = 8$

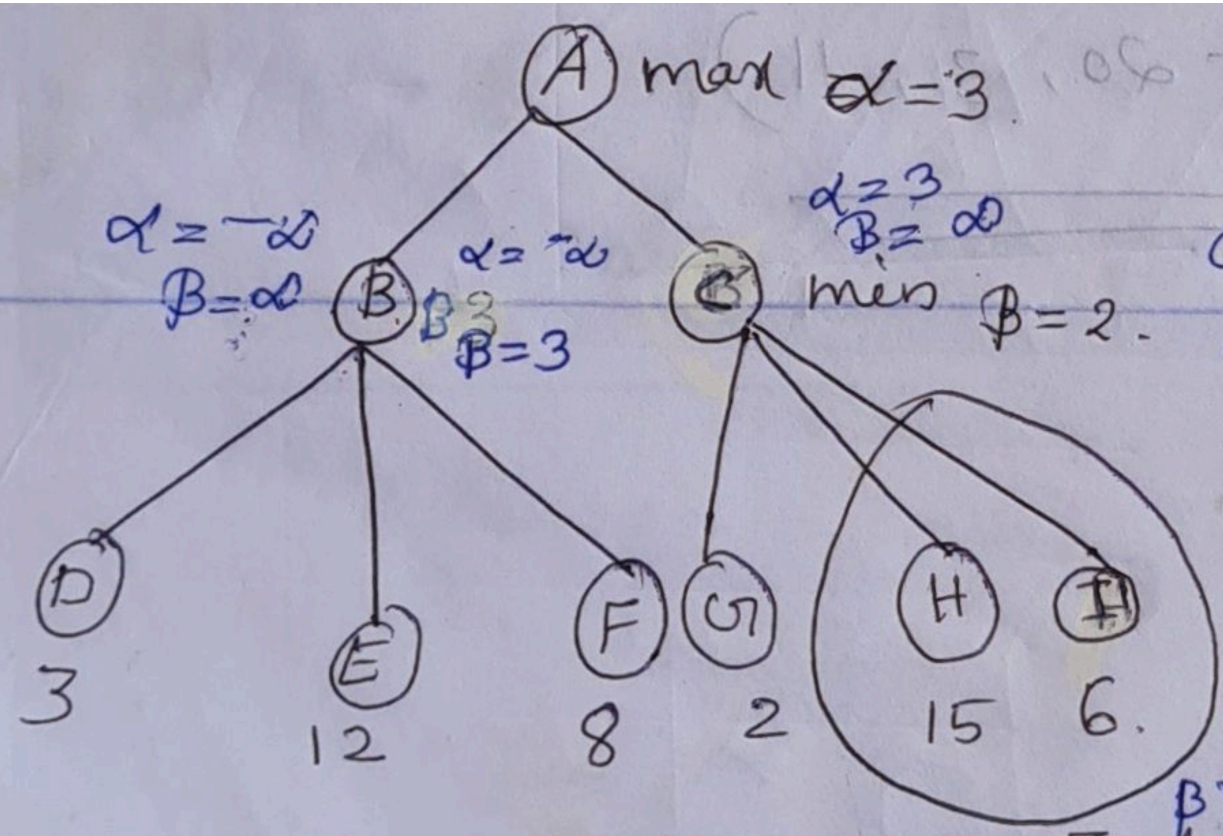
Alpha – Beta pruning ($\alpha - \beta$)

→ Cutoff search by exploring less number of nodes.

α – Max node (Maximizer – lower bound)

β – Min node. (β minizer – upper Bound)

Qn



Compare the value of G ie less than α value ie $3 \ngtr 2$ $\therefore$ 2 copied into β and compare ie $3 \geq 2 \Rightarrow$ True prune. So prune it H and I

β prune.

$\alpha \geq \beta$, $\alpha = -\infty$, $\beta = \infty$ [always check this default value is $\alpha = -\infty$ and $\beta = +\infty$]

$-\infty \geq 3 \Rightarrow$ false ie $\beta = 3$, $\alpha = 3$

ie A value = 3

$\beta = 2$.   pruning = HI