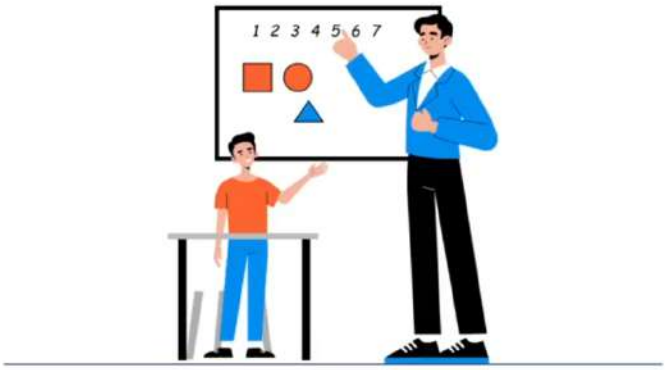
EIOV

Topics to be covered...

State space approach
Search strategies
Informed vs Uninformed
BFS and DFS
Heuristic Function
Hill climbing algorithm
Water jug problem
Constraint Satisfaction Problems
Backtracking
Game playing good candidate
Min-Max Procedure
Alpha-beta pruning
Happy Ending!



EIOV

State space approach

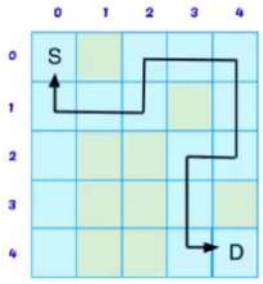
State space approach

- State space search is a process used in the field of computer science, including artificial intelligence (AI), in which successive configurations or states of an instance are considered, with the intention of finding a goal state with the desired property.
- A State space is the set of all states reachable from the initial state.
- A state space forms a graph in which the nodes are states.
- In the state space, a path is a sequence of states connected by a sequence of actions.

EIOV

State space approach example

- A maze problem can be represented as a state-space
- Each state represents “where you are” that is the current position in the maze
- The start state or initial state represents your starting position
- The goal state represents the exit from the maze.
- Rules (for a rectangular maze) are: move north, move south, move east, and move west.



EIOV

Search strategies

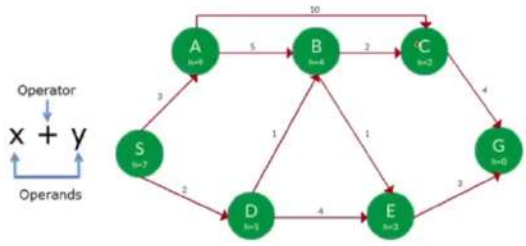
EIOV

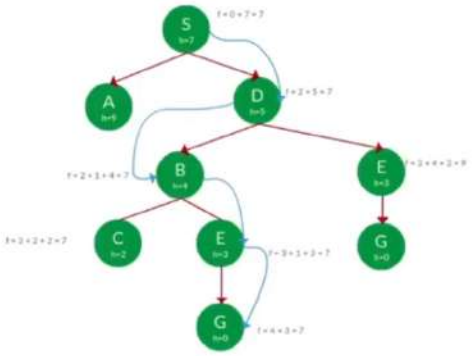
Searching Process

Searching

- Searching is the sequence of steps that transforms the initial state to the goal state.

- The process of search includes:
  - Initial state description of the problem.
  - A set of legal operators that change the state.
  - The finel or goal state.





EIOV

Parameters used to evaluate a search technique

1. Completeness

- A search algorithm is said to be complete if it guarantees to return a solution if at least any solution exists for any random input.

2. Optimality

- If a solution found for an algorithm is guaranteed to be the best solution (lowest path cost) among all other solutions.

3. Time Complexity

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

4. Space Complexity

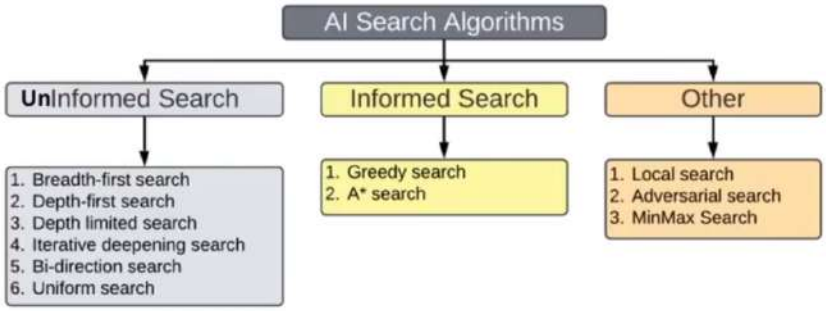
- It is the maximum storage space required at any point during the search, as the complexity of the problem.

EIOV

Informed vs Uninformed

EIOV

Informed vs Uninformed



EIOV

Informed vs Uninformed

Uninformed

- The uninformed search does not contain any domain knowledge such as closeness, the location of the goal.
- It operates in a brute-force way.
- 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.

Informed

- Informed search algorithms use domain knowledge.
- A heuristic is a way which might not always be guaranteed for best solutions but guaranteed to find a good solution in reasonable time.

Uninformed (Blind search) search and informed search (Heuristic search) algorithms.

EIOV

Informed vs Uninformed

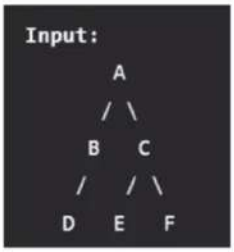
EIOV

BFS and DFS

EIOV

BFS(Breadth First Search)

- BFS stands for Breadth First Search.
- It is also known as level order traversal.
- The Queue data structure is used for the Breadth First Search traversal.
- When we use the BFS algorithm for the traversal in a graph, we can consider any node as a root node.
- It is slower than DFS.



EIOV

BFS(Breadth First Search)

unit-3

Advantage

- If there is more than one solution for a given problem, then BFS provides the minimal solution which requires the least number of steps.

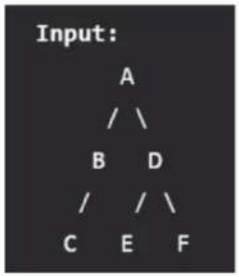
Disadvantage

- BFS needs lots of time if solution far.

EIOV

DFS(Depth First Search)

- DFS stands for Depth First Search.
- In DFS traversal, the stack data structure is used, which works on the LIFO principle.
- In DFS, traversing can be started from any node, or we can say that any node can be considered as a root node until the root node is not mentioned in the problem.



A, B, C, D, E, F

EIOV

DFS(Depth First Search)

Advantage

- It takes less time to reach to the goal node.

Disadvantage

- It goes for deep down and sometimes in infinite loop.

EIOV

BFS(Breadth First Search)

Sr. No.	Key	BFS	DFS
1	Definition	BFS, stands for Breadth First Search.	DFS, stands for Depth First Search.
2	Data structure	BFS uses Queue to find the shortest path.	DFS uses Stack to find the shortest path.
3	Source	BFS is better when target is closer to Source.	DFS is better when target is far from source.
4	Suitablity for decision tree	As BFS considers all neighbour so it is not suitable for decision tree used in puzzle games.	DFS is more suitable for decision tree. As with one decision, we need to traverse further to augment the decision. If we reach the conclusion, we won.
5	Speed	BFS is slower than DFS.	DFS is faster than BFS.
6	Time Complexity	Time Complexity of BFS = O(V+E) where V is vertices and E is edges.	Time Complexity of DFS is also O(V+E) where V is vertices and E is edges.


EIOV
AI
Heuristic Function

EIOV

Heuristic Function

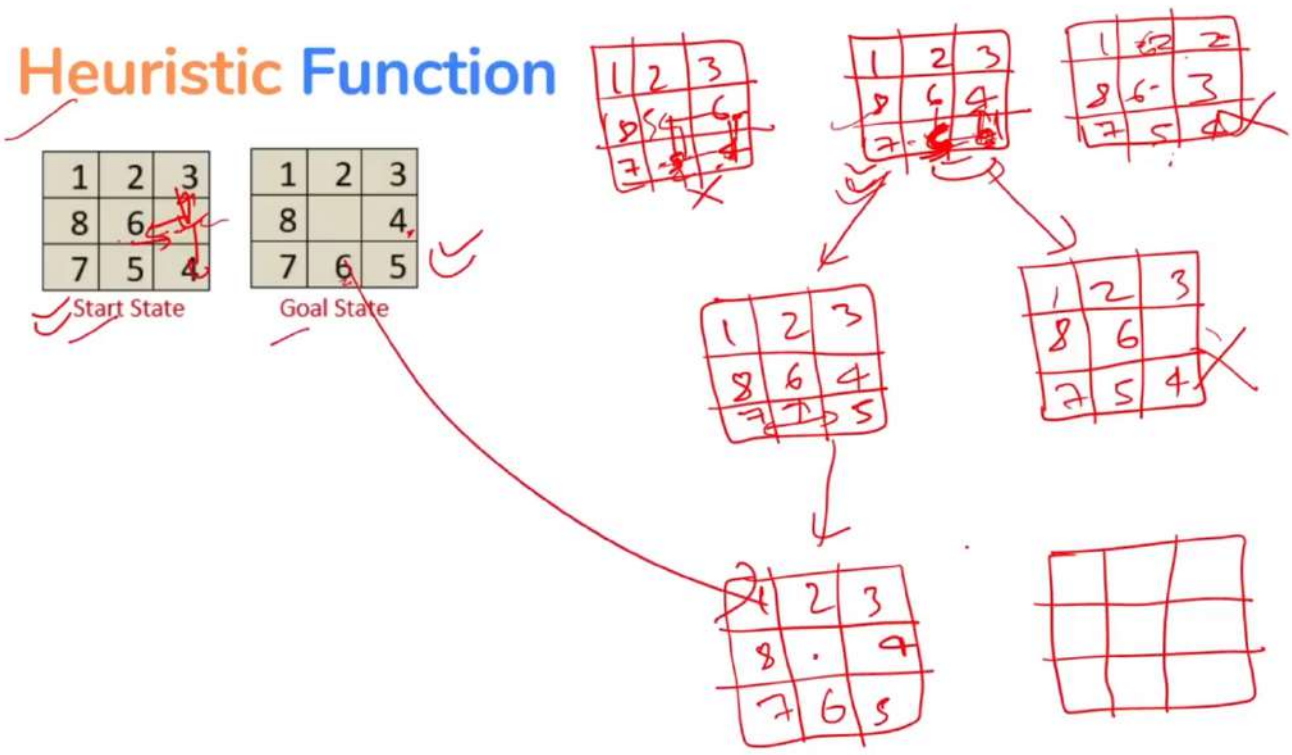
- Heuristic is a function which is used in Informed Search, and it finds the most promising path.
- It takes the current state of the agent as its input and produces the estimation of how close agent is from the goal.
- Not always give the best solution, but it guaranteed to find a good solution in reasonable time.
- It estimates how close a state is to the goal.
- Represented by h(n).
- The value of the heuristic function is always positive.

$$h(n) <= h^*(n)$$

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

- Best First Search Algorithm(Greedy search)
- A* Search Algorithm

Heuristic Function



EIOV

Heuristic Function

•**Blind search** → traversing the search space until the goal nodes is found (might be doing exhaustive search). •*Techniques* : **Breadth First Uniform Cost ,Depth first, Interactive Deepening search**. •Guarantees solution.	•**Heuristic search** → search process takes place by traversing search space with applied rules (information). •*Techniques*: **Greedy Best First Search, A* Algorithm** •There is no guarantee that solution is found.

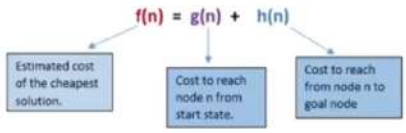
EIOV

Heuristic Function

EIOV

Heuristic Function

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



EIOV

Heuristic Function

- Example

watch Previous example of dice in A* Algo

EIOV

Hill climbing algorithm

Airplane mode enabled

EIOV

Hill climbing algorithm Concept

- It is a local search algorithm which continuously moves in the direction of increasing value to find the peak of the mountain.
- It terminates when it reaches a peak value where no neighbor has a higher value.
- **Example:** Traveling-salesman Problem in which we need to minimize the distance traveled by the salesman.
- Also called greedy local search.
- A node of hill climbing algorithm has two components which are state and value.

Features:

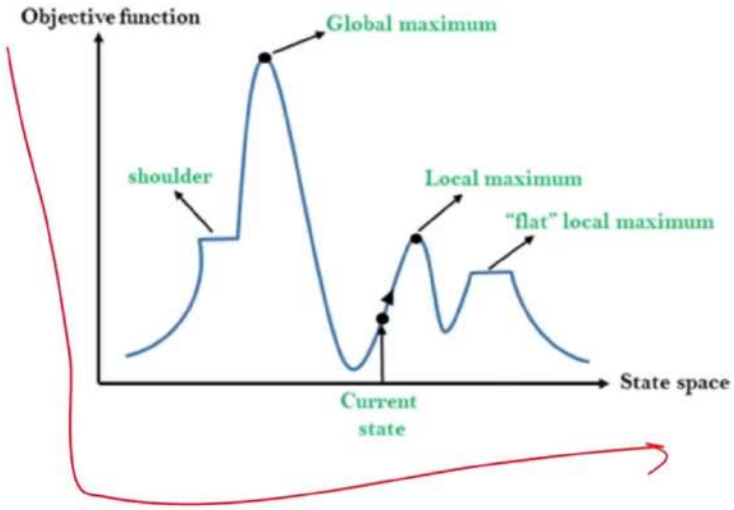
- **Greedy approach:** Hill-climbing algorithm search moves in the direction which optimizes the cost.
- **No backtracking:** It does not backtrack the search space, as it does not remember the previous states.

Hill climbing algorithm

- **Step 1:** Evaluate the initial state, if it is goal state then return success and Stop.
- **Step 2:** Loop Until a solution is found or there is no new operator left to apply.
- **Step 3:** Select and apply an operator to the current state.
- **Step 4:** Check new state:
  - If it is goal state, then return success and quit.
  - Else if it is better than the current state then assign new state as a current state.
  - Else if not better than the current state, then return to step2.
- **Step 5:** Exit.

EIOV

Hill climbing algorithm

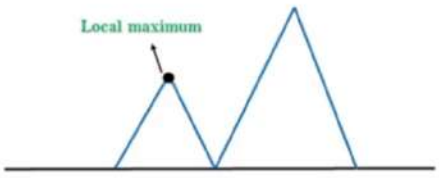


EIOV

Drawbacks and Solution

1. Local Maximum:

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



Solution:

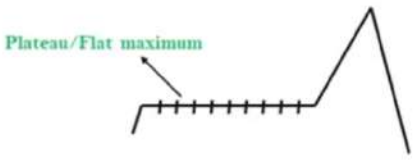
- Backtracking technique can be a solution of the local maximum in state space landscape.

EIOV

Drawbacks and Solution

2. Plateau:

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



Solution:

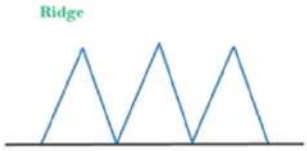
- Take big steps or very little steps while searching.

EIOV

Drawbacks and Solution

3. Ridge:

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



Solution:

- With the use of bidirectional search.

EIOV

Water jug problem

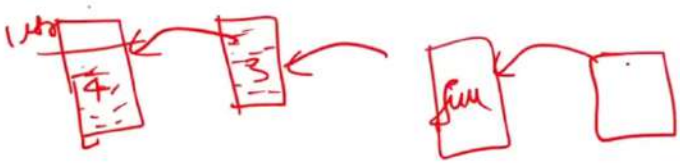
EIOV

Water jug problem

“You are given two jugs, a 4-liter one and a 3-liter one. Neither has any measuring markers on it. There is a pump that can be used to fill the jugs with water. How can you get exactly 2 liters of water into a 4-liter jug.”

State: (x, y)
where x represents the quantity of water in a 4-liter jug and y represents the quantity of water in a 3-liter jug.
That is, x = 0, 1, 2, 3, or 4 y = 0, 1, 2, 3
Start state: (0, 0).
Goal state: (2, n) for any n.

Water jug problem



EIOV

1	(x, y) is X<4 ->(4, Y)	Fill the 4-liter jug
2	(x, y) if Y<3 -> (x, 3)	Fill the 3-liter jug
3	(x, y) if x>0 -> (x-d, d)	Pour some water out of the 4-liter jug.
4	(x, y) if Y>0 -> (d, y-d)	Pour some water out of the 3-liter jug.
5	(x, y) if x>0 -> (0, y)	Empty the 4-liter jug on the ground
6	(x, y) if y>0 -> (x,0)	Empty the 3-liter jug on the ground
7	(x, y) if X+Y >= 4 and y>0 -> (4, y-(4-x))	Pour water from the 3-liter jug into the 4-liter jug until the 4-liter jug is full

EIOV

Water jug problem

8	(x, y) if X+Y>=3 and x>0 -> (x-(3-y), 3))	Pour water from the 4-liter jug into the 3-liter jug until the 3-liter jug is full.
9	(x, y) if X+Y <=4 and y>0 -> (x+y, 0)	Pour all the water from the 3-liter jug into the 4-liter jug.
10	(x, y) if X+Y<=3 and x>0 -> (0, x+y)	Pour all the water from the 4-liter jug into the 3-liter jug.
11	(0, 2) -> (2, 0)	Pour the 2-liter water from the 3-liter jug into the 4-liter jug.
12	(2, Y) -> (0, y)	Empty the 2-liter in the 4-liter jug on the ground.

EIOV

Constraint Satisfaction Problems

EIOV

Constraint Satisfaction Problems

Constraint satisfaction is the process of finding a solution through a set of constraints that impose conditions that the variables must satisfy.

- Consider a Sudoku game with some numbers filled initially in some squares.
- You are expected to fill the empty squares with numbers ranging from 1 to 9 in such a way that no row, column or a block has a number repeating itself.
- This is a very basic constraint satisfaction problem.
- You are supposed to solve a problem keeping in mind some constraints.
- The remaining squares that are to be filled are known as variables, and the range of numbers (1-9) that can fill them is known as a domain.
- Variables take on values from the domain.
- The conditions governing how a variable will choose its domain are known as constraints.

EIOV

Constraint Satisfaction Problems

A constraint satisfaction problem (CSP) is a problem that requires its solution within some limitations or conditions also known as constraints. It consists of the following:

- A finite set of **variables** which stores the solution (V = {V1, V2, V3,....., Vn})
- A set of **discrete** values known as **domain** from which the solution is picked (D = {D1, D2, D3,.....,Dn})
- A finite set of **constraints** (C = {C1, C2, C3,....., Cn})

Suppose that a row, column and block already have 3, 5 and 7 filled in. Then the domain for all the variables in that row, column and block will be {1, 2, 4, 6, 8, 9}.

EIOV

Constraint Satisfaction Problems

The following problems are some of the popular problems that can be solved using CSP:

1. CryptArithmetic
2. n-Queen
3. Map Coloring
4. Crossword
5. Sudoku
6. Latin Square Problem

EIOV

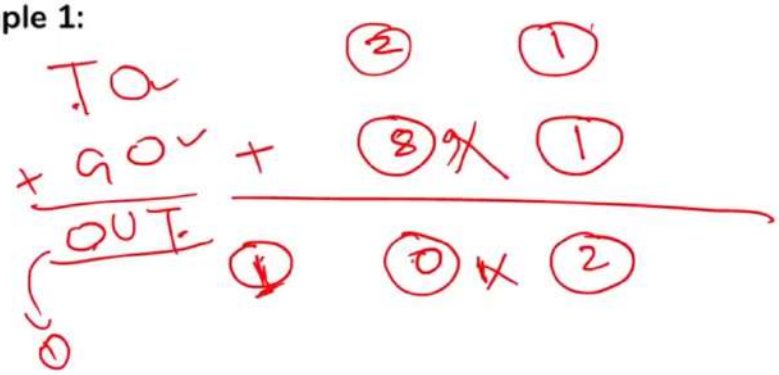
CSP CryptArithmetic

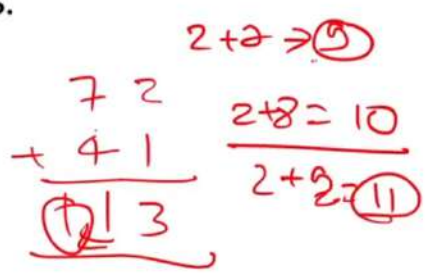
Types: CSP

Constraints:

- The result should satisfy the predefined arithmetic rules, i.e., 2+2 =4, nothing else.
- Digits should be from **0-9** only.
- The problem can be solved from both sides, i.e., **L.H.S, or R.H.S.**

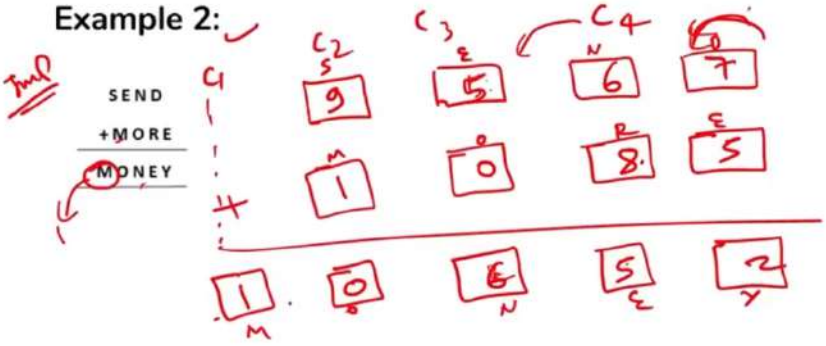
Example 1:





CSP CryptArithmetic

Example 2:



$S + 1 = O \rightarrow (1) \rightarrow C$

$8 + 1 = 9$

$9 + 1 = (0)$

$E + 0 = N$

$N + R = E$

$6 + 8 = 5 = (14)$

$6 + 9 = 15$

(15)

$D + E = Y$

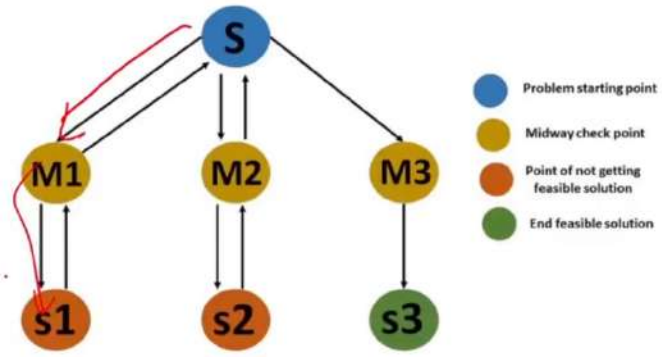
EIOV

Backtracking

- In artificial intelligence, backtracking is often used in search algorithms to explore the search space and find a solution to a problem.
- It involves trying a set of possible choices, and if a choice leads to a dead end or a solution cannot be found, then backtracking to the previous step and trying a different choice.
- Backtracking algorithms are typically used when the solution to a problem cannot be determined in advance and needs to be discovered through trial and error.
- **For example**
  - Backtracking can be used in constraint satisfaction problems, where variables are subject to constraints, and a value must be found for each variable that satisfies all of the constraints. In this case, the algorithm would try different values for each variable and backtrack if it determines that a particular value does not satisfy the constraints.

EIOV

Backtracking



EIOV

Game playing good candidate

Game playing is a good candidate for artificial intelligence (AI) because it involves many of the challenges that AI systems are designed to tackle. Games often require decision-making, problem-solving, and strategy, which are all areas where AI can excel.

Some specific reasons why game playing is a good candidate for AI include:

1. Games have well-defined rules and objectives: The rules of a game provide a clear framework for AI systems to operate within. This makes it easier to design algorithms and evaluate their performance.
2. Games provide a controlled environment: In a game, the AI system knows all the rules and has access to all the information it needs to make decisions. This makes it easier to design and test AI algorithms, as compared to more open-ended environments where the AI may need to handle unexpected events or incomplete information.
3. Games offer a rich source of data: Games provide a large amount of data that can be used to train and evaluate AI algorithms. For example, chess has been used extensively to test AI algorithms because it provides a large number of well-defined positions that can be used to train and evaluate the algorithms.

EIOV

Min-Max Procedure

EIOV

Min-Max Algorithm

- Mini-max algorithm is a recursive or backtracking algorithm which is used in decision-making and game theory.
- Mini-Max algorithm uses recursion to search through the game-tree.
- Min-Max algorithm is mostly used for game playing in AI. Such as Chess, Checkers, tic-tac-toe, go, and various tow-players game. This Algorithm computes the minimax decision for the current state.
- In this algorithm two players play the game, one is called MAX and other is called MIN.
- Both the players fight it as the opponent player gets the minimum benefit while they get the maximum benefit.
- Both Players of the game are opponent of each other, where MAX will select the maximized value and MIN will select the minimized value.
- The minimax algorithm performs a depth-first search algorithm for the exploration of the complete game tree.
- The minimax algorithm proceeds all the way down to the terminal node of the tree, then backtrack the tree as the recursion.

EIOV

Properties Min-Max Algorithm

- **Complete**- Min-Max algorithm is Complete. It will definitely find a solution (if exist), in the finite search tree.
- **Optimal**- Min-Max algorithm is optimal if both opponents are playing optimally.
- **Time complexity**- O(bm), where b is branching factor of the game-tree, and m is the maximum depth of the tree.
- **Space Complexity**- O(bm).

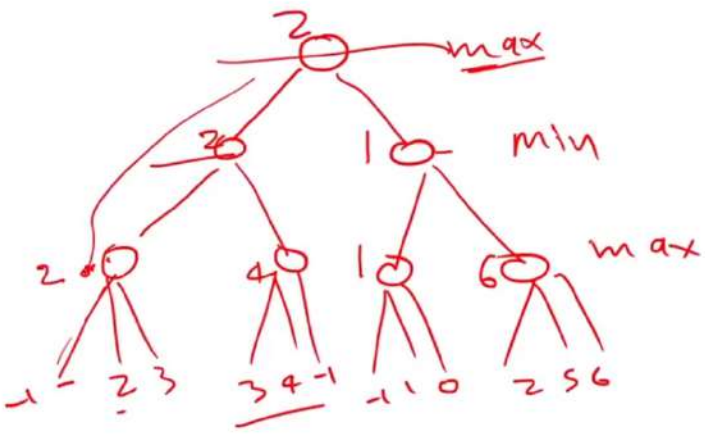
Limitation of the minimax Algorithm:

The main drawback of the minimax algorithm is that it gets really slow for complex games such as Chess, go, etc.

EIOV

Properties Min-Max Algorithm

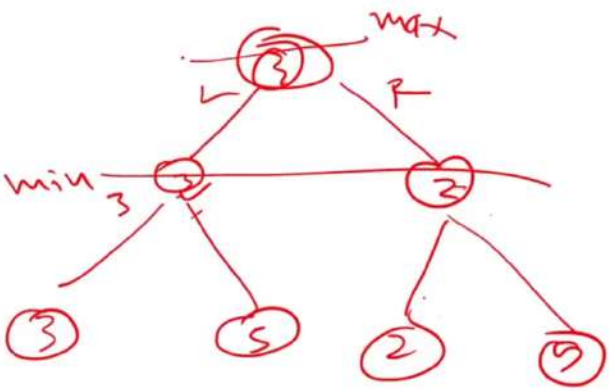
- **Example 1:**



EIOV

Properties Min-Max Algorithm

- **Example 2:**



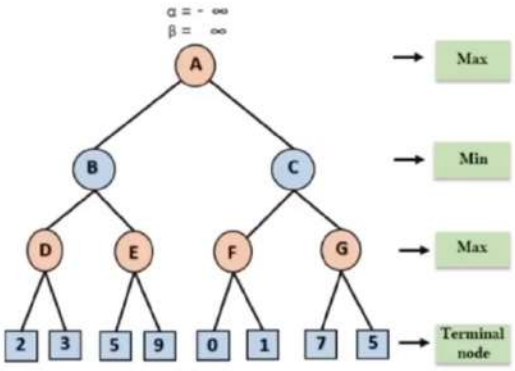
EIOV

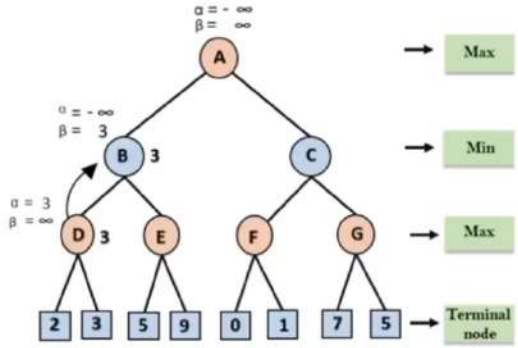
Alpha-beta pruning

- Alpha-beta pruning is a modified version of the minimax algorithm.
- It is an optimization technique for the minimax algorithm.
- There is a technique by which without checking each node of the game tree we can compute the correct minimax decision, and this technique is called **pruning**.
- This involves two threshold parameter Alpha and beta for future expansion, so it is called **alpha-beta pruning**.
- It is also called as **Alpha-Beta Algorithm**.
  - **Alpha**: The best (highest-value). The initial value of alpha is -∞.
  - **Beta**: The best (lowest-value). The initial value of beta is +∞.
- **Condition:**
  - $\alpha >= \beta$
- **Key points:**
  - The Max player will only update the value of alpha.
  - The Min player will only update the value of beta.
  - We will only pass the alpha, beta values to the child nodes.

EIOV

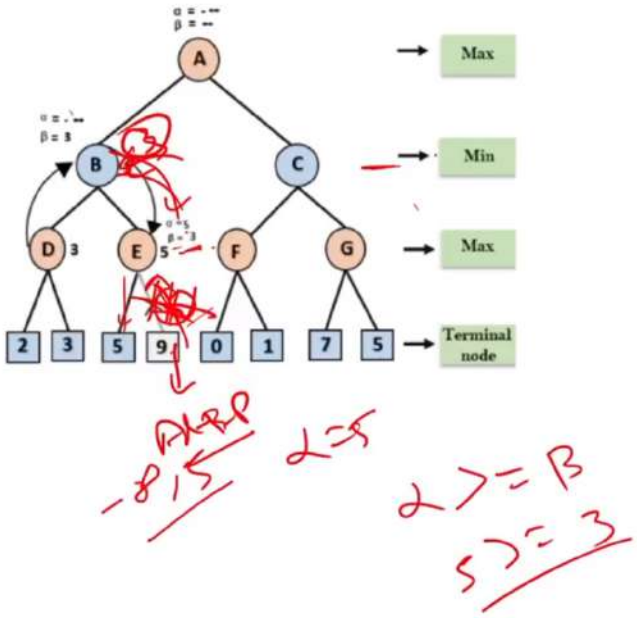
Alpha-beta pruning

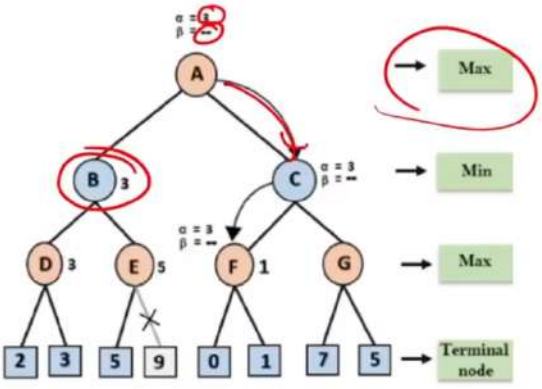




EIOV

Alpha-beta pruning





EIOV

Alpha-beta pruning