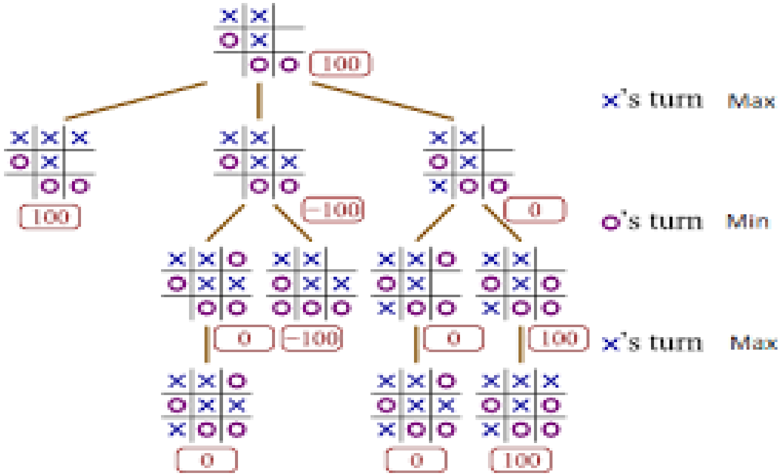
AMIRAJ

COLLEGE OF ENGINEERING & TECHNOLOGY

CHAPTER – 10
GAME PLAYING :OVERVIEW AND EXAMPLE DOMAIN



Subject : AI

Code : 2180703

Prepared By:
Asst. Prof. Twinkal Panchal
(CSE Department, ACET)

AMIRAJ
COLLEGE OF ENGINEERING & TECHNOLOGY

Mini-Max Algorithm

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

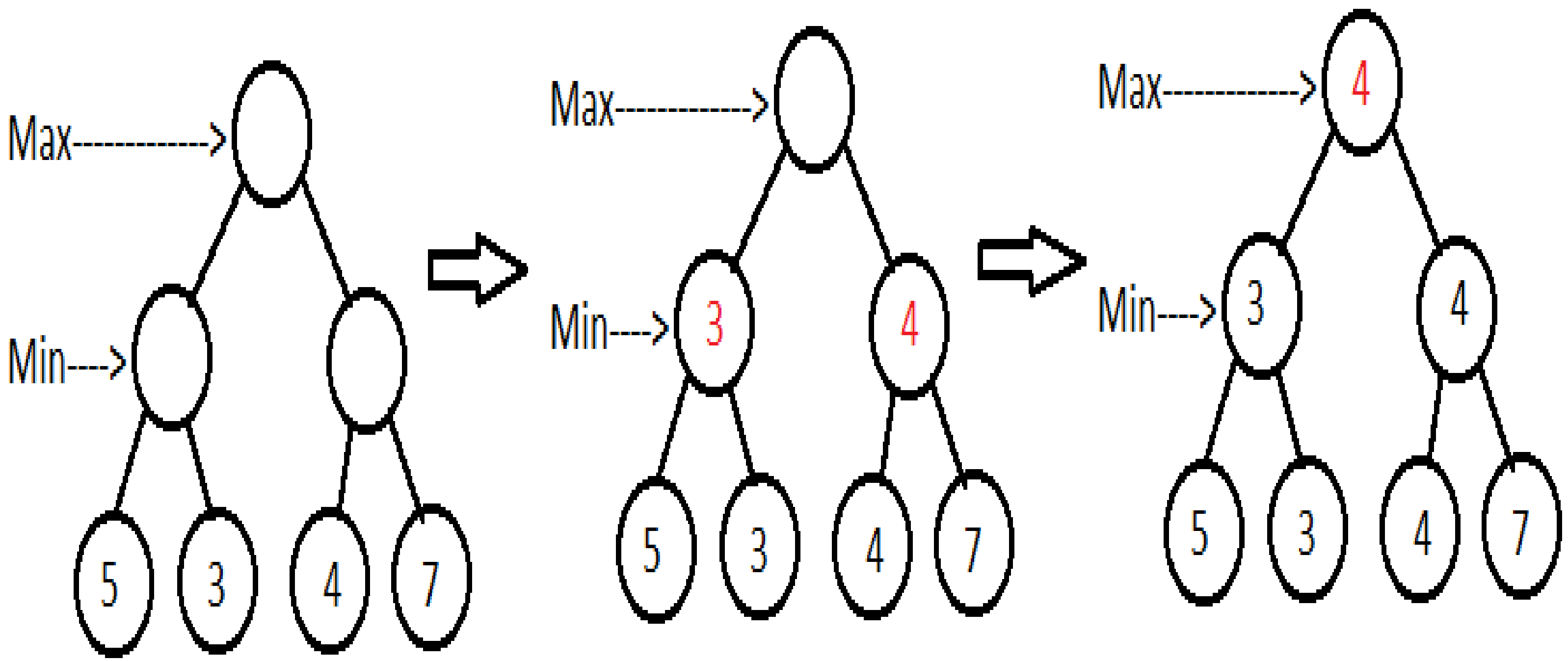
- In this algorithm two players play the game, one is called MAX and other is called MIN.

AMIRAJ
COLLEGE OF ENGINEERING & TECHNOLOGY

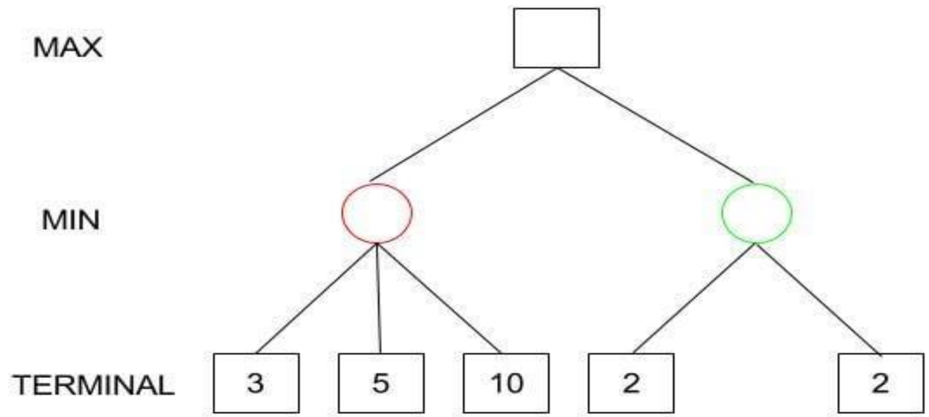
- Both Players of the game are opponent of each other, where MAX will select the maximized value and MIN will select the minimized value.

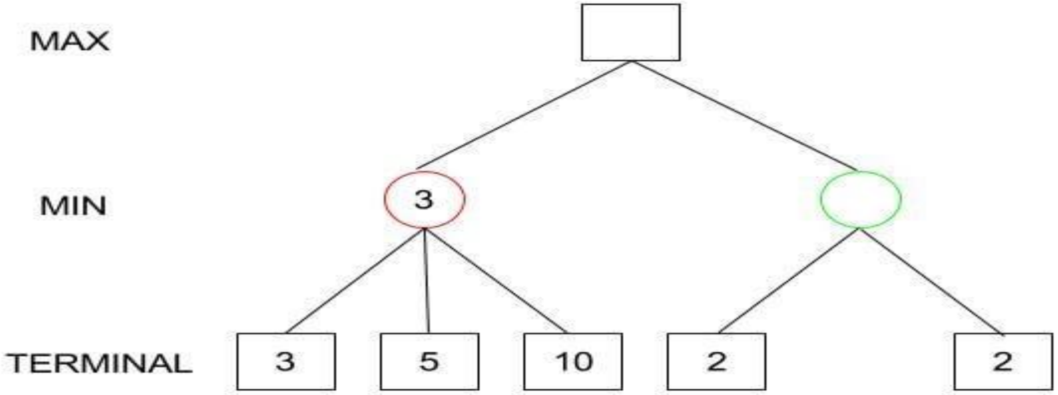
- Minimax is a decision-making algorithm, typically used in a turn-based, two player games. The goal of the algorithm is to find the optimal next move.

- In the algorithm, one player is called the maximizer, and the other player is a minimizer. If we assign an evaluation score to the game board, one player tries to choose a game state with the maximum score, while the other chooses a state with the minimum score.


Max------------->
Min---->
5
3
4
7
Max------------->
Min---->
3
4
5
3
4
7
Max------------->
4
Min---->
3
4
5
3
4
7

EX –





Minimax Decision = MAX{MIN{3,5,10},MIN{2,2}}
= MAX{3,2}
= 3

Alpha-Beta Pruning

- Alpha-beta pruning is a modified version of the minimax algorithm. It is an optimization technique for the minimax algorithm.

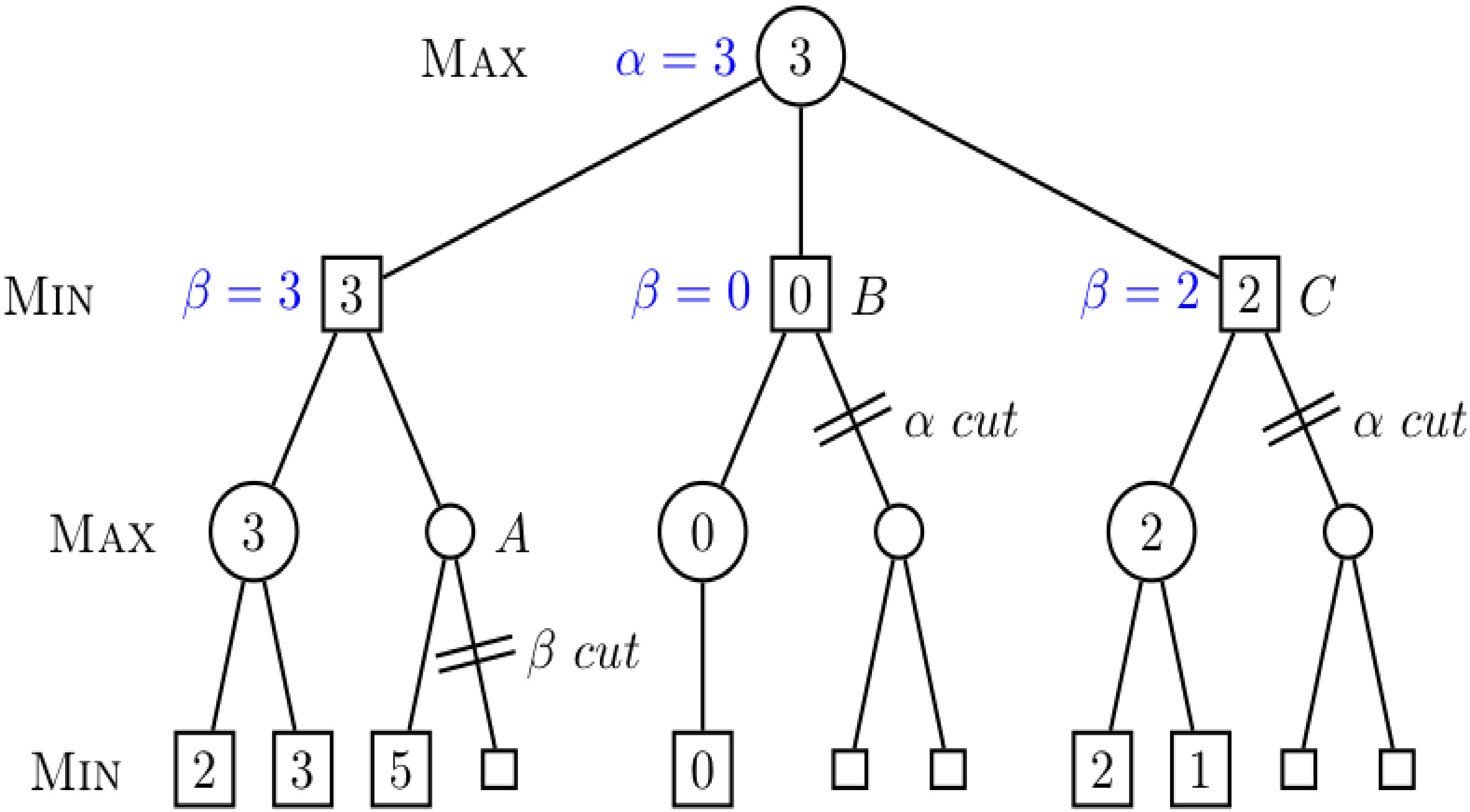
- As we have seen in the minimax search algorithm that the number of game states it has to examine are exponential in depth of the tree. Since we cannot eliminate the exponent, but we can cut it to half.

- Hence 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.

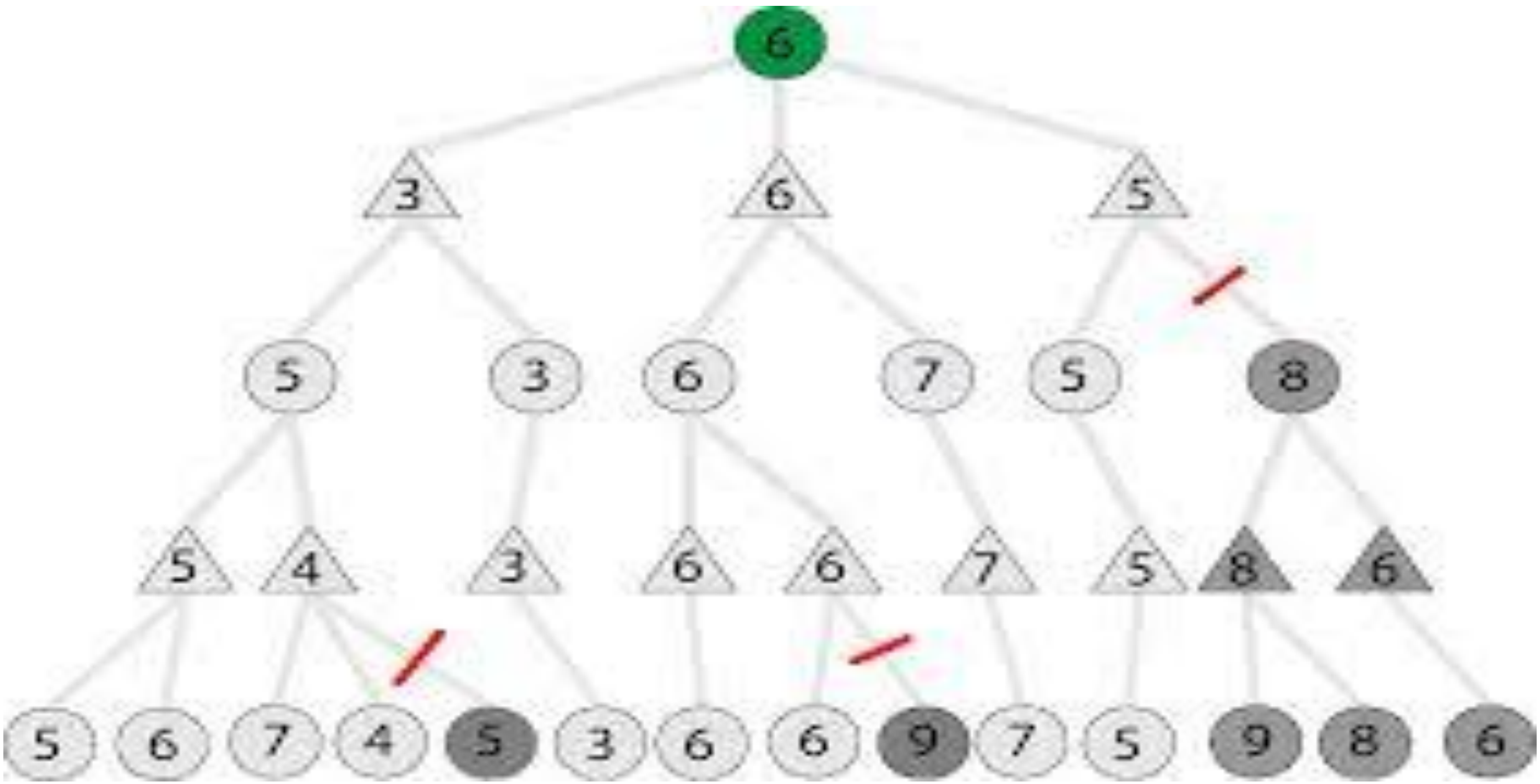
- This involves two threshold parameter Alpha and beta for future expansion, so it is called alpha-beta pruning. It is also called as Alpha-Beta Algorithm.

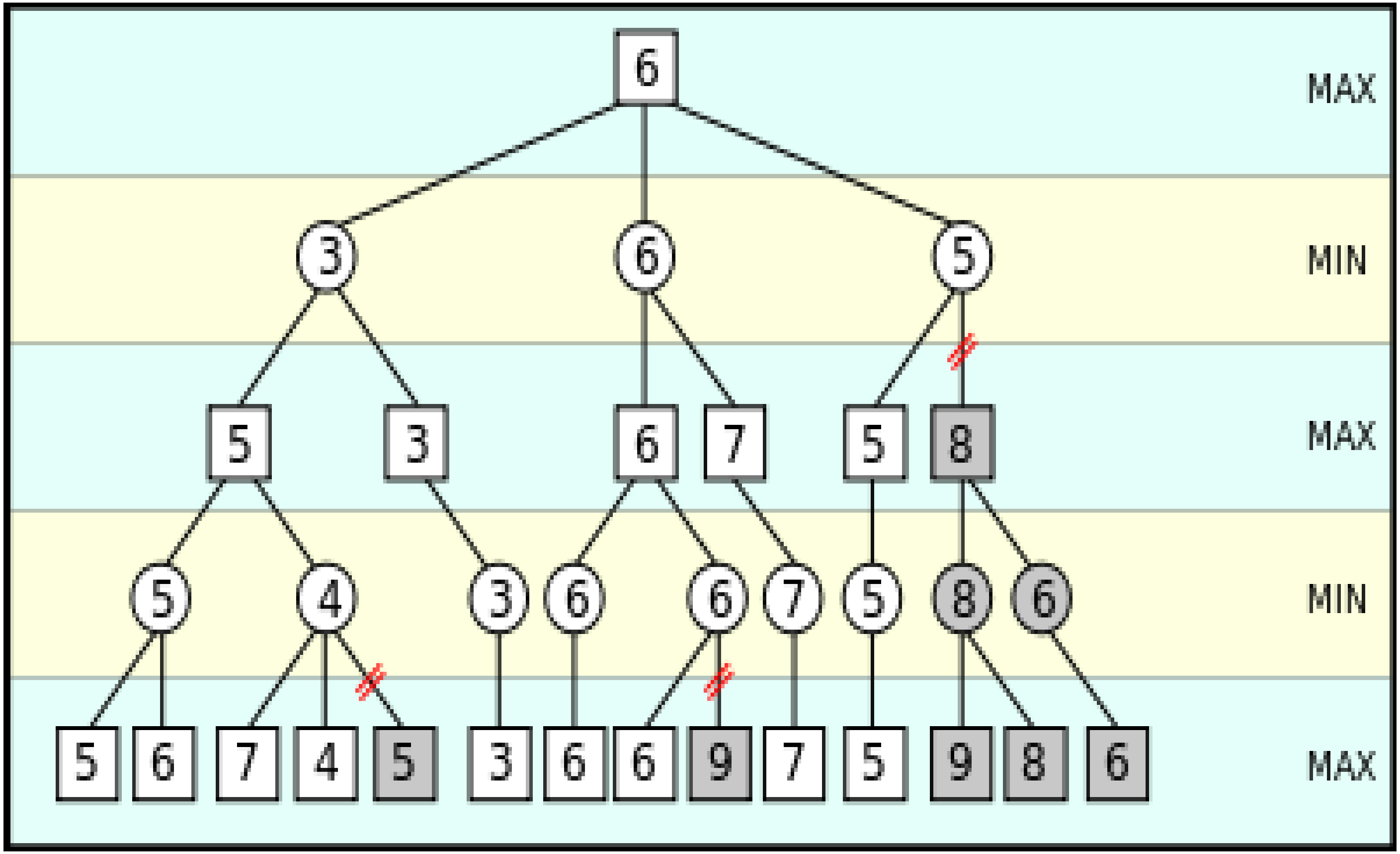
- Alpha-beta pruning can be applied at any depth of a tree, and sometimes it not only prune the tree leaves but also entire sub-tree.

- The two-parameter can be defined as:
  1. **Alpha:** The best (highest-value) choice we have found so far at any point along the path of Maximizer.
  2. **Beta:** The best (lowest-value) choice we have found so far at any point along the path of Minimizer.



MAX
α = 3
3
MIN
β = 3
3
β = 0
0
B
β = 2
2
C
α cut
α cut
MAX
3
A
0
2
β cut
MIN
2
3
5
0
2
1


6
3 6 5
5 3 6 7 5 8
5 4 3 6 6 7 5 8 6
5 6 7 4 5 3 6 6 9 7 5 9 8 6


MAX
MIN
MAX
MIN
MAX

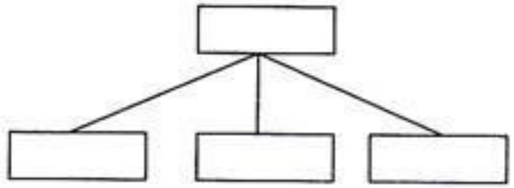
Iterative Deepening

- The minimax search is then initiated up to a depth of two plies and to more plies and so on. The name "iterative deepening" derives its name from the fact that on each iteration, the tree is searched one level deeper. This method is also called progressive deepening.

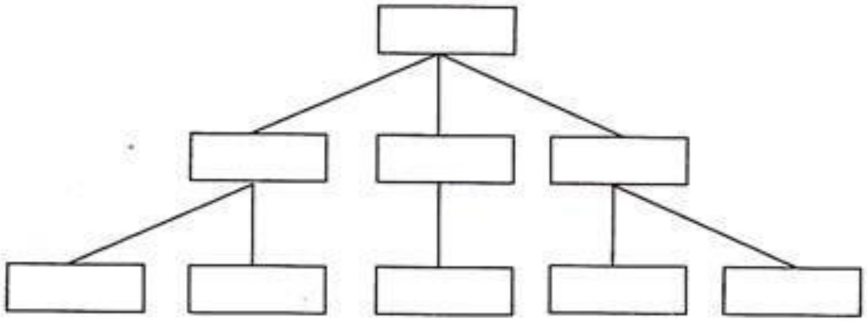
Advantages

1. Find the shortest path to the goal state.
2. The maximum amount of memory used by DFID is proportional to the number of nodes in that solution path.

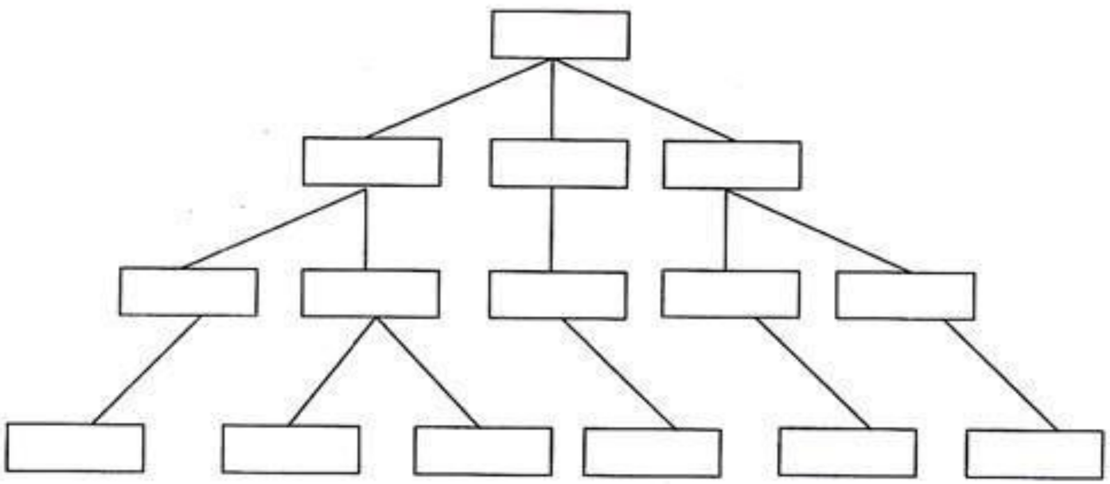
AMIRAJ COLLEGE OF ENGINEERING & TECHNOLOGY



Iteration 1
(*a*)

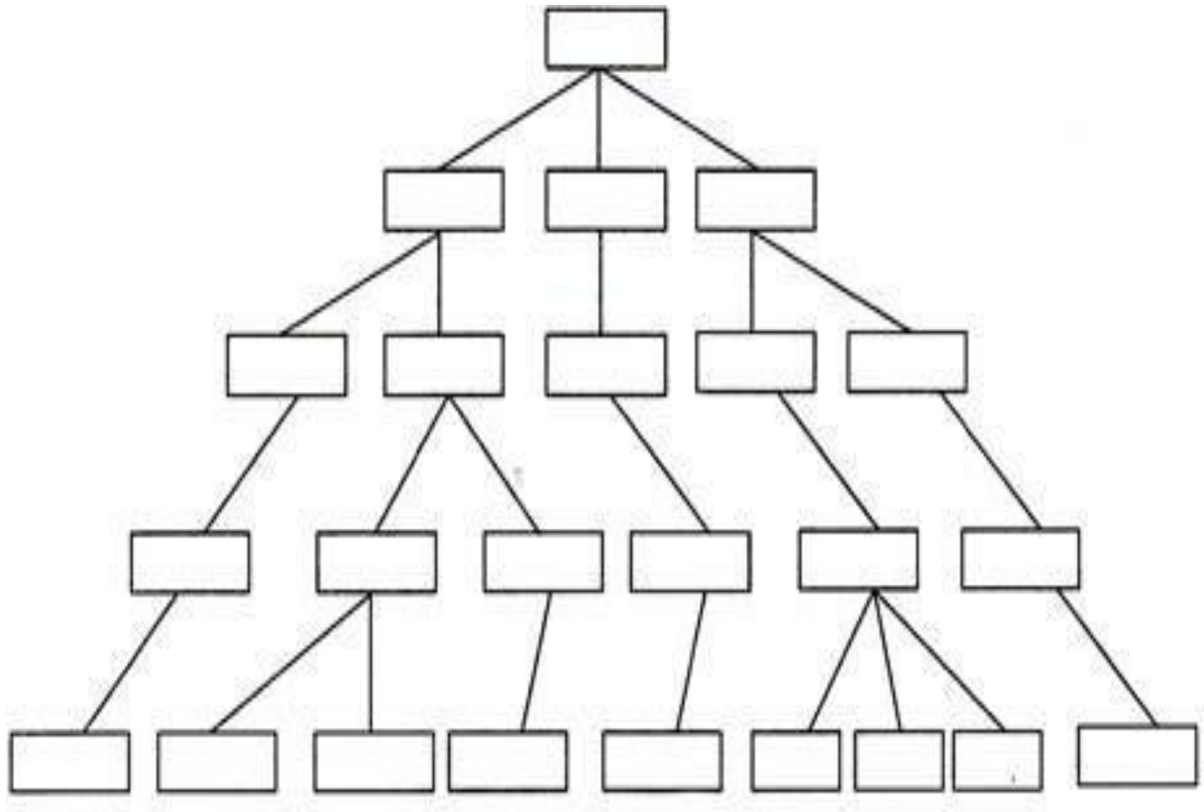


Iteration 2
(*b*)



Iteration 3
(*c*)

AMIRAJ
COLLEGE OF ENGINEERING & TECHNOLOGY



Iteration 4
(*d*)

Fig. 5.18. *Iterative deepening.*

Blocks World

- The blocks world has two kinds of components:A table top with three places **p**, **q**, and **r**

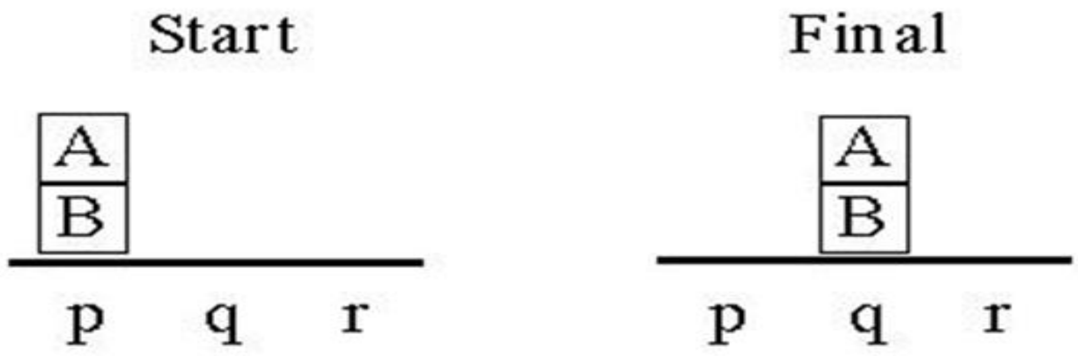
- A variable number of blocks **A**, **B**, **C**, etc., that can be arranged in places on the table or stacked on one another

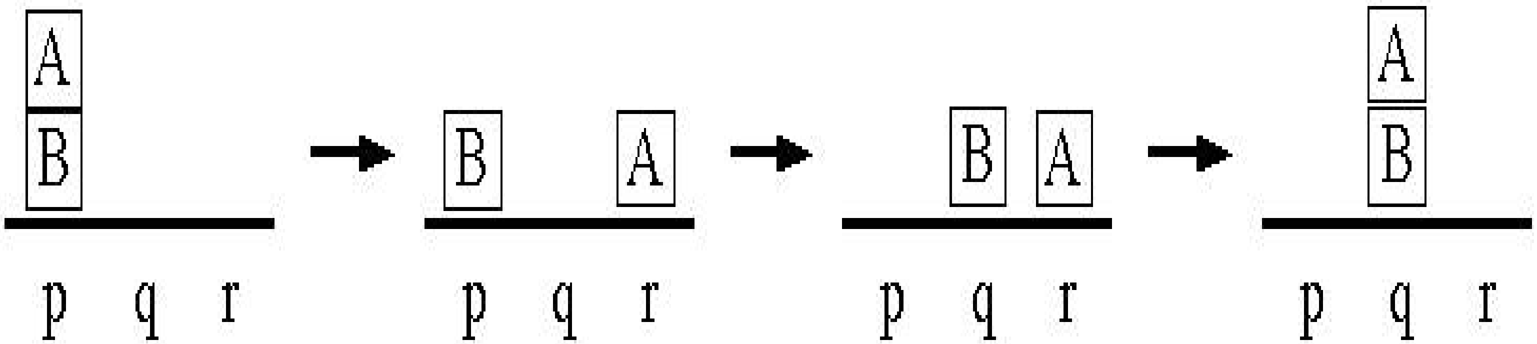
- A legal move is to transfer a block from one place or block onto another place or block, with these restrictions:The moved block must not have another block on top of it

- No other blocks are moved in the process

Ex-

- Here is a simple blocks world problem:


Start
Final
A
B
p q r
A
B
p q r


A
B
p q r
B
A
p q r
B
A
p q r
A
B
p q r

Components of Planning System

1. Choose the best rule for applying the next rule based on the best available heuristics.
2. Apply the chosen rule for computing the new problem state.
3. Detect when a solution has been found.
4. Detect dead ends so that they can be abandoned and the system's effort is directed in more fruitful directions.
5. Detect when an almost correct solution has been found.

AMIRAJ COLLEGE OF ENGINEERING & TECHNOLOGY

Goal stack planning

- This is one of the most important planning algorithms, which is specifically used by STRIPS**.**
- The stack is used in an algorithm to hold the action and satisfy the goal. A knowledge base is used to hold the current state, actions.
- Goal stack is similar to a node in a search tree, where the branches are created if there is a choice of an action.

AMIRAJ
COLLEGE OF ENGINEERING & TECHNOLOGY

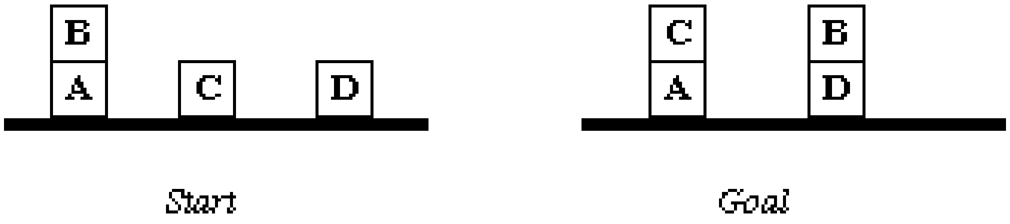
1. Start by pushing the original goal on the stack. Repeat this until the stack becomes empty. If stack top is a compound goal, then push its unsatisfied subgoals on the stack.

2. If stack top is a single unsatisfied goal then, replace it by an action and push the action's precondition on the stack to satisfy the condition.

3. If stack top is an action, pop it from the stack, execute it and change the knowledge base by the effects of the action.

4. If stack top is a satisfied goal, pop it from the stack.

Consider the following where wish to proceed from the start to goal state.



start state:

ON(B, A) ONTABLE(A) ONTABLE(C) ONTABLE(D) ARMEMPTY

goal state:

ON(C, A) ON(B,D) ONTABLE(A) ONTABLE(D)

- Initially the goal stack is the goal state.
- We then split the problem into four subproblems
- Two are solved as they already are true in the initial state -- ONTABLE(A), ONTABLE(D).

- With the other two -- there are two ways to proceed:

  ON(C,A) ON(*B*,*D*)

  ON(*C*,*A*) ON(*B*,*D*)

  ONTABLE(*A*) ONTABLE(*D*)

  ON(*B*,*D*) ON(*C*,*A*)

  ON(*C*,*A*) ON(*B*,*D*)

  ONTABLE(*A*) ONTABLE(*D*)

- The first goal ON(*C*,*A*) is not true and the only operator that would make it true is STACK (*C*,*A*) which replaces ON(*C*,*A*) giving:

  B<>STACK (*C*,*A*) ON(*B*,*D*)

  ON(*C*,*A*) ON(*B*,*D*)

  ONTABLE(*A*) ONTABLE(*D*)

- STACK has prerequisites that must be met which means that block A is clear and the arm is holding block C. So we must do:

  B<>CLEAR(*A*) HOLDING(*C*)

  CLEAR(*A*) HOLDING(*C*)

  STACK (*C*,*A*)

  ON(*B*,*D*)

  ON(*C*,*A*) ON(*B*,*D*)

  ONTABLE(*A*) ONTABLE(*D*)

- Now top goal is false and can only be made true by unstacking B. This leads to:

  B<>ON(*B*,*A*) CLEAR(*B*)

  ARMEMPTY

  ON(*B*,*A*) CLEAR(*B*)

ARMEMPTY

UNSTACK(*B*,*A*)

HOLDING(*C*)

CLEAR(*A*) HOLDING(*C*)

- The goal stack becomes HOLDING(C) CLEAR(A) HOLDING(C)

STACK (C,A)

ON(B,D)

ON(C,A) ON(B,D) ONTABLE(A)

ONTABLE(D)

There are two ways we can achieve HOLDING(*C*) by using the operators PICKUP(*C*) or UNSTACK(*C*,*x*) where *x* is an unspecified block. This leads to two alternative paths:

ON(C, x) CLEAR(C)

ARMEMPTY

ON(C, x) CLEAR(C)

ARMEMPTY

UNSTACK(C,x)

CLEAR(A) HOLDING(C)

STACK (C,A)

ON(B,D)

ON(C,A) ON(B,D) ONTABLE(A)

ONTABLE(D)

ONTABLE(C) CLEAR(C)

ARMEMPTY

ONTABLE(C) CLEAR(C)

ARMEMPTY

PICKUP(C)

CLEAR(*A*) HOLDING(*C*)

STACK (*C*,*A*)

ON(*B*,*D*)

ON(*C*,*A*) ON(*B*,*D*) ONTABLE(*A*)

ONTABLE(*D*)

- CLEAR(*x*) HOLDING(*C*)

  CLEAR(*x*) HOLDING(*C*)

  STACK (*C*, *x*)

  CLEAR(*C*)

  ARMEMPTY

- The new goal stack becomes: CLEAR(*D*) HOLDING(*B*)

  CLEAR(*D*) HOLDING(*B*)

  STACK (*B*, *D*)

  ONTABLE(*C*) CLEAR(*C*) ARMEMPTY

  PICKUP(*C*)

- At this point the top goal is true and the next and thus the combined goal leading to the application of STACK (*B*,*D*), which means that the world model becomes

  ONTABLE(*A*) ONTABLE(*C*) ONTABLE(*D*) ON(*B*,*D*) ARMEMPTY

- This means that we can perform PICKUP(*C*) and then STACK (*C*,*A*)

- Now coming to the goal ON(*B*,*D*) we realise that this has already been achieved and checking the final goal we derive the following plan

  UNSTACK(*B*,*A*)

  STACK (*B*,*D*)

  PICKUP(*C*)

  STACK (*C*,*A*)

Non-linear planning

- This planning is used to set a goal stack and is included in the search space of all possible subgoal orderings. It handles the goal interactions by interleaving method.

Advantage

- Non-linear planning may be an optimal solution with respect to plan length (depending on search strategy used).

Disadvantages

- It takes larger search space, since all possible goal orderings are taken into consideration.
- Complex algorithm to understand.

AMIRAJ COLLEGE OF ENGINEERING & TECHNOLOGY

AMIRAJ
COLLEGE OF ENGINEERING & TECHNOLOGY