| | |
|---|---|
| **Batch:** C1 **Roll No.:** 16010121221 | |
| **Experiment / assignment / tutorial No. 7** | |

**Title:** Implementation of Alpha Beta Pruning.

**Objective:** Implementation of Alpha-Beta Pruning algorithm

**Expected Outcome of Experiment:**

| Course Outcome | After successful completion of the course students should be able to |
|---|---|
| CO2 | Analyse and solve problems for goal based agent architecture (searching and planning algorithms). |

**Books/ Journals/ Websites referred:**

1. **"Artificial Intelligence: a Modern Approach" by Russel and Norving, Pearson education Publications**
2. **"Artificial Intelligence" By Rich and knight, Tata Mcgraw Hill Publications**
3. **www.cs.sfu.ca/CourseCentral/310/oschulte/mychapter5.pdf**
4. **http://cs.lmu.edu/~ray/notes/asearch/**
5. **www.cs.cornell.edu/courses/cs4700/2011fa/.../06_adversarialsearch.pdf**

**Pre Lab/ Prior Concepts:** Two/Multi player Games and rules, state-space tree, searching algorithms and their analysis properties

**Historical Profile: -** The game playing has been integral part of human life. The multiplayer games are competitive environment in which everyone tries to gain more points for himself and wishes the opponent to gain minimum.

The game can be represented in form of a state space tree and one can follow the path from root to some goal node, for either of the player.

**New Concepts to be learned:** Adversarial search, minmax algorithm, minmax pruning,

## Adversarial Search:-

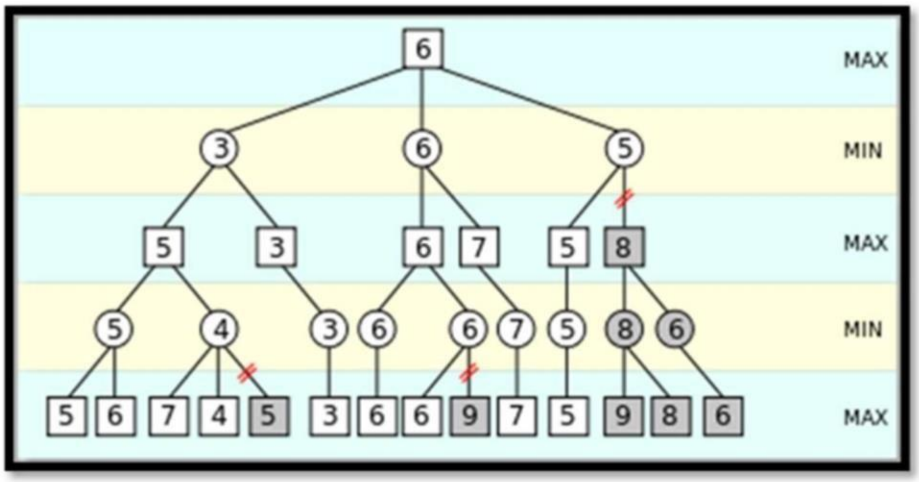
Adversarial search has more than one entity, and each entity has conflicting goals and objectives. These entities are pitted against each other in a game like situation, and the strategy or game approach of each player varies as per the opponent's move. A player in Adversarial Search may take a maximizer role or minimizer role depending on the game situation and the intent of the opponent. Maximizer will always try to maximize the gain, and Minimizer will try to control the damage and minimize the loss. Adversarial search is used in many fields to find optimal solutions in a competitive environment and few of the applications are:

- In the legal system where two advocates argue their stand representing their party's cases and Judges or Jury takes cues from the above-explained techniques, analyze and deliver judgment.
- These techniques are used in the business environment in sourcing suppliers and buyers for Products/services in a competitive manner.
- Wherever hard negotiations are required, these concepts help to maximize the gain of those who can negotiate intelligently and prudently.
- This concept is the base for further development of new technologies like Alpha-Beta pruning,

## Alpha-beta pruning algorithm:

- 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:
    - Alpha: The best (highest-value) choice we have found so far at any point along the path of Maximizer. The initial value of alpha is -∞.
    - Beta: The best (lowest-value) choice we have found so far at any point along the path of Minimizer. The initial value of beta is +∞.
- The Alpha-beta pruning to a standard minimax algorithm returns the same move as the standard algorithm does, but it removes all the nodes which are not really

affecting the final decision but making algorithm slow. Hence by pruning thesenodes, it makes the algorithm fast.

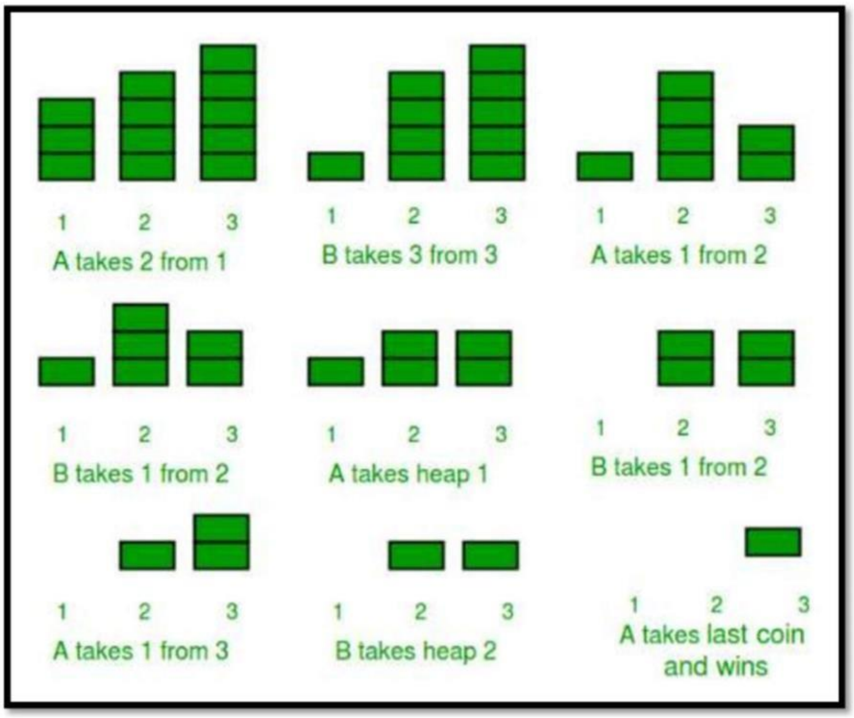


## Chosen Problem:

The Game of Nim is described by the following rules-
Given a number of piles in which each pile contains some numbers of stones/coins. In each turn, a player can choose only one pile and remove any number of stones (at leastone) from that pile. The player who cannot move is considered to lose the game (i.e., one who take the last stone is the winner).
You are playing the following Nim Game with your friend:

- Initially, there is a heap of stones on the table.
- You and your friend will alternate taking turns, and you go first.
- On each turn, the person whose turn it is will remove 1 to 3 stones from theheap.
- The one who removes the last stone is the winner.

Given n, the number of stones in the heap, return true if you can win the gameassuming both you and your friend play optimally, otherwise return false.
For example, consider that there are two players- A and B, and initially there are three piles of coins initially having 3, 4, 5 coins in each of them as shown below. We assumethat first move is made by A. See the below figure for clear understanding of the wholegame play.

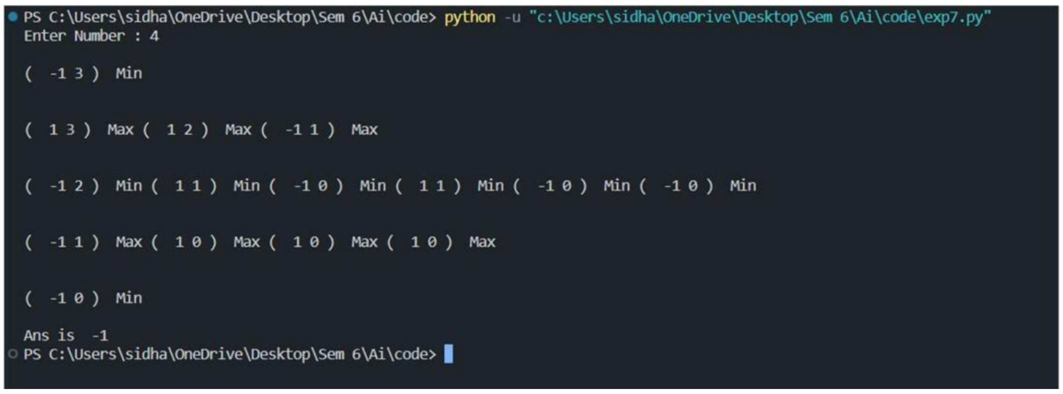


A Won the match (Note: A made the first move)

**Solution tree for chosen Problem:**

```
class Node:
 def__init__(self, rem_stones):
  self.rem_stones = rem_stones
  self.children = []
  self.value = 0
  self.level = 0

 def addChild(self, child):
  self.children.append(child)

 def setValue(self, value):
  self.value = value

 def setLevel(self, level):
  self.level = level
n = int(input("Enter Number : "))
root = Node(n)
# Odd level - Max
queue = []
def dfs(node, level):
 rem_stones = node.rem_stones
 if rem_stones == 0:
  if level % 2 == 0:
   node.setValue(-1)
 else:
  node.setValue(1)
 return
 for i in range(1, min(rem_stones+1, 4)):
  new_stones = rem_stones-i
 child = Node(new_stones)
child.setLevel(level+1)
node.addChild(child)
dfs(child, level+1)
dfs(root, 0)
def getValue(node):
 n = len(node.children)

 l = node.level
 all = []
 for i in range(n):
  if node.children[i].value != 0:
   all.append(node.children[i].value)
 else:
 all.append(getValue(node.children[i]))
 if node.level % 2 == 0:
 node.setValue(min(all))
 return min(all)
 else:
 node.setValue(max(all))
 return max(all)
ans = getValue(root)
queue.append(root)
while queue:
 n = len(queue)
 print(" ")
 lvl = "Min"
 for i in range(n):
  node = queue.pop(0)
 if node.level % 2 == 1:
  lvl = "Max"
 c = len(node.children)
 print("( ", node.value, c, end=" )")
 for j in range(c):
 queue.append(node.children[j])
 print(" ", lvl, end=(" "))
 print(" ")
 print()
print("Ans is ", root.value)
```

**OUTPUT:**



```
PS C:\Users\sidha\OneDrive\Desktop\Sem 6\Ai\code> python -u "c:\Users\sidha\OneDrive\Desktop\Sem 6\Ai\code\exp7.py"
Enter Number : 4

(  -1 3 )  Min

(  1 3 )  Max (  1 2 )  Max (  -1 1 )  Max

(  -1 2 )  Min (  1 1 )  Min (  -1 0 )  Min (  1 1 )  Min (  -1 0 )  Min (  -1 0 )  Min

(  -1 1 )  Max (  1 0 )  Max (  1 0 )  Max (  1 0 )  Max

(  -1 0 )  Min

Ans is  -1
PS C:\Users\sidha\OneDrive\Desktop\Sem 6\Ai\code> 
```

## Post Lab objective Questions:

1. **Which search is equal to minmax search but eliminates the branches that can’t influence the final decision?**
   a. Breadth-first search
   b. Depth first search
   c. Alpha-beta pruning
   d. None of the above
2. **Which values are independent in minmax search alogirthm?**
   a. Pruned leaves x and y
   b. Every states are dependant
   c. Root is independent
   d. None of the above

## Post Lab Subjective Questions:

1. **Explain the concept of adversarial search**
   **Ans:** Adversarial search is a search, where we examine the problem which arises when we try to plan ahead of the world and other agents are planning against us. Adversarial search is a game-playing technique where the agents are surrounded by a competitive environment. A conflicting goal is given to the agents (multiagent). These agents compete with one another and try to defeat one another in order to win the game. Such conflicting goals give rise to the adversarial search. Here, game-playing means discussing those games where human intelligence and logic factor is used, excluding other factors such as luck factor. Tic-tac-toe, chess, checkers, etc., are such type of gameswhere no luck factor works, only mind works
2. **Explain how alpha-beta pruning improves memory efficiency of algorithm**
   **Ans:** The alpha-beta pruning is a standard min-max algorithm that returns the same moveas the standard algorithm does, but it removes all the nodes which are not

affecting thefinal decision but making the algorithm slow. Hence by pruning these nodes, it makesthe algorithm fast.

3. **Explain how a game of chess may benefit from min-max and alpha-beta pruning algorithms.**

**Ans:** Minimax algorithm is used to determine the score in a zero-sum game after a certain number of moves, with best play according to an evaluation function. The algorithm can be explained like this:

In a one-ply search, where only move sequences with length one are examined, the sideto move (max player) can simply look at the evaluation after playing all possible moves. The move with the best evaluation is chosen. But for a two-ply search, when the opponentalso moves, things become more complicated. The opponent (min player) also chooses the move that gets the best score. Therefore, the score of each move is now the score of the worst that the opponent can do.

The Alpha-Beta algorithm is a significant enhancement to the minimax search algorithm that eliminates the need to search large portions of the game tree applying a branch-and- bound technique. Remarkably, it does this without any potential of overlooking a better move. If one already has found a quite good move and search for alternatives, one refutation is enough to avoid it. No need to look for even stronger refutations. The algorithm maintains two values, alpha and beta. They represent the minimum score that the maximizing player is assured of and the maximum score that the minimizing playeris assured of respectively.