LAB 6 - IMPLEMENTATION OF MINIMAX ALGORITHM

AI LAB

RAJAT KUMAR

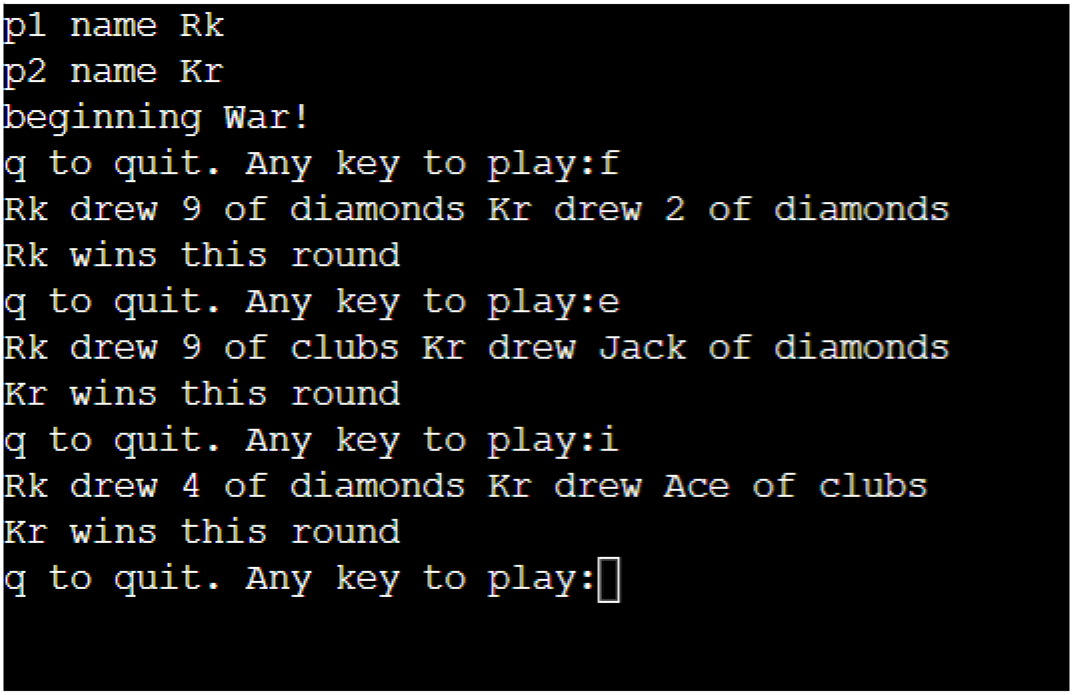
RA1911003010652

CODE FOR MINIMAX ALGORITHM:

```
from random import shuffle

class Card:

    suits = ["spades",

             "hearts",

             "diamonds",

             "clubs"]


    values = [None, None,"2", "3",

              "4", "5", "6", "7",

              "8", "9", "10",

              "Jack", "Queen",

              "King", "Ace"]


    def __init__(self, v, s):

        """suit + value are ints"""

        self.value = v

        self.suit = s


    def __lt__(self, c2):
        if self.value < c2.value:
            return True
        if self.value == c2.value:
            if self.suit < c2.suit:
                return True
            else:
                return False
        return False

    def __gt__(self, c2):
        if self.value > c2.value:
            return True
        if self.value == c2.value:
            if self.suit > c2.suit:
                return True
            else:
                return False
        return False

    def __repr__(self):
        v = self.values[self.value] +\
            " of " + \
            self.suits[self.suit]
        return v
```

```
class Deck:
    def __init__(self):
        self.cards = []
        for i in range(2, 15):
            for j in range(4):
                self.cards\
                    .append(Card(i,
                                  j))
        shuffle(self.cards)

    def rm_card(self):
        if len(self.cards) == 0:
            return
        return self.cards.pop()


class Player:
    def __init__(self, name):
        self.wins = 0
        self.card = None
        self.name = name


class Game:
    def __init__(self):
        name1 = input("p1 name ")
    name2 = input("p2 name ")
    self.deck = Deck()
    self.p1 = Player(name1)
    self.p2 = Player(name2)

def wins(self, winner):
    w = "{} wins this round"
    w = w.format(winner)
    print(w)

def draw(self, p1n, p1c, p2n, p2c):
    d = "{} drew {} {} drew {}"
    d = d.format(p1n,
                p1c,
                p2n,
                p2c)
    print(d)

def play_game(self):
    cards = self.deck.cards
    print("beginning War!")
    while len(cards) >= 2:
        m = "q to quit. Any " + \
            "key to play:"
        response = input(m)
        if response == 'q':
            break
        p1c = self.deck.rm_card()
        p2c = self.deck.rm_card()
        p1n = self.p1.name
        p2n = self.p2.name
        self.draw(p1n,
                  p1c,
                  p2n,
                  p2c)
        if p1c > p2c:
            self.p1.wins += 1
            self.wins(self.p1.name)
        else:
            self.p2.wins += 1
            self.wins(self.p2.name)

    win = self.winner(self.p1,
                      self.p2)
    print("War is over.{} wins"
          .format(win))

def winner(self, p1, p2):
    if p1.wins > p2.wins:
        return p1.name
    if p1.wins < p2.wins:
        return p2.name
    return "It was a tie!"


game = Game()

game.play_game()
```

OUTPUT:



```
p1 name Rk
p2 name Kr
beginning War!
q to quit. Any key to play:f
Rk drew 9 of diamonds Kr drew 2 of diamonds
Rk wins this round
q to quit. Any key to play:e
Rk drew 9 of clubs Kr drew Jack of diamonds
Kr wins this round
q to quit. Any key to play:i
Rk drew 4 of diamonds Kr drew Ace of clubs
Kr wins this round
q to quit. Any key to play:
```

RESULT: Hence, the implementation of Minimax algorithm is successfully done.

RAJAT KUMAR

RA1911003010652

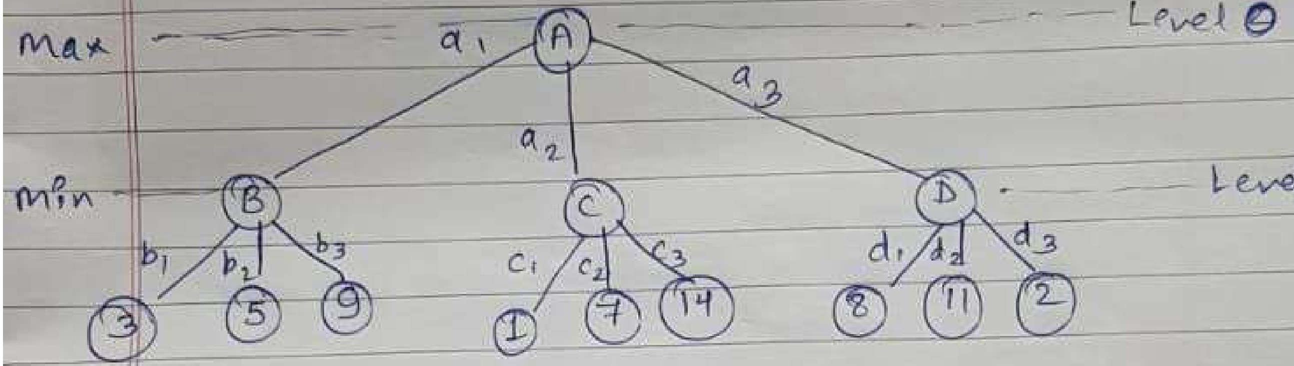
AI LAB-6

MINIMAX ALGORITHM

ALGORITHM:

consider an example.

Two players, min and max are playing a game that can be represented by a tree



step 1: choose min() in level 1

$B = \min(3,5,9)$, $C = \min(1,7,14)$, $D = \min(8,11,2)$

step: proceed to choose the optimal for max().

$A = \max(B,C,D) = \max(\min(3,5,9), \min(1,7,14), \min(8,11,2)$

step: Calculate the above expression
we will get the terminal value for this
problem = 3.