Experiment 4

Foundations of AI

Minimax algorithm

By- Shivansh Jain,VIT Chennai

AIM:-

To find the optimal score from the scores of leaf nodes using minimax algorithm

Concept:-

Minimax is a kind of backtracking algorithm that is used in decision making and game theory to find the optimal move for a player, assuming that your opponent also plays optimally.

In Minimax the two players are called maximizer and minimizer. The **maximizer** tries to get the highest score possible while the **minimizer** tries to do the opposite and get the lowest score possible.

Algorithm:-

1. Enter an array of scores find its length and thus its height
2. Check the terminating statement else continue
3. If its max player turn find max attainable from the children nodes else find minimum
4. Give the inputs values and call the function
5. Display the optimum score

STEP BY STEP IMPLEMENTATION:-

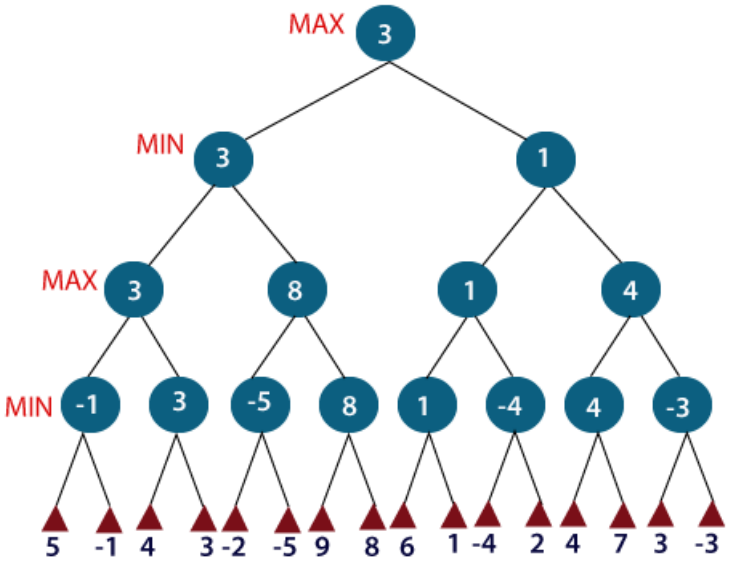
```
#R version 4.1.0

#RStudio version 1.4.1717


rm(list = ls())

#to ensure a clean environment before executing the code


minimax <- function(curDepth, nodeIndex,maxTurn, scores,targetDepth){

#function minimax containing 5 arguments

#curDepth beng the current depth of the node which changes ecery recursion

#nodeIndex being the index of node in array of nodes

#maxTurn a boolean value which showcases if its max players turn or not

#scores an array storing all the scores

#targetDepth being the maximum depth code will run till
```

```
  #Terminating condition
  #if we the target depth return the score of current node
  if (curDepth == targetDepth){
    return(scores[nodeIndex])}

  #if its maximizing players turn
  #return the max attainable value of two sets of children of current node
  #recursive algorithm
  if (maxTurn){
    return(max(minimax(curDepth + 1, nodeIndex* 2,FALSE, scores, targetDepth),minimax(curDepth + 1,
nodeIndex* 2 - 1,FALSE, scores, targetDepth)))



  }
  #minimizing turn
  #return the min attainable value of two sets of children of current node
  else{
    return(min(minimax(curDepth + 1, nodeIndex* 2,TRUE, scores, targetDepth),minimax(curDepth + 1,
nodeIndex * 2 - 1,TRUE, scores, targetDepth)))


  }
}
#driver code


test_case1 = c(3, 5, 6, 9, 1, 2, 0, -1)
len1=length(test_case1)
treeDepth1 = ceiling(log(len1, base=2))
minimax(0, 1, TRUE, test_case1, treeDepth1)


test_case2 = c(5, -1,4,3,-2,-5,9,8,6,1,-4,2,4,7,3,-3)
len2=length(test_case2)
treeDepth2 = ceiling(log(len2, base=2))
```

minimax(0, 1, TRUE, test_case2, treeDepth2)

RESULTS AND OUTPUT:-

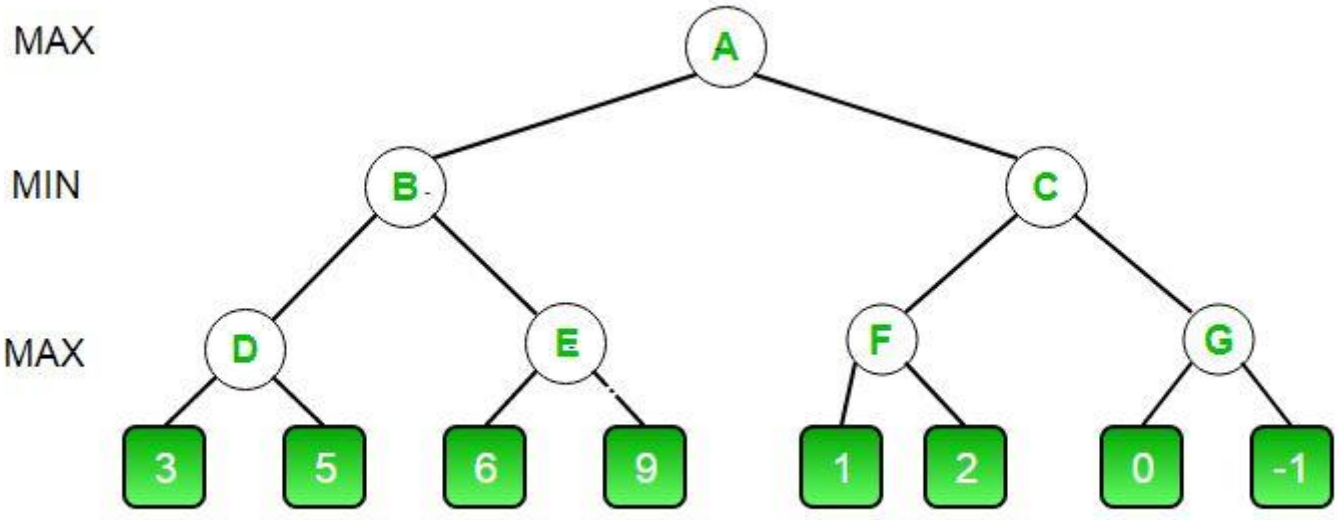
Test case 1:-



Output

```
> test_case2 = c(5, -1,4,3,-2,-5,9,8,6,1,-4,2,4,7,3,-3)
> len2=length(test_case2)
> treeDepth2 = ceiling(log(len2, base=2))
> minimax(0, 1, TRUE, test_case2, treeDepth2)
[1] 3
>
```

Test case 2:-



Output:-

```
> test_case1 = c(3, 5, 6, 9, 1, 2, 0, -1)
> len1=length(test_case1)
> treeDepth1 = ceiling(log(len1, base=2))
> minimax(0, 1, TRUE, test_case1, treeDepth1)
[1] 5
>
```