

SRM INSTITUTE OF SCIENCE & TECHNOLOGY DEPARTMENT OF NETWORKING & COMMUNICATIONS **18CSC305J-ARTIFICIAL INTELLIGENCE**

SEMESTER – 6 BATCH-2

|  |  |
| --- | --- |
| **REGISTRATION NUMBER** | **RA1911003011044** |
| **NAME** | **Sk.Mahammad Khaja Firoz** |

# B.Tech- CSE / CC, Third Year (Section: H2)

**Faculty In charge: Dr. S. Prabakeran, B.Tech, M.E, PhD**

**Assistant Professor**

**School of Computing - Department of**

**Networking and Communications**

**Year 2021-2022 / Even Semester**

**INDEX**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Ex No** | **DATE** | **Title** | **Page No** | **Marks** |
| 1 | 22/02/22 | Min-Max Algorithm |  |  |

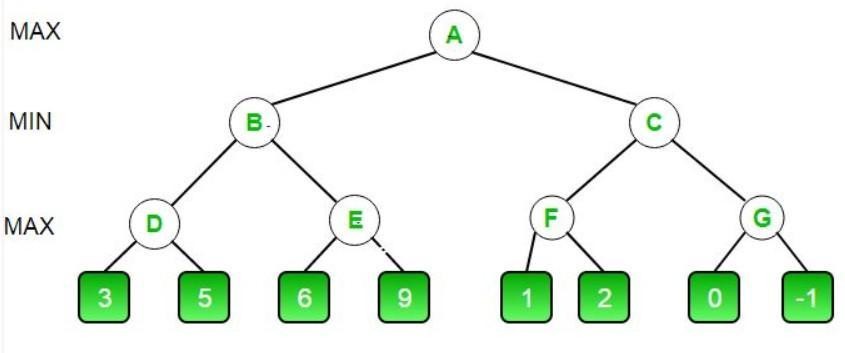
## Experiment No: 6

**Date : 22-02-2022**

**MINIMAX ALGORITHM**

**AIM :** Developing a mini max algorithm for real world problems.

**PROBLEM :** Find the optimal value in the given tree of integer values in the most optimal way possible under the time complexity O(B^D).



## ALGORITHM MINIMAX APPROACH :

1. Start traversing the given tree in top to bottom manner.
2. If node is a leaf node then return the value of the node.
3. If isMaximizingPlayer exist then bestVal = -INFINITY
4. For each child node, value = minimax(node, depth+1, false, alpha, beta)
5. bestVal = max( bestVal, value) and alpha = max( alpha, bestVal)
6. If beta <= alpha then stop traversing and return bestVal
7. Else, bestVal = +INFINITY
8. For each child node, value = minimax(node, depth+1, true, alpha, beta)
9. bestVal = min( bestVal, value) and beta = min( beta, bestVal)
10. if beta <= alpha the stop traversing and return bestVal

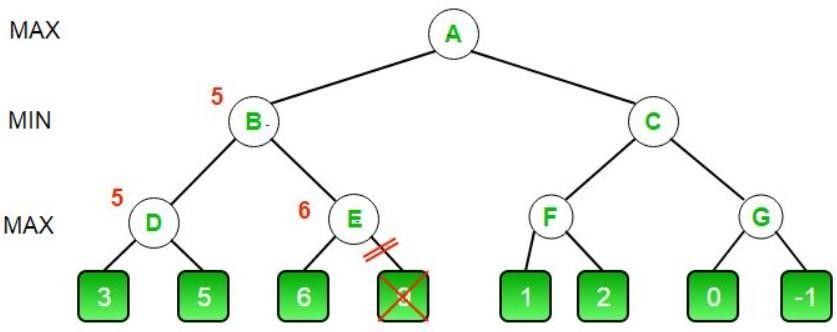
## OPTIMIZATION TECHNIQUE :

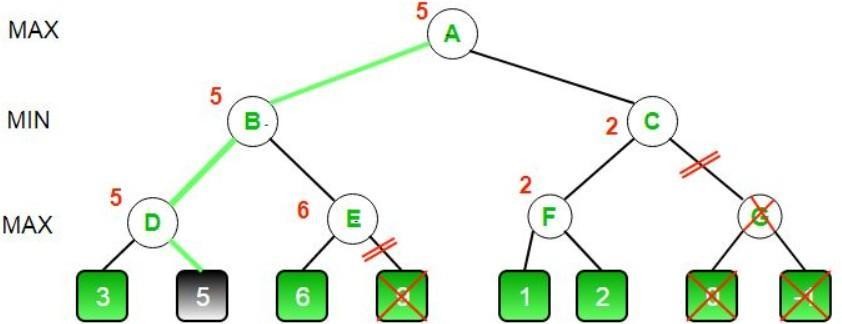
Alpha-Beta pruning is not actually a new algorithm, rather an optimization technique for minimax algorithms. It reduces the computation time by a huge factor. This allows us to search much faster and even go into deeper levels in the game tree. It cuts off branches in the game tree which need not be searched because there already exists a better move available. It is called Alpha-Beta pruning because it passes 2 extra parameters in the minimax function, namely alpha and beta.

Let’s define the parameters alpha and beta.

**Alpha** is the best value that the **maximizer** currently can guarantee at that level or above.

**Beta** is the best value that the **minimizer** currently can guarantee at that level or above.





## CODE (MINIMAX ALGORITHM) :

MAX, MIN = 1000, -1000

def minimax(depth, nodeIndex, maximizingPlayer,

values, alpha, beta):

if depth == 3:

return values[nodeIndex]

if maximizingPlayer:

best = MIN

for i in range(0, 2):

val = minimax(depth + 1, nodeIndex \* 2 + i,

False, values, alpha, beta)

best = max(best, val)

alpha = max(alpha, best)

if beta <= alpha:

break

return best

else:

best = MAX

for i in range(0, 2):

val = minimax(depth + 1, nodeIndex \* 2 + i,

True, values, alpha, beta)

best = min(best, val)

beta = min(beta, best)

if beta <= alpha:

break

return best

if \_\_name\_\_ == "\_\_main\_\_":

values = []

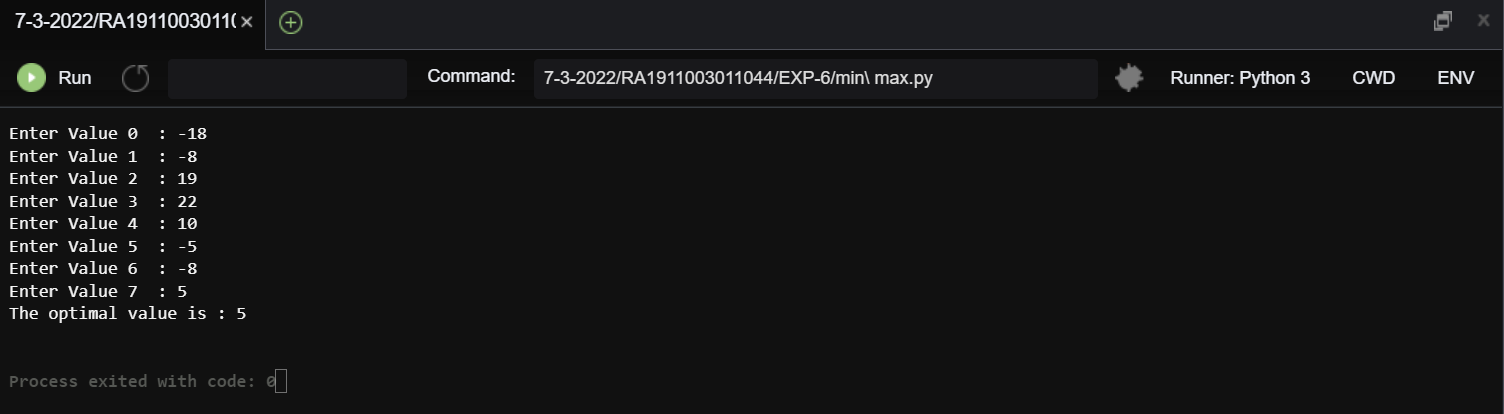
for i in range(0, 8):

x = int(input(f"Enter Value {i} : "))

values.append(x)

print ("The optimal value is :", minimax(0, 0, True, values, MIN, MAX))

## OUTPUT :

****

**RESULT :** The Optimal value of the given tree successfully found using Minimax Algorithm with Alpha Beta Pruning in time complexity O(B^D).