VIVEK BURUGUDDA RA1911003010316

HARSHIL AGARWAL RA1911003010325

RITESH RAI RA1911003010333

PUNEET SHARMA RA1911003010331

Prakhar Vijay RA1911003010337

**18CSC305J - ARTIFICIAL INTELLIGENCE**

**Week 6: Developing Alpha-Beta pruning using Min-Max Algorithm**

Aim: Developing Alpha-Beta pruning using Min-Max Algorithm

**Min-Max Algorithm:**

**Algorithm:**

Let's define steps of the algorithm in general:

1. Construct the complete game tree
2. Evaluate scores for leaves using the evaluation function
3. Back-up scores from leaves to root, considering the player type:
4. For max player, select the child with the maximum score
5. For min player, select the child with the minimum score
6. At the root node, choose the node with max value and perform the corresponding move

**Alpha-Beta pruning:**

**Algorithm:**

* Start DFS traversal from the root of game tree
* Set initial values of alpha and beta as follows:

a. alpha = INT\_MIN(-INFINITY)

b. beta = INT\_MAX(+INFINITY)

* Traverse tree in DFS fashion where maximizer player tries to get the highest score possible while the minimizer player tries to get the lowest score possible.
* While traversing update the alpha and beta values accordingly

**Code:** C++ code for Alpha-Beta pruning is as follows:

#include <iostream>

#include <algorithm>

#include <cmath>

#include <climits>

#define SIZE(arr) (sizeof(arr) / sizeof(arr[0]))

using namespace std;

int getHeight(int n) {

return (n == 1) ? 0 : 1 + log2(n / 2);

}

int minmax(int height, int depth, int nodeIndex,

bool maxPayer, int values[], int alpha,

int beta) {

if (depth == height) {

return values[nodeIndex];

}

if (maxPayer) {

int bestValue = INT\_MIN;

for (int i = 0; i < height - 1; i++) {

int val = minmax(height, depth + 1, nodeIndex \* 2 + i, false, values, alpha, beta);

bestValue = max(bestValue, val);

alpha = max(alpha, bestValue);

if (beta <= alpha)

break;

}

return bestValue;

}

else {

int bestValue = INT\_MAX;

for (int i = 0; i < height - 1; i++) {

int val = minmax(height, depth + 1, nodeIndex \* 2 + i, true, values, alpha, beta);

bestValue = min(bestValue, val);

beta = min(beta, bestValue);

if (beta <= alpha)

break;

}

return bestValue;

}

}

int main() {

int values[] = {13, 8, 24, -5, 23, 15, -14, -20};

int height = getHeight(SIZE(values));

int result = minmax(height, 0, 0, true, values, INT\_MIN, INT\_MAX);

cout <<"Result : " << result << "\n";

return 0;

}

**Output:**

