Minimax algortihm

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. It is widely used in two player turn-based games such as Tic-Tac-Toe, Backgammon, Mancala, Chess, etc.

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.

Every board state has a value associated with it. In a given state if the maximizer has upper hand then, the score of the board will tend to be some positive value. If the minimizer has the upper hand in that board state then it will tend to be some negative value. The values of the board are calculated by some heuristics which are unique for every type of game.

Propesed alogrithm

```
#include <bits/stdc++.h>
using namespace std;
int player_turn(int a[7][8], int depth);
int computer_turn(int a[7][8], int depth);
void print(int a[7][8])
{
	char ch;
	cout << endl;
	for (int i = 1; i <= 6; ++i)
	{
		cout << " ";
		for (int j = 1; j <= 7; ++j)
		{
			if (a[i][j] == 0)
				ch = ' ';
			if (a[i][j] == 1)
				ch = 'O';
			if (a[i][j] == 2)
				ch = 'X';
			cout << " | " << ch;
		}
		cout << " | " << endl;
	}
	cout << " -----------------------------\n";
	cout << "   1  2  3  4  5  6  7\n";
}
int check(int a[7][8])
{
	for (int i = 1; i <= 6; ++i)
	{
		for (int j = 1; j <= 4; ++j)
		{
			if (a[i][j] == a[i][j + 1] && a[i][j + 1] == a[i][j + 2] && a[i][j] == a[i][j + 3] && a[i][j] != 0)
				return 1;
		}
	}

	for (int j = 1; j <= 7; ++j)
	{
		for (int i = 1; i <= 3; ++i)
		{
			if (a[i][j] == a[i + 1][j] && a[i][j] == a[i + 2][j] && a[i][j] == a[i + 3][j] && a[i][j] != 0)
				return 1;
		}
	}
	for (int i = 6; i >= 4; --i)
	{
```

```
		for (int j = 1; j <= 4; ++j)
		{
			if (a[i][j] == a[i - 1][j + 1] && a[i][j] == a[i - 2][j + 2] && a[i][j] == a[i - 3][j + 3] && a[i][j] != 0)
				return 1;
		}
	}
	for (int i = 6; i >= 4; --i)
	{
		for (int j = 7; j >= 4; --j)
		{
			if (a[i][j] == a[i - 1][j - 1] && a[i][j] == a[i - 2][j - 2] && a[i][j] == a[i - 3][j - 3] && a[i][j] != 0)
				return 1;
		}
	}
	return 0;
}
int score(int a[7][8])
{
	for (int i = 1; i <= 6; ++i)
	{
		for (int j = 1; j <= 4; ++j)
		{
			if (a[i][j] == a[i][j + 1] && a[i][j + 1] == a[i][j + 2] && a[i][j] == a[i][j + 3] && a[i][j] != 0 && a[i][j]
== 2)
				return 5000;
		}
	}

	for (int j = 1; j <= 7; ++j)
	{
		for (int i = 1; i <= 3; ++i)
		{
			if (a[i][j] == a[i + 1][j] && a[i][j] == a[i + 2][j] && a[i][j] == a[i + 3][j] && a[i][j] != 0 && a[i][j] ==
2)
				return 5000;
		}
	}
	for (int i = 6; i >= 4; --i)
	{
		for (int j = 1; j <= 4; ++j)
		{
			if (a[i][j] == a[i - 1][j + 1] && a[i][j] == a[i - 2][j + 2] && a[i][j] == a[i - 3][j + 3] && a[i][j] != 0 &&
a[i][j] == 2)
				return 5000;
		}
	}
	for (int i = 6; i >= 4; --i)
	{
		for (int j = 7; j >= 4; --j)
		{
			if (a[i][j] == a[i - 1][j - 1] && a[i][j] == a[i - 2][j - 2] && a[i][j] == a[i - 3][j - 3] && a[i][j] != 0 &&
a[i][j] == 2)
				return 5000;
		}
```

```
	}

	// PLAYER LOSING

	for (int i = 1; i <= 6; ++i)
	{
		for (int j = 1; j <= 4; ++j)
		{
			if (a[i][j] == a[i][j + 1] && a[i][j + 1] == a[i][j + 2] && a[i][j] == a[i][j + 3] && a[i][j] != 0 && a[i][j]
== 1)
				return -5000;
		}
	}

	for (int j = 1; j <= 7; ++j)
	{
		for (int i = 1; i <= 3; ++i)
		{
			if (a[i][j] == a[i + 1][j] && a[i][j] == a[i + 2][j] && a[i][j] == a[i + 3][j] && a[i][j] != 0 && a[i][j] ==
1)
				return -5000;
		}
	}
	for (int i = 6; i >= 4; --i)
	{
		for (int j = 1; j <= 4; ++j)
		{
			if (a[i][j] == a[i - 1][j + 1] && a[i][j] == a[i - 2][j + 2] && a[i][j] == a[i - 3][j + 3] && a[i][j] != 0 &&
a[i][j] == 1)
				return -5000;
		}
	}
	for (int i = 6; i >= 4; --i)
	{
		for (int j = 7; j >= 4; --j)
		{
			if (a[i][j] == a[i - 1][j - 1] && a[i][j] == a[i - 2][j - 2] && a[i][j] == a[i - 3][j - 3] && a[i][j] != 0 &&
a[i][j] == 1)
				return -5000;
		}
	}
	// Checking for three in a row

	for (int i = 1; i <= 6; ++i)
	{
		for (int j = 1; j <= 4; ++j)
		{
			if (a[i][j] == a[i][j + 1] && a[i][j + 1] == a[i][j + 2] && a[i][j] != 0 && a[i][j] == 2)
			{
				if (a[i][j - 1] == 0 || a[i][j + 3] == 0)
					return 1200;
				else
					return 1000;
			}
```

```
		}
}

for (int j = 1; j <= 7; ++j)
{
		for (int i = 1; i <= 3; ++i)
		{
			if (a[i][j] == a[i + 1][j] && a[i][j] == a[i + 2][j] && a[i][j] != 0 && a[i][j] == 2)
			{
				if (a[i - 1][j] == 0 || a[i + 3][j] == 0)
					return 1200;
				else
					return 1000;
			}
		}
}
for (int i = 6; i >= 4; --i)
{
		for (int j = 1; j <= 4; ++j)
		{
			if (a[i][j] == a[i - 1][j + 1] && a[i][j] == a[i - 2][j + 2] && a[i][j] != 0 && a[i][j] == 2)
			{
				if (a[i - 3][j + 3] == 0 || a[i + 1][j - 1] == 0)
					return 1200;
				else
					return 1000;
			}
		}
}
for (int i = 6; i >= 4; --i)
{
		for (int j = 7; j >= 4; --j)
		{
			if (a[i][j] == a[i - 1][j - 1] && a[i][j] == a[i - 2][j - 2] && a[i][j] != 0 && a[i][j] == 2)
			{
				if (a[i + 1][j + 1] == 0 || a[i - 3][j - 3] == 0)
					return 1200;
				else
					return 1000;
			}
		}
}

// Blocking 3 in a row

for (int i = 1; i <= 6; ++i)
{
		for (int j = 1; j <= 4; ++j)
		{
			if (a[i][j] == a[i][j + 1] && a[i][j + 1] == a[i][j + 2] && a[i][j] != 0 && a[i][j] == 1)
			{
				if (a[i][j - 1] == 0 || a[i][j + 3] == 0)
					return -1200;
				else
```

```
                                        return -1000;
                        }
                }
        }

        for (int j = 1; j <= 7; ++j)
        {
                for (int i = 1; i <= 3; ++i)
                {
                        if (a[i][j] == a[i + 1][j] && a[i][j] == a[i + 2][j] && a[i][j] != 0 && a[i][j] == 1)
                        {
                                if (a[i - 1][j] == 0 || a[i + 3][j] == 0)
                                        return -1200;
                                else
                                        return -1000;
                        }
                }
        }
        for (int i = 6; i >= 4; --i)
        {
                for (int j = 1; j <= 4; ++j)
                {
                        if (a[i][j] == a[i - 1][j + 1] && a[i][j] == a[i - 2][j + 2] && a[i][j] != 0 && a[i][j] == 1)
                        {
                                if (a[i + 1][j - 1] == 0 || a[i - 3][j + 3] == 0)
                                        return -1200;
                                else
                                        return -1000;
                        }
                }
        }
        for (int i = 6; i >= 4; --i)
        {
                for (int j = 7; j >= 4; --j)
                {
                        if (a[i][j] == a[i - 1][j - 1] && a[i][j] == a[i - 2][j - 2] && a[i][j] != 0 && a[i][j] == 1)
                        {
                                if (a[i + 1][j + 1] == 0 || a[i - 3][j - 3] == 0)
                                        return -1200;
                                else
                                        return -1000;
                        }
                }
        }

        // Making 2 in a row

        for (int i = 1; i <= 6; ++i)
        {
                for (int j = 1; j <= 4; ++j)
                {
                        if (a[i][j] == a[i][j + 1] && a[i][j] != 0 && a[i][j] == 2)
                                return 250;
                }
```

```
}

for (int j = 1; j <= 7; ++j)
{
	for (int i = 1; i <= 3; ++i)
	{
		if (a[i][j] == a[i + 1][j] && a[i][j] != 0 && a[i][j] == 2)
			return 250;
	}
}
for (int i = 6; i >= 4; --i)
{
	for (int j = 1; j <= 4; ++j)
	{
		if (a[i][j] == a[i - 1][j + 1] && a[i][j] != 0 && a[i][j] == 2)
			return 250;
	}
}
for (int i = 6; i >= 4; --i)
{
	for (int j = 7; j >= 4; --j)
	{
		if (a[i][j] == a[i - 1][j - 1] && a[i][j] != 0 && a[i][j] == 2)
			return 250;
	}
}

// Blocking two in a row

for (int i = 1; i <= 6; ++i)
{
	for (int j = 1; j <= 4; ++j)
	{
		if (a[i][j] == a[i][j + 1] && a[i][j] != 0 && a[i][j] == 1)
			return -250;
	}
}

for (int j = 1; j <= 7; ++j)
{
	for (int i = 1; i <= 3; ++i)
	{
		if (a[i][j] == a[i + 1][j] && a[i][j] != 0 && a[i][j] == 1)
			return -250;
	}
}
for (int i = 6; i >= 4; --i)
{
	for (int j = 1; j <= 4; ++j)
	{
		if (a[i][j] == a[i - 1][j + 1] && a[i][j] != 0 && a[i][j] == 1)
			return -250;
	}
}
```

```
	for (int i = 6; i >= 4; --i)
	{
		for (int j = 7; j >= 4; --j)
		{
			if (a[i][j] == a[i - 1][j - 1] && a[i][j] != 0 && a[i][j] == 1)
				return -250;
		}
	}

	return 0;
}
void copy_array(int a[7][8], int temp[7][8])
{
	for (int i = 1; i <= 6; ++i)
	{
		for (int j = 1; j <= 7; ++j)
		{
			temp[i][j] = a[i][j];
		}
	}
}
void add(int a[7][8], int col, int turn)
{ // cout<<"Here"<<col;
	for (int i = 6; i >= 1; --i)
	{ // cout<<a[i][col];
		if (a[i][col] == 0)
		{ // cout<<i<<" "<<col;
			if (turn % 2 == 0)
			{
				a[i][col] = 1;
				break;
			}
			else
			{
				a[i][col] = 2;
				break;
			}
		}
	}
}
int computer_choice(int a[7][8], int depth)
{
	int temp[7][8], maximum = -100000000;
	int count = 0;
	int pointer = 100;

	for (int i = 1; i <= 7; ++i)
	{
		if (a[1][i] == 0)
		{
			copy_array(a, temp);
			add(temp, i, 1);
			if (player_turn(temp, depth - 1) + score(temp) >= maximum)
			{
```

```
                                        if (player_turn(temp, depth - 1) + score(temp) == maximum)
                                        {
                                                if (rand() % 2 == 1)
                                                        pointer = i;
                                        }
                                        else
                                        {
                                                pointer = i;
                                                maximum = player_turn(temp, depth - 1) + score(temp);
                                        }
                                }
                        }
                }
                return pointer;
}
int player_turn(int a[7][8], int depth)
{

                int minimum = 1000000;
                if (depth == 0)
                        return score(a);
                if (check(a) == 1)
                        return score(a) + depth * 250;
                // if(score(a) != 0)
                //      return score(a);
                int temp[7][8];
                int count = 0;
                for (int i = 1; i <= 7; ++i)
                {
                        if (a[1][i] == 0)
                        {
                                copy_array(a, temp);
                                add(temp, i, 2);
                                // print(temp);//cout<<computer_turn(temp,depth-1);
                                if (computer_turn(temp, depth - 1) < minimum)
                                {
                                        minimum = computer_turn(temp, depth - 1);
                                }
                        }
                }
                // cout<<minimum;
                return minimum;
}

int computer_turn(int a[7][8], int depth)
{
                int temp[7][8], maximum = -100000;
                int count = 0;
                if (depth == 0)
                {
                        return score(a);
                }
                if (check(a) == 1)
```

```
                return score(a) - depth * 250;
        // if(score(a) != 0)
        //      return score(a);
        for (int i = 1; i <= 7; ++i)
        {
                if (a[1][i] == 0)
                {
                        copy_array(a, temp);
                        add(temp, i, 1);
                        // print(temp);
                        if (player_turn(temp, depth - 1) > maximum)
                        {
                                maximum = player_turn(temp, depth - 1);
                        }
                }
        }
        return maximum;
}

void disp(int a[7][8])
{
        for (int i = 1; i <= 6; ++i)
        {
                for (int j = 1; j <= 7; ++j)
                {
                        cout << a[i][j] << " ";
                }
                cout << endl;
        }
}

int main()
{
        int a[7][8] = {0}, level;
        int flag = 0, col;
        string player1, player2;
        cout << "Enter player's name: ";
        cin >> player1;
        cout << "Enter level:";
        cout << "\n1.Easy";
        cout << "\n2.Medium";
        cout << "\n3.Hard\n";
        cin >> level;
        level += 4;
        int turn = 0;
        int total = 42;
        print(a);
        while (turn < total)
        {

                if (turn % 2 == 0)
                {
                        cout << "\n"
                                << player1 << "'s turn: \n            ";
```

```
                cin >> col;
                if (col < 1 || col > 7 || a[1][col] != 0)
                {
                        cout << "Wrong input try again! \n";
                        continue;
                }
                add(a, col, turn);
                // disp(a);
        }
        else
        {
                cout << "\n \n "
                         << " "
                         << "wait.."
                         << "\n \n";
                int point;
                point = computer_choice(a, level);
                if (point != 100)
                        add(a, point, 1);
                else
                {
                        cout << "Game draw \n";
                        return 0;
                }
        }
        print(a);
        if (check(a) == 1)
        {
                if (turn % 2 == 0)
                {
                        cout << player1 << " wins\n";
                        flag = 1;
                        break;
                }
                else
                {
                        cout << "Computer wins\n";
                        flag = 1;
                        break;
                }
        }

        turn++;
    }
    if (flag == 0)
        cout << "Its a draw";
    return 0;
}
```

PROPOSED OUTPUT

Enter player's name: a
Enter level:
1.Easy
2.Medium
3.Hard
3

```
|  |  |  |  |  |  |  |
 |  |  |  |  |  |  |  |
 |  |  |  |  |  |  |  |
 |  |  |  |  |  |  |  |
 |  |  |  |  |  |  |  |
 |  |  |  |  |  |  |  |
---------------------------
  1  2  3  4  5  6  7
```

a's turn:
1

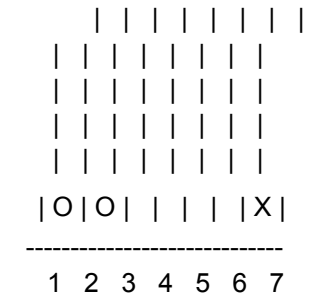
```
      |  |  |  |  |  |  |  |
 |  |  |  |  |  |  |  |
 |  |  |  |  |  |  |  |
 |  |  |  |  |  |  |  |
 |  |  |  |  |  |  |  |
|O |  |  |  |  |  |  |
---------------------------
  1  2  3  4  5  6  7
```

wait..

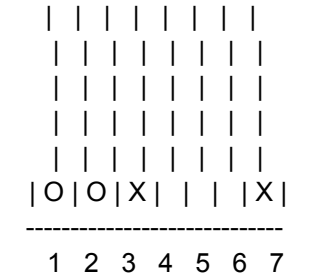
```
|  |  |  |  |  |  |  |
 |  |  |  |  |  |  |  |
 |  |  |  |  |  |  |  |
 |  |  |  |  |  |  |  |
 |  |  |  |  |  |  |  |
|O |  |  |  |  |  |X |
---------------------------
  1  2  3  4  5  6  7
```

a's turn:

2



wait..



a's turn:

Conclusion

With the basis of minimax algorithm for mathematical analysis alongside speeding up the computation by alpha beta pruning concept and optimizing the utility function using heuristic function, the 4x4 tic tac toe game was developed. We explored that a 4x4 tic tac toe, an adversary search technique game in artificial intelligence,can be developed using these techniques. Increasing the size of this game would create a huge time complexity issue with the same algorithm and techniques, for which other logics must be further researche