Experiment Number: 01

**Problem Statement: Tic-tac-toe AI and NON-AI approach.**

**NAME: Priti Patil ROLLNO: 80**

# CLASS: TY(IT) BATCH: 3

**DATE OF PERFORMANCE: 9/ 10 / 2022**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**1.Non-AI Approach-**

**Code-**

import java.util.\*;

import java.lang.Math;

public class tictactoe {

public static void main(String[] args) {

int[] arr= {1,0,0,2,0,0,1,1,2};

System.out.println("Board : ");

System.out.println(Arrays.toString(arr));

Scanner sc=new Scanner(System.in);

int XCount = 0;

int YCount = 0;

for(int i=0;i<9;i++)

{

if(arr[i]==1)

{

XCount++;

}

else if(arr[i]==2)

{

YCount++;

}

}

if (XCount < YCount || XCount - YCount > 1) {

System.out.println("This is invalid");

}

else {

System.out.println("This is valid");

System.out.println( "Decimal Index is : " +decimalvalue(arr) );

}

int ZCount = 9 - ( XCount + YCount );

int[][] matrix=new int[ZCount][9];

for(int i=0;i<ZCount;i++)

{

for(int j=0;j<9;j++)

{

matrix[i][j]=arr[i];

}

}

}

static int evaluateMove(int[] arr)

{

//if there are 2 elements in one row and one is blank then and if that is the turn of that player then it's an winning pos

if(arr[0]==1 && arr[1]==1 && arr[2]==1 || (arr[0]==0 || arr[1]==1) )

{

System.out.println("X is winner");

}

return 0;

}

static int decimalvalue(int[] arr)

{

int dec=0;

for (int i=0; i<9; i++)

{

dec = dec + (arr[i] \* (int) Math.pow(3, 8-i));

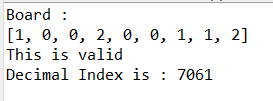
}

return dec;

}

}

**Output-**

****

**2.AI Approach-**

**Code-**

import java.util.\*;

public class tictacAi {

public static void main(String[] args) {

// TODO Auto-generated method stub

Scanner sc=new Scanner(System.in);

int arr[]={2, 1, 2, 1, 1, 2, 0, 0, 0};

int count=0, turn=1, countZ=0, countOne=0, countTwo=0, diff, ret=-1;

boolean x;

for(int i=0;i<arr.length;i++)

{

if(arr[i]==0)

{

count++;

}

}

int weight[][]=new int[count][9];

for(int i=0;i<9;i++)

{

if(arr[i]==1)

{

countOne++;

}

else if(arr[i]==2)

{

countTwo++;

}

else{

countZ++;

}

}

diff=Math.abs(countOne-countTwo);

if(diff==1 || diff==0)

{

ret=valid(weight, arr, turn);

if(ret==1)

{

return;

}

else if (ret==0)

{

if(turn==1)

{

ret=isItValid(weight, arr, 2);

}

else if(turn==2)

{

ret=isItValid(weight, arr, 1);

}

if(ret==-1)

{

validStep3(weight, arr, turn);

}

else {

System.out.println("\n\nRow " + ret);

int c=0, posn=-1;

for(int z=0;z<arr.length;z++)

{

if(arr[z]==0)

{

c++;

}

if(c==ret+1)

{

posn=z;

break;

}

}

System.out.println("Position to block for turn other than " + turn + " is " + posn);

arr[posn]=turn;

System.out.println("Therefore the position after turn " + turn + " has played after blocking should be");

printArray(arr);

}

}

else if(ret==-1)

{

validStep3(weight, arr, turn);

}

}

else{

System.out.println("Invalid input");

}

}

public static int valid(int weight[][], int arr[], int turn)

{

int pos=0, x=-1;

//this is to fill the nx9 matrix with our original array values in each row..

for(int i=0;i<weight.length;i++)

{

for(int j=0;j<9;j++)

{

weight[i][j]=arr[j];

}

}

//fill in all the 0 positions with the turn value and find the win position/row

for(int i=0;i<weight.length;i++)

{

for(int j=0;j<9;j++)

{

if(weight[i][j]==0 && j>=pos)//\*\*w[i][pos]

{

weight[i][j]=turn;//\*\*w[i][pos]=turn;

pos=j;

break;

}

}

pos=pos+1;

}

print(weight);

x=maxWin(weight, turn);

return x;

}

public static int isItValid(int w[][], int arr[], int turn)

{

int pos=0, x=-1;

//this is to fill the nx9 matrix with our original array values in each row

for(int i=0;i<w.length;i++)

{

for(int j=0;j<9;j++)

{

w[i][j]=arr[j];

}

}

//fill in all the 0 positions with the turn value and find the win position/row

for(int i=0;i<w.length;i++)

{

for(int j=0;j<9;j++)

{

if(w[i][j]==0 && j>=pos)//\*\*w[i][pos]

{

w[i][j]=turn;//\*\*w[i][pos]=turn;

pos=j;

break;

}

}

pos=pos+1;

}

print(w);

x=maxWinBlock(w, turn);

return x;

}

public static void validStep3(int weight[][], int arr[], int turn)

{

int pos=0, x=-1;

//this is to fill the n by 9 matrix positions with our original array values in each row

for(int i=0;i<weight.length;i++)

{

for(int j=0;j<9;j++)

{

weight[i][j]=arr[j];

}

}

System.out.println("We have " + weight.length + " moves to fill");

//fill in all the 0 positions with the turn value and find the win position/row

for(int i=0;i<weight.length;i++)

{

for(int j=0;j<9;j++)

{

if(weight[i][j]==0 && j>=pos)//\*\*w[i][pos]

{

System.out.println("Position of the move is " + j + " in the " + i + "th row");

weight[i][j]=turn;//\*\*w[i][pos]=turn;

pos=j;

break;

}

}

pos=pos+1;

}

System.out.println();

print(weight);

// x=maxWin(w, turn);

// return x;

if(arr[4]==0)

{

System.out.println("Score for the move is 4..which is the maximum");

arr[4]=turn;

}

for(int i=0;i<arr.length;i++)

{

if(arr[i]==0)

{

if(i==4 || i==0 || i==2 || i==6 || i==8)

{

System.out.println("Score for the move is 3");

System.out.println("Choosing this position as its score is 2nd highest and pos=4/score=4 not possible");

arr[i]=turn;

break;

}

else if(i==1 || i==3 || i==5 || i==7)

{

System.out.println("Score for the move is " + i);

arr[i]=turn;

break;

}

}

}

printArray(arr);

//return 1;

}

public static void print(int a[][])

{

for(int i=0;i<a.length;i++)

{

System.out.println();

System.out.println("Step " +(i+1));

System.out.println();

for(int j=0;j<9;j++)

{

System.out.print(a[i][j]+ " " );

}

System.out.println();

}

}

public static int maxWin(int a[][], int turn)

{

int win=0,x=-1;

int row[]=new int[9];

//int mat[][]=new int[3][3];

for(int i=0;i<a.length;i++)

{

x=i;

for(int j=0;j<9;j++)

{

row[j]=a[i][j];

}

win=checkWin(row, turn);

if(win==1)

{

System.out.println("\nWinning position found in the row " + x + " for turn " + turn);

printArray(row);

return 1;

//break;

}

}

System.out.println("\nNo winning position found for turn " + turn + "\n");

return 0;

}

public static int maxWinBlock(int a[][], int turn)

{

int win=0,x=-1;

int row[]=new int[9];

//int mat[][]=new int[3][3];

for(int i=0;i<a.length;i++)

{

x=i;

for(int j=0;j<9;j++)

{

row[j]=a[i][j];

}

win=checkWin(row, turn);

if(win==1)

{

System.out.println("\nWinning position found in the row " + x + " for turn " + turn);

printArray(row);

return x;

//break;

}

}

//System.out.println("\nNo winning position found for turn " + turn + "\n");

System.out.println("\nNothing to block\n");

return -1;

}

public static int checkWin(int board[], int turn)

{

int win=0;

if (

((board[0] == board[1]) && (board[0]==board[2]) && (board[0]==turn)) ||

((board[3] == board[4]) && (board[3]==board[5]) && (board[3]==turn)) ||

((board[6] == board[7]) && (board[6]==board[8]) && (board[6]==turn)) ||

((board[0] == board[3]) && (board[0]==board[6]) && (board[0]==turn)) ||

((board[1] == board[4]) && (board[1]==board[7]) && (board[1]==turn)) ||

((board[2] == board[5]) && (board[2]==board[8]) && (board[2]==turn)) ||

((board[0] == board[4]) && (board[0]==board[8]) && (board[0]==turn)) ||

((board[2] == board[4]) && (board[2]==board[6]) && (board[2]==turn))

)

{

win=1;

}

return win;

}

public static void printArray(int a[])

{

for(int i=0;i<a.length;i++)

{

System.out.print(a[i] + " ");

}

}

public static int[] printSingleArrayBlock(int a[])

{

for(int i=0;i<a.length;i++)

{

System.out.print(a[i] + " ");

}

return a;

}

public static int block(int turn)

{

if(turn==1)

{

turn=2;

}

else if(turn==2)

{

turn=1;

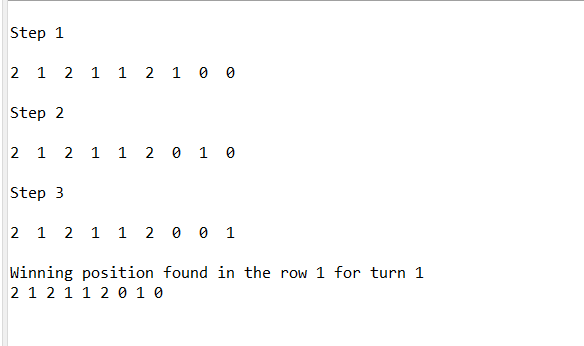
}

return turn;

}

}

**Output-**

****