**AI Lab – 1**

**Implement AI vs Non-AI Technique**

Name: Mokshit Oswal

Class: AI-B

Roll Number – 32

PRN: 12111304

Batch B2

1. **Non-AI Technique:**

#include <stdio.h>

char gridChar(int i) {

switch(i) {

case -1:

return 'X';

case 0:

return ' ';

case 1:

return 'O';

}

}

void draw(int b[9]) {

printf(" %c | %c | %c\n",gridChar(b[0]),gridChar(b[1]),gridChar(b[2]));

printf("---+---+---\n");

printf(" %c | %c | %c\n",gridChar(b[3]),gridChar(b[4]),gridChar(b[5]));

printf("---+---+---\n");

printf(" %c | %c | %c\n",gridChar(b[6]),gridChar(b[7]),gridChar(b[8]));

}

int win(const int board[9]) {

//determines if a player has won, returns 0 otherwise.

unsigned wins[8][3] = {{0,1,2},{3,4,5},{6,7,8},{0,3,6},{1,4,7},{2,5,8},{0,4,8},{2,4,6}};

int i;

for(i = 0; i < 8; ++i) {

if(board[wins[i][0]] != 0 &&

board[wins[i][0]] == board[wins[i][1]] &&

board[wins[i][0]] == board[wins[i][2]])

return board[wins[i][2]];

}

return 0;

}

int minimax(int board[9], int player) {

//How is the position like for player (their turn) on board?

int winner = win(board);

if(winner != 0) return winner\*player;

int move = -1;

int score = -2;//Losing moves are preferred to no move

int i;

for(i = 0; i < 9; ++i) {//For all moves,

if(board[i] == 0) {//If legal,

board[i] = player;//Try the move

int thisScore = -minimax(board, player\*-1);

if(thisScore > score) {

score = thisScore;

move = i;

}//Pick the one that's worst for the opponent

board[i] = 0;//Reset board after try

}

}

if(move == -1) return 0;

return score;

}

void computerMove(int board[9]) {

int move = -1;

int score = -2;

int i;

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

if(board[i] == 0) {

board[i] = 1;

int tempScore = -minimax(board, -1);

board[i] = 0;

if(tempScore > score) {

score = tempScore;

move = i;

}

}

}

//returns a score based on minimax tree at a given node.

board[move] = 1;

}

void playerMove(int board[9]) {

int move = 0;

do {

printf("\nInput move ([0..8]): ");

scanf("%d", &move);

printf("\n");

} while (move >= 9 || move < 0 && board[move] == 0);

board[move] = -1;

}

int main() {

int board[9] = {0,0,0,0,0,0,0,0,0};

//computer squares are 1, player squares are -1.

printf("Computer: O, You: X\nPlay (1)st or (2)nd? ");

int player=0;

scanf("%d",&player);

printf("\n");

unsigned turn;

for(turn = 0; turn < 9 && win(board) == 0; ++turn) {

if((turn+player) % 2 == 0)

computerMove(board);

else {

draw(board);

playerMove(board);

}

}

switch(win(board)) {

case 0:

printf("A draw. How droll.\n");

break;

case 1:

draw(board);

printf("You lose.\n");

break;

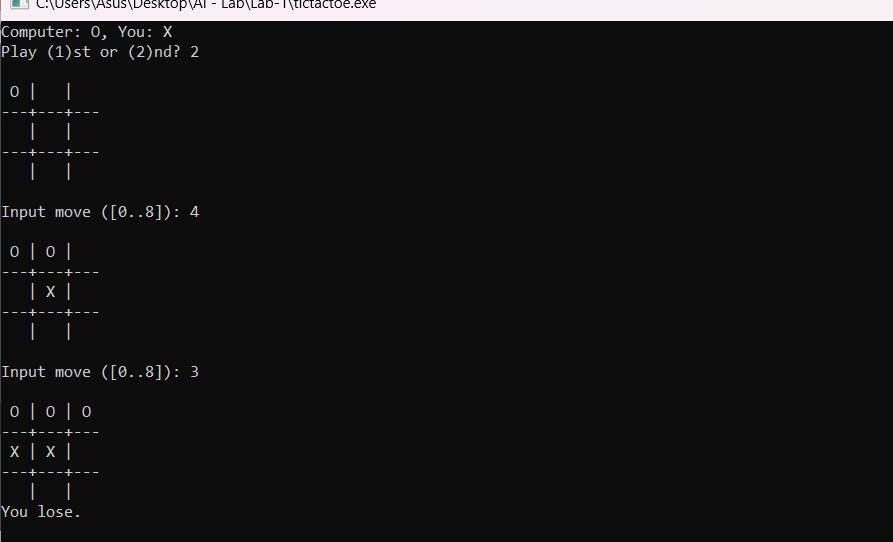
case -1:

printf("You win. Inconceivable!\n");

break;

}

}

  
  
  
**AI Technique:**

#include <stdio.h>

int flag[] = {0,0};

int indexm[][3] = {{9,9,9},{9,9,9},{9,9,9}};

int cost\_mat[][3] = {{0,0,0},{0,0,0},{0,0,0}};

int path\_matrix[8][3];

int flag\_draw = 0;

int matrix\_drawing\_flag = 0;

int flag\_extreme = 0;

int path\_cost=0, min\_path\_cost, max\_path\_cost;

int min\_path\_number, max\_path\_number;

int i,j,k;

int person\_chance=0;

int position;

int blankColumn, blankRow, row, column;

int flagBlank = 0;

void chancePerson()

{

person\_chance = 1;

place(position);

}

void place(int pos)

{

if(pos>9 || pos<1)

{

printf("Position out of bound... please put position b/w 1-9\n");

person\_chance = 0;

return;

}

column = (pos - 1) % 3;

row = (pos - 1) / 3;

int pos\_value = indexm[row][column];

if(pos\_value != 9|| pos\_value == 0 || pos\_value == 1)

{

printf("Sorry ... You can't place the X here !\n");

person\_chance = 0;

}

else

{

indexm[row][column] = 1;

// chance[1] = 0;

}

}

int isWinningAI() //AI is winning

{

if(indexm[0][0]+indexm[1][0]+indexm[2][0] == 0 || indexm[0][1]+indexm[1][1]+indexm[2][1] == 0 || indexm[0][2]+indexm[1][2]+indexm[2][2] == 0)

{

//return 1;

flag[0] = 1;

}

if(indexm[0][0]+indexm[0][1]+indexm[0][2] == 0 || indexm[1][0]+indexm[1][1]+indexm[1][2] == 0 || indexm[2][0]+indexm[2][1]+indexm[2][2] == 0)

{

//return 1;

flag[0] = 1;

}

if(indexm[0][0]+indexm[1][1]+indexm[2][2] == 0 || indexm[0][2]+indexm[1][1]+indexm[2][0] == 0)

{

//return 1;

flag[0] = 1;

}

if(flag[0])return 1;

else return 0;

}

int isWinningPerson()

{

if(indexm[0][0]+indexm[1][0]+indexm[2][0] == 3 || indexm[0][1]+indexm[1][1]+indexm[2][1] == 3 || indexm[0][2]+indexm[1][2]+indexm[2][2] == 3)

{

//return 1;

flag[1] = 1;

}

if(indexm[0][0]+indexm[0][1]+indexm[0][2] == 3 || indexm[1][0]+indexm[1][1]+indexm[1][2] == 3 || indexm[2][0]+indexm[2][1]+indexm[2][2] == 3)

{

//return 1;

flag[1] = 1;

}

if(indexm[0][0]+indexm[1][1]+indexm[2][2] == 3 || indexm[0][2]+indexm[1][1]+indexm[2][0] == 3)

{

//return 1;

flag[1] = 1;

}

if(flag[1])return 1;

else return 0;

}

int isDraw()

{

if(!isWinningAI() || !isWinningPerson())

{

for(int i = 0; i < 3; i++)

{

for(int j = 0; j < 3; j++)

{

if(indexm[i][j] == 9)

{

flag\_draw = 0;

break;

}

else

{

flag\_draw = 1;

}

}

if(!flag\_draw)

{

break;

}

}

}

if(flag\_draw)

{

return 1;

}

else

{

return 0;

}

}

void goAI()

{

for(i=0; i<3; i++)

{

for(j=0; j<3; j++)

{

path\_cost += cost\_mat[i][j];

if(indexm[i][j] == 9 && flagBlank != 1)

{

blankRow = i;

blankColumn = j;

flagBlank = 1;

}

}

path\_matrix[k][0] = path\_cost;

path\_matrix[k][1] = blankRow;

path\_matrix[k][2] = blankColumn;

//printf("Path cost at path %d is %d\n", k, path\_cost);

flagBlank = 0;

path\_cost = 0;

k++;

}

for(j=0; j<3; j++)

{

for(i=0; i<3; i++)

{

path\_cost += cost\_mat[i][j];

if(indexm[i][j] == 9 && flagBlank != 1)

{

blankRow = i;

blankColumn = j;

flagBlank = 1;

}

}

path\_matrix[k][0] = path\_cost;

path\_matrix[k][1] = blankRow;

path\_matrix[k][2] = blankColumn;

// printf("Path cost at path %d is %d\n", k, path\_cost);

flagBlank = 0;

path\_cost = 0;

k++;

}

for(i=0; i<3; i++)

{

path\_cost += cost\_mat[i][i];

if(indexm[i][i] == 9 && flagBlank != 1)

{

blankRow = blankColumn = i;

flagBlank = 1;

}

path\_matrix[k][0] = path\_cost;

path\_matrix[k][1] = blankRow;

path\_matrix[k][2] = blankColumn;

}

//printf("Path cost at path %d is %d\n", k, path\_cost);

flagBlank = 0;

path\_cost = 0;

k++;

for(i=0; i<3; i++)

{

path\_cost += cost\_mat[2-i][i];

if(indexm[2-i][i] == 9 && flagBlank != 1)

{

blankRow = 2-i;

blankColumn = i;

flagBlank = 1;

}

path\_matrix[k][0] = path\_cost;

path\_matrix[k][1] = blankRow;

path\_matrix[k][2] = blankColumn;

}

//printf("Path cost at path %d is %d\n", k, path\_cost);

flagBlank = 0;

path\_cost = 0;

k=0;

// FOR LOOP FOR MINIMUM COST AND BLANK SPOTS AT THAT PATH

for(k=0; k<8; k++)

{

if(k==0)

{

min\_path\_cost = path\_matrix[k][0];

}

if(path\_matrix[k][0] <= min\_path\_cost)

{

min\_path\_cost = path\_matrix[k][0];

min\_path\_number = k;

}

}

// FOR LOOP FOR MAXIMUM COST AND BLANK SPOTS AT THAT PATH

for(k=0; k<8; k++)

{

if(k==0)

{

max\_path\_cost = path\_matrix[k][0];

}

if(path\_matrix[k][0] >= max\_path\_cost) //greater than or equal to just becuase k is still 0 in first iteration

{

max\_path\_cost = path\_matrix[k][0];

max\_path\_number = k;

}

}

k=0;

//printf("Minimum path number = %d, Maximum path number = %d, Minimum path cost = %d, Maximum path cost = %d\n\n", min\_path\_number, max\_path\_number, min\_path\_cost, max\_path\_cost);

//printf("Min Blank row = %d, Min blank column = %d, Max blank row = %d, Max blank row = %d\n", path\_matrix[min\_path\_number][1], path\_matrix[min\_path\_number][2], path\_matrix[max\_path\_number][1], path\_matrix[max\_path\_number][2]);

if(!flag\_extreme)

{

if(min\_path\_cost + max\_path\_cost == 0)

{

indexm[path\_matrix[min\_path\_number][1]][path\_matrix[min\_path\_number][2]] = 0;

}

if(max\_path\_cost + min\_path\_cost > 0)

{

indexm[path\_matrix[max\_path\_number][1]][path\_matrix[max\_path\_number][2]] = 0;

}

if(min\_path\_cost + max\_path\_cost < 0)

{

indexm[path\_matrix[min\_path\_number][1]][path\_matrix[min\_path\_number][2]] = 0;

}

}

flag\_extreme = 0;

}

void costMatrix()

{

for(i=0; i<3; i++)

{

for(j=0; j<3; j++)

{

// If triangulation has already happened

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

{

flag\_extreme = 0;

}

// Defying advantage of corner

else if(((i==0 && j==0) || (i==0 && j==2) || (i==2 && j==0) || (i==2 && j==2))&& indexm[1][1] == 9)

{

flag\_extreme = 1;

indexm[1][1] = 0;

}

// Avoiding triangulation

else if((indexm[0][0]==1 && indexm[2][2]==1) && indexm[0][1] == 9)

{

flag\_extreme = 1;

indexm[0][1] = 0;

}

//Avoiding triangulatoin

else if((indexm[0][2]==1 && indexm[2][0]==1) && indexm[0][1] == 9)

{

flag\_extreme = 1;

indexm[0][1] = 0;

}

// Cost assignment to each cell

if(indexm[i][j]==0)

{

cost\_mat[i][j] = -1;

}

if(indexm[i][j]==1)

{

cost\_mat[i][j] = 1;

}

else if(indexm[i][j]==9)

{

cost\_mat[i][j] = 0;

}

}

}

}

void drawMatrix()

{

if(matrix\_drawing\_flag)

{

for(int i = 0; i < 3; i++)

{

printf("|");

for(int j = 0; j < 3; j++)

{

if(indexm[i][j] == 0)

{

printf(" O |");

}

else if(indexm[i][j] == 1)

{

printf(" X |");

}

else

{

printf(" |");

}

}

printf("\n\n");

}

}

else

{

int pos\_counter = 1;

for(int i = 0; i < 3; i++)

{

printf("|");

for(int j = 0; j < 3; j++)

{

printf(" %d |", pos\_counter);

pos\_counter++;

}

printf("\n\n");

}

}

}

void main()

{

printf("WELCOME TO TIC TAC TOE, YOU ARE PLAYING AGAINST A VIRTUAL AI ENABLED AGENT 'BATMAN'\n\n");

printf("BATMAN DARES YOU TO WIN !\n\n");

drawMatrix();

while(!isWinningAI() || !isWinningAI() || !isDraw)

{

matrix\_drawing\_flag = 1;

while(person\_chance != 1)

{

printf("PERSON.... please enter the position where you want to place X: - ");

scanf("%d", &position);

chancePerson();

}

drawMatrix();

person\_chance = 0;

if(isWinningAI())

{

printf("THE AI HAS WON !!!!!!!!!!!!!!!!!!!!!!!!!!!\n");

break;

}

if(isWinningPerson())

{

printf("THE PERSON HAS WON !!!!!!!!!!!!!!!!!!!!!!!!!!!\n");

break;

}

if(isDraw())

{

printf("THE MATCH IS DRAW !!!!!!!!!!!!!!!!!!!!!!!!!!!\n");

break;

}

printf("Now its chance for the AI\n");

costMatrix();

k=0;

goAI();

drawMatrix();

if(isWinningAI())

{

printf("THE AI HAS WON !!!!!!!!!!!!!!!!!!!!!!!!!!!\n");

break;

}

if(isWinningPerson())

{

printf("THE PERSON HAS WON !!!!!!!!!!!!!!!!!!!!!!!!!!!\n");

break;

}

if(isDraw())

{

printf("THE MATCH IS DRAW !!!!!!!!!!!!!!!!!!!!!!!!!!!\n");

break;

}

}

