VIT Vidyalankar Institute of Technology Accredited A+ by NAAC	**DEPARTMENT OF COMPUTER ENGINEERING**

Experiment No. 4B

Semester	T.E. Semester VI
Subject	ARTIFICIAL INTELLIGENCE (CSL 604)
Subject Professor In-charge	Prof. Avinash Shrivas
Assisting Teachers	Prof. Avinash Shrivas

Student Name	Deep Salunkhe
Roll Number	21102A0014
Lab Number	310A

Title:

8 Puzzle Problem using Hill climbing

Theory:

Hill Climbing Algorithm:
Hill climbing is a local search algorithm used in optimization problems. It starts with an initial solution and iteratively moves to the neighboring solutions with higher values until it reaches a peak where no neighbor has a higher value. In the context of the 8-puzzle problem, hill climbing aims to find the optimal sequence of moves to reach the goal state by evaluating the heuristic value of each move.

Heuristic Function:
A heuristic function is used to estimate the cost or value of reaching the goal state from a given state in a search problem. In the 8-puzzle problem, the Manhattan distance heuristic is commonly used, which calculates the sum of the distances each tile is from its goal position. This heuristic provides an admissible estimate of the minimum number of moves required to reach the goal state.

Program Code:

```
#include<iostream>
#include<vector>
#include<math.h>
using namespace std;
```

```
void ISGS(vector<vector<int> >&IS,vector<vector<int> >&GS){
    cout<<"NOTE: To enter blank use 0"<<endl;
    cout<<"Pls enter the Goal state"<<endl;
    for(int i=0;i<3;i++){
        for(int j=0;j<3;j++){
            cin>>GS[i][j];
        }
    }

    cout<<"Pls enter the Initial state"<<endl;
    for(int i=0;i<3;i++){
        for(int j=0;j<3;j++){
            cin>>IS[i][j];
        }
    }

}

void display2d(vector<vector<int> >v){
    for(int i=0;i<3;i++){
        for(int j=0;j<3;j++){
            cout<<v[i][j]<<" ";
        }
        cout<<endl;
    }
}

bool goalReached(vector<vector<int> >&IS,vector<vector<int> >&GS){
    for(int i=0;i<3;i++){
        for(int j=0;j<3;j++){
          if(IS[i][j]!=GS[i][j])
          return false;
        }

    }

    return true;
}

void moveUp(vector<vector<int> >&IS,vector<vector<int> >&GS){

    int xb;
    int yb;

    for(int i=0;i<3;i++){
        for(int j=0;j<3;j++){
            if(IS[i][j]==0){
```

```
                xb=i;
                yb=j;
            }
        }
    }

    if(xb==0)
    cout<<"Cant go any up"<<endl;
    else{
        swap(IS[xb][yb],IS[xb-1][yb]);
    }

}

void moveRight(vector<vector<int> >&IS,vector<vector<int> >&GS){

    int xb;
    int yb;

    for(int i=0;i<3;i++){
        for(int j=0;j<3;j++){
            if(IS[i][j]==0){
                xb=i;
                yb=j;
            }
        }
    }

    if(yb==2)
    cout<<"Cant go any right"<<endl;
    else{
        swap(IS[xb][yb],IS[xb][yb+1]);
    }

}
void moveDown(vector<vector<int> >&IS,vector<vector<int> >&GS){

    int xb;
    int yb;

    for(int i=0;i<3;i++){
        for(int j=0;j<3;j++){
            if(IS[i][j]==0){
                xb=i;
                yb=j;
            }
        }
```

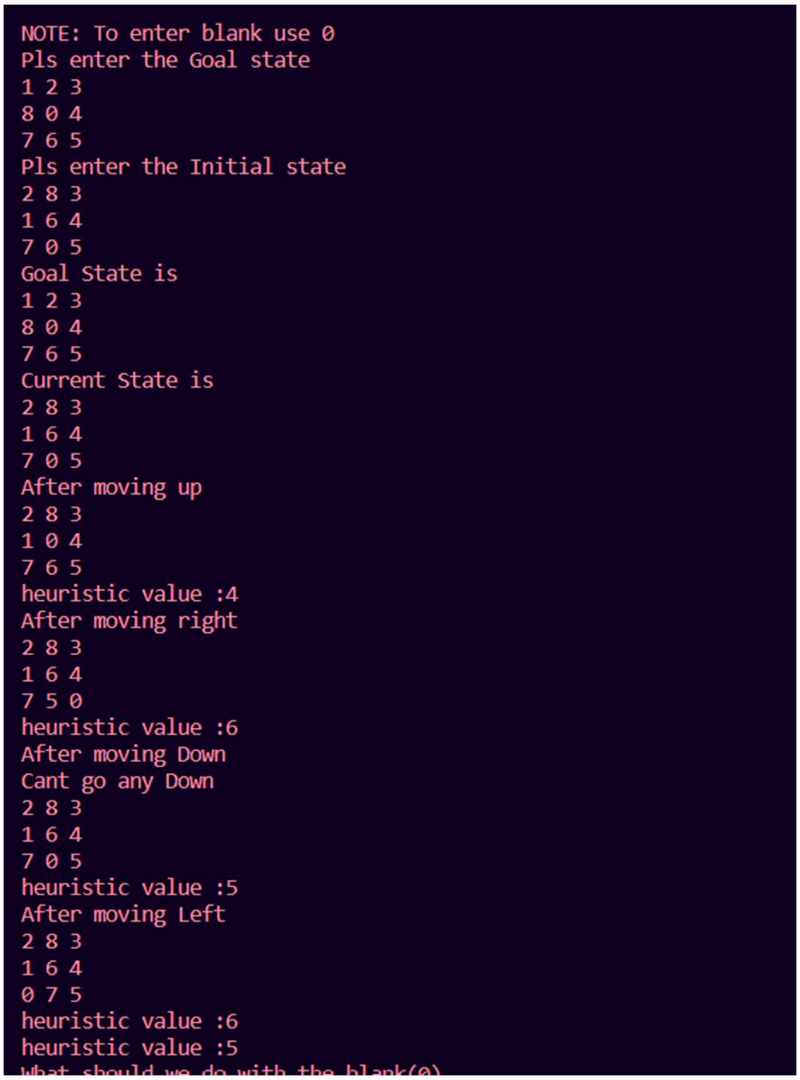
```
    }

    if(xb==2)
    cout<<"Cant go any Down"<<endl;
    else{
        swap(IS[xb][yb],IS[xb+1][yb]);
    }

}
void moveLeft(vector<vector<int> >&IS,vector<vector<int> >&GS){

    int xb;
    int yb;

    for(int i=0;i<3;i++){
        for(int j=0;j<3;j++){
            if(IS[i][j]==0){
                xb=i;
                yb=j;
            }
        }
    }

    if(yb==0)
    cout<<"Cant go any Left"<<endl;
    else{
        swap(IS[xb][yb],IS[xb][yb-1]);
    }

}

int heuristic(vector<vector<int> >&IS,vector<vector<int> >&GS){
    int value=0;
    for(int i=0;i<3;i++){
        for(int j=0;j<3;j++){
            int curr_ele=IS[i][j];
            int cex=i;
            int cey=j;
            int fx;
            int fy;
            if(curr_ele==0)
            continue;
            for(int x=0;x<3;x++){
                for(int y=0;y<3;y++){
                    if(GS[x][y]==curr_ele){
                        value=value+abs(cex-x)+abs(cey-y);
                    }
```

```
                }
            }
        }
    }

    return value;
}

void displaypossible(vector<vector<int> >&IS,vector<vector<int> >&GS){

    int xb;
    int yb;
    //complete fromhere
    int minhval=INT_MAX;

    for(int i=0;i<3;i++){
        for(int j=0;j<3;j++){
            if(IS[i][j]==0){
                xb=i;
                yb=j;
            }
        }
    }
    int nextx=xb;
    int nexty=yb;
    int dir[4][2]={{0,1},{1,0},{-1,0},{0,-1}};


    for(int i=0;i<4;i++){




                vector<vector<int> >temp(3,vector<int>(3,0));
                for(int p=0;p<3;p++){
                    for(int q=0;q<3;q++){
                        temp[p][q]=IS[p][q];
                    }
                }
                if(i==0){
                    cout<<"After moving up"<<endl;
                    moveUp(temp,GS);
                }
                 if(i==1){
                    cout<<"After moving right"<<endl;
                    moveRight(temp,GS);
                }
```

```
                if(i==2){
                    cout<<"After moving Down"<<endl;
                    moveDown(temp,GS);
                }
                if(i==3){
                    cout<<"After moving Left"<<endl;
                    moveLeft(temp,GS);
                }

                display2d(temp);

                int hvalue=heuristic(temp,GS);
                cout<<"heuristic value :"<<hvalue<<endl;









    }

}

int main(){
    vector<vector<int> >IS(3,vector<int>(3,0));
    vector<vector<int> >GS(3,vector<int>(3,0));
    ISGS(IS,GS);


    while(!goalReached(IS,GS)){
        cout<<"Goal State is"<<endl;
        display2d(GS);
        cout<<"Current State is"<<endl;
        display2d(IS);
        displaypossible(IS,GS);
        int hvalue=heuristic(IS,GS);
        cout<<"heuristic value :"<<hvalue<<endl;


        cout<<"What should we do with the blank(0)"<<endl;
        cout<<"1. Move UP"<<endl;
        cout<<"2. Move RIGHT"<<endl;
        cout<<"3. Move DOWN"<<endl;
        cout<<"4. Move LEFT"<<endl;

        int choice;
        cin>>choice;
```

```
        switch(choice){
            case 1: moveUp(IS,GS);
                    break;
            case 2: moveRight(IS,GS);
                    break;
            case 3: moveDown(IS,GS);
                    break;
            case 4: moveLeft(IS,GS);
                    break;
            default:cout<<"Invalid choice"<<endl;
        }

        if(!goalReached(IS,GS)){
            cout<<"Goal State is"<<endl;
            display2d(GS);
            cout<<"Current State is"<<endl;
            display2d(IS);
            int hvalue=heuristic(IS,GS);
            cout<<"heuristic value :"<<hvalue<<endl;
        }


    }

    cout<<"You have completed task"<<endl;


    return 0;
}
```

Output:



Conclusion:

In this lab exercise, we implemented a hill-climbing approach to solve the 8-puzzle problem. The problem involves finding the optimal sequence of moves to rearrange tiles in a 3x3 grid to reach a specified goal state from an initial state.

The hill-climbing algorithm starts with an initial state and iteratively evaluates neighboring states by applying possible moves (up, down, left, right). It selects the move that results in the highest heuristic value, i.e., the state closest to the goal state according to the heuristic function.

While hill climbing is a simple and intuitive approach, it is prone to getting stuck in local optima. In our implementation, this limitation was evident as the algorithm might get stuck in a state where no neighboring state has a higher heuristic value, leading to suboptimal solutions.

Despite its limitations, hill climbing provides a fundamental understanding of local search algorithms and serves as a starting point for exploring more sophisticated optimization techniques in artificial intelligence.

By implementing and experimenting with hill climbing in the context of the 8-puzzle problem, we gained insights into the challenges and trade-offs involved in heuristic search algorithms, paving the way for further exploration in the field of artificial intelligence.