**CS-304-AI -LAB(LAB TASK-4)**

**ROLL-423135**

**NAME-HARSHITH SUDA**

**CODE-1**

#include<bits/stdc++.h>

using namespace std;

vector<vector<int>>goal={{0,1,2},{3,4,5},{6,7,8}};

unordered\_map<int,pair<int,int>>loc{{0,{0,0}},{1,{0,1}},{2,{0,2}},{3,{1,0}},{4,{1,1}},{5,{1,2}},{6,{2,0}},{7,{2,1}},{8,{2,2}}};

void printVectorOfVector(vector<vector<int>>&v){

    for(int i=0;i<v.size();i++){

        for(int j=0;j<v[0].size();j++){

*// cout<<"value of i:"<<i<<"value of j:"<<j<<" ";*

            cout<<v[i][j]<<" ";

        }

        cout<<endl;

    }

    return;

}

bool isSolvable(vector<vector<int>>&puzzle){

    vector<int>v;

    for(int i=0;i<puzzle.size();i++){

        for(int j=0;j<puzzle[0].size();j++){

            v.push\_back(puzzle[i][j]);

        }

    }

    int invcnt=0;

    for(int i=0;i<8;i++){

        for(int j=i+1;j<9;j++){

            if(v[i] && v[j] && v[i]>v[j]){

                invcnt++;

            }

        }

    }

    return (invcnt%2)==0;

}

bool valid(int x,int y,int n){

    if(x>=0 && y>=0 && x<n && y<n){

        return true;

    }

    return false;

}

int costManhatten(vector<vector<int>>&puzzle){

    int cost=0;

    for(int i=0;i<3;i++){

        for(int j=0;j<3;j++){

            cost+=abs(i-loc[puzzle[i][j]].first)+abs(j-loc[puzzle[i][j]].second);

        }

    }

    return cost;

}

vector<int>X={-1,1,0,0};

vector<int>Y={0,0,1,-1};

void hillClimbing8Puzzle(vector<vector<int>>&puzzle,int x,int y){

    int bestCost=costManhatten(puzzle);

    vector<vector<int>>bestState=puzzle;

    vector<vector<int>>current=puzzle;

    for(int i=0;i<20000;i++){

        int z=rand()%4;

        int newX=x+X[z];

        int newY=y+Y[z];

        if(valid(newX,newY,3)){

            vector<vector<int>>newPuzzle=current;

            swap(newPuzzle[x][y],newPuzzle[newX][newY]);

            if(costManhatten(newPuzzle)<bestCost){

                bestCost=costManhatten(newPuzzle);

                x=newX;y=newY;

                bestState=newPuzzle;

                current=bestState;

            }

        }

    }

    printVectorOfVector(bestState);

    cout<<bestCost<<endl;

    return;

    }

int main(){

    ios\_base::sync\_with\_stdio(false);

    cin.tie(0);

    vector<vector<int>>puzzle={{1,4,2},{3,0,5},{6, 7, 8}};

    if(isSolvable(puzzle)){

        hillClimbing8Puzzle(puzzle,1,1);

    }

    else{

        printVectorOfVector(puzzle);

        cout<<"IMPOSSIBLE TO SOLVE";

    }

    return 0;

}

**CODE-2**

#include<bits/stdc++.h>

using namespace std;

int heuristic(vector<int>&queen){

    int cost=0;

    int n=queen.size();

    for(int i=0;i<n-1;i++){

        for(int j=i+1;j<n;j++){

            if(queen[i]==queen[j] || abs(i-j)==abs(queen[i]-queen[j])){

                cost++;

            }

        }

    }

    return cost;

}

vector<int> hillClimbing8Queens(vector<int>&queen){

    int bestCost=heuristic(queen);

    vector<int>bestBoard=queen;

    vector<int>current=queen;

    for(int i=0;i<550000;i++){

        int x=rand()%8;

        int y=rand()%8;

        int previousValueX=current[y];

        current[y]=x;

        if(heuristic(current)<bestCost){

            bestCost=heuristic(current);

            bestBoard=current;

            continue;

        }

        current[y]=previousValueX;

    }

    cout<<"COST:"<<heuristic(bestBoard)<<endl;

    return bestBoard;

}

void printVector(vector<int>&v){

    for(auto &x : v){

        cout<<x<<" ";

    }

    cout<<endl;

}

int main(){

    ios\_base::sync\_with\_stdio(false);

    cin.tie(0);

*// srand(time(NULL));*

    vector<int>result;

    vector<int>board(8);

    int ans=INT\_MAX;

    for(int i=0;i<8;i++){

        board[i]=rand()%8;

    }

    result=hillClimbing8Queens(board);

    printVector(result);

    return 0;

}