**Artificial Intelligence**

**DL2**

**Practical 3**

**19BCE248**

AIM: Program to implement hill climbing (for 8 queen problem)

Code:

import java.util.\*;

class Prac3{

public static void main(String[] args) {

int[][] dp;

int n=4;

dp=getRandom(n);

int curr\_conflicts=getCost(dp);

if (curr\_conflicts==0) {

System.out.println("N-Queen Problem Solved");

printSol(dp);

}

PriorityQueue<State> q=new PriorityQueue<>(new Comparator<State>(){

public int compare(State s1,State s2){

return s1.cost-s2.cost;

}

});

int[][] ans=new int[n][n];

q.add(new State(dp,curr\_conflicts));

outer:while (!q.isEmpty()) {

int z=q.size();

while (z-->0) {

State curr=q.poll();

int[][] currBoard=curr.curr;

int currCost=curr.cost;

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

for (int j=0;j<n;j++) {

ans[i][j]=currBoard[i][j];

}

}

if (currCost==0) {

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

for (int j=0;j<n;j++) {

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

}

System.out.println("");

}

}

for (int col=0;col<n;col++) {

for (int row=0;row<n;row++) {

int[][] next=Arrays.stream(currBoard).map(int[]::clone).toArray(int[][]::new);

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

next[col][i]=0;

}

next[col][row]=1;

int nextCost=getCost(next);

if (nextCost<currCost) {

q.clear();

q.add(new State(next,nextCost));

continue outer;

}

}

}

}

System.out.println("Reached To Final State");

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

for (int j=0;j<n;j++) {

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

}

System.out.println("");

}

}

}

public static void printSol(int[][] dp){

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

for (int j=0;j<dp.length;j++) {

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

}

System.out.println("");

}

}

public static int[][] getRandom(int n){

int[][] dp=new int[4][4];

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

int ind=(int)(Math.random()\*(4));

dp[i][ind]=1;

}

return dp;

}

public static int getCost(int[][] curr){

int n=curr.length;

ArrayList<Pair> qPos=new ArrayList<>();

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

for (int j=0;j<n;j++) {

if (curr[i][j]==1) {

qPos.add(new Pair(i,j));

}

}

}

int ct\_attacks=0;

for (int i=0;i<qPos.size();i++) {

for(int j=i+1;j<qPos.size();j++){

if (getConflict(qPos.get(i).x,qPos.get(j).x,qPos.get(i).y,qPos.get(j).y)) {

ct\_attacks++;

}

}

}

return ct\_attacks;

}

public static boolean getConflict(int x1,int x2,int y1,int y2){

if (x1==x2 || y1==y2) {

return true;

}

if (Math.abs(x1-x2)==Math.abs(y1-y2)) {

return true;

}

return false;

}

}

class Pair{

int x,y;

Pair(int x,int y){

this.x=x;

this.y=y;

}

}

class State{

int[][] curr;

int cost;

State(int[][] curr,int cost){

this.curr=curr;

this.cost=cost;

}

}

Output:

