

Mini- Project report on

**N-queens problem, Hot chilly game and Program for encryption and decryption of message**

Submitted by

**KISHORE S Ganesh Swaroop V Darshan C V**

**1CR16IS041 1CR16IS033 1CR16IS027**

**CONTENTS**

**CHAPTER 1 : THE N-QUEENS PROBLEM**

1.1 Introduction and Plan of solution

1.2 Algorithm

1.3 Implementation

**CHAPTER 2 : THE HOT CHILLI GAME**

2.1 Introduction and Plan of solution

2.2 Algorithm

2.3 Implementation

**CHAPTER 3 : ENCRYPTION AND DECRYPTION OF MESSAGE**

3.1 Introduction and Plan of solution

3.2 Algorithm

3.3 Implementation

3.4 Future Enhancements

**INTRODUCTION**

In Third semester we were very much interested in the Data Structures in which we dealt with lot of interesting problems. We were also hoping for a project then, since we were asked to indulge ourselves in any 3 projects in this semester , our interest in DAA made us select 3 problems provided by Megha Ma’am ,which became our 1st mini project of the semester and DAA . Now coming to the 3 problems, The first one is N-Queens wherein we could learn the principle of Back tracking and analysing the complexity of recursive algorithms .Our second problem was Data Encryption and Decryption, where we wanted to design our own cipher and learn from the problems we encountered in the designing stage . This also led us to ponder over the present cryptographic techniques used in the industries. Finally over third problem was Hot Chilly Game, where we had to find the logic using which a player would never lose.

**CHAPTER 1: THE N-QUEENS PROBLEM**

1.1 INTRODUCTION AND PLAN OF SOLUTION

PROBLEM STATEMENT : Given a chess board of 8\*8 (say) and 8 8 queens ,the task is to place all the queens on the board such that no queen should be able to kill another queen . But the catch is chess board is of size 'n'\*'n' and there are n queens . We probably can solve 8 queens problem but it would be difficult to solve for randomly given number. So We need an algorithm using which we can solve this problem with ease.

APPROACH

Though this problem can be solved using various methods Back Tracking is the most basic one. Backtracking is natural way in which puzzles and maze are solved. When we jump into a maze we move in a path and if it is found wrong we go back until there is an other way which we ignored or the only other way . In the same way in n queens problem we go on placing queens in each row in a safe place ,if we find that the next queen don't have a safe position to place then we Backtrack and replace the previously placed queen to next safe position and if such position is not found then we go back and replace one more queen which was placed before this. We can make use of stacks to store position of each queen in as soon as it is placed so that we can use it during backtracking

PRESENTATION

The n\*n chess board is represented using a matrix of order n\*n with zero(0)s and one(1)s where 1 representing queens ie. position where queens are placed is indicated with one and the remaining with zero.

1.2 ALGORITHM

NQUEEN( queen[1,2,....n][1,2,...n] ,)

Input : Order of Chess board - 'n'

output : n\*n matrix with n number of one(1)s representing queens

1. start

2. set row = column = n

3. while row <= n repeat

4. { set flag = 0;

5. for i from 1 to n repeat

6. { if CHECK(row,i) is true then

a. queue[row][i] = 1;

b. increment flag = 1 and row =row+1;

c. PUSH( i) to stack and break;

7. }

8. if flag != 0 then decrement row and

9. k = POP() REPLACE(row,k);

10. }

11. DISPLAY matrix;

12. end

REPLACE( row,column)

/\* here row and column is position of previously placed queen\*/

1. set queue[row][column] <= 0; //removing previous queen

2. set flag = 0;

3. for i from column+1 to n repeat

4. {

5. if CHECK(row,i) is true then

a. queue[row][i] = 1;

b. increment flag = 1 and row =row+1;

c. PUSH( i) to stack and break;

6. }

7. if flag != 0 then decrement row and

8. k = POP() REPLACE(row,k);

9. return ;

CHECK(row,column) //check if position is safe to place the queen

2. if row and column have a queen then return false

3. check for right upper diagonal has any queen if yes return false;

4. check for right lower diagonal has any queen if yes return false;

5. check for left upper diagonal has any queen if yes return false;

6. check for left lower diagonal has any queen if yes return false;

7. else return true //position is safe to place queen ;

1.3 IMPLEMENTATION

The algorithm is implemented using JAVA Programming language.

Following is the JAVA program for N-queens problem.

import java.util.Scanner;

import java.util.Stack;

public class Nqueen{

static int[][] qu = new int[50][50];

static Scanner ss=new Scanner(System.in);

static int n;

static int row=0;

static Stack<Integer> s = new Stack<Integer>();

public static void main(String[] args)

{

System.out.println("enter n value");

n = ss.nextInt();

nqueen();

for(int i=0;i<n;i++){for(int j=0;j<n;j++){ System.out.print(qu[i][j]+" ");}

System.out.print("\n");}

}

public static void nqueen(){

int flag;

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

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

qu[i][j]=0;

}

while(row<n)

{ flag=0;

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

if(check(row,k))

{

qu[row][k]=1;

flag++;row++;

s.push(k);

break;

}

}

if(flag==0) {

replace(--row,s.pop());}

}

return ;

}

public static void replace(int r,int c)

{

qu[r][c]=0;

int flag=0;

for(int i=c+1;i<n;i++){

if(check(r,i)==true){

qu[r][i]=1;

flag++; row++;

s.push(i);break;

}

}

if(flag==0) replace(--row,s.pop());

return;

}

public static boolean check(int r,int c)

{

int i=0,j=r,k=c,l=c;

for(i=0;i<n;i++){ if(qu[r][i]==1||qu[i][c]==1){ return false; }

}

while(j>=0){ if(k>=0 && qu[j][k--]==1) return false;

if(l<n && qu[j][l++]==1) return false;

j--;

}

j=r;k=c;l=c;

while(j<n){ if(k<n && qu[j][k++]==1) return false;

if(l>=0 && qu[j][l--]==1) return false;

j++;

}

return true;

}

}

Comment on complexity

Since n queens have function with recursive call, complexity cannot be calculated by number of loops it contains. The algorithm is non homogenous and intrceable for larger inputs. It contains two loops where inner loop runs for maximum of n times, but outer loop depends on row count which changes when replace function is called and is also recursive due to backtracking. Thus best case of this algorithm is n2, but this is very rare in this case. The recursion case runs in worst case T(n!). By analysis of Recursive algorithms, order of growth is calculated by solving recursive definition and found to be

n\*T(n2) as there are n\*n positions to be evaluated. It results in order of growth n2 + 2n where n2 is insignificant. Thus order of growth of this algorithm is 2n .

**CHAPTER 2 : THE HOT CHILLI GAME**

**2.1 INTRODUCTION AND PLAN OF SOLUTION**

PROBLEM STATEMENT

Given 'n' chocolates with 1 chilly , 2 players [ Computer vs User ] has to pick either 1 OR 2 OR 3 chocolates alternatively until all chocolates are taken.

The player who remains to pick the chilli loses the game .Objective is to make computer always win in any case.

Hot chilli is a game where in there are two players and a bowl of chocolates. And!!! There is a red hot chilli too. Each player has to pick either 1, 2 or at most 3 chocolates. As the game goes on whoever remain with the chilly must eat the chilly.

APPROACH

There is a logic using which one can always win this game which is designed and implemented as a computer game wherein user will play with the computer and computer always wins. Say there are ‘n’ chocolates and one chilli ,first the player must get to know the value ‘n’ and then start picking the chocolates first such that ‘n’ value reduces to multiple of 4. If ‘n’ is multiple of four then let other player to begin first. In the preceding rounds this player must pick the chocolates such that the total chocolates picked out of the bowl by both the players is equal to 4 . So when player makes ‘n’ value equal to multiple of 4,then in each round the player makes count to lower multiple of four finally zero making the other player pick the chilli.

Say for 13 chocolates 13= 1 + 12 , on removing one chocolate we have 4n number of chocolates

>< >< ><

>< + >< + >< + >< + ()

>< >< ><

>< >< ><

PRESENTATION

The game between computer and the user is presented as an interactive session where computer describes the rules of the game , replying for each input ,guiding for invalid choices and finally laugh at user informing about his defeat.

**2.2 ALGORITHM**

HotChilly( )

1. Start
2. READ() the number of chocolates
3. Set n=1
4. While 4n < chocolates repeat
   1. If chocolate mod 4 != 0 , then
      1. Set multi = 4n-4
      2. First = chocolate – multi

**CHAPTER 2 : ENCRYPTION AND DECRYPTION**

2.1 INTRODUCTION AND PLAN OF SOLUTION

PROBLEM STATEMENT: Given a sentence/message, the message should be encrypted at the transmitting end and decrypted at the receiving end.

Security is the major requirement in today’s trend. Everyone expect at least minimum security for exchange/storage of information. With this project we have attempted to encrypt and decrypt text messages or files strong enough to secure data from theft. This technique can be used efficiently to encrypt and store personal information which requires medium security in notepad or ‘notes’ like applications which do not have access to internet.

APPROACH

here we have used self designed cipher which includes one – one mapping of 10 alphabets and hash function for remaining 16 alphabets which makes our cipher difficult to decode with basic cryptographic knowledge. Since ‘e’ is the alphabet used most frequently in English so it is mapped to 2 values which makes the encoded message even more difficult to decode .

PRESENTATION

This program reads the message stored in a mentioned file encrypts it and stores in a new file. During decryption it reads the encrypted file and creates a new decoded file using C.