|  |
| --- |
| **Subject: DAA Class: S.E.(Comp)**    **Practical No.: 4 Date:** |

**AIM:** Implement back tracking by n-queen problem.

**Title:** Study and implement back tracking by n-queen problem.

**Theory:**

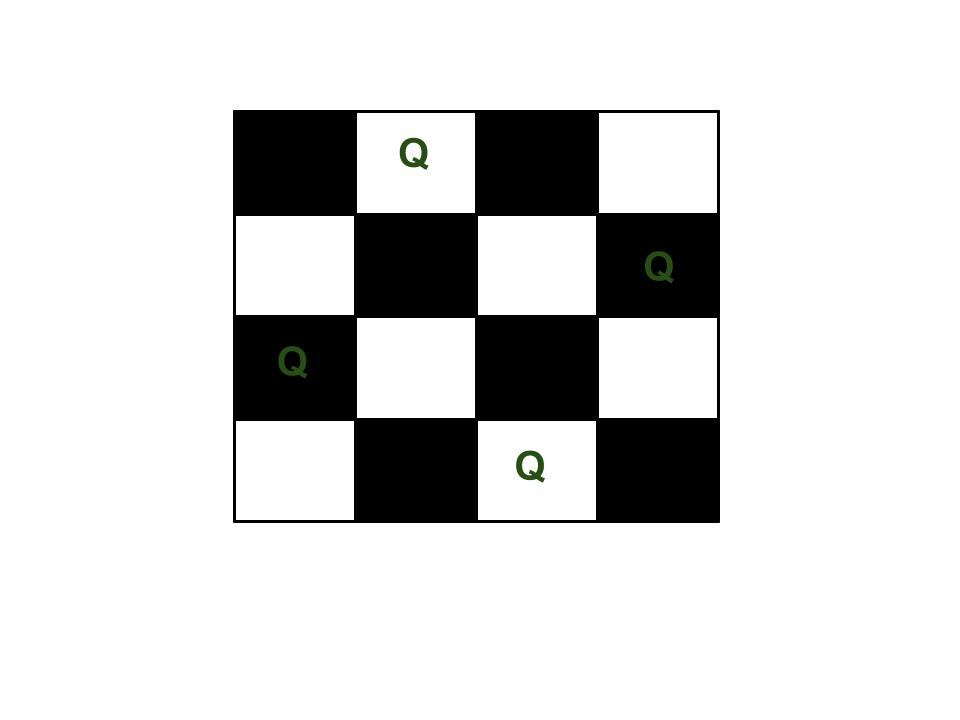
**Backtracking Algorithms:**

Backtracking is an algorithmic-technique for solving problems recursively by trying to build a solution incrementally, one piece at a time, removing those solutions that fail to satisfy the constraints of the problem at any point of time (by time, here, is referred to the time elapsed till reaching any level of the search tree).

**N Queen Problem:**

We have discussed Knight’s tour and Rat in a Maze problems in [Set 1](https://www.geeksforgeeks.org/backtracking-set-1-the-knights-tour-problem/) and [Set 2](https://www.geeksforgeeks.org/backttracking-set-2-rat-in-a-maze/) respectively. Let us discuss N Queen as another example problem that can be solved using Backtracking.

The N Queen is the problem of placing N chess queens on an N×N chessboard so that no two queens attack each other. For example, following is a solution for 4 Queen problem.



The expected output is a binary matrix which has 1s for the blocks where queens are placed. For example, following is the output matrix for above 4 queen solution.

{0 1 0 0}

{0 0 0 1}

{1 0 0 0}

{0 0 1 0}

**Algorithm:**

1. Place the queens column wise, start from the left most column
2. If all queens are placed.
   1. return true and print the solution matrix.
3. Else
   1. Try all the rows in the current column.
   2. Check if queen can be placed here safely if yes mark the current cell in solution matrix as 1 and try to solve the rest of the problem recursively.
   3. If placing the queen in above step leads to the solution return true.
   4. If placing the queen in above step does not lead to the solution , BACKTRACK, mark the current cell in solution matrix as 0 and return false.
4. If all the rows are tried and nothing worked, return false and print NO SOLUTION.

**Source Code:**

#include<stdio.h>

#include<conio.h>

#include<math.h>

int a[30],count=0;

int place(int pos) {

int i;

for (i=1;i<pos;i++) {

if((a[i]==a[pos])||((abs(a[i]-a[pos])==abs(i-pos))))

return 0;

}

return 1;

}

void print\_sol(int n) {

int i,j;

count++;

printf("\n\nSolution #%d:\n",count);

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

for (j=1;j<=n;j++) {

if(a[i]==j)

printf("Q\t"); else

printf("\*\t");

}

printf("\n");

}

}

void queen(int n) {

int k=1;

a[k]=0;

while(k!=0) {

a[k]=a[k]+1;

while((a[k]<=n)&&!place(k))

a[k]++;

if(a[k]<=n) {

if(k==n)

print\_sol(n); else {

k++;

a[k]=0;

}

} else

k--;

}

}

void main() {

int i,n;

clrscr();

printf("Enter the number of Queens:\n");

scanf("%d",&n);

queen(n);

printf("\nTotal solutions=%d",count);

getch();

}

**Output:**

