Batch: B1 **Roll No.: 1711072**

Experiment No. 8

Grade: AA / AB / BB / BC / CC / CD /DD

Signature of the Staff In-charge with date

Title: Implementation of Backtracking Algorithm

Objective: To learn the Backtracking strategy of problem solving for 8-Queens problem

CO to be achieved:

Sr. No	Objective
CO 1	Compare and demonstrate the efficiency of algorithms using asymptotic complexity notations.
CO 2	Analyze and solve problems for divide and conquer strategy, greedy method, dynamic programming approach and backtracking and branch & bound policies.
CO 3	Analyze and solve problems for different string matching algorithms.

Books/ Journals/ Websites referred:

1. **Ellis horowitz, Sarataj Sahni, S.Rajsekaran," Fundamentals of computer algorithm", University Press**
2. **T.H.Cormen ,C.E.Leiserson,R.L.Rivest and C.Stein," Introduction to algortihtms",2nd Edition ,MIT press/McGraw Hill,2001**
3. **http://www.math.utah.edu/~alfeld/queens/queens.html**
4. **http://www-isl.ece.arizona.edu/ece175/assignments275/assignment4a/Solving%208%20queen%20problem.pdf**
5. **http://www.slideshare.net/Tech_MX/8-queens-problem-using-back-tracking**
6. **http://www.mathcs.emory.edu/~cheung/Courses/170.2010/Syllabus/Backtracking/8queens.html**
7. **http://www.geeksforgeeks.org/backtracking-set-3-n-queen-problem/**

8. **http://www.hbmeyer.de/backtrack/achtdamen/eight.htm**

Pre Lab/ Prior Concepts:

Data structures, Concepts of algorithm analysis

Historical Profile:

The **N-Queens puzzle** is the problem of placing N queens on an N×N chessboard so that no two queens attack each other. Thus, a solution requires that no two queens share the same row, column, or diagonal.

New Concepts to be learned:

Application of algorithmic design strategy to any problem, Backtracking method of problem solving Vs other methods of problem solving,8- Queens problem and its applications.

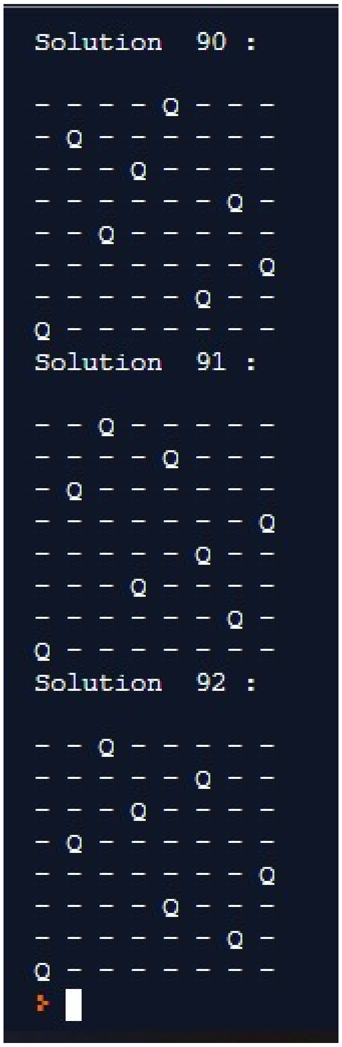
Algorithm N Queens Problem:-

```
void NQueens(int k, int n)
// Using backtracking, this procedure prints all possible placements of n queens on an n X n
chessboard so that they are nonattacking.
{       for (int i=1; i<=n; i++)
     {
            if (Place(k, i))
             {
               x[k] = i;
               if (k==n)
                      for (int j=1;j<=n;j++)          Print  x[j] ;
               else NQueens(k+1, n);
             }
        }
}

Boolean Place(int k, int i)
// Returns true if a queen can be placed in kth row and ith column.  Otherwise it returns false.
// x[] is a global array whose first (k-1) values have been set. abs(r) returns absolute value of r.
{
for (int j=1; j < k; j++)

        if ((x[j] == i)  // Two in the same column

      || (abs(x[j]-i) == abs(j-k)))                  // or in the same diagonal

          return(false);

return(true);

 }
```

Solution Using Backtracking Approach:

```
global N,k
k=0
N=int(input("Enter board size: "))
def printSolution(board):
  for i in range(N):
    for j in range(N):
      print ("Q" if board[i][j]==1 else "-", end=' ')
    print()
def isSafe(board, row, col):
  for i in range(col):  #checking if pos already occupied
    if board[row][i] == 1:
      return False
  #upper diagonal left side
  for i,j in zip(range(row,-1,-1), range(col,-1,-1)):
    if board[i][j] == 1:
      return False
  #lower diagonal left side
  for i,j in zip(range(row,N,1), range(col,-1,-1)):
    if board[i][j] == 1:
      return False
  return True
def PlaceQueens(board, col):
  global k
  if col == N:
    print('Solution ',k+1,':\n')
    k=k+1
    printSolution(board)
    return True
  for i in range(N):
    res=False
    if isSafe(board, i, col):
      board[i][col] = 1
      res = PlaceQueens(board, col + 1) or res
      board[i][col] = 0
  return res
def NQueens():
  board = [[0 for i in range(N)] for i in range(N)]
  if PlaceQueens(board, 0) == False and k==0:
    print("Solution does not exist")
    return False
  return True
NQueens()
```

Screenshot:



```
Solution  90 :

- - - - Q - - -
- Q - - - - - -
- - - Q - - - -
- - - - - - Q -
- - Q - - - - -
- - - - - - - Q
- - - - - Q - -
Q - - - - - - -
Solution  91 :

- - Q - - - - -
- - - - Q - - -
- Q - - - - - -
- - - - - - - Q
- - - - - Q - -
- - - Q - - - -
- - - - - - Q -
Q - - - - - - -
Solution  92 :

- - Q - - - - -
- - - - - Q - -
- - - Q - - - -
- Q - - - - - -
- - - - - - - Q
- - - - Q - - -
- - - - - - Q -
Q - - - - - - -
> 
```

For chess board size input of 8, we got 92 solutions.

Analysis of Backtracking solution for 8-Queens Problem:

Instead of generating one optimal solution, we are generating all possible and feasible solutions for a given chess board size. The time complexity of the program is:

$$\mathbf{T(n) = O(n^n)}$$

Where n is the size of the chessboard. The complexity is approximately n! for higher values of n.

CONCLUSION: The program for determining solution to N-Queen problem was solved using backtracking and all feasible solutions were found.