**K. J. Somaiya College of Engineering**

**Topic: Backtracking**

**Theory:** In many applications of the backtrack method, the desired solution is expressible as an n-tuple *(x1,...,Xn),* where the x*i* are chosen from some finite set Si. Often the problem to be solved calls for finding one vector that maximizes (or minimizes or satisfies) a *criterion function P(x1,…..* . , *xn). Sometime*s it seeks all vectors that satisfy *P.* For example, sorting the array of integers in. *a[1* : n] is a problem whose solution is expressible by an *n- tuple, w*here x*i* is the index in *a* of the ith smallest element. The criterion function P is the inequality *a[xi]* ≤ *a[xi+1]* for 1 ≤ i < *n.* The set *Si* is finite and includes the integers 1 through *n.* Though sorting is not usually one of the problems solved by backtracking, it is one example of a familiar problem whose solution can be formulated as an n-tuple.

**Control abstraction**:

void Backtrack( int k )

// This is a schema that describes the backtracking process //using recursion. On entering, the first k-1 values x[1], x[2], //…., x[k-1] of the solution vector x[1:n] have been //assigned. x[] and n are global.

{

for (each x[k] such that x[k] Є T(x[1], …, x[k-1])

{

if (Bk (x[1], x[2], …, x[k]))

{

if (x[1], x[2], …, x[k] is a path to an answer node)

output x[1:k];

if (k < n) Backtrack(k+1);

}

}

}



|  |
| --- |
| **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**](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**](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**](http://www.mathcs.emory.edu/~cheung/Courses/170.2010/Syllabus/Backtracking/8queens.html)
7. [**http://www.geeksforgeeks.org/backtracking-set-3-n-queen-problem/**](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);

}

**Example 8-Queens Problem:**

**Solution Using Backtracking Approach:**

**Always placing 1st queen on 1st position of 1st row**

|  |  |  |  |
| --- | --- | --- | --- |
| **Q** |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

**2ND queen cannot be on 1st row and 1st column and along the diagonal**

**Hence we have 2 safe positions ( 2 , 3 ) and ( 2 , 4 ). We’ll choose ( 2 , 3 )**

|  |  |  |  |
| --- | --- | --- | --- |
| **Q** |  |  |  |
|  |  |  |  |
|  | **Q** |  |  |
|  |  |  |  |

**3rd queen cannot be on 1st row and column and also on 3rd row and column and not diagonal to both queens.**

**Hence we have 0 psoitions on 3rd column which means that only 3 queens can be placed.**

**Hence backtrack to 2nd queen.**

**We will take 2nd queen to her last position ( 2 , 4 ) so safe position for 3rd queen**

**will be ( 3 , 2 )**

|  |  |  |  |
| --- | --- | --- | --- |
| **Q** |  |  |  |
|  |  | **Q** |  |
|  |  |  |  |
|  | **Q** |  |  |

**Now there will be no safe position for 4th queen.**

**Hence backtrack to queen 1 since 2nd and 3rd queen’s all positions are utilised.**

**We will take 1st queen to her next position ( 1 , 2 )**

|  |  |  |  |
| --- | --- | --- | --- |
|  |  |  |  |
| **Q** |  |  |  |
|  |  |  |  |
|  |  |  |  |

**Now 2nd queen has 1 safe position ( 2, 4 ) and**

**hence 3rd queen will also have one position ( 3 , 1)**

|  |  |  |  |
| --- | --- | --- | --- |
|  |  | **Q** |  |
| **Q** |  |  |  |
|  |  |  |  |
|  | **Q** |  |  |

**Now for the 4th queen only 3rd row is remaining hence ( 3 , 3 ) will be the only safe position .**

|  |  |  |  |
| --- | --- | --- | --- |
|  |  | **Q** |  |
| **Q** |  |  |  |
|  |  |  | **Q** |
|  | **Q** |  |  |

**Now the problem is solved since all queens are placed at safe positions**

**Analysis of Backtracking solution for 8-Queens Problem:**

TIME COMPLEXITY OF N-QUEEN PROBLEM IS GREATER THAN O(N!)

If we add all this up and define the run time as T(N). Then T(N) = O(N2) + N\*T(N-1). If you draw a recursion tree using this recurrence, the final term will be something like n3+ n!O(1). By the definition of Big O, this can be reduced to O(n!) running time.

**Implementation :**

**import java.util.\*;**

**public class Queens {**

**int[] x;**

**int count=0;**

**public Queens(int N) {**

**x = new int[N];**

**}**

**public boolean canPlaceQueen(int r, int c) {**

**for (int i = 0; i < r; i++) {**

**if (x[i] == c || (i - r) == (x[i] - c) ||(i - r) == (c - x[i]))**

**{**

**return false;**

**}**

**}**

**return true;**

**}**

**public void printQueens(int[] x) {**

**int N = x.length;**

**count=count+1;**

**for (int i = 0; i < N; i++) {**

**for (int j = 0; j < N; j++) {**

**if (x[i] == j) {**

**System.out.print("Q ");**

**} else {**

**System.out.print("\* ");**

**}**

**}**

**System.out.println();**

**}**

**System.out.println();**

**}**

**public void placeNqueens(int r, int n) {**

**for (int c = 0; c < n; c++) {**

**if (canPlaceQueen(r, c)) {**

**x[r] = c;**

**if (r == n - 1) {**

**printQueens(x);**

**} else {**

**placeNqueens(r + 1, n);**

**}**

**}**

**}**

**}**

**public static void main(String args[]) {**

**Scanner sc=new Scanner(System.in);**

**System.out.println("Enter number of queens");**

**Queens Q = new Queens(sc.nextInt());**

**Q.placeNqueens(0, Q.x.length);**

**System.out.println("total possiblities = "+Q.count);**

**}**

**}**

**Output:**

**/\***

**C:\Users\PRIYAM SHAH\Desktop\AOAD>javac Queens.java**

**C:\Users\PRIYAM SHAH\Desktop\AOAD>java Queens**

**Enter number of queens**

**8**

**Q \* \* \* \* \* \* \***

**\* \* \* \* Q \* \* \***

**\* \* \* \* \* \* \* Q**

**\* \* \* \* \* Q \* \***

**\* \* Q \* \* \* \* \***

**\* \* \* \* \* \* Q \***

**\* Q \* \* \* \* \* \***

**\* \* \* Q \* \* \* \***

**Q \* \* \* \* \* \* \***

**\* \* \* \* \* Q \* \***

**\* \* \* \* \* \* \* Q**

**\* \* Q \* \* \* \* \***

**\* \* \* \* \* \* Q \***

**\* \* \* Q \* \* \* \***

**\* Q \* \* \* \* \* \***

**\* \* \* \* Q \* \* \***

**Q \* \* \* \* \* \* \***

**\* \* \* \* \* \* Q \***

**\* \* \* Q \* \* \* \***

**\* \* \* \* \* Q \* \***

**\* \* \* \* \* \* \* Q**

**\* Q \* \* \* \* \* \***

**\* \* \* \* Q \* \* \***

**\* \* Q \* \* \* \* \***

**Q \* \* \* \* \* \* \***

**\* \* \* \* \* \* Q \***

**\* \* \* \* Q \* \* \***

**\* \* \* \* \* \* \* Q**

**\* Q \* \* \* \* \* \***

**\* \* \* Q \* \* \* \***

**\* \* \* \* \* Q \* \***

**\* \* Q \* \* \* \* \***

**\* Q \* \* \* \* \* \***

**\* \* \* Q \* \* \* \***

**\* \* \* \* \* Q \* \***

**\* \* \* \* \* \* \* Q**

**\* \* Q \* \* \* \* \***

**Q \* \* \* \* \* \* \***

**\* \* \* \* \* \* Q \***

**\* \* \* \* Q \* \* \***

**\* Q \* \* \* \* \* \***

**\* \* \* \* Q \* \* \***

**\* \* \* \* \* \* Q \***

**Q \* \* \* \* \* \* \***

**\* \* Q \* \* \* \* \***

**\* \* \* \* \* \* \* Q**

**\* \* \* \* \* Q \* \***

**\* \* \* Q \* \* \* \***

**\* Q \* \* \* \* \* \***

**\* \* \* \* Q \* \* \***

**\* \* \* \* \* \* Q \***

**\* \* \* Q \* \* \* \***

**Q \* \* \* \* \* \* \***

**\* \* \* \* \* \* \* Q**

**\* \* \* \* \* Q \* \***

**\* \* Q \* \* \* \* \***

**\* Q \* \* \* \* \* \***

**\* \* \* \* \* Q \* \***

**Q \* \* \* \* \* \* \***

**\* \* \* \* \* \* Q \***

**\* \* \* Q \* \* \* \***

**\* \* \* \* \* \* \* Q**

**\* \* Q \* \* \* \* \***

**\* \* \* \* Q \* \* \***

**\* Q \* \* \* \* \* \***

**\* \* \* \* \* Q \* \***

**\* \* \* \* \* \* \* Q**

**\* \* Q \* \* \* \* \***

**Q \* \* \* \* \* \* \***

**\* \* \* Q \* \* \* \***

**\* \* \* \* \* \* Q \***

**\* \* \* \* Q \* \* \***

**\* Q \* \* \* \* \* \***

**\* \* \* \* \* \* Q \***

**\* \* Q \* \* \* \* \***

**\* \* \* \* \* Q \* \***

**\* \* \* \* \* \* \* Q**

**\* \* \* \* Q \* \* \***

**Q \* \* \* \* \* \* \***

**\* \* \* Q \* \* \* \***

**\* Q \* \* \* \* \* \***

**\* \* \* \* \* \* Q \***

**\* \* \* \* Q \* \* \***

**\* \* \* \* \* \* \* Q**

**Q \* \* \* \* \* \* \***

**\* \* \* Q \* \* \* \***

**\* \* \* \* \* Q \* \***

**\* \* Q \* \* \* \* \***

**\* Q \* \* \* \* \* \***

**\* \* \* \* \* \* \* Q**

**\* \* \* \* \* Q \* \***

**Q \* \* \* \* \* \* \***

**\* \* Q \* \* \* \* \***

**\* \* \* \* Q \* \* \***

**\* \* \* \* \* \* Q \***

**\* \* \* Q \* \* \* \***

**\* \* Q \* \* \* \* \***

**Q \* \* \* \* \* \* \***

**\* \* \* \* \* \* Q \***

**\* \* \* \* Q \* \* \***

**\* \* \* \* \* \* \* Q**

**\* Q \* \* \* \* \* \***

**\* \* \* Q \* \* \* \***

**\* \* \* \* \* Q \* \***

**\* \* Q \* \* \* \* \***

**\* \* \* \* Q \* \* \***

**\* Q \* \* \* \* \* \***

**\* \* \* \* \* \* \* Q**

**Q \* \* \* \* \* \* \***

**\* \* \* \* \* \* Q \***

**\* \* \* Q \* \* \* \***

**\* \* \* \* \* Q \* \***

**\* \* Q \* \* \* \* \***

**\* \* \* \* Q \* \* \***

**\* Q \* \* \* \* \* \***

**\* \* \* \* \* \* \* Q**

**\* \* \* \* \* Q \* \***

**\* \* \* Q \* \* \* \***

**\* \* \* \* \* \* Q \***

**Q \* \* \* \* \* \* \***

**\* \* Q \* \* \* \* \***

**\* \* \* \* Q \* \* \***

**\* \* \* \* \* \* Q \***

**Q \* \* \* \* \* \* \***

**\* \* \* Q \* \* \* \***

**\* Q \* \* \* \* \* \***

**\* \* \* \* \* \* \* Q**

**\* \* \* \* \* Q \* \***

**\* \* Q \* \* \* \* \***

**\* \* \* \* Q \* \* \***

**\* \* \* \* \* \* \* Q**

**\* \* \* Q \* \* \* \***

**Q \* \* \* \* \* \* \***

**\* \* \* \* \* \* Q \***

**\* Q \* \* \* \* \* \***

**\* \* \* \* \* Q \* \***

**\* \* Q \* \* \* \* \***

**\* \* \* \* \* Q \* \***

**\* Q \* \* \* \* \* \***

**\* \* \* \* Q \* \* \***

**\* \* \* \* \* \* \* Q**

**Q \* \* \* \* \* \* \***

**\* \* \* \* \* \* Q \***

**\* \* \* Q \* \* \* \***

**\* \* Q \* \* \* \* \***

**\* \* \* \* \* Q \* \***

**\* Q \* \* \* \* \* \***

**\* \* \* \* \* \* Q \***

**Q \* \* \* \* \* \* \***

**\* \* \* Q \* \* \* \***

**\* \* \* \* \* \* \* Q**

**\* \* \* \* Q \* \* \***

**\* \* Q \* \* \* \* \***

**\* \* \* \* \* Q \* \***

**\* Q \* \* \* \* \* \***

**\* \* \* \* \* \* Q \***

**\* \* \* \* Q \* \* \***

**Q \* \* \* \* \* \* \***

**\* \* \* \* \* \* \* Q**

**\* \* \* Q \* \* \* \***

**\* \* Q \* \* \* \* \***

**\* \* \* \* \* Q \* \***

**\* \* \* Q \* \* \* \***

**Q \* \* \* \* \* \* \***

**\* \* \* \* \* \* \* Q**

**\* \* \* \* Q \* \* \***

**\* \* \* \* \* \* Q \***

**\* Q \* \* \* \* \* \***

**\* \* Q \* \* \* \* \***

**\* \* \* \* \* Q \* \***

**\* \* \* Q \* \* \* \***

**\* Q \* \* \* \* \* \***

**\* \* \* \* \* \* \* Q**

**\* \* \* \* Q \* \* \***

**\* \* \* \* \* \* Q \***

**Q \* \* \* \* \* \* \***

**\* \* Q \* \* \* \* \***

**\* \* \* \* \* Q \* \***

**\* \* \* \* \* \* \* Q**

**Q \* \* \* \* \* \* \***

**\* \* \* Q \* \* \* \***

**\* \* \* \* \* \* Q \***

**\* \* \* \* Q \* \* \***

**\* Q \* \* \* \* \* \***

**\* \* Q \* \* \* \* \***

**\* \* \* \* \* Q \* \***

**\* \* \* \* \* \* \* Q**

**Q \* \* \* \* \* \* \***

**\* \* \* \* Q \* \* \***

**\* \* \* \* \* \* Q \***

**\* Q \* \* \* \* \* \***

**\* \* \* Q \* \* \* \***

**\* \* Q \* \* \* \* \***

**\* \* \* \* \* Q \* \***

**\* \* \* \* \* \* \* Q**

**\* Q \* \* \* \* \* \***

**\* \* \* Q \* \* \* \***

**Q \* \* \* \* \* \* \***

**\* \* \* \* \* \* Q \***

**\* \* \* \* Q \* \* \***

**\* \* Q \* \* \* \* \***

**\* \* \* \* \* \* Q \***

**\* Q \* \* \* \* \* \***

**\* \* \* \* \* \* \* Q**

**\* \* \* \* Q \* \* \***

**Q \* \* \* \* \* \* \***

**\* \* \* Q \* \* \* \***

**\* \* \* \* \* Q \* \***

**\* \* Q \* \* \* \* \***

**\* \* \* \* \* \* Q \***

**\* Q \* \* \* \* \* \***

**\* \* \* \* \* \* \* Q**

**\* \* \* \* \* Q \* \***

**\* \* \* Q \* \* \* \***

**Q \* \* \* \* \* \* \***

**\* \* \* \* Q \* \* \***

**\* \* Q \* \* \* \* \***

**\* \* \* \* \* \* \* Q**

**\* \* \* Q \* \* \* \***

**\* \* \* \* \* \* Q \***

**Q \* \* \* \* \* \* \***

**\* \* \* \* \* Q \* \***

**\* Q \* \* \* \* \* \***

**\* \* \* \* Q \* \* \***

**\* \* \* Q \* \* \* \***

**Q \* \* \* \* \* \* \***

**\* \* \* \* Q \* \* \***

**\* \* \* \* \* \* \* Q**

**\* Q \* \* \* \* \* \***

**\* \* \* \* \* \* Q \***

**\* \* Q \* \* \* \* \***

**\* \* \* \* \* Q \* \***

**\* \* \* Q \* \* \* \***

**Q \* \* \* \* \* \* \***

**\* \* \* \* Q \* \* \***

**\* \* \* \* \* \* \* Q**

**\* \* \* \* \* Q \* \***

**\* \* Q \* \* \* \* \***

**\* \* \* \* \* \* Q \***

**\* Q \* \* \* \* \* \***

**\* \* \* Q \* \* \* \***

**\* Q \* \* \* \* \* \***

**\* \* \* \* Q \* \* \***

**\* \* \* \* \* \* \* Q**

**\* \* \* \* \* Q \* \***

**Q \* \* \* \* \* \* \***

**\* \* Q \* \* \* \* \***

**\* \* \* \* \* \* Q \***

**\* \* \* Q \* \* \* \***

**\* Q \* \* \* \* \* \***

**\* \* \* \* \* \* Q \***

**\* \* Q \* \* \* \* \***

**\* \* \* \* \* Q \* \***

**\* \* \* \* \* \* \* Q**

**Q \* \* \* \* \* \* \***

**\* \* \* \* Q \* \* \***

**\* \* \* Q \* \* \* \***

**\* Q \* \* \* \* \* \***

**\* \* \* \* \* \* Q \***

**\* \* Q \* \* \* \* \***

**\* \* \* \* \* Q \* \***

**\* \* \* \* \* \* \* Q**

**\* \* \* \* Q \* \* \***

**Q \* \* \* \* \* \* \***

**\* \* \* Q \* \* \* \***

**\* Q \* \* \* \* \* \***

**\* \* \* \* \* \* Q \***

**\* \* \* \* Q \* \* \***

**Q \* \* \* \* \* \* \***

**\* \* \* \* \* \* \* Q**

**\* \* \* \* \* Q \* \***

**\* \* Q \* \* \* \* \***

**\* \* \* Q \* \* \* \***

**\* Q \* \* \* \* \* \***

**\* \* \* \* \* \* \* Q**

**\* \* \* \* Q \* \* \***

**\* \* \* \* \* \* Q \***

**Q \* \* \* \* \* \* \***

**\* \* Q \* \* \* \* \***

**\* \* \* \* \* Q \* \***

**\* \* \* Q \* \* \* \***

**\* Q \* \* \* \* \* \***

**\* \* \* \* \* \* \* Q**

**\* \* \* \* \* Q \* \***

**Q \* \* \* \* \* \* \***

**\* \* Q \* \* \* \* \***

**\* \* \* \* Q \* \* \***

**\* \* \* \* \* \* Q \***

**\* \* \* Q \* \* \* \***

**\* \* \* \* \* Q \* \***

**Q \* \* \* \* \* \* \***

**\* \* \* \* Q \* \* \***

**\* Q \* \* \* \* \* \***

**\* \* \* \* \* \* \* Q**

**\* \* Q \* \* \* \* \***

**\* \* \* \* \* \* Q \***

**\* \* \* Q \* \* \* \***

**\* \* \* \* \* Q \* \***

**\* \* \* \* \* \* \* Q**

**\* Q \* \* \* \* \* \***

**\* \* \* \* \* \* Q \***

**Q \* \* \* \* \* \* \***

**\* \* Q \* \* \* \* \***

**\* \* \* \* Q \* \* \***

**\* \* \* Q \* \* \* \***

**\* \* \* \* \* Q \* \***

**\* \* \* \* \* \* \* Q**

**\* \* Q \* \* \* \* \***

**Q \* \* \* \* \* \* \***

**\* \* \* \* \* \* Q \***

**\* \* \* \* Q \* \* \***

**\* Q \* \* \* \* \* \***

**\* \* \* Q \* \* \* \***

**\* \* \* \* \* \* Q \***

**Q \* \* \* \* \* \* \***

**\* \* \* \* \* \* \* Q**

**\* \* \* \* Q \* \* \***

**\* Q \* \* \* \* \* \***

**\* \* \* \* \* Q \* \***

**\* \* Q \* \* \* \* \***

**\* \* \* Q \* \* \* \***

**\* \* \* \* \* \* Q \***

**\* \* Q \* \* \* \* \***

**\* \* \* \* \* \* \* Q**

**\* Q \* \* \* \* \* \***

**\* \* \* \* Q \* \* \***

**Q \* \* \* \* \* \* \***

**\* \* \* \* \* Q \* \***

**\* \* \* Q \* \* \* \***

**\* \* \* \* \* \* Q \***

**\* \* \* \* Q \* \* \***

**\* Q \* \* \* \* \* \***

**\* \* \* \* \* Q \* \***

**Q \* \* \* \* \* \* \***

**\* \* Q \* \* \* \* \***

**\* \* \* \* \* \* \* Q**

**\* \* \* Q \* \* \* \***

**\* \* \* \* \* \* Q \***

**\* \* \* \* Q \* \* \***

**\* \* Q \* \* \* \* \***

**Q \* \* \* \* \* \* \***

**\* \* \* \* \* Q \* \***

**\* \* \* \* \* \* \* Q**

**\* Q \* \* \* \* \* \***

**\* \* \* Q \* \* \* \***

**\* \* \* \* \* \* \* Q**

**Q \* \* \* \* \* \* \***

**\* \* Q \* \* \* \* \***

**\* \* \* \* \* Q \* \***

**\* Q \* \* \* \* \* \***

**\* \* \* \* \* \* Q \***

**\* \* \* \* Q \* \* \***

**\* \* \* Q \* \* \* \***

**\* \* \* \* \* \* \* Q**

**Q \* \* \* \* \* \* \***

**\* \* \* \* Q \* \* \***

**\* \* \* \* \* \* Q \***

**\* Q \* \* \* \* \* \***

**\* \* \* \* \* Q \* \***

**\* \* Q \* \* \* \* \***

**\* \* \* Q \* \* \* \***

**\* \* \* \* \* \* \* Q**

**\* \* \* \* Q \* \* \***

**\* \* Q \* \* \* \* \***

**Q \* \* \* \* \* \* \***

**\* \* \* \* \* \* Q \***

**\* Q \* \* \* \* \* \***

**\* \* \* \* \* Q \* \***

**\* \* \* \* Q \* \* \***

**Q \* \* \* \* \* \* \***

**\* \* \* Q \* \* \* \***

**\* \* \* \* \* Q \* \***

**\* \* \* \* \* \* \* Q**

**\* Q \* \* \* \* \* \***

**\* \* \* \* \* \* Q \***

**\* \* Q \* \* \* \* \***

**\* \* \* \* Q \* \* \***

**Q \* \* \* \* \* \* \***

**\* \* \* \* \* \* \* Q**

**\* \* \* Q \* \* \* \***

**\* Q \* \* \* \* \* \***

**\* \* \* \* \* \* Q \***

**\* \* Q \* \* \* \* \***

**\* \* \* \* \* Q \* \***

**\* \* \* \* Q \* \* \***

**Q \* \* \* \* \* \* \***

**\* \* \* \* \* \* \* Q**

**\* \* \* \* \* Q \* \***

**\* \* Q \* \* \* \* \***

**\* \* \* \* \* \* Q \***

**\* Q \* \* \* \* \* \***

**\* \* \* Q \* \* \* \***

**\* \* \* \* Q \* \* \***

**\* Q \* \* \* \* \* \***

**\* \* \* Q \* \* \* \***

**\* \* \* \* \* Q \* \***

**\* \* \* \* \* \* \* Q**

**\* \* Q \* \* \* \* \***

**Q \* \* \* \* \* \* \***

**\* \* \* \* \* \* Q \***

**\* \* \* \* Q \* \* \***

**\* Q \* \* \* \* \* \***

**\* \* \* Q \* \* \* \***

**\* \* \* \* \* \* Q \***

**\* \* Q \* \* \* \* \***

**\* \* \* \* \* \* \* Q**

**\* \* \* \* \* Q \* \***

**Q \* \* \* \* \* \* \***

**\* \* \* \* Q \* \* \***

**\* Q \* \* \* \* \* \***

**\* \* \* \* \* Q \* \***

**Q \* \* \* \* \* \* \***

**\* \* \* \* \* \* Q \***

**\* \* \* Q \* \* \* \***

**\* \* \* \* \* \* \* Q**

**\* \* Q \* \* \* \* \***

**\* \* \* \* Q \* \* \***

**\* Q \* \* \* \* \* \***

**\* \* \* \* \* \* \* Q**

**Q \* \* \* \* \* \* \***

**\* \* \* Q \* \* \* \***

**\* \* \* \* \* \* Q \***

**\* \* Q \* \* \* \* \***

**\* \* \* \* \* Q \* \***

**\* \* \* \* Q \* \* \***

**\* \* Q \* \* \* \* \***

**Q \* \* \* \* \* \* \***

**\* \* \* \* \* Q \* \***

**\* \* \* \* \* \* \* Q**

**\* Q \* \* \* \* \* \***

**\* \* \* Q \* \* \* \***

**\* \* \* \* \* \* Q \***

**\* \* \* \* Q \* \* \***

**\* \* Q \* \* \* \* \***

**Q \* \* \* \* \* \* \***

**\* \* \* \* \* \* Q \***

**\* Q \* \* \* \* \* \***

**\* \* \* \* \* \* \* Q**

**\* \* \* \* \* Q \* \***

**\* \* \* Q \* \* \* \***

**\* \* \* \* Q \* \* \***

**\* \* Q \* \* \* \* \***

**\* \* \* \* \* \* \* Q**

**\* \* \* Q \* \* \* \***

**\* \* \* \* \* \* Q \***

**Q \* \* \* \* \* \* \***

**\* \* \* \* \* Q \* \***

**\* Q \* \* \* \* \* \***

**\* \* \* \* Q \* \* \***

**\* \* \* \* \* \* Q \***

**Q \* \* \* \* \* \* \***

**\* \* Q \* \* \* \* \***

**\* \* \* \* \* \* \* Q**

**\* \* \* \* \* Q \* \***

**\* \* \* Q \* \* \* \***

**\* Q \* \* \* \* \* \***

**\* \* \* \* Q \* \* \***

**\* \* \* \* \* \* Q \***

**Q \* \* \* \* \* \* \***

**\* \* \* Q \* \* \* \***

**\* Q \* \* \* \* \* \***

**\* \* \* \* \* \* \* Q**

**\* \* \* \* \* Q \* \***

**\* \* Q \* \* \* \* \***

**\* \* \* \* Q \* \* \***

**\* \* \* \* \* \* Q \***

**\* Q \* \* \* \* \* \***

**\* \* \* Q \* \* \* \***

**\* \* \* \* \* \* \* Q**

**Q \* \* \* \* \* \* \***

**\* \* Q \* \* \* \* \***

**\* \* \* \* \* Q \* \***

**\* \* \* \* Q \* \* \***

**\* \* \* \* \* \* Q \***

**\* Q \* \* \* \* \* \***

**\* \* \* \* \* Q \* \***

**\* \* Q \* \* \* \* \***

**Q \* \* \* \* \* \* \***

**\* \* \* Q \* \* \* \***

**\* \* \* \* \* \* \* Q**

**\* \* \* \* Q \* \* \***

**\* \* \* \* \* \* Q \***

**\* Q \* \* \* \* \* \***

**\* \* \* \* \* Q \* \***

**\* \* Q \* \* \* \* \***

**Q \* \* \* \* \* \* \***

**\* \* \* \* \* \* \* Q**

**\* \* \* Q \* \* \* \***

**\* \* \* \* Q \* \* \***

**\* \* \* \* \* \* Q \***

**\* \* \* Q \* \* \* \***

**Q \* \* \* \* \* \* \***

**\* \* Q \* \* \* \* \***

**\* \* \* \* \* \* \* Q**

**\* \* \* \* \* Q \* \***

**\* Q \* \* \* \* \* \***

**\* \* \* \* Q \* \* \***

**\* \* \* \* \* \* \* Q**

**\* \* \* Q \* \* \* \***

**Q \* \* \* \* \* \* \***

**\* \* Q \* \* \* \* \***

**\* \* \* \* \* Q \* \***

**\* Q \* \* \* \* \* \***

**\* \* \* \* \* \* Q \***

**\* \* \* \* Q \* \* \***

**\* \* \* \* \* \* \* Q**

**\* \* \* Q \* \* \* \***

**Q \* \* \* \* \* \* \***

**\* \* \* \* \* \* Q \***

**\* Q \* \* \* \* \* \***

**\* \* \* \* \* Q \* \***

**\* \* Q \* \* \* \* \***

**\* \* \* \* \* Q \* \***

**Q \* \* \* \* \* \* \***

**\* \* \* \* Q \* \* \***

**\* Q \* \* \* \* \* \***

**\* \* \* \* \* \* \* Q**

**\* \* Q \* \* \* \* \***

**\* \* \* \* \* \* Q \***

**\* \* \* Q \* \* \* \***

**\* \* \* \* \* Q \* \***

**\* Q \* \* \* \* \* \***

**\* \* \* \* \* \* Q \***

**Q \* \* \* \* \* \* \***

**\* \* Q \* \* \* \* \***

**\* \* \* \* Q \* \* \***

**\* \* \* \* \* \* \* Q**

**\* \* \* Q \* \* \* \***

**\* \* \* \* \* Q \* \***

**\* Q \* \* \* \* \* \***

**\* \* \* \* \* \* Q \***

**Q \* \* \* \* \* \* \***

**\* \* \* Q \* \* \* \***

**\* \* \* \* \* \* \* Q**

**\* \* \* \* Q \* \* \***

**\* \* Q \* \* \* \* \***

**\* \* \* \* \* Q \* \***

**\* \* Q \* \* \* \* \***

**Q \* \* \* \* \* \* \***

**\* \* \* \* \* \* Q \***

**\* \* \* \* Q \* \* \***

**\* \* \* \* \* \* \* Q**

**\* Q \* \* \* \* \* \***

**\* \* \* Q \* \* \* \***

**\* \* \* \* \* Q \* \***

**\* \* Q \* \* \* \* \***

**Q \* \* \* \* \* \* \***

**\* \* \* \* \* \* \* Q**

**\* \* \* Q \* \* \* \***

**\* Q \* \* \* \* \* \***

**\* \* \* \* \* \* Q \***

**\* \* \* \* Q \* \* \***

**\* \* \* \* \* Q \* \***

**\* \* Q \* \* \* \* \***

**Q \* \* \* \* \* \* \***

**\* \* \* \* \* \* \* Q**

**\* \* \* \* Q \* \* \***

**\* Q \* \* \* \* \* \***

**\* \* \* Q \* \* \* \***

**\* \* \* \* \* \* Q \***

**\* \* \* \* \* Q \* \***

**\* \* Q \* \* \* \* \***

**\* \* \* \* Q \* \* \***

**\* \* \* \* \* \* Q \***

**Q \* \* \* \* \* \* \***

**\* \* \* Q \* \* \* \***

**\* Q \* \* \* \* \* \***

**\* \* \* \* \* \* \* Q**

**\* \* \* \* \* Q \* \***

**\* \* Q \* \* \* \* \***

**\* \* \* \* Q \* \* \***

**\* \* \* \* \* \* \* Q**

**Q \* \* \* \* \* \* \***

**\* \* \* Q \* \* \* \***

**\* Q \* \* \* \* \* \***

**\* \* \* \* \* \* Q \***

**\* \* \* \* \* Q \* \***

**\* \* Q \* \* \* \* \***

**\* \* \* \* \* \* Q \***

**\* Q \* \* \* \* \* \***

**\* \* \* Q \* \* \* \***

**\* \* \* \* \* \* \* Q**

**Q \* \* \* \* \* \* \***

**\* \* \* \* Q \* \* \***

**\* \* \* \* \* Q \* \***

**\* \* Q \* \* \* \* \***

**\* \* \* \* \* \* Q \***

**\* Q \* \* \* \* \* \***

**\* \* \* \* \* \* \* Q**

**\* \* \* \* Q \* \* \***

**Q \* \* \* \* \* \* \***

**\* \* \* Q \* \* \* \***

**\* \* \* \* \* Q \* \***

**\* \* Q \* \* \* \* \***

**\* \* \* \* \* \* Q \***

**\* \* \* Q \* \* \* \***

**Q \* \* \* \* \* \* \***

**\* \* \* \* \* \* \* Q**

**\* Q \* \* \* \* \* \***

**\* \* \* \* Q \* \* \***

**\* \* \* \* \* Q \* \***

**\* \* \* Q \* \* \* \***

**Q \* \* \* \* \* \* \***

**\* \* \* \* Q \* \* \***

**\* \* \* \* \* \* \* Q**

**\* Q \* \* \* \* \* \***

**\* \* \* \* \* \* Q \***

**\* \* Q \* \* \* \* \***

**\* \* \* \* \* Q \* \***

**\* \* \* Q \* \* \* \***

**\* Q \* \* \* \* \* \***

**\* \* \* \* \* \* \* Q**

**\* \* \* \* Q \* \* \***

**\* \* \* \* \* \* Q \***

**Q \* \* \* \* \* \* \***

**\* \* Q \* \* \* \* \***

**\* \* \* \* \* Q \* \***

**\* \* \* Q \* \* \* \***

**\* \* \* \* \* \* Q \***

**Q \* \* \* \* \* \* \***

**\* \* Q \* \* \* \* \***

**\* \* \* \* Q \* \* \***

**\* Q \* \* \* \* \* \***

**\* \* \* \* \* \* \* Q**

**\* \* \* \* \* Q \* \***

**\* \* \* Q \* \* \* \***

**\* \* \* \* \* \* Q \***

**Q \* \* \* \* \* \* \***

**\* \* \* \* \* \* \* Q**

**\* Q \* \* \* \* \* \***

**\* \* \* \* Q \* \* \***

**\* \* Q \* \* \* \* \***

**\* \* \* \* \* Q \* \***

**\* \* \* \* \* \* \* Q**

**\* Q \* \* \* \* \* \***

**\* \* \* Q \* \* \* \***

**Q \* \* \* \* \* \* \***

**\* \* \* \* \* \* Q \***

**\* \* \* \* Q \* \* \***

**\* \* Q \* \* \* \* \***

**\* \* \* \* \* \* Q \***

**Q \* \* \* \* \* \* \***

**\* \* Q \* \* \* \* \***

**\* \* \* \* \* \* \* Q**

**\* \* \* \* \* Q \* \***

**\* \* \* Q \* \* \* \***

**\* Q \* \* \* \* \* \***

**\* \* \* \* Q \* \* \***

**\* \* \* \* \* \* Q \***

**\* Q \* \* \* \* \* \***

**\* \* \* Q \* \* \* \***

**Q \* \* \* \* \* \* \***

**\* \* \* \* \* \* \* Q**

**\* \* \* \* Q \* \* \***

**\* \* Q \* \* \* \* \***

**\* \* \* \* \* Q \* \***

**\* \* \* \* \* \* Q \***

**\* Q \* \* \* \* \* \***

**\* \* \* \* \* Q \* \***

**\* \* Q \* \* \* \* \***

**Q \* \* \* \* \* \* \***

**\* \* \* Q \* \* \* \***

**\* \* \* \* \* \* \* Q**

**\* \* \* \* Q \* \* \***

**\* \* \* \* \* \* Q \***

**\* \* Q \* \* \* \* \***

**Q \* \* \* \* \* \* \***

**\* \* \* \* \* Q \* \***

**\* \* \* \* \* \* \* Q**

**\* \* \* \* Q \* \* \***

**\* Q \* \* \* \* \* \***

**\* \* \* Q \* \* \* \***

**\* \* \* \* \* \* Q \***

**\* \* Q \* \* \* \* \***

**\* \* \* \* \* \* \* Q**

**\* Q \* \* \* \* \* \***

**\* \* \* \* Q \* \* \***

**Q \* \* \* \* \* \* \***

**\* \* \* \* \* Q \* \***

**\* \* \* Q \* \* \* \***

**\* \* \* \* \* \* Q \***

**\* \* \* Q \* \* \* \***

**\* Q \* \* \* \* \* \***

**\* \* \* \* Q \* \* \***

**\* \* \* \* \* \* \* Q**

**Q \* \* \* \* \* \* \***

**\* \* Q \* \* \* \* \***

**\* \* \* \* \* Q \* \***

**\* \* \* \* \* \* Q \***

**\* \* \* Q \* \* \* \***

**\* Q \* \* \* \* \* \***

**\* \* \* \* \* \* \* Q**

**\* \* \* \* \* Q \* \***

**Q \* \* \* \* \* \* \***

**\* \* Q \* \* \* \* \***

**\* \* \* \* Q \* \* \***

**\* \* \* \* \* \* Q \***

**\* \* \* \* Q \* \* \***

**\* \* Q \* \* \* \* \***

**Q \* \* \* \* \* \* \***

**\* \* \* \* \* Q \* \***

**\* \* \* \* \* \* \* Q**

**\* Q \* \* \* \* \* \***

**\* \* \* Q \* \* \* \***

**\* \* \* \* \* \* \* Q**

**\* Q \* \* \* \* \* \***

**\* \* \* Q \* \* \* \***

**Q \* \* \* \* \* \* \***

**\* \* \* \* \* \* Q \***

**\* \* \* \* Q \* \* \***

**\* \* Q \* \* \* \* \***

**\* \* \* \* \* Q \* \***

**\* \* \* \* \* \* \* Q**

**\* Q \* \* \* \* \* \***

**\* \* \* \* Q \* \* \***

**\* \* Q \* \* \* \* \***

**Q \* \* \* \* \* \* \***

**\* \* \* \* \* \* Q \***

**\* \* \* Q \* \* \* \***

**\* \* \* \* \* Q \* \***

**\* \* \* \* \* \* \* Q**

**\* \* Q \* \* \* \* \***

**Q \* \* \* \* \* \* \***

**\* \* \* \* \* Q \* \***

**\* Q \* \* \* \* \* \***

**\* \* \* \* Q \* \* \***

**\* \* \* \* \* \* Q \***

**\* \* \* Q \* \* \* \***

**\* \* \* \* \* \* \* Q**

**\* \* \* Q \* \* \* \***

**Q \* \* \* \* \* \* \***

**\* \* Q \* \* \* \* \***

**\* \* \* \* \* Q \* \***

**\* Q \* \* \* \* \* \***

**\* \* \* \* \* \* Q \***

**\* \* \* \* Q \* \* \***

**total possiblities = 92**

**\*/**

**TIME COMPLEXITY OF N-QUEEN PROBLEM IS O(N!)**

**Explanation: If we add all this up and define the run time as T(N). Then T(N) = O(N^2) + N\*T(N-1). If you draw a recursion tree using this recurrence, the final term will be something like n^3+ n!O(1). By the definition of Big O, this can be reduced to O(n!) running time.**

**CONCLUSION: N queen's** problem was **implemented and analyzed successfully.**