



Includes paid promotion

Normal text

Arial

-

11

+

B

U

I

L

M

T

E

H

A

C

P

F

R

V

W

X

Y

Z

a

b

c

d

e

f

g

h

i

To exit full screen, press Esc

[ONnb1wdx2Ma&index=51](#)

## 6. K-th permutation Sequence

[https://www.youtube.com/watch?v=wT7gcXLYoao&list=PLgUwDviBlf0p4ozDR\\_kJJk](https://www.youtube.com/watch?v=wT7gcXLYoao&list=PLgUwDviBlf0p4ozDR_kJJk)[ONnb1wdx2Ma&index=55](#)

## Day10: (Recursion and Backtracking)

### 1. Print all Permutations of a string/array

[https://www.youtube.com/watch?v=f2ic2Rsc9pU&list=PLgUwDviBlf0p4ozDR\\_kJJkO](https://www.youtube.com/watch?v=f2ic2Rsc9pU&list=PLgUwDviBlf0p4ozDR_kJJkO)[Nnb1wdx2Ma&index=52](#)

### 2. N queens Problem

### 3. Sudoku

### 4. M coloring Problem (Graph prob)

### 5. Rat in a Maze

### 6. Word Break (print all ways)

## Day11: (Divide and Conquer)

### 1. 1/N-th root of an integer (use binary search) (square root, cube root, ..)

### 2. Matrix Median

### 3. Find the element that appears once in sorted array, and rest element appears twice (Binary search)

12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 | 32 | 33 | 34 | 35 | 36 | 37 | 38 | 39 | 40 | 41 | 42 | 43 | 44 | 45 | 46 | 47 | 48 | 49 | 50 | 51 | 52 | 53 | 54 | 55 | 56 | 57 | 58 | 59 | 60 | 61 | 62 | 63 | 64 | 65 | 66 | 67 | 68 | 69 | 70 | 71 | 72 | 73 | 74 | 75 | 76 | 77 | 78 | 79 | 80 | 81 | 82 | 83 | 84 | 85 | 86 | 87 | 88 | 89 | 90 | 91 | 92 | 93 | 94 | 95 | 96 | 97 | 98 | 99 | 100

TUF



0:06 / 36:54





100%

Normal text

Arial

-

11

+

**B***I*UA[ONnb1wdx2Ma&index=51](#)

#### 6. K-th permutation Sequence

[https://www.youtube.com/watch?v=wT7gcXLYoao&list=PLgUwDviBlf0p4ozDR\\_kJJk](https://www.youtube.com/watch?v=wT7gcXLYoao&list=PLgUwDviBlf0p4ozDR_kJJk)[ONnb1wdx2Ma&index=55](#)

#### Day10: (Recursion and Backtracking)

##### 1. Print all Permutations of a string/array

[https://www.youtube.com/watch?v=f2ic2Rsc9pU&list=PLgUwDviBlf0p4ozDR\\_kJJkO](https://www.youtube.com/watch?v=f2ic2Rsc9pU&list=PLgUwDviBlf0p4ozDR_kJJkO)[Nnb1wdx2Ma&index=52](#)

##### 2. N queens Problem

##### 3. Sudoku

##### 4. M coloring Problem (Graph prob)

##### 5. Rat in a Maze

##### 6. Word Break (print all ways)

#### Day11: (Divide and Conquer)

##### 1. 1/N-th root of an integer (use binary search) (square root, cube root, ..)

##### 2. Matrix Median

##### 3. Find the element that appears once in sorted array, and rest element appears twice (Binary search)

  
USE COUPON

ON ANY COURSE

  
TAKEUFORWARD

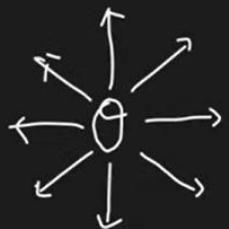
$N=4$   $\Theta$  events

	↓	↓	↓	↓
→	Θ			
→		Θ		
→			Θ	
→	Θ			

→ Every row  $\rightarrow 1 \Theta$

→ Every col  $\rightarrow 1 \Theta$

→ None of the  $\Theta$  attack each other



$N=4$     θ even

↓    ↓    ↓    ↓

→		0	
→			0
→	0		
→		0	

→ Every row  $\rightarrow 1 \theta$

→ Every col  $\rightarrow 1 \theta$

→ None of the θ attack each other

$N=4$      $\Theta$  events

↓    ↓    ↓    ↓

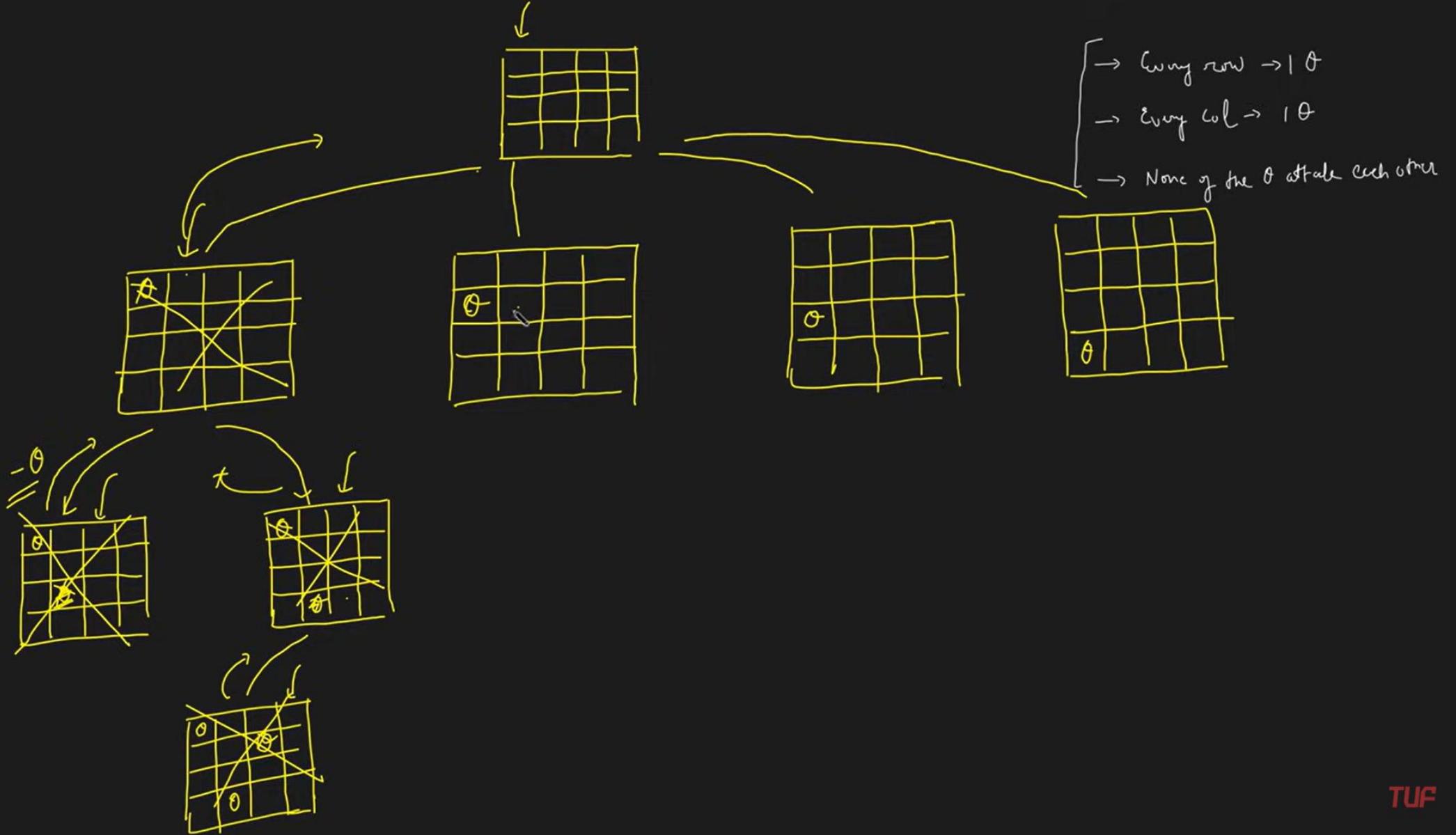
→		0	
→			0
→	0		
→		0	

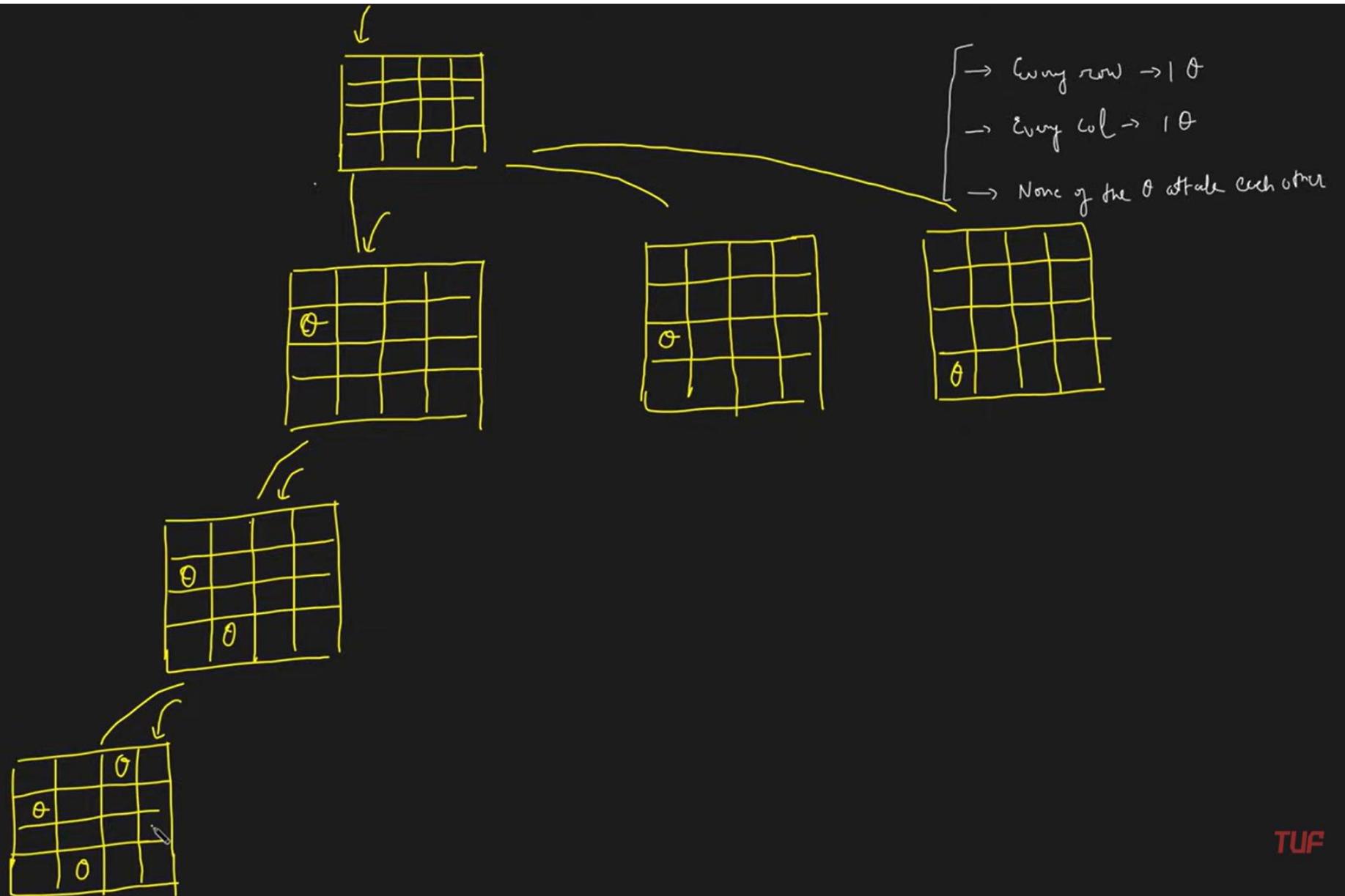
- Every row  $\rightarrow 1 \Theta$   
→ Every col  $\rightarrow 1 \Theta$   
→ None of the  $\Theta$  attack each other

0		0	
			0
0			

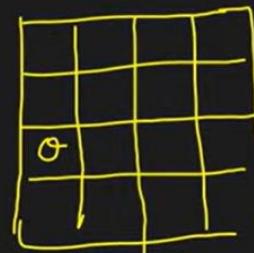
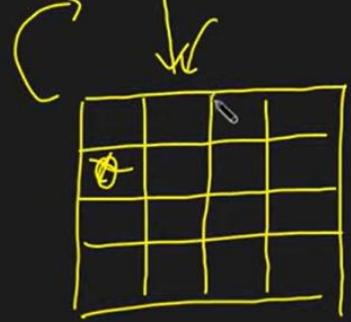
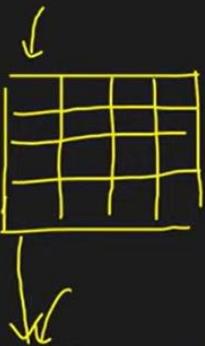
$N$

$N \times N$

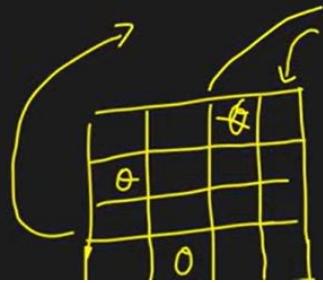
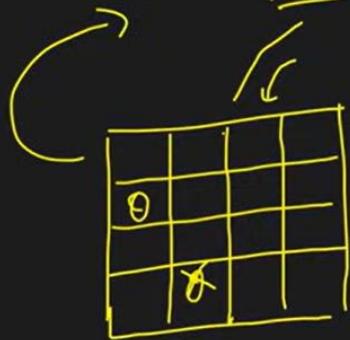
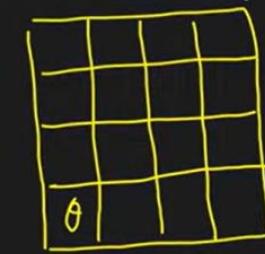




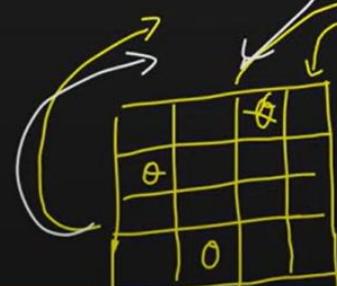
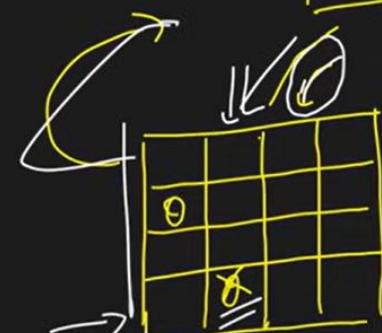
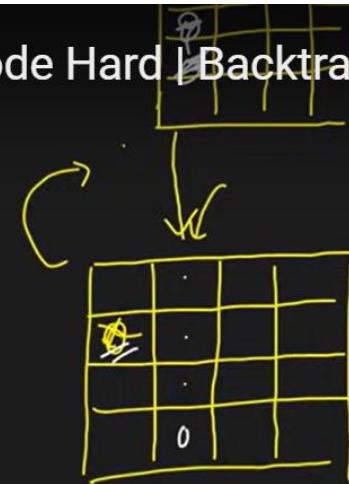
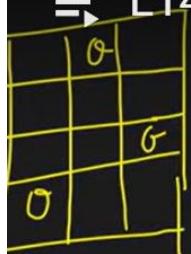
	0	
		0
0		



$\begin{cases} \rightarrow \text{Every row} \rightarrow 1 \theta \\ \rightarrow \text{Every col} \rightarrow 1 \theta \\ \rightarrow \text{None of the } \theta \text{ attack each other} \end{cases}$



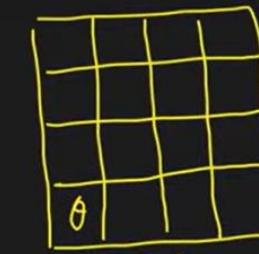
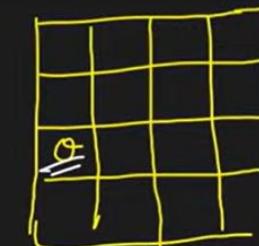
## L14. N-Queens | Leetcode Hard | Backtracking



→ Every col → 1



→ None of the Q attack each other



↓ X

$f_{col}$

$f_n(i=0 \rightarrow n-1)$

y (fill → v)

now col = 0  
 $f_{col+1}$  = empty

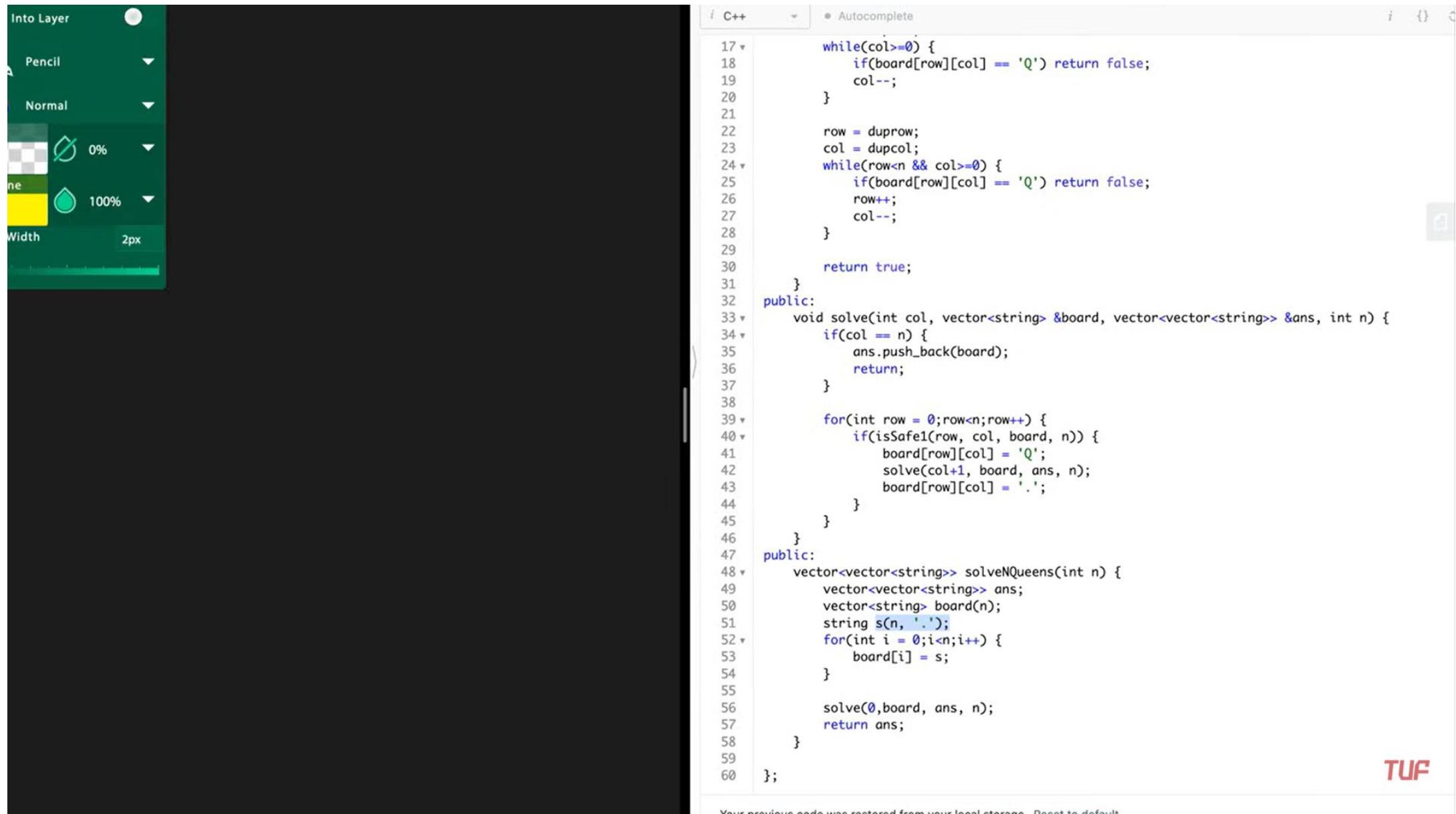
TUF

i C++ • Autocomplete i {} c

```
17 v     while(col>=0) {
18       if(board[row][col] == 'Q') return false;
19       col--;
20     }
21
22     row = duprow;
23     col = dupcol;
24 v     while(row<n && col>=0) {
25       if(board[row][col] == 'Q') return false;
26       row++;
27       col--;
28     }
29
30     return true;
31   }
32 public:
33 v   void solve(int col, vector<string> &board, vector<vector<string>> &ans, int n) {
34 v     if(col == n) {
35       ans.push_back(board);
36       return;
37     }
38
39 v     for(int row = 0;row<n;row++) {
40 v       if(isSafe1(row, col, board, n)) {
41         board[row][col] = 'Q';
42         solve(col+1, board, ans, n);
43         board[row][col] = '.';
44       }
45     }
46   }
47 public:
48 v   vector<vector<string>> solveNQueens(int n) {
49   vector<vector<string>> ans;
50   vector<string> board(n);
51   string s(n, '.');
52 v   for(int i = 0;i<n;i++) {
53     board[i] = s;
54   }
55
56   solve(0,board, ans, n);
57   return ans;
58 }
59
60 };
```

Your previous code was restored from your local storage. Reset to default

TUF



Intro Layer

Pencil

Normal

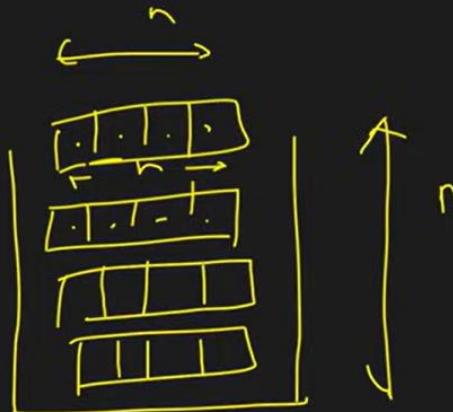
Width: 2px

Autocomplete

```
i C++ • Autocomplete
17 while(col>=0) {
18     if(board[row][col] == 'Q') return false;
19     col--;
20 }
21
22 row = duprow;
23 col = dupcol;
24 while(row<n && col>=0) {
25     if(board[row][col] == 'Q') return false;
26     row++;
27     col--;
28 }
29
30 return true;
31 }
32 public:
33 void solve(int col, vector<string> &board, vector<vector<string>> &ans, int n) {
34     if(col == n) {
35         ans.push_back(board);
36         return;
37     }
38
39 for(int row = 0;row<n;row++) {
40     if(isSafe1(row, col, board, n)) {
41         board[row][col] = 'Q';
42         solve(col+1, board, ans, n);
43         board[row][col] = '.';
44     }
45 }
46 }
47 public:
48 vector<vector<string>> solveNQueens(int n) {
49     vector<vector<string>> ans;
50     vector<string> board(n);
51     string s(n, '.');
52 for(int i = 0;i<n;i++) {
53     board[i] = s;
54 }
55
56 solve(0,board, ans, n);
57 return ans;
58 }
59 }
60 }
```

Your previous code was restored from your local storage. Reset to default

TUF

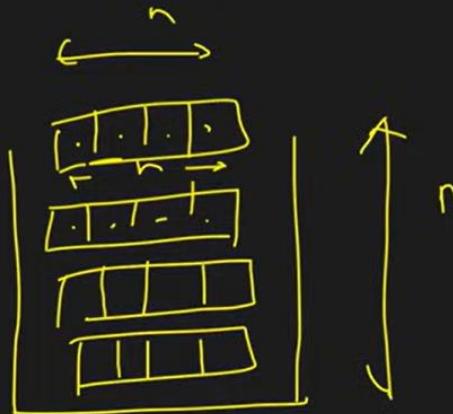


$n=4$

```
i C++ • Autocomplete i {} c
17 while(col>=0) {
18     if(board[row][col] == 'Q') return false;
19     col--;
20 }
21
22 row = duprow;
23 col = dupcol;
24 while(row<n && col>=0) {
25     if(board[row][col] == 'Q') return false;
26     row++;
27     col--;
28 }
29
30 return true;
31 }
32 public:
33 void solve(int col, vector<string> &board, vector<vector<string>> &ans, int n) {
34     if(col == n) {
35         ans.push_back(board);
36         return;
37     }
38
39 for(int row = 0;row<n;row++) {
40     if(isSafe1(row, col, board, n)) {
41         board[row][col] = 'Q';
42         solve(col+1, board, ans, n);
43         board[row][col] = '.';
44     }
45 }
46 }
47 public:
48 vector<vector<string>> solveNQueens(int n) {
49     vector<vector<string>> ans;
50     vector<string> board(n);
51     string s(n, '.');
52     for(int i = 0;i<n;i++) {
53         board[i] = s;
54     }
55
56     solve(0,board, ans, n);
57     return ans;
58 }
59
60 };
```

TUF

Your previous code was restored from your local storage. Reset to default

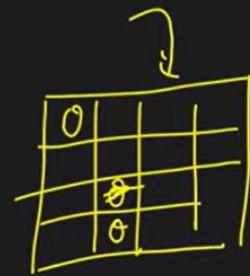
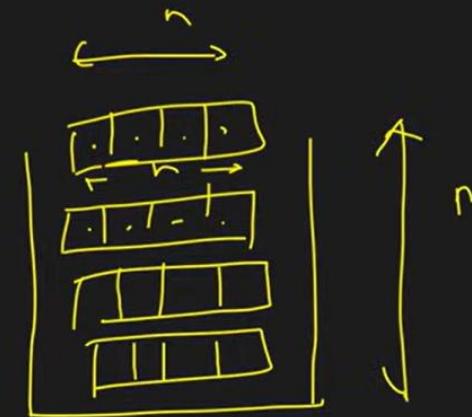


```

i C++ • Autocomplete i {}
17 while(col>=0) {
18     if(board[row][col] == 'Q') return false;
19     col--;
20 }
21
22 row = duprow;
23 col = dupcol;
24 while(row<n && col>=0) {
25     if(board[row][col] == 'Q') return false;
26     row++;
27     col--;
28 }
29
30     return true;
31 }
32 public:
33 void solve(int col, vector<string> &board, vector<vector<string>> &ans, int n) {
34     if(col == n) {
35         ans.push_back(board);
36         return;
37     }
38
39 for(int row = 0;row<n;row++){
40     if(isSafe1(row, col, board, n)) {
41         board[row][col] = 'Q';
42         solve(col+1, board, ans, n);
43         board[row][col] = '.';
44     }
45 }
46
47 public:
48 vector<vector<string>> solveNQueens(int n) {
49     vector<vector<string>> ans;
50     vector<string> board(n);
51     string s(n, '.');
52 for(int i = 0;i<n;i++) {
53     board[i] = s;
54 }
55
56     solve(0,board, ans, n);
57     return ans;
58 }
59
60 };

```

Your previous code was restored from your local storage. Reset to default



```
i C++ • Autocomplete
17 while(col>=0) {
18     if(board[row][col] == 'Q') return false;
19     col--;
20 }
21
22 row = duprow;
23 col = dupcol;
24 while(row<n && col>=0) {
25     if(board[row][col] == 'Q') return false;
26     row++;
27     col--;
28 }
29
30 return true;
31 }
32 public:
33 void solve(int col, vector<string> &board, vector<vector<string>> &ans, int n) {
34     if(col == n) {
35         ans.push_back(board);
36         return;
37     }
38
39 for(int row = 0;row<n;row++) {
40     if(isSafe1(row, col, board, n)) {
41         board[row][col] = 'Q';
42         solve(col+1, board, ans, n);
43         board[row][col] = '.';
44     }
45 }
46 public:
47 vector<vector<string>> solveNQueens(int n) {
48     vector<vector<string>> ans;
49     vector<string> board(n);
50     string s(n, '.');
51     for(int i = 0;i<n;i++) {
52         board[i] = s;
53     }
54
55     solve(0,board, ans, n);
56     return ans;
57 }
58 }
59 }
60 }

TUF
```

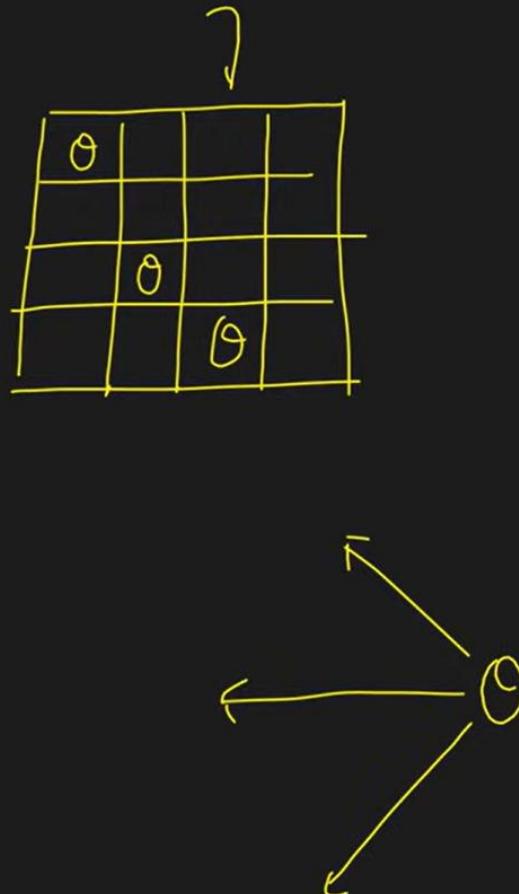
Your previous code was restored from your local storage. Reset to default

```
i C++ • Autocomplete i {} c
17     while(col>=0) {
18         if(board[row][col] == 'Q') return false;
19         col--;
20     }
21
22     row = duprow;
23     col = dupcol;
24     while(row<n && col>=0) {
25         if(board[row][col] == 'Q') return false;
26         row++;
27         col--;
28     }
29
30     return true;
31 }
32 public:
33 void solve(int col, vector<string> &board, vector<vector<string>> &ans, int n) {
34     if(col == n) {
35         ans.push_back(board);
36         return;
37     }
38
39     for(int row = 0;row<n;row++) {
40         if(isSafe1(row, col, board, n)) {
41             board[row][col] = 'Q';
42             solve(col+1, board, ans, n);
43             board[row][col] = '.';
44         }
45     }
46 }
47 public:
48 vector<vector<string>> solveNQueens(int n) {
49     vector<vector<string>> ans;
50     vector<string> board(n);
51     string s(n, '.');
52     for(int i = 0;i<n;i++) {
53         board[i] = s;
54     }
55
56     solve(0,board, ans, n);
57     return ans;
58 }
59
60 };

```

Your previous code was restored from your local storage. Reset to default

TUF

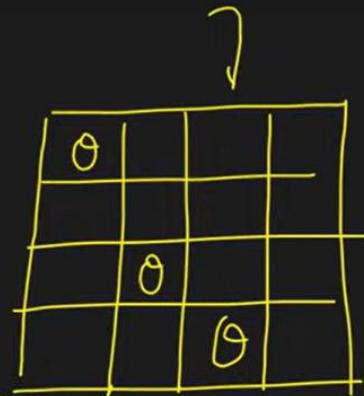


```
i C++ Autocomplete

1 class Solution {
2     public:
3         bool isSafe1(int row, int col, vector<string> board, int n) {
4             // check upper diagonal
5             int duprow = row;
6             int dupcol = col;
7
8             while(row >= 0 && col >= 0) {
9                 if(board[row][col] == 'Q') return false;
10                row--;
11                col--;
12            }
13
14
15            col = dupcol;
16            row = duprow;
17            while(col >= 0) {
18                if(board[row][col] == 'Q') return false;
19                col--;
20            }
21
22            row = duprow;
23            col = dupcol;
24            while(row < n && col >= 0) {
25                if(board[row][col] == 'Q') return false;
26                row++;
27                col--;
28            }
29
30            return true;
31        }
32    public:
33        void solve(int col, vector<string> &board, vector<vector<string>> &ans, int n) {
34            if(col == n) {
35                ans.push_back(board);
36                return;
37            }
38
39            for(int row = 0; row < n; row++) {
40                if(isSafe1(row, col, board, n)) {
41                    board[row][col] = 'Q';
42                    solve(col + 1, board, ans, n);
43                    board[row][col] = '.';
44                }
45            }
46        }
47    }
```

Your previous code was restored from your local storage. Reset to default

## L14. N-Queens | Leetcode Hard | Backtracking



Q (row, col)

```
C++ Autocomplete
1 class Solution {
2     public:
3         bool isSafe1(int row, int col, vector<string> board, int n) {
4             // check upper diagonal
5             int duprow = row;
6             int dupcol = col;
7
8             while(row >= 0 && col >= 0) {
9                 if(board[row][col] == 'Q') return false;
10                row--;
11                col--;
12            }
13
14            col = dupcol;
15            row = duprow;
16            while(col >= 0) {
17                if(board[row][col] == 'Q') return false;
18                col--;
19            }
20
21            row = duprow;
22            col = dupcol;
23            while(row < n && col >= 0) {
24                if(board[row][col] == 'Q') return false;
25                row++;
26                col--;
27            }
28
29            return true;
30        }
31
32        void solve(int col, vector<string> &board, vector<vector<string>> &ans, int n) {
33            if(col == n) {
34                ans.push_back(board);
35                return;
36            }
37
38            for(int row = 0; row < n; row++) {
39                if(isSafe1(row, col, board, n)) {
40                    board[row][col] = 'Q';
41                    solve(col + 1, board, ans, n);
42                    board[row][col] = '.';
43                }
44            }
45        }
46    }
```

TUF

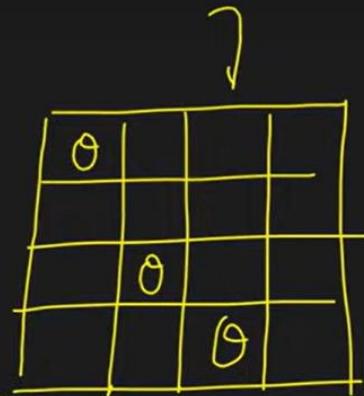


26:03 / 36:54

Your previous code was restored from your local storage. Reset to default



## L14. N-Queens | Leetcode Hard | Backtracking



Q (row, col)

```
C++ Autocomplete
1 class Solution {
2     public:
3         bool isSafe1(int row, int col, vector<string> board, int n) {
4             // check upper diagonal
5             int duprow = row;
6             int dupcol = col;
7
8             while(row >= 0 && col >= 0) {
9                 if(board[row][col] == 'Q') return false;
10                row--;
11                col--;
12            }
13
14            col = dupcol;
15            row = duprow;
16            while(col >= 0) {
17                if(board[row][col] == 'Q') return false;
18                col--;
19            }
20
21            row = duprow;
22            col = dupcol;
23            while(row < n && col >= 0) {
24                if(board[row][col] == 'Q') return false;
25                row++;
26                col--;
27            }
28
29            return true;
30        }
31
32        void solve(int col, vector<string> &board, vector<vector<string>> &ans, int n) {
33            if(col == n) {
34                ans.push_back(board);
35                return;
36            }
37
38            for(int row = 0; row < n; row++) {
39                if(isSafe1(row, col, board, n)) {
40                    board[row][col] = 'Q';
41                    solve(col + 1, board, ans, n);
42                    board[row][col] = '.';
43                }
44            }
45        }
46    }
```

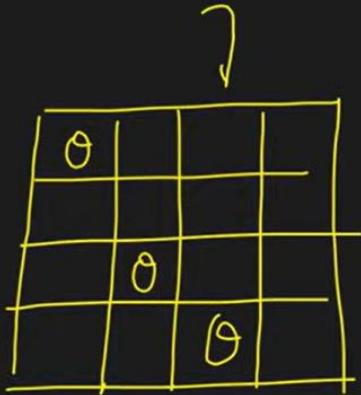
TUF



26:03 / 36:54

Your previous code was restored from your local storage. Reset to default

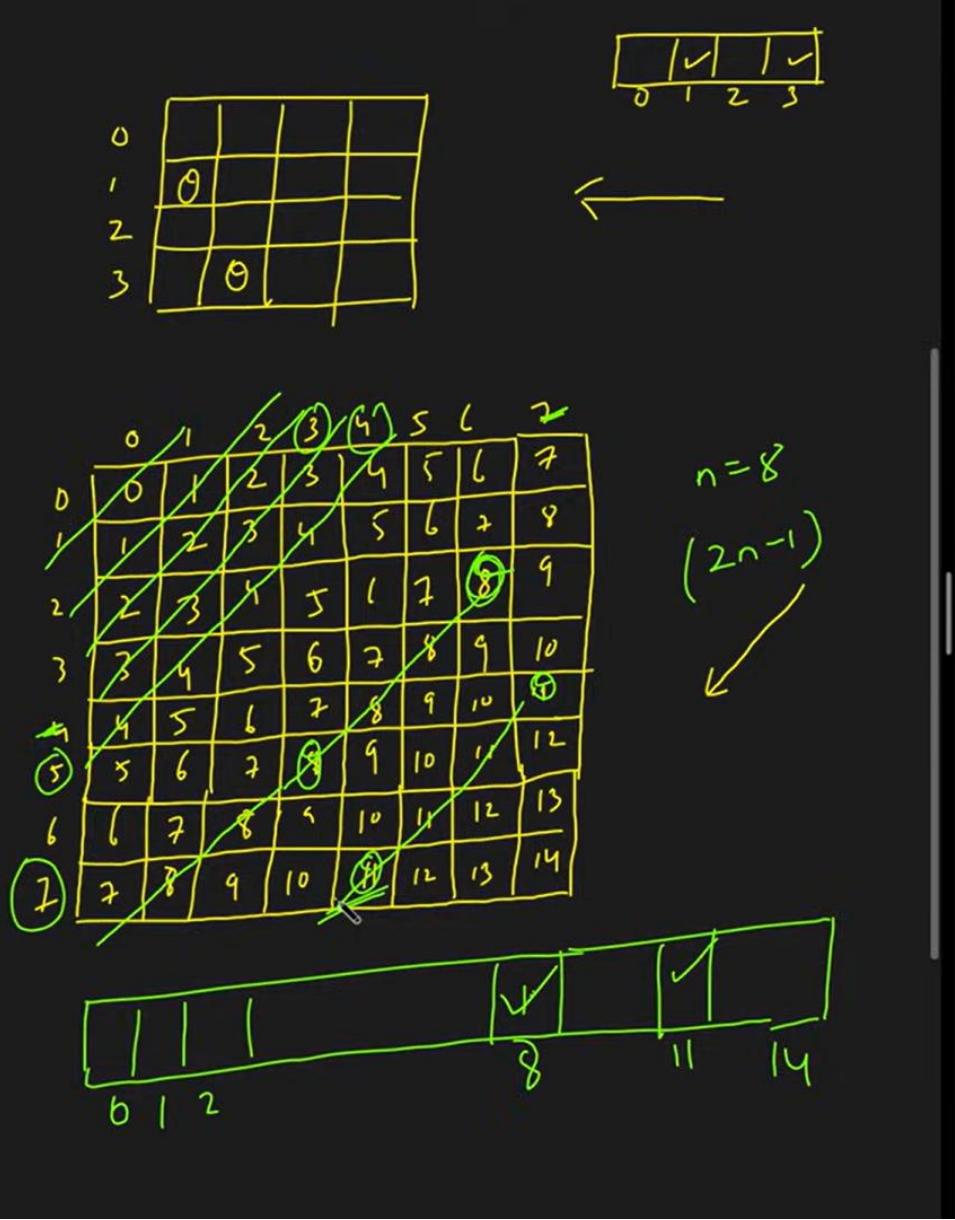




$\Theta(n \times m)$

```
i C++ Autocomplete
1 class Solution {
2 public:
3     bool isSafe1(int row, int col, vector<string> board, int n) {
4         // check upper diagonal
5         int duprow = row;
6         int dupcol = col;
7
8         while(row >= 0 && col >= 0) {
9             if(board[row][col] == 'Q') return false;
10            row--;
11            col--;
12        }
13
14        col = dupcol;
15        row = duprow;
16        while(col >= 0) {
17            if(board[row][col] == 'Q') return false;
18            col--;
19        }
20
21        row = duprow;
22        col = dupcol;
23        while(row < n && col >= 0) {
24            if(board[row][col] == 'Q') return false;
25            row++;
26            col--;
27        }
28
29        return true;
30    }
31
32    void solve(int col, vector<string> &board, vector<vector<string>> &ans, int n) {
33        if(col == n) {
34            ans.push_back(board);
35            return;
36        }
37
38        for(int row = 0; row < n; row++) {
39            if(isSafe1(row, col, board, n)) {
40                board[row][col] = 'Q';
41                solve(col + 1, board, ans, n);
42                board[row][col] = '.';
43            }
44        }
45    }
46 }
```

Your previous code was restored from your local storage. Reset to default



LeetCode Explore Day 10 Problems Interview Contest We're hiring, apply today! Premium

```
i C++ Autocomplete
class Solution {
public:
    void solve(int col, vector<string> &board, vector<vector<string>> &ans,
               vector<int> &leftRow,
               vector<int> &upperDiagonal, vector<int> &lowerDiagonal, int n) {
        if(col == n) {
            ans.push_back(board);
            return;
        }

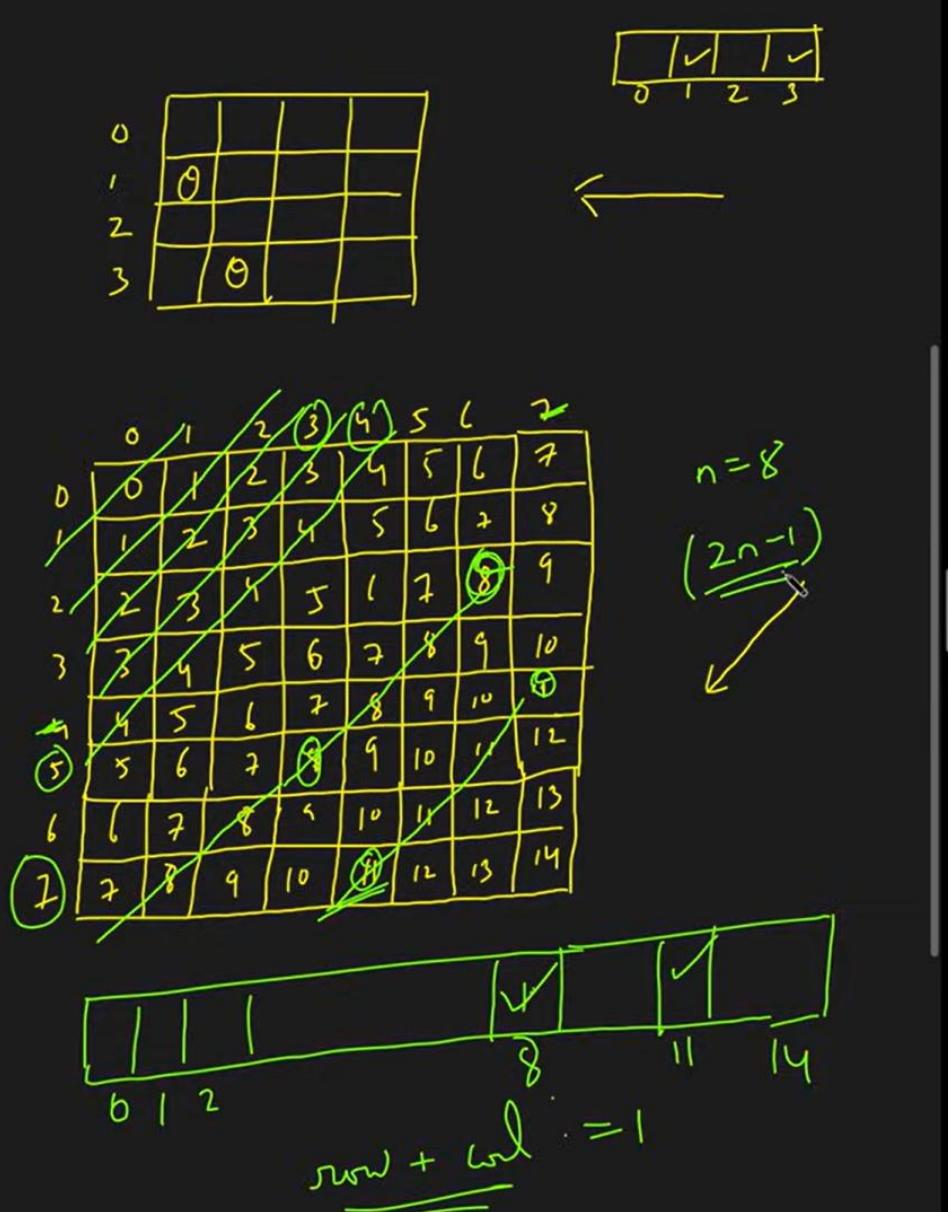
        for(int row = 0; row < n; row++) {
            if(leftRow[row] == 0 && lowerDiagonal[row + col] == 0
               && upperDiagonal[n - 1 + col - row] == 0) {

                board[row][col] = 'Q';
                leftRow[row] = 1;
                lowerDiagonal[row + col] = 1;
                upperDiagonal[n - 1 + col - row] = 1;
                solve(col + 1, board, ans, leftRow, upperDiagonal, lowerDiagonal, n);
                board[row][col] = '.';
                leftRow[row] = 0;
                lowerDiagonal[row + col] = 0;
                upperDiagonal[n - 1 + col - row] = 0;
            }
        }
    }

    public:
    vector<vector<string>> solveNQueens(int n) {
        vector<vector<string>> ans;
        vector<string> board(n);
        string s(n, '.');
        for(int i = 0; i < n; i++) {
            board[i] = s;
        }
        vector<int> leftRow(n, 0), upperDiagonal(2 * n - 1, 0), lowerDiagonal(2 * n - 1, 0);
        solve(0, board, ans, leftRow, upperDiagonal, lowerDiagonal, n);
        return ans;
    }
};
```

Your previous code was restored from your local storage. Reset to default

TUF



LeetCode Explore Day 10 Problems Interview New Contest We're hiring, apply today! Premium

```
i C++ Autocomplete
1 class Solution {
2
3     public:
4         void solve(int col, vector<string> &board, vector<vector<string>> &ans,
5                 vector<int> &leftRow,
6                 vector<int> &upperDiagonal, vector<int> &lowerDiagonal, int n) {
7             if(col == n) {
8                 ans.push_back(board);
9                 return;
10            }
11
12            for(int row = 0; row < n; row++) {
13                if(leftRow[row] == 0 && lowerDiagonal[row + col] == 0
14                  && upperDiagonal[n - 1 + col - row] == 0) {
15
16                    board[row][col] = 'Q';
17                    leftRow[row] = 1;
18                    lowerDiagonal[row + col] = 1;
19                    upperDiagonal[n - 1 + col - row] = 1;
20                    solve(col + 1, board, ans, leftRow, upperDiagonal, lowerDiagonal, n);
21                    board[row][col] = '.';
22                    leftRow[row] = 0;
23                    lowerDiagonal[row + col] = 0;
24                    upperDiagonal[n - 1 + col - row] = 0;
25                }
26            }
27        }
28
29    public:
30        vector<vector<string>> solveNQueens(int n) {
31            vector<vector<string>> ans;
32            vector<string> board(n);
33            string s(n, '.');
34            for(int i = 0; i < n; i++) {
35                board[i] = s;
36            }
37            vector<int> leftRow(n, 0), upperDiagonal(2 * n - 1, 0), lowerDiagonal(2 * n - 1, 0);
38            solve(0, board, ans, leftRow, upperDiagonal, lowerDiagonal, n);
39            return ans;
40        }
41    };
42 }
```

Your previous code was restored from your local storage. Reset to default

TUF

	0	1	2	3	4	5	6	7
0	7	8	9	10	11	12	13	14
1	6	7	8	9	10	11	12	13
2	5	6	7	8	9	10	11	12
3	4	5	6	7	8	9	10	11
4	3	4	5	6	7	8	9	10
5	2	3	4	5	6	7	8	9
6	1	2	3	4	5	6	7	8
7	0	1	2	3	4	5	6	7

$n=8$

$$(n-1) + (col - row)$$

$$7 + (7 - 0) = 14$$

LeetCode Explore Day 10 Problems Interview New Contest We're hiring, apply today! Premium

```
i C++ Autocomplete
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
```

```
class Solution {
public:
    void solve(int col, vector<string> &board, vector<vector<string>> &ans,
               vector<int> &leftRow,
               vector<int> &upperDiagonal, vector<int> &lowerDiagonal, int n) {
        if(col == n) {
            ans.push_back(board);
            return;
        }

        for(int row = 0; row < n; row++) {
            if(leftRow[row] == 0 && lowerDiagonal[row + col] == 0
               && upperDiagonal[n - 1 + col - row] == 0) {

                board[row][col] = 'Q';
                leftRow[row] = 1;
                lowerDiagonal[row + col] = 1;
                upperDiagonal[n - 1 + col - row] = 1;
                solve(col + 1, board, ans, leftRow, upperDiagonal, lowerDiagonal, n);
                board[row][col] = '.';
                leftRow[row] = 0;
                lowerDiagonal[row + col] = 0;
                upperDiagonal[n - 1 + col - row] = 0;
            }
        }
    }

public:
    vector<vector<string>> solveNQueens(int n) {
        vector<vector<string>> ans;
        vector<string> board(n);
        string s(n, '.');
        for(int i = 0; i < n; i++) {
            board[i] = s;
        }
        vector<int> leftRow(n, 0), upperDiagonal(2 * n - 1, 0), lowerDiagonal(2 * n - 1, 0);
        solve(0, board, ans, leftRow, upperDiagonal, lowerDiagonal, n);
        return ans;
    }
};
```

Your previous code was restored from your local storage. Reset to default

TUF

	0	1	2	3	4	5	6	7
0	7	8	9	10	11	12	13	14
1	6	7	8	9	10	11	12	13
2	5	6	7	8	9	10	11	12
3	4	5	6	7	8	9	10	11
4	3	4	5	6	7	8	9	10
5	2	3	4	5	6	7	8	9
6	1	2	3	4	5	6	7	8
7	0	1	2	3	4	5	6	7

$n=8$



$2^{n-1}$

$$\ln \left[ \frac{(n-1)}{2} + \frac{(n-1)(n-2)}{2} \right] = 1$$

$$\frac{7}{2} + \frac{7 \cdot 6}{2} = 14$$



```
i C++ Autocomplete
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42

class Solution {
public:
    void solve(int col, vector<string> &board, vector<vector<string>> &ans,
               vector<int> &leftRow,
               vector<int> &upperDiagonal, vector<int> &lowerDiagonal, int n) {
        if(col == n) {
            ans.push_back(board);
            return;
        }

        for(int row = 0; row < n; row++) {
            if(leftRow[row] == 0 && lowerDiagonal[row + col] == 0
               && upperDiagonal[n - 1 + col - row] == 0) {

                board[row][col] = 'Q';
                leftRow[row] = 1;
                lowerDiagonal[row + col] = 1;
                upperDiagonal[n - 1 + col - row] = 1;
                solve(col + 1, board, ans, leftRow, upperDiagonal, lowerDiagonal, n);
                board[row][col] = '.';
                leftRow[row] = 0;
                lowerDiagonal[row + col] = 0;
                upperDiagonal[n - 1 + col - row] = 0;
            }
        }
    }

public:
    vector<vector<string>> solveNQueens(int n) {
        vector<vector<string>> ans;
        vector<string> board(n);
        string s(n, '.');
        for(int i = 0; i < n; i++) {
            board[i] = s;
        }
        vector<int> leftRow(n, 0), upperDiagonal(2 * n - 1, 0), lowerDiagonal(2 * n - 1, 0);
        solve(0, board, ans, leftRow, upperDiagonal, lowerDiagonal, n);
        return ans;
    }
};

TUF
```

Your previous code was restored from your local storage. Reset to default

	0	1	2	3	4	5	6	7
0	7	8	9	10	11	12	13	14
1	6	7	8	9	10	11	12	13
2	5	6	7	8	9	10	11	12
3	4	5	6	7	8	9	10	11
4	3	4	5	6	7	8	9	10
5	2	3	4	5	6	7	8	9
6	1	2	3	4	5	6	7	8
7	0	1	2	3	4	5	6	7

$n=8$



$2^{n-1}$

$$\text{hand} \left[ \cancel{(n-1)} + \cancel{(col - row)} \right] = 1$$

$$\cancel{7} + (7 - 0) = 1$$



```
i C++ Autocomplete
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42

class Solution {
public:
    void solve(int col, vector<string> &board, vector<vector<string>> &ans,
               vector<int> &leftRow,
               vector<int> &upperDiagonal, vector<int> &lowerDiagonal, int n) {
        if(col == n) {
            ans.push_back(board);
            return;
        }

        for(int row = 0; row < n; row++) {
            if(leftRow[row] == 0 && lowerDiagonal[row + col] == 0
               && upperDiagonal[n - 1 + col - row] == 0) {

                board[row][col] = 'Q';
                leftRow[row] = 1;
                lowerDiagonal[row + col] = 1;
                upperDiagonal[n - 1 + col - row] = 1;
                solve(col + 1, board, ans, leftRow, upperDiagonal, lowerDiagonal, n);
                board[row][col] = '.';
                leftRow[row] = 0;
                lowerDiagonal[row + col] = 0;
                upperDiagonal[n - 1 + col - row] = 0;
            }
        }
    }

public:
    vector<vector<string>> solveNQueens(int n) {
        vector<vector<string>> ans;
        vector<string> board(n);
        string s(n, '.');
        for(int i = 0; i < n; i++) {
            board[i] = s;
        }
        vector<int> leftRow(n, 0), upperDiagonal(2 * n - 1, 0), lowerDiagonal(2 * n - 1, 0);
        solve(0, board, ans, leftRow, upperDiagonal, lowerDiagonal, n);
        return ans;
    }
};

TUF
```

Your previous code was restored from your local storage. Reset to default

	0	1	2	3	4	5	6	7
0	7	8	9	10	11	12	13	14
1	6	7	8	9	10	11	12	13
2	5	6	7	8	9	10	11	12
3	4	5	6	7	8	9	10	11
4	3	4	5	6	7	8	9	10
5	2	3	4	5	6	7	8	9
6	1	2	3	4	5	6	7	8
7	0	1	2	3	4	5	6	7

$n=8$



$2^{n-1}$

$$\ln \left[ \frac{(n-1)}{2} + \frac{(n! - n!)^2}{2} \right] = 1$$

$$\frac{7}{2} + (7-0) = 14$$



```

1 C++ Autocomplete
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42

class Solution {
public:
    void solve(int col, vector<string> &board, vector<vector<string>> &ans,
               vector<int> &leftRow,
               vector<int> &upperDiagonal, vector<int> &lowerDiagonal, int n) {
        if(col == n) {
            ans.push_back(board);
            return;
        }

        for(int row = 0; row < n; row++) {
            if(leftRow[row] == 0 && lowerDiagonal[row + col] == 0
               && upperDiagonal[n - 1 + col - row] == 0) {

                board[row][col] = 'Q';
                leftRow[row] = 1;
                lowerDiagonal[row + col] = 1;
                upperDiagonal[n - 1 + col - row] = 1;
                solve(col + 1, board, ans, leftRow, upperDiagonal, lowerDiagonal, n);
                board[row][col] = '.';
                leftRow[row] = 0;
                lowerDiagonal[row + col] = 0;
                upperDiagonal[n - 1 + col - row] = 0;
            }
        }
    }

    public:
        vector<vector<string>> solveNQueens(int n) {
            vector<vector<string>> ans;
            vector<string> board(n);
            string s(n, '.');
            for(int i = 0; i < n; i++) {
                board[i] = s;
            }
            vector<int> leftRow(n, 0), upperDiagonal(2 * n - 1, 0), lowerDiagonal(2 * n - 1, 0);
            solve(0, board, ans, leftRow, upperDiagonal, lowerDiagonal, n);
            return ans;
        }
};

TUF

```

Your previous code was restored from your local storage. Reset to default

	0	1	2	3	4	5	6	7
0	7	8	9	10	11	12	13	14
1	6	7	8	9	10	11	12	13
2	5	6	7	8	9	10	11	
3	4	5	6	7	8	9	10	
4	3	4	5	6	7	8	9	10
5	2	3	4	5	6	7	8	9
6	1	2	3	4	5	6	7	8
7	0	1	2	3	4	5	6	7

$n=8$



$2^{n-1}$

$$\ln \left[ \frac{(n-1)}{2} + \frac{(n! - n^n)}{2} \right] = 1$$

$$\frac{7}{2} + (7 - 0) = 14$$



```
i C++ Autocomplete
1  class Solution {
2
3  public:
4      void solve(int col, vector<string> &board, vector<vector<string>> &ans,
5                  vector<int> &leftRow,
6                  vector<int> &upperDiagonal, vector<int> &lowerDiagonal, int n) {
7          if(col == n) {
8              ans.push_back(board);
9              return;
10         }
11
12         for(int row = 0; row < n; row++) {
13             if(leftRow[row] == 0 && lowerDiagonal[row + col] == 0
14                 && upperDiagonal[n - 1 + col - row] == 0) {
15
16                 board[row][col] = 'Q';
17                 leftRow[row] = 1;
18                 lowerDiagonal[row + col] = 1;
19                 upperDiagonal[n - 1 + col - row] = 1;
20                 solve(col + 1, board, ans, leftRow, upperDiagonal, lowerDiagonal, n);
21                 board[row][col] = '.';
22                 leftRow[row] = 0;
23                 lowerDiagonal[row + col] = 0;
24                 upperDiagonal[n - 1 + col - row] = 0;
25             }
26         }
27     }
28
29     public:
30     vector<vector<string>> solveNQueens(int n) {
31         vector<vector<string>> ans;
32         vector<string> board(n);
33         string s(n, '.');
34         for(int i = 0; i < n; i++) {
35             board[i] = s;
36         }
37         vector<int> leftRow(n, 0), upperDiagonal(2 * n - 1, 0), lowerDiagonal(2 * n - 1, 0);
38         solve(0, board, ans, leftRow, upperDiagonal, lowerDiagonal, n);
39         return ans;
40     }
41
42 };

```

Your previous code was restored from your local storage. Reset to default

TUF