



SDE-PROBLEMS



# L14: N-Queens | Leetcode Hard | Backtracking



Includes paid promotion &gt;

Normal text

Arial

11

To exit full screen, press Esc

Esc

[ONnb1wdx2Ma&index=51](#)

## 6. K-th permutation Sequence

[https://www.youtube.com/watch?v=wT7gcXLYoao&list=PLgUwDviBIf0p4ozDR\\_kJJK](https://www.youtube.com/watch?v=wT7gcXLYoao&list=PLgUwDviBIf0p4ozDR_kJJK)[ONnb1wdx2Ma&index=55](#)

## Day10: (Recursion and Backtracking)

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

[https://www.youtube.com/watch?v=f2ic2Rsc9pU&list=PLgUwDviBIf0p4ozDR\\_kJJKO](https://www.youtube.com/watch?v=f2ic2Rsc9pU&list=PLgUwDviBIf0p4ozDR_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

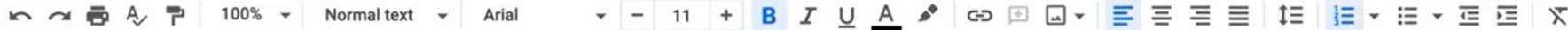
### 3. Find the element that appears once in sorted array, and rest element appears twice (Binary search)

TUF



0:06 / 36:54





[ONnb1wdx2Ma&index=51](#)

#### 6. K-th permutation Sequence

[https://www.youtube.com/watch?v=wT7gcXLYoao&list=PLgUwDviBIf0p4ozDR\\_kJJkONnb1wdx2Ma&index=55](https://www.youtube.com/watch?v=wT7gcXLYoao&list=PLgUwDviBIf0p4ozDR_kJJkONnb1wdx2Ma&index=55)

#### Day10: (Recursion and Backtracking)

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

[https://www.youtube.com/watch?v=f2ic2Rsc9pU&list=PLgUwDviBIf0p4ozDR\\_kJJkONnb1wdx2Ma&index=52](https://www.youtube.com/watch?v=f2ic2Rsc9pU&list=PLgUwDviBIf0p4ozDR_kJJkONnb1wdx2Ma&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)

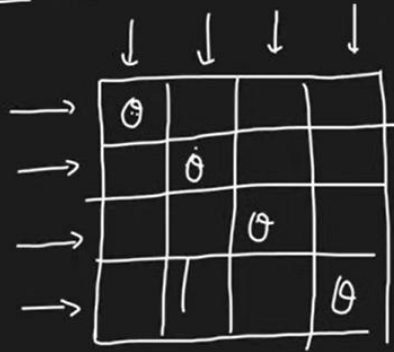
1. 1/N-th root of an integer (use binary search) (square root, cube root, ...)
2. Matrix Median
3. Find the element that appears once in sorted array, and rest element appears twice (Binary search)

USE COUPON

ON ANY COURSE

TAKEUFORWARD

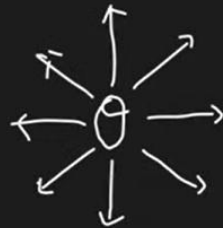
N=4 Queens



→ Every row → 1 Q

→ Every col → 1 Q

→ None of the Q attack each other



N=4 Queens

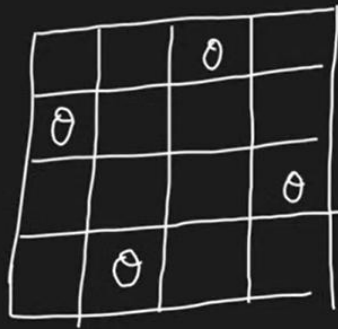
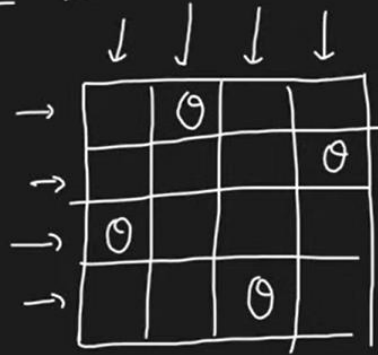
	↓	↓	↓	↓
→		Q		
→				Q
→	Q			
→			Q	

→ Every row → 1 Q

→ Every col → 1 Q

→ None of the Q attack each other

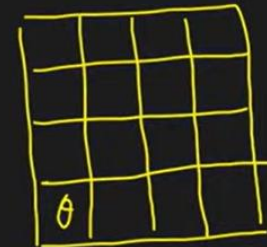
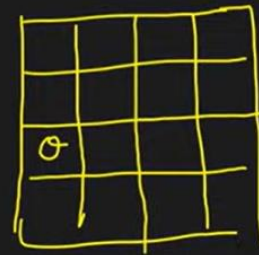
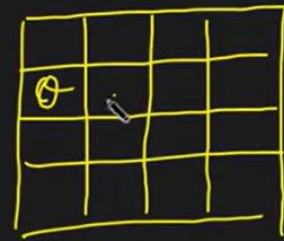
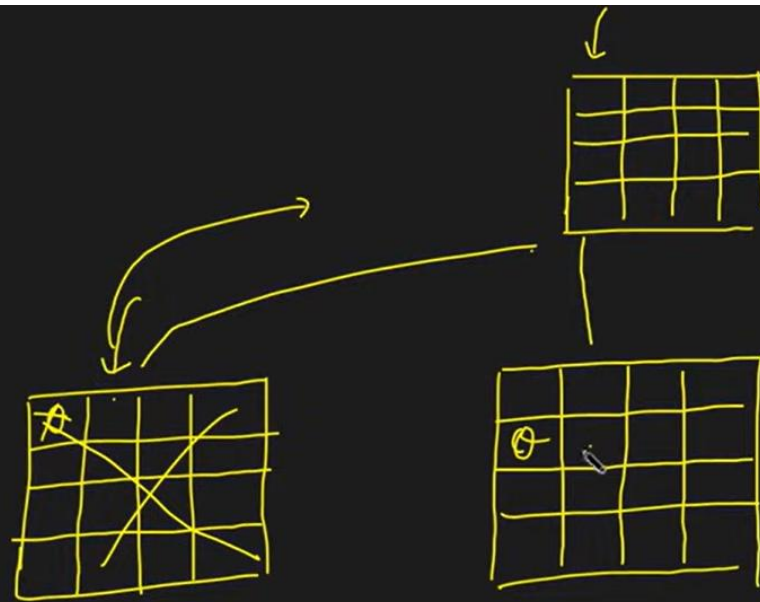
N=4 Queens



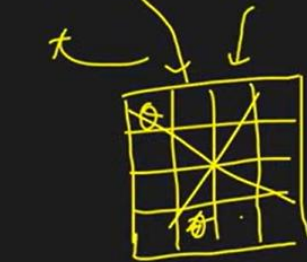
- Every row  $\rightarrow 1$  Q
- Every col  $\rightarrow 1$  Q
- None of the Q attack each other

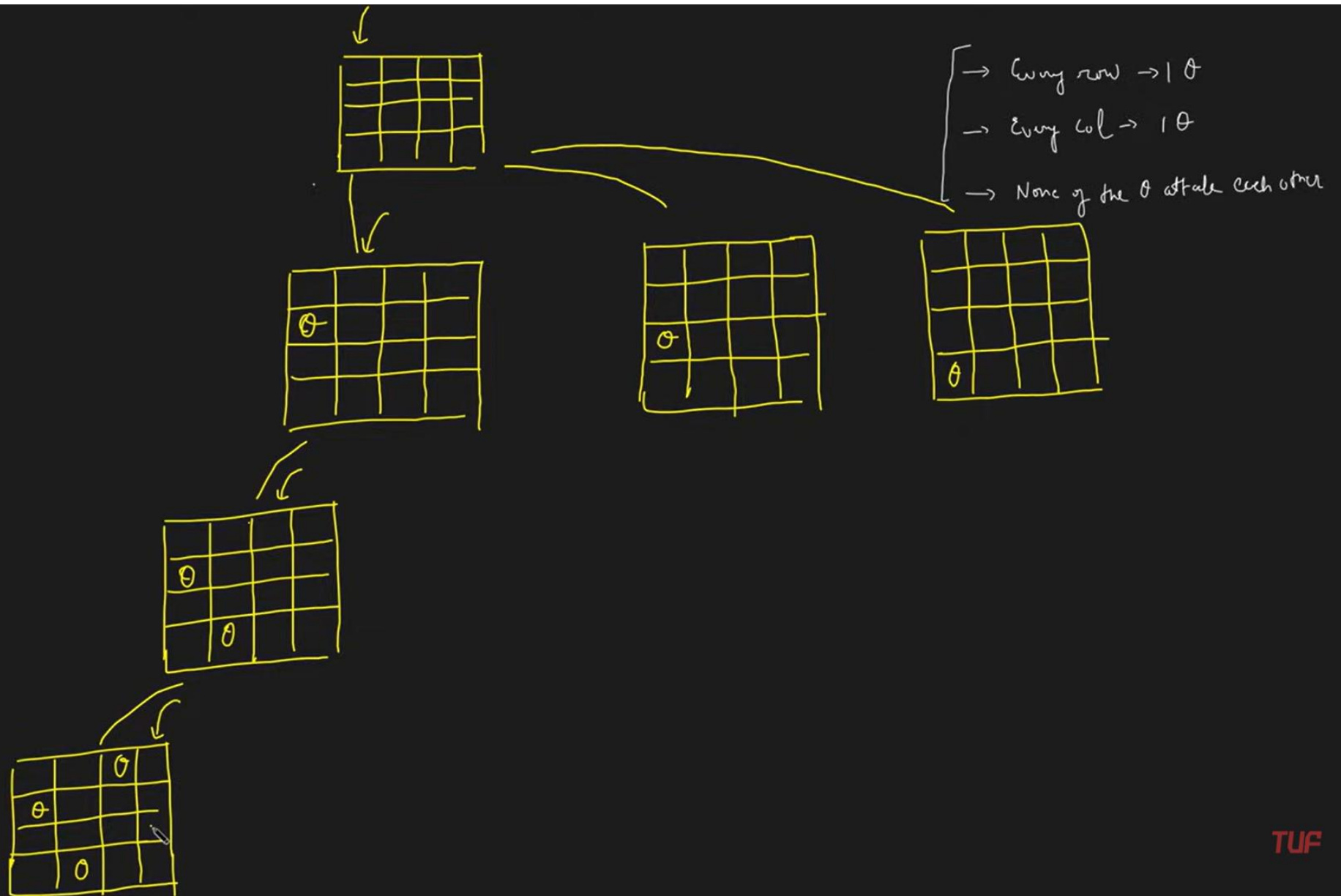
N

N x N



- Every row  $\rightarrow 1 \ 0$
- Every col  $\rightarrow 1 \ 0$
- None of the 0 attack each other







		0	
			0
0			


0			

0			
		0	

		0	
0			
0			

0			

0			

- Every row  $\rightarrow 1 \oplus$
- Every col  $\rightarrow 1 \oplus$
- None of the 0 attack each other



# L14. N-Queens | Leetcode Hard | Backtracking

→ Every col → 1  
→ None of the 0 attack each other

$f(col)$

$f(col+1)$

now col = 0

$f(col+1)$  = empty

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

$y (fill \rightarrow v)$

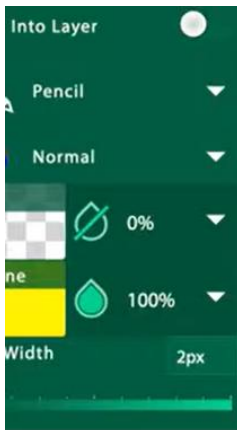
TUF

```

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 };

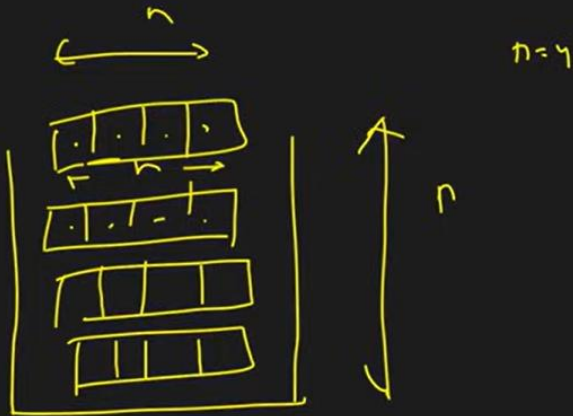
```

Your previous code was restored from your local storage. [Reset to default](#)



```
i C++ Autocomplete 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 };
```

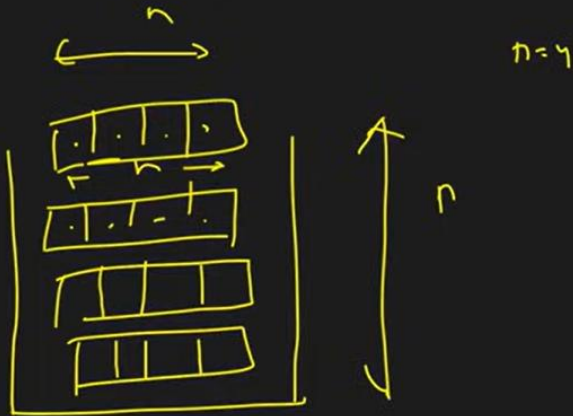
Your previous code was restored from your local storage. [Reset to default](#)



```

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 }
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 };

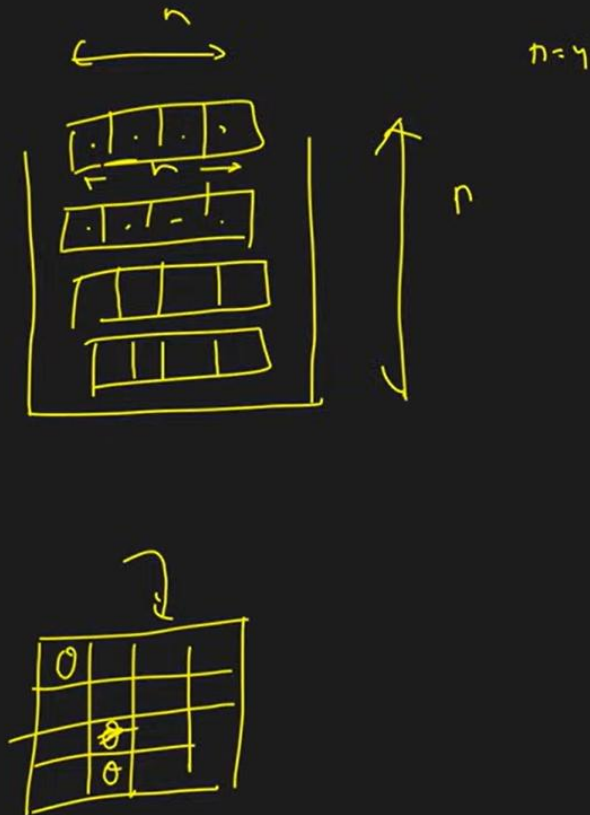
```



```

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 };

```



```

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 };

```



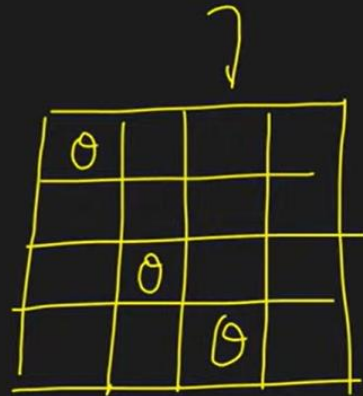
```

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](#)



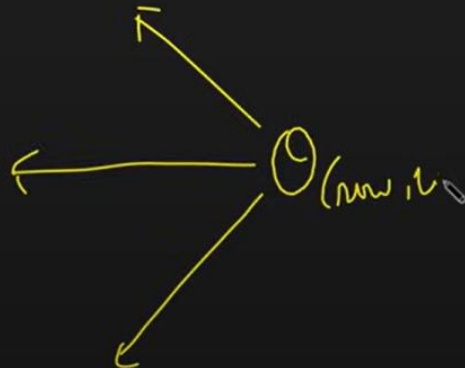
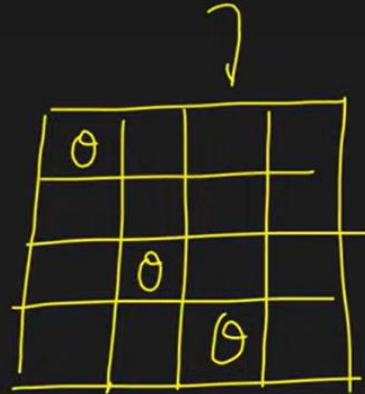


```

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 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 }

```

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

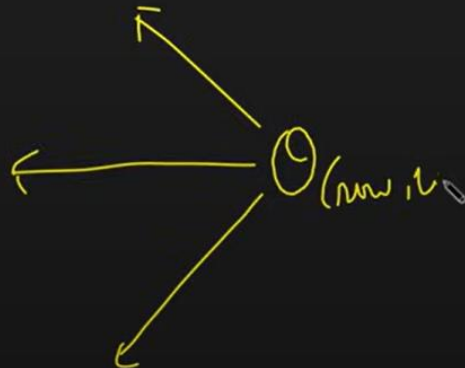
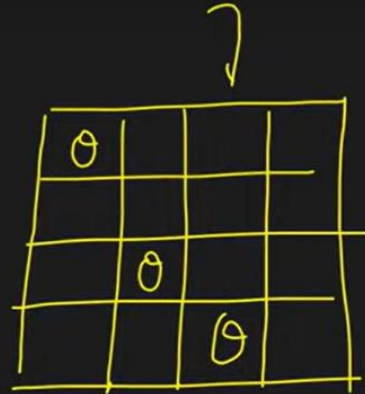


```
1 class Solution {
2 public:
3     bool isSafe1(int row, int col, vector<string> board, int n) {
4         // check upper diagonal
5         int duprow = row;
6         int dupcol = col;
7
8         while(row >= 0 && col >= 0) {
9             if(board[row][col] == 'Q') return false;
10            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 public:
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](#)

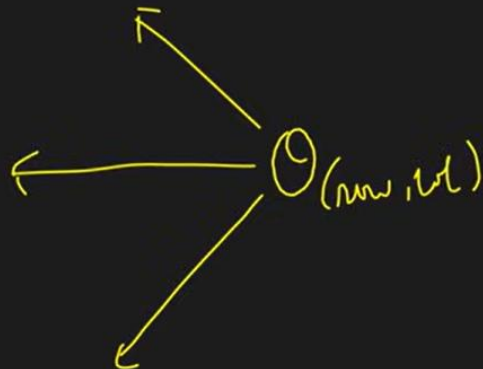
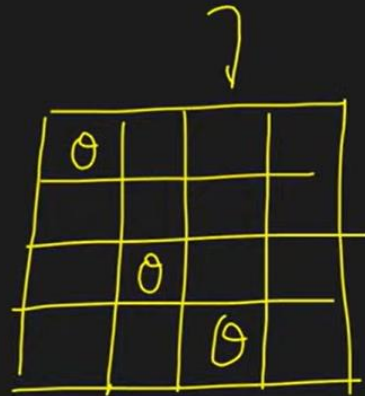
TUF

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



```
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 public:
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

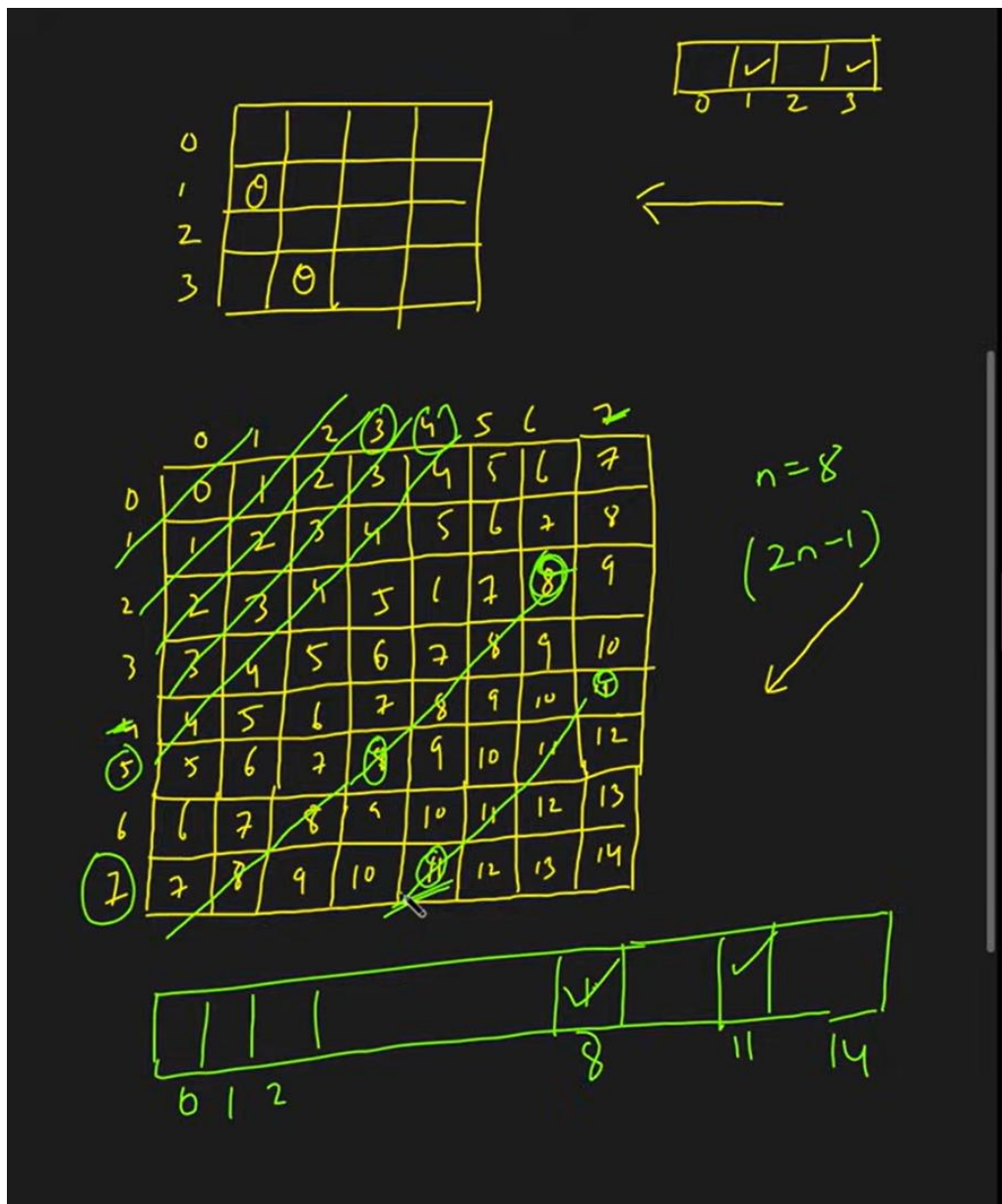


```

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 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 }

```





```

LeetCode Explore Problems Interview We're hiring, apply today! Premium
C++ Autocomplete

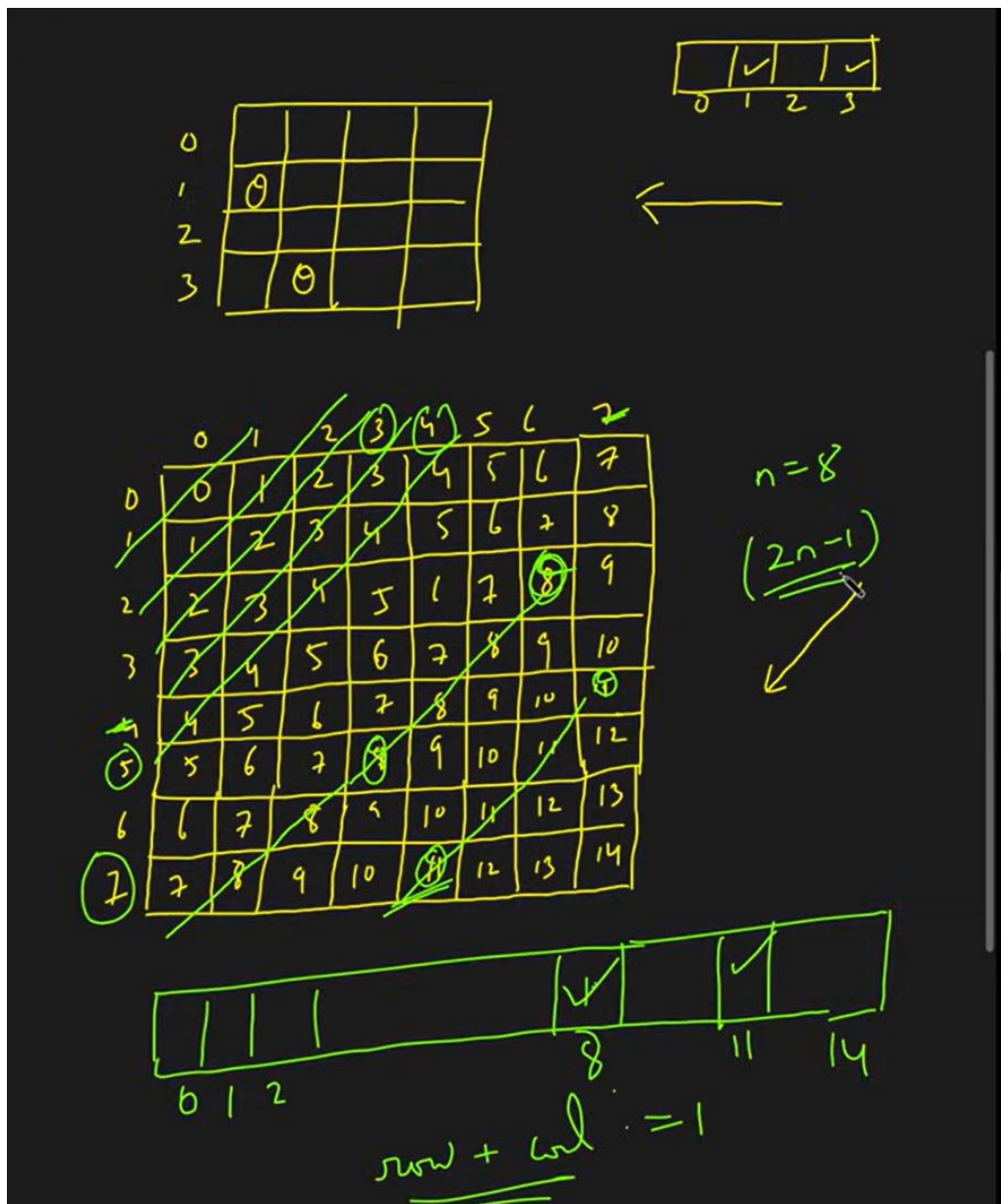
class Solution {
public:
    void solve(int col, vector<string> &board, vector<vector<string>> &ans,
               vector<int> &leftRow,
               vector<int> &upperDiagonal, vector<int> &lowerDiagonal, int n) {
        if(col == n) {
            ans.push_back(board);
            return;
     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

        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**

$n=8$

	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

$$(n-1) + (col-row) = 17$$

$$7 + (7-0) = 17$$

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

```

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 };

```

**TUF**

Your previous code was restored from your local storage. [Reset to default](#)



$n=8$

	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

$2n-1$

$$\text{len}[(n-1) + (col-row)] = 1$$

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

✓

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

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 };

```

TUF

Your previous code was restored from your local storage. [Reset to default](#)

$n=8$

	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

$2n-1$

$\text{bool}[(n-1) + (col-row)] = 1$

$7 + (7-0) = 14$

✓

LeetCode Explore Problems Interview **New** We're hiring, apply today! **Premium**

```

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 };

```

TUF

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

$n=8$

	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

$2n-1$

$\text{len}[(n-1) + (col-row)] = 1$

$7 + (7-0) = 14$

✓

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

```

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 };

```

TUF

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



$n=8$

	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

$2n-1$

$\text{bool}[(n-1) + (col-row)] = 1$   
 $7 + (7-0) = 14$

✓

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

```

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 };

```

TUF

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