02

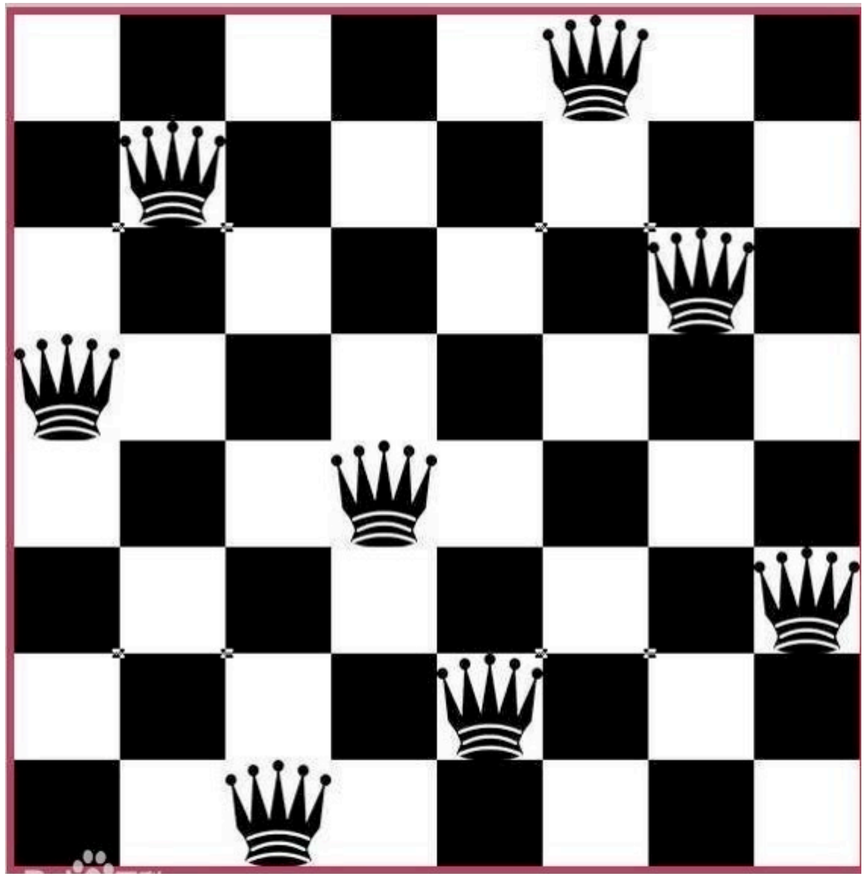
OPEN ORIENTED

凹 凸 实 验 室

八皇后

回溯&递归&动态规划

八皇后问题描述



八皇后问题国际西洋棋棋手马克斯·贝瑟尔于1848年提出：

在8×8格的国际象棋上摆放八个皇后，使其不能互相攻击，即任意两个皇后都不能处于同一行、同一列或同一斜线上，问有多少种摆法。

问题分析

02

- 如何表示
- 如何判断是否符合

问题分析 如何表示

	1	2	3	4
0		1		
1				1
2	1			
3			1	

1	1-4	2
2	1-4	4
3	1-4	1
4	1-4	3

问题分析

02

- 如何表示
- 如何判断是否符合

问题分析

如何判断是否符合

	1	2	3	4
0		1		
1				1
2	1			
3			1	

0	1-4	2
1	1-4	4
2	1-4	1
3	1-4	3

```
A[i] === A[j]
```

```
|A[i] - A[j]| === i - j
```

问题解决 暴力枚举


```
function checkQueen(a, n) {
  for(var i = 0; i < n; i++) {
    for(var j = i + 1; j < n; j++) {
      if((a[i] == a[j]) || Math.abs(a[i] - a[j]) == j - i) {
        return false;
      }
    }
  }
  return true;
}
function searchQueen() {
  var a = [];
  for(a[0] = 1; a[0] <= 4; a[0]++) {
    for(a[1] = 1; a[1] <= 4; a[1]++) {
      for(a[2] = 1; a[2] <= 4; a[2]++) {
        for(a[3] = 1; a[3] <= 4; a[3]++) {
          if(!checkQueen(a, 4)) {
            continue;
          } else {
            console.log(a);
          }
        }
      }
    }
  }
}
searchQueen();
//[ 2, 4, 1, 3 ]
//[ 3, 1, 4, 2 ]
```

问题解决 回溯思想


```
function checkQueen(a, n) {
  for(var i = 0; i <= n - 1; i++) {
    if((Math.abs(a[i] - a[n]) == n - i) || (a[i] == a[n])) {
      return false;
    }
  }
  return true;
}

function queen2() {
  var a = [];

  for(a[0] = 1; a[0] <= 4; a[0]++) {
    for(a[1] = 1; a[1] <= 4; a[1]++) {
      if(!checkQueen(a, 1)) continue;
      for(a[2] = 1; a[2] <= 4; a[2]++) {
        if(!checkQueen(a, 2)) continue;
        for(a[3] = 1; a[3] <= 4; a[3]++) {
          if(!checkQueen(a, 3)) {
            continue;
          } else {
            console.log(a);
          }
        }
      }
    }
  }
}

queen2();
```

问题解决 递归思想

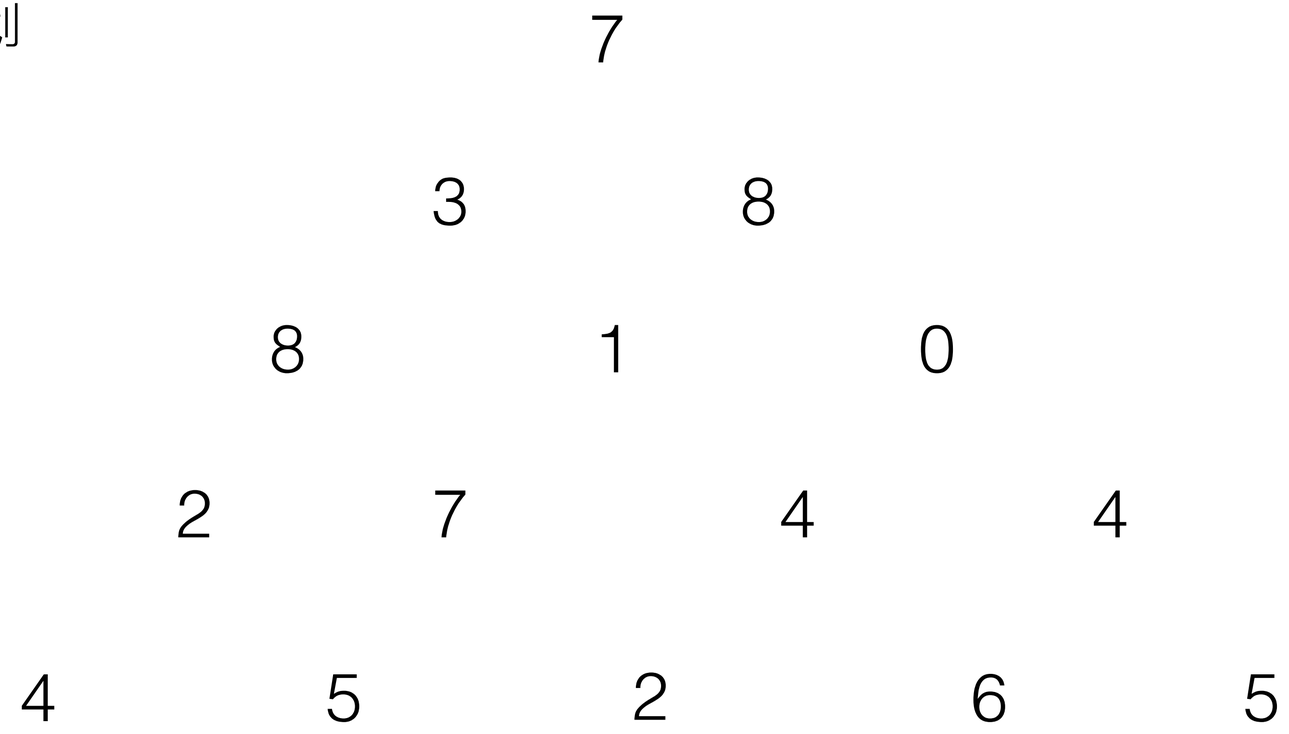

```
function checkQueen(a, n) {
  for(var i = 0; i <= n - 1; i++) {
    if((Math.abs(a[i] - a[n]) == n - i) || (a[i] == a[n])) {
      return false;
    }
  }
  return true;
}

var a = [], n = 4;
function recursion(k) {
  if(k > n) {
    console.log(a);
  } else {
    for(var i = 1;i <= n; i++) {
      a[k-1] = i;
      if(checkQueen(a, k-1)) {
        recursion(k + 1);
      }
    }
  }
}

recursion(1);
//[ 2, 4, 1, 3 ]
//[ 3, 1, 4, 2 ]
```

问题解决

动态规划



在上面的数字三角形中寻找一条从顶部到底边的路径，使得路径上所经过的数字之和最大。
路径上的每一步都只能往左下或 右下走。
要求出这个最大和

问题解决

动态规划

7

3 8

8 1 0

2 7 4 4

4 5 2 6 5

4 5 2 6 5

7 12 10 10

4 5 2 6 5

问题解决 动态规划


```
var A = [[7],[3,8],[8,1,0],[2,7,4,4],[4,5,2,6,5]];
var n = 5;
var maxSum = [[],[],[],[],[]];
var max = function(){
    for(var i = 0; i < n; i++){
        maxSum[n-1][i] = A[n-1][i];
    }
    for(var k = n-2; k >= 0; k--){
        for(var l = 0; l < i; l++){
            maxSum[k][l] = Math.max(maxSum[k+1][l],maxSum[k+1][l+1]) + A[k][l];
        }
    }
    console.log(maxSum[0][0]);
}
max();
```

THANKS

FOR YOUR WATCHING

02

OPEN ORIENTED

凹 凸 实 验 室