基础计算方法分析

jinguochong@gmail.com

金国充

Sudoku

一种逻辑性的数字填充游戏，玩家须以数字填进每一格，
而每行、每列和每个宫（即3x3的大格）有齐1至9所有数字。

5	3			7				
6			1	9	5			
	9	8					6	
8				6				3
4			8		3			1
7				2				6
	6					2	8	
			4	1	9			5
				8			7	9

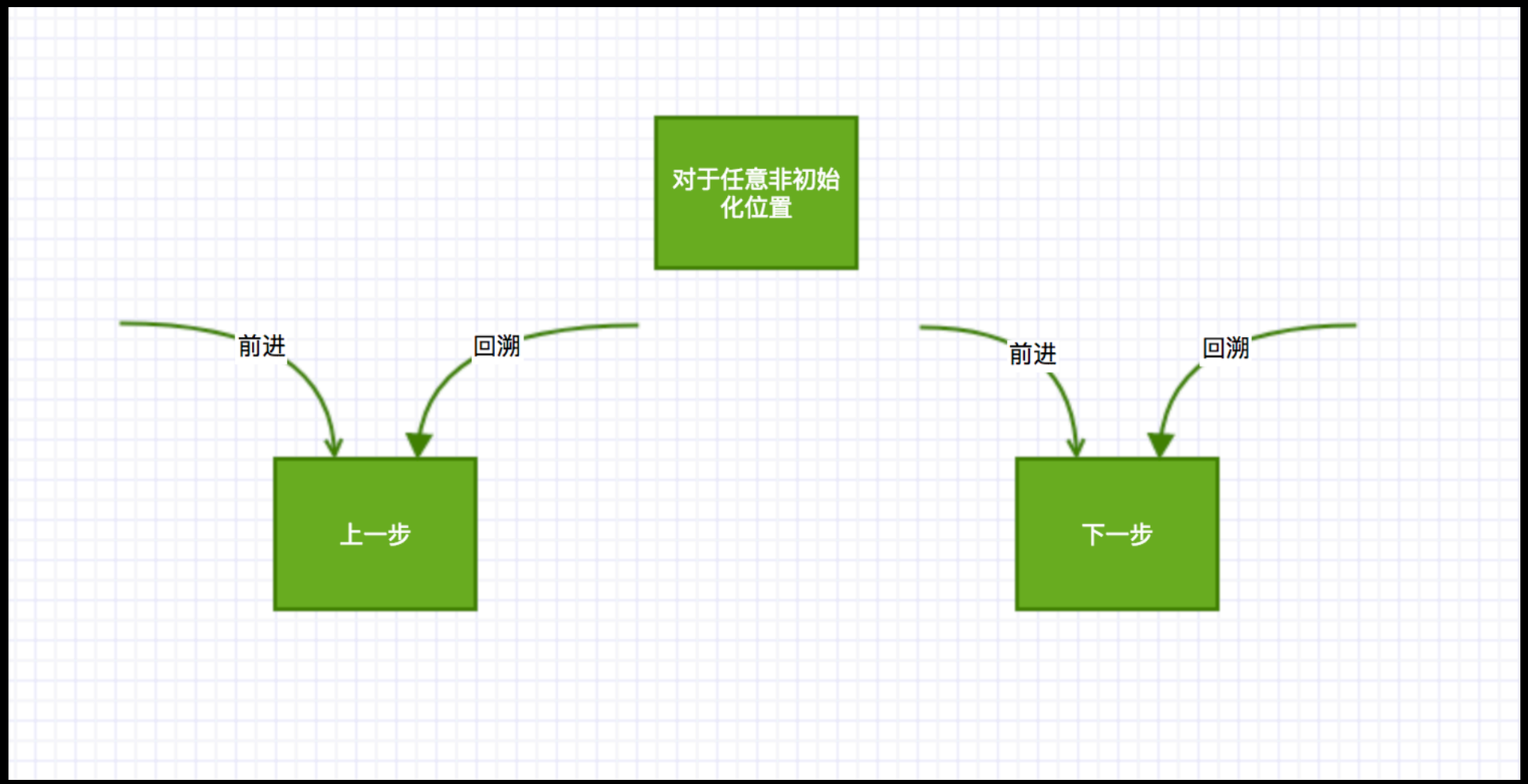
5	3	4	6	7	8	9	1	2
6	7	2	1	9	5	3	4	8
1	9	8	3	4	2	5	6	7
8	5	9	7	6	1	4	2	3
4	2	6	8	5	3	7	9	1
7	1	3	9	2	4	8	5	6
9	6	1	5	3	7	2	8	4
2	8	7	4	1	9	6	3	5
3	4	5	2	8	6	1	7	9

Sudoku

```
/**
 * 回溯法求解:
 * 认为数独位置有序;
 * 从0一直往后遍历
 * 暴力求解的略微改进;
 * guess从小到大进行
 * 思路:从0-80 顺序guess, 不需要维护guess表, 产生guess之后, 立即计算下一个位置的可能.
 * 如果产生回溯, 只需把当前guess的值++,
 *
 * 对于任意空位:
 * 1.关注上一步是前进还是回溯
 * 2.关注当前步是前进还是回溯
 * @param input 原始数独
 * @param position 当前位置
 * @param isBacktrack 上一步是否是回溯
 */
private static void backTrack(int[][] input, int position, boolean isBacktrack) {
```

没有什么算法是一个循环解决不了的,如果有,那就两个

Sudoku



SudokuTest3.java

Improve

对于空位:
input[0]是否正确依赖input[1]正确,
...
input[i]是否正确依赖input[i+1]正确,
...
直至input[80]是正确的,input[79]才是正确的.

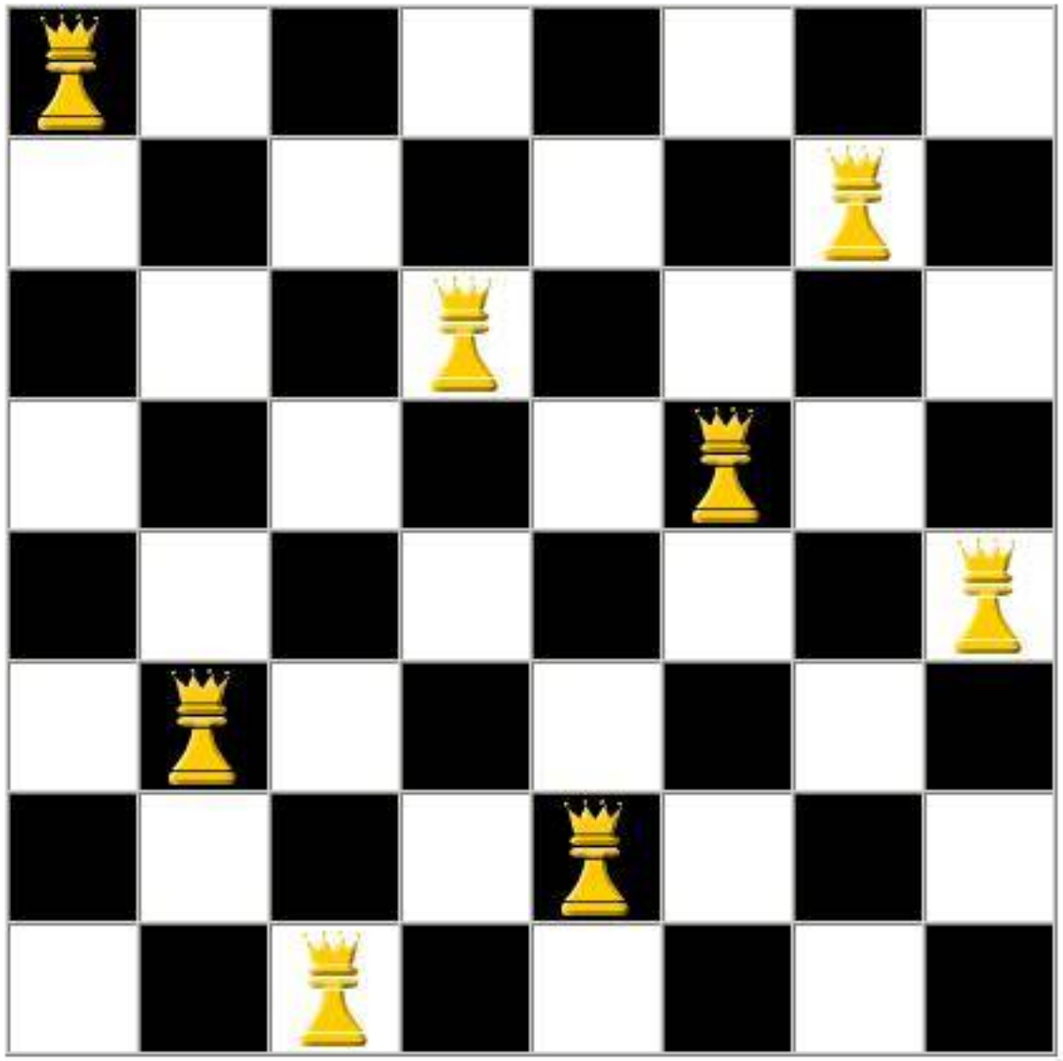
```java
private static boolean backTrack(int[][] input, int position) {
    if (position > 80) {
        return true;
    }
    int x = position % 9;//行
    int y = position / 9;//列
    if (input[y][x] != 0) {
        return backTrack(input, ++position);
    }
    for (int guess = 1; guess < 10; guess++) {
        if (isValid(input, position, guess)) {
            input[y][x] = guess;

            if (backTrack(input, ++position)) {
                return true;
            } else {
                input[y][x] = 0;
            }
        }
    }
    return false;
}
```

没有什么算法是一个循环解决不了的,如果有,那就两个

Queen8

背景：如何能够在8×8的国际象棋棋盘上放置八个皇后，使得任何一个皇后都无法直接吃掉其他的皇后？为了达到此目的，任两个皇后都不能处于同一条横行、纵行或斜线上。



后 是国际象棋棋局中
实力最强的一种棋子。
后 可横直斜走，且格数不限
类似中国象棋的 车

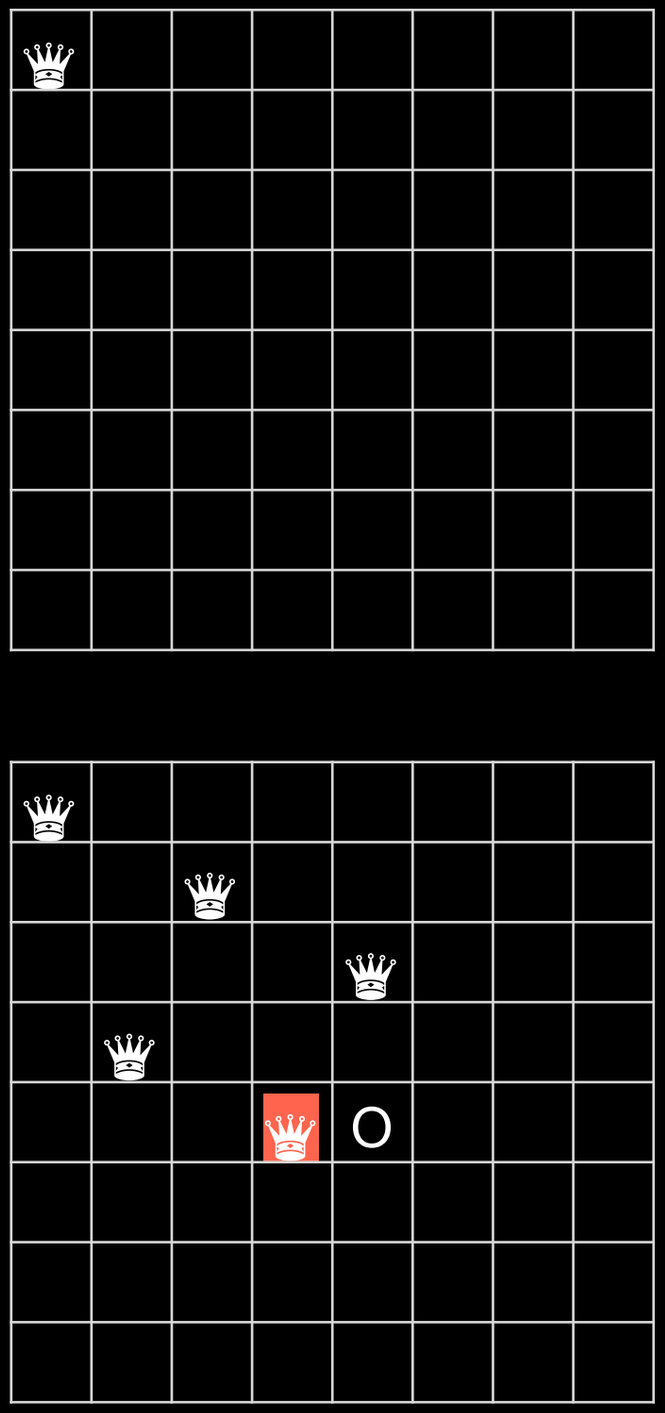
分析

每行每列都有一个,恰好8行8列,每一行,每一列都有一个

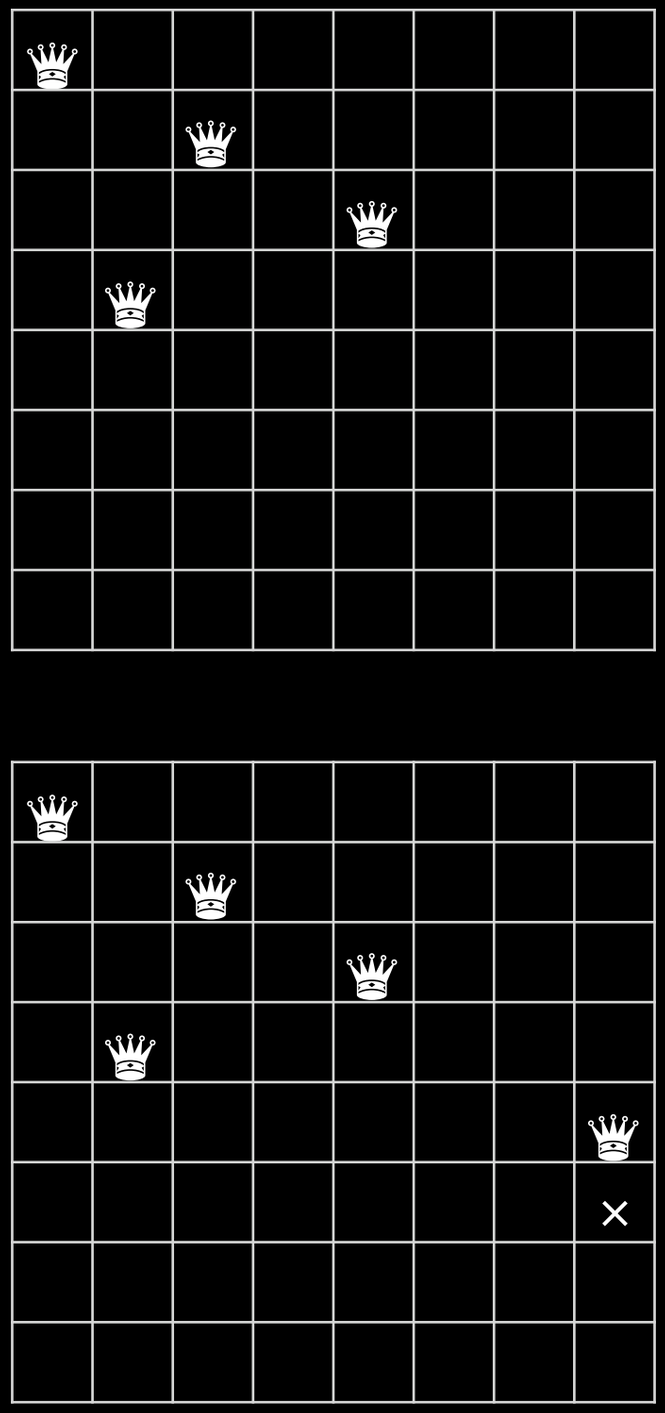
思路:从头开始遍历,
把合法的位置记录到数组里,然后跳转下一行;
把合法的位置记录到数组里,然后跳转下一行;
...
直至最后一行,记录合法位置,跳出循环,打印结果.
如果当前行没有合法的位置,说明上一行的位置不对,或者上上行的位置不对.
需要产生回溯,(和数独一样的思路)
清空上一行,或者上一行和上上行的值.

没有什么算法是一个循环解决不了的,如果有,那就两个

演进

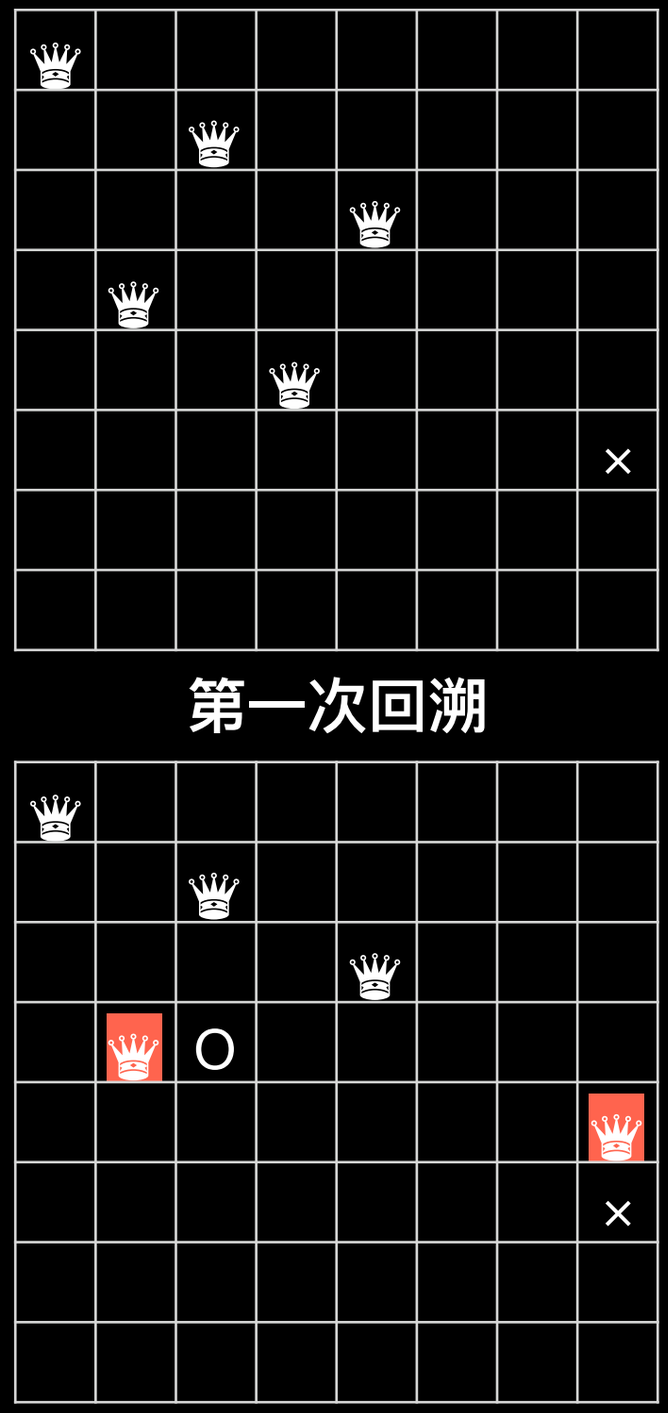


清空上一行皇后位置,从O位置开始



第二次回溯

第一次回溯

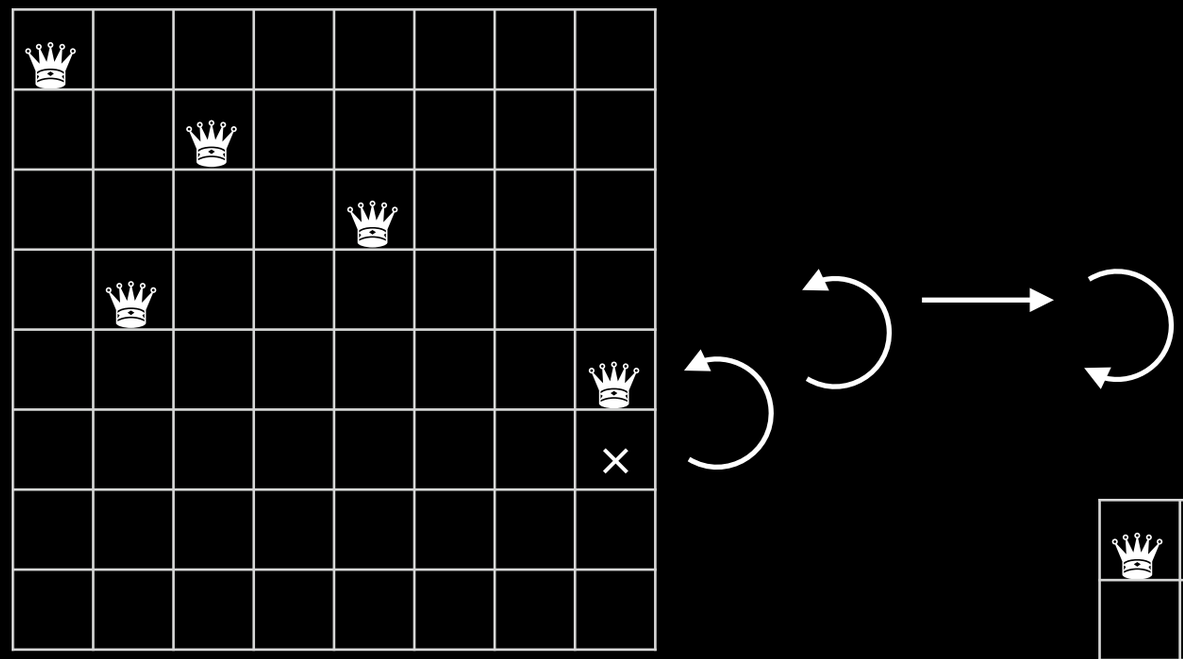


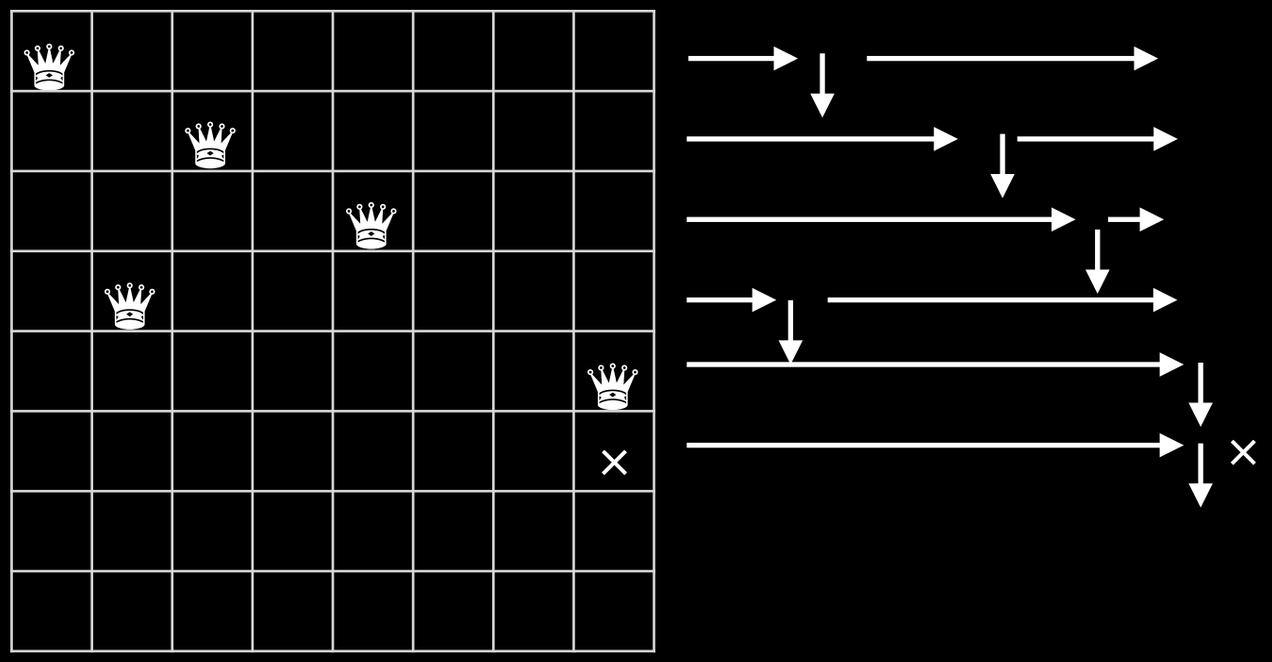
清空两个位置,从O位置开始

算法改进

能否不回溯呢?

不进行Backtracking,类比深度遍历树;





```java
/**
 * 每层递归设一个指针，从左往右，如果找到合法的就赋值，往下进行
 * 不会产生回溯过程
 */
private void cal(int pos) {
    int x = pos % 8;
    final int Y = pos / 8;
    if (Y == 8) {
        print(existPos);
        return;
    }
    while (x < 8) {
        if (isValid(pos)) {
            existPos[Y] = pos;
            int next = 8 * (Y + 1);
            cal(next);//向下
            //深度优先的精髓,需要清空当前行的赋值
            existPos[Y] = -1;
        }
        //向右
        x++;
        pos = x + Y * 8;
    }
}
```

拓展

八皇后问题可以推广为更一般的n皇后摆放问题：
这时棋盘的大小变为n×n，而皇后个数也变成n。
当且仅当n = 1或n ≥ 4时问题有解。

深度优先算法

深度优先搜索算法（英语：Depth-First-Search，简称DFS）是一种用于遍历或搜索树或图的算法。沿着树的深度遍历树的节点，尽可能深的搜索树的分支。当节点v的所在边都己被探寻过，搜索将回溯到发现节点v的那条边的起始节点。这一过程一直进行到已发现从源节点可达的所有节点为止。如果还存在未被发现的节点，则选择其中一个作为源节点并重复以上过程，整个进程反复进行直到所有节点都被访问为止。属于盲目搜索。

广度优先算法 Breadth-first algorithm

Dynamic Programing

Problem18
Problem81
Problem82

Problem18

给定一个三角形,从顶点到叶子节点,找出最大值对应的路径,求最大值是多少?

Find the maximum total from top to bottom of the triangle below:

```
                75
              95  64
            17  47  82
          18  35  87  10
        20  04  82  47  65
      19  01  23  75  03  34
    88  02  77  73  07  63  67
  99  65  04  28  06  16  70  92
41  41  26  56  83  40  80  70  33
41  48  72  33  47  32  37  16  94  29
53  71  44  65  25  43  91  52  97  51  14
70  11  33  28  77  73  17  78  39  68  17  57
91  71  52  38  17  14  91  43  58  50  27  29  48
63  66  04  68  89  53  67  30  73  16  69  87  40  31
04  62  98  27  23  09  70  98  73  93  38  53  60  04  23
```

思路

思路：如果使用Brute Force算法，需要把每条路径遍历一遍
如果使用dp思路，如下图：
//　　1
//　2　3
// 4　5　6
最后是要从2或者3出发，选取2+4，2+5，3+5，3+6中最大的
可简化为
//　　1
// 7　 9　// max(2+4，2+5)　max(3+5，3+6)
从后往前递推，得解。

$$a[i][j] += max(a[i+1][j], a[i+1][j+1])$$

时间复杂度

BF : 有 {2 pow(h -1) //h 树高} 这么多条路径，乘以（h-1）次的加法

DP : 第n层，需要计算2n加法，(n-1)次比较，
总：3n-1+3(n-1)-1+...+1 //n是树宽，正好等于树高
DP把BF算法中的pow降低到n平方

Problem81

题意：给出一个80×80的矩阵，找出在只允许向右和向下移动的情况下，
该矩阵中从左上角到右下角的最小路径和

目的找最短路径

$$\begin{pmatrix} 131 & 673 & 234 & 103 & 18 \\ 201 & 96 & 342 & 965 & 150 \\ 630 & 803 & 746 & 422 & 111 \\ 537 & 699 & 497 & 121 & 956 \\ 805 & 732 & 524 & 37 & 331 \end{pmatrix}$$

实现

arrs[i][j] += Math.min(arrs[i - 1][j], arrs[i][j - 1]);

```
private void traverseByLine(int[][] arrs) {
    for (int i = 1; i < 80; i++) {
        arrs[i][0] += arrs[i - 1][0];
        arrs[0][i] += arrs[0][i - 1];
    }

    for (int i = 1; i < 80; i++) {// row
        for (int j = 1; j < 80; j++) {// column

            arrs[i][j] += Math.min(arrs[i - 1][j], arrs[i][j - 1]);

        }
    }
}
```

没有什么算法是一个循环解决不了的,如果有,那就两个

Problem82

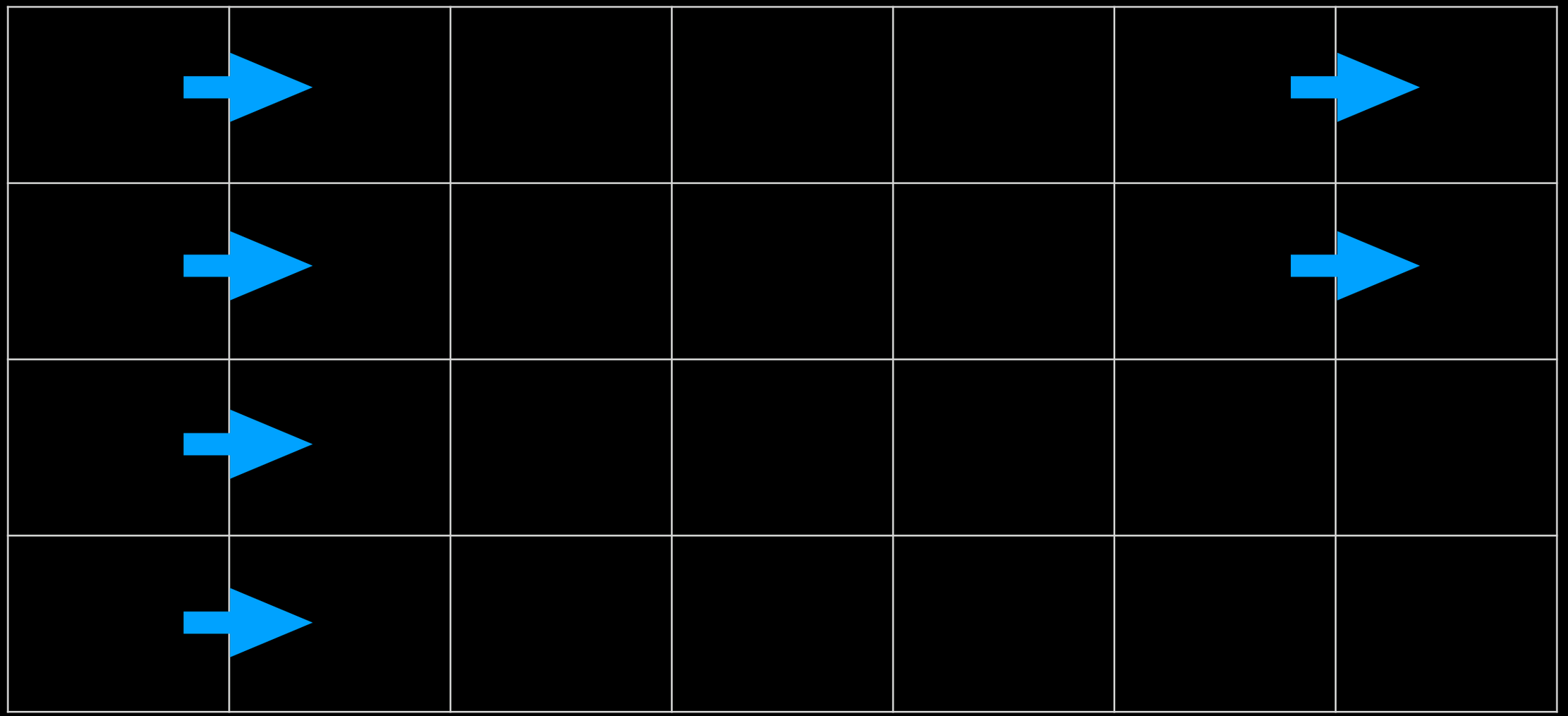
题意：给出一个80×80的矩阵，找出在只允许向上、向右和向下移动的情况下，
该矩阵中从左上角到右下角的最小路径和

目的找最短路径

$$\begin{pmatrix} 131 & 673 & 234 & 103 & 18 \\ 201 & 96 & 342 & 965 & 150 \\ 630 & 803 & 746 & 422 & 111 \\ 537 & 699 & 497 & 121 & 956 \\ 805 & 732 & 524 & 37 & 331 \end{pmatrix}$$

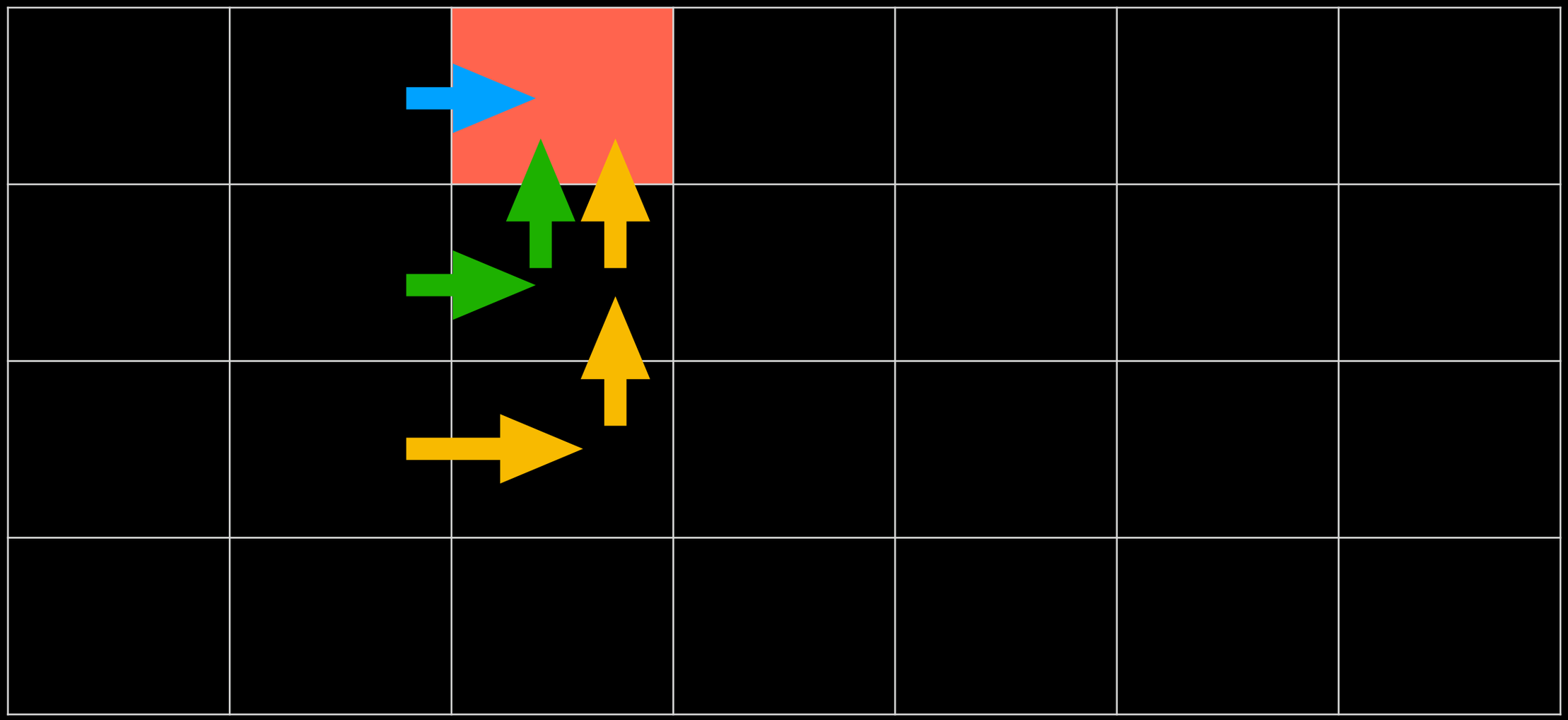
Problem82

第一列前往第二列和倒数第二列前往最后一列
前进方向只能是向右.



Problem82

一般性



Problem82

思路：81题的加强版
arr[i][j] = min(arr[i][k - 1]+sum[k, j])
其中，sum[k, j]表示同一列中，从k行到j行的和
三重循环后得到答案，再从arr[i][79]中选一个最小的。这个方法不是效率最高的方法，但是是最容易理解的...
这个思路是从其他位置到当前位置的算法;
应该也有从当前位置触发,判断是去哪个位置的

Problem82 Improve

空间换时间,少一个for循环,原理一样.属于算法优化

```
distance = new int[h][w];
//按列循环
for (int x = 0; x < w; x++) {
    //内循环向下
    for (int y = 0; y < h; y++) {
        //从上还是从左
        distance[y][x] = GRID[y][x] +
            Math.min(getValue(x - 1, y), getValue(x, y - 1));
    }
    //内循环向上
    for (int y = h - 1; y >= 0; y--) {
        //从下还是保持原样(上左最小)
        distance[y][x] = Math.min(GRID[y][x] + getValue(x, y + 1),
            distance[y][x]);
    }
}
```

谢谢