

当一个数组中大部分元素为0，或者为同一个值的数组时，可以使用稀疏数组来保存改数组

稀疏数组的处理方法是：

- 1. 记录数组一共有几行几列，有多少个不同的值
- 2. 把具有不同值的元素的行列及值记录在一个小规模数组中，从而缩小程序的规模

稀疏数组举例介绍

原始的二维数组


0	0	0	22	0	0	15
0	11	0	0	0	17	0
0	0	0	-6	0	0	0
0	0	0	0	0	39	0
91	0	0	0	0	0	0
0	0	28	0	0	0	0

转换为稀疏数组

	行 (row)	列 (col)	值 (value)
[0]	6	7	8
[1]	0	3	22
[2]	0	6	15
[3]	1	1	11
[4]	1	5	17
[5]	2	3	-6
[6]	3	5	39
[7]	4	0	91
[8]	5	2	28

- [0]记录原始数据有6行7列8个值
- [1]记录第一个值的坐标0行3列值为22以此类推

应用实例



- 稀疏sparsearray数组

先看一个实际的需求

➤ 编写的五子棋程序中，有存盘退出和续上盘的功能。

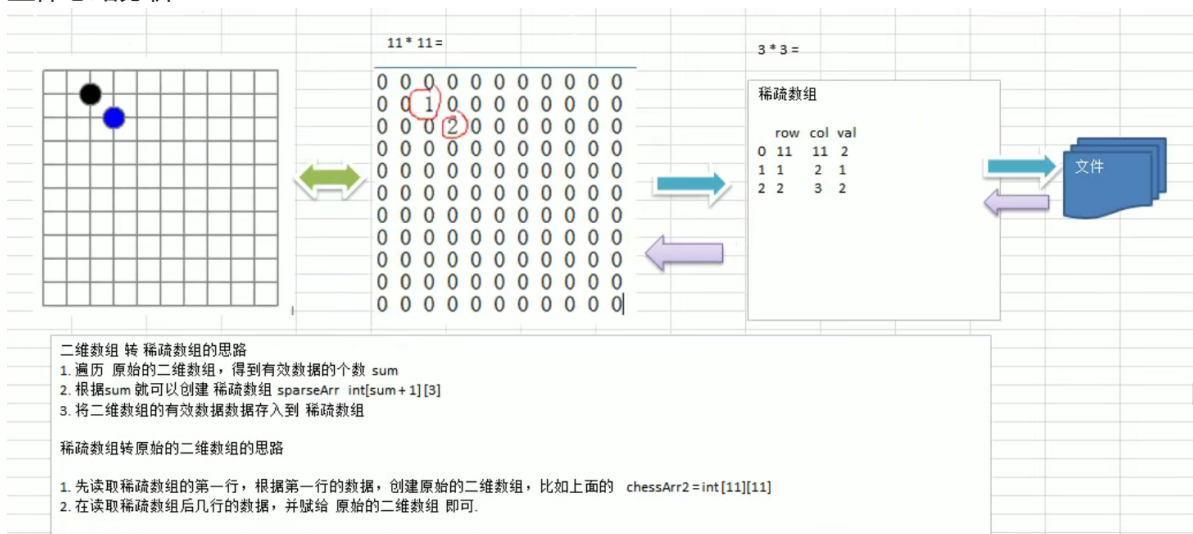


使用二维数组记录棋盘



➤ 分析问题:
因为该二维数组的很多值是默认值0, 因此记录了很多没有意义的数据.->稀疏数组。

1. 使用稀疏数组，来保留类似前面的二维数组（棋盘、地图等等）
2. 把稀疏数组存盘，并且可以重新恢复原来的二维数组
3. 整体思路分析



代码实现

```
/**
 * 稀疏数组
 *
 * @program: data-structure-algorithm
 * @author: ke
 * @create: 2021-09-03 16:59
 */
```

```

public class SparseArray {

    public static void main(String[] args) {
        /*
         * 创建一个原始的二维数组 11 * 11
         * 0表示没有棋子, 1表示黑子, 2表示篮子
         */
        int[][] chessArr1 = new int[11][11];
        chessArr1[1][2] = 1;
        chessArr1[2][3] = 2;
        chessArr1[3][5] = 1;
        // 输出原始的二维数组
        System.out.println("原始的二维数组");
        for (int[] row : chessArr1) {
            for (int data : row) {
                System.out.printf("%d\t", data);
            }
            System.out.println();
        }

        /*
         * 将二维数组转换为稀疏数组
         * 1. 先遍历二维数组, 得到非 0 数据的个数
         */
        int sum = 0;
        for (int i = 0; i < chessArr1.length; i++) {
            for (int j = 0; j < chessArr1[i].length; j++) {
                if (chessArr1[i][j] != 0) {
                    sum++;
                }
            }
        }

        // 创建对应的稀疏数组
        int[][] sparseArr = new int[sum + 1][3];

        // 给稀疏数组赋值
        sparseArr[0][0] = chessArr1.length;
        sparseArr[0][1] = chessArr1[0].length;
        sparseArr[0][2] = sum;

        // 用于记录是第几个非 0 数据
        int count = 0;
        // 遍历二维数组, 将非 0 的值存放倒稀疏数组中
        for (int i = 0; i < chessArr1.length; i++) {
            for (int j = 0; j < chessArr1[i].length; j++) {
                if (chessArr1[i][j] != 0) {
                    count++;
                    sparseArr[count][0] = i;
                    sparseArr[count][1] = j;
                    sparseArr[count][2] = chessArr1[i][j];
                }
            }
        }

        // 输出稀疏数组的形式
        System.out.println();
    }
}

```

```

        System.out.println("得到的稀疏数组为");
        for (int i = 0; i < sparseArr.length; i++) {
            System.out.printf("%d\t%d\t%d\t\n", sparseArr[i][0], sparseArr[i][1],
sparseArr[i][2]);
        }

        /*
         * 将稀疏数组恢复成原始的数组
         * 1.先读取稀疏数组的第一行，根据第一行的数据，创建原始的二维数组
         * 2.再读取稀疏数组后几行的数据，并赋值给原始的二维数组即可
         */
        // 1.先读取稀疏数组的第一行，根据第一行的数据，创建原始的二维数组
        int[][] chessArr2 = new int[sparseArr[0][0]][sparseArr[0][1]];

        // 2.再读取稀疏数组后几行的数据，并赋值给原始的二维数组即可
        for (int i = 1; i < sparseArr.length; i++) {
            chessArr2[sparseArr[i][0]][sparseArr[i][1]] = sparseArr[i][2];
        }

        // 输出恢复后的二维数组
        System.out.println();
        System.out.println("恢复后的二维数组");
        for (int[] row : chessArr2) {
            for (int data : row) {
                System.out.printf("%d\t", data);
            }
            System.out.println();
        }
    }
}

```

运行结果

原始的二维数组

```

0  0  0  0  0  0  0  0  0  0  0
0  0  1  0  0  0  0  0  0  0  0
0  0  0  2  0  0  0  0  0  0  0
0  0  0  0  0  1  0  0  0  0  0
0  0  0  0  0  0  0  0  0  0  0
0  0  0  0  0  0  0  0  0  0  0
0  0  0  0  0  0  0  0  0  0  0
0  0  0  0  0  0  0  0  0  0  0
0  0  0  0  0  0  0  0  0  0  0
0  0  0  0  0  0  0  0  0  0  0
0  0  0  0  0  0  0  0  0  0  0

```

得到的稀疏数组为：

```

11  11  3
1   2   1
2   3   2
3   5   1

```

恢复后的二维数组

```

0  0  0  0  0  0  0  0  0  0  0

```

0	0	1	0	0	0	0	0	0	0	0
0	0	0	2	0	0	0	0	0	0	0
0	0	0	0	0	1	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0

Process finished with exit code 0