52、图像物体的边界，考点 or 实现——数据结构/并查集

题目描述

给定一个二维数组M行N列，二维数组里的数字代表图片的像素，为了简化问题，仅包含像素1和5两种像素，每种像素代表一个物体，2个物体相邻的格子为边界，求像素1代表的物体的边界个数。

像素1代表的物体的边界指与像素5相邻的像素1的格子，边界相邻的属于同一个边界，相邻需要考虑8个方向（上，下，左，右，左上，左下，右上，右下）。

其他约束

地图规格约束为：

0<M<100
0<N<100

1）如下图，与像素5的格子相邻的像素1的格子（0,0）、（0,1）、（0,2）、（1,0）、（1,2）、（2,0）、（2,1）、（2,2）、（4,4）、（4,5）、（5,4）为边界，另（0,0）、（0,1）、（0,2）、（1,0）、（1,2）、（2,0）、（2,1）、（2,2）相邻，为1个边界，（4,4）、（4,5）、（5,4）相邻，为1个边界，所以下图边界个数为2。

	1	1	1	1	1	1
	1	5	1	1	1	1
	1	1	1	1	1	1
	1	1	1	1	1	1
	1	1	1	1	1	1
	1	1	1	1	1	5

2）如下图，与像素5的格子相邻的像素1的格子（0,0）、（0,1）、（0,2）、（1,0）、（1,2）、（2,0）、（2,1）、（2,2）、（3,3）、（3,4）、（3,5）、（4,3）、（4,5）、（5,3）、（5,4）、（5,5）为边界，另这些边界相邻，所以下图边界个数为1。

	1	1	1	1	1	1
	1	5	1	1	1	1
	1	1	1	1	1	1
	1	1	1	1	1	1
	1	1	1	1	5	1
	1	1	1	1	1	1

注：（2,2）、（3,3）相邻。

输入描述

第一行，行数M，列数N

第二行开始，是M行N列的像素的二维数组，仅包含像素1和5

输出描述

像素1代表的物体的边界个数。

如果没有边界输出0（比如只存在像素1，或者只存在像素5）。

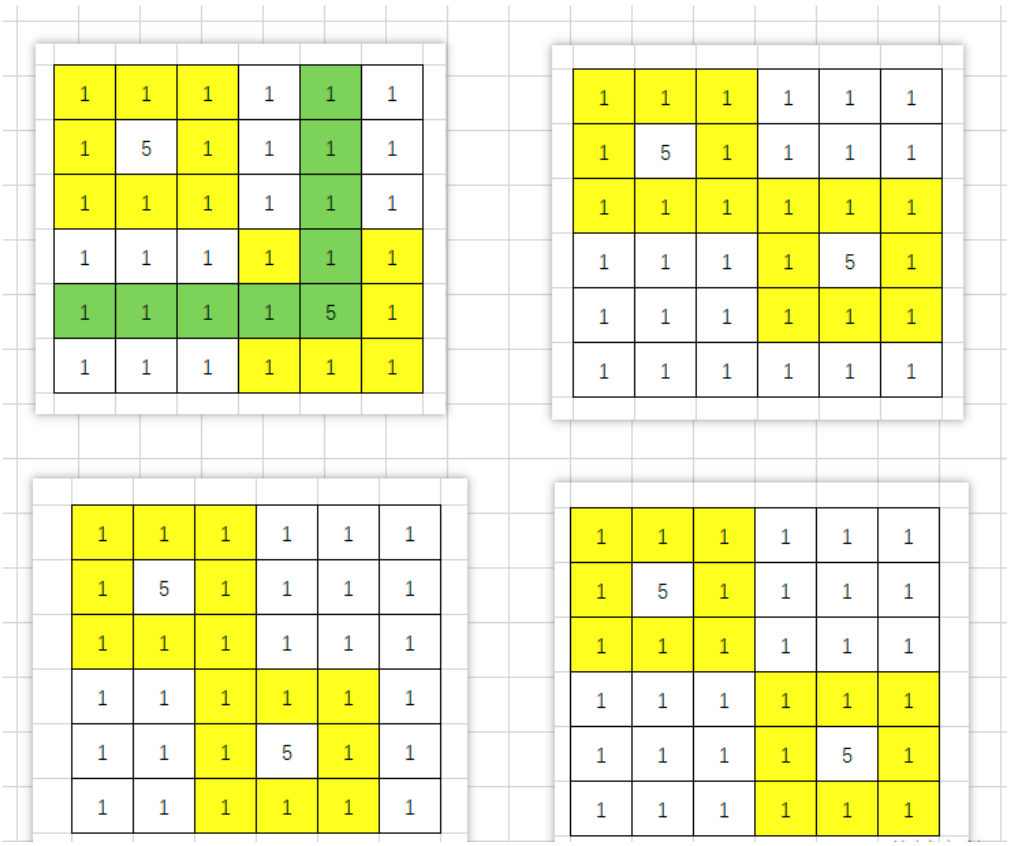
用例

输入	6 6 1 1 1 1 1 1 1 5 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 5
输出	2
说明	参考题目描述部分

输入	6 6 1 1 1 1 1 1 1 5 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 5 1 1 1 1 1 1 1
输出	1
说明	参考题目描述部分

题目解析

本题可以使用并查集。



而判断两个边界相邻的条件是：两个像素5坐标满足：|x1-x2| <=3 && |y1-y2| <=3

即如上图绿色线就是另一个像素5的移动范围边界。

因此，本题可以转化为求解像素5是否联通的并查集求解。

Java算法源码

```java
import java.util.ArrayList;
import java.util.Scanner;

public class Main {
  public static void main(String[] args) {
    Scanner sc = new Scanner(System.in);

    int m = sc.nextInt();
    int n = sc.nextInt();

    int[][] matrix = new int[m][n];
    for (int i = 0; i < m; i++) {
      for (int j = 0; j < n; j++) {
        matrix[i][j] = sc.nextInt();
      }
    }

    System.out.println(getResult(matrix, m, n));
  }

  public static int getResult(int[][] matrix, int m, int n) {
    ArrayList<int[]> brands = new ArrayList<>();

    for (int i = 0; i < m; i++) {
      for (int j = 0; j < n; j++) {
        if (matrix[i][j] == 5) {
          brands.add(new int[] {i, j});
        }
```

```
      }
    }

    int k = brands.size();
    if (k == 0 || k == n * m) return 0;

    UnionFindSet ufs = new UnionFindSet(k);

    for (int i = 0; i < k; i++) {
      int x1 = brands.get(i)[0], y1 = brands.get(i)[1];
      for (int j = i + 1; j < k; j++) {
        int x2 = brands.get(j)[0], y2 = brands.get(j)[1];

        if (Math.abs(x2 - x1) <= 3 && Math.abs(y2 - y1) <= 3) {
          ufs.union(i, j);
        }
      }
    }

    return ufs.count;
  }
}

class UnionFindSet {
  int[] fa;
  int count;

```

```
  public UnionFindSet(int n) {
    this.count = n;
    this.fa = new int[n];
    for (int i = 0; i < n; i++) this.fa[i] = i;
  }

  public int find(int x) {
    if (x != this.fa[x]) {
      return (this.fa[x] = this.find(this.fa[x]));
    }
    return x;
  }

  public void union(int x, int y) {
    int x_fa = this.find(x);
    int y_fa = this.find(y);

    if (x_fa != y_fa) {
      this.fa[y_fa] = x_fa;
      this.count--;
    }
  }
}
```

JS算法源码

```
/* JavaScript Node ACM模式 控制台输入获取 */
const readline = require("readline");

const rl = readline.createInterface({
  input: process.stdin,
  output: process.stdout,
});

const lines = [];
let m;
let n;
rl.on("line", (line) => {
  lines.push(line);

  if (lines.length === 1) {
    [m, n] = lines[0].split(" ").map(Number);
  }

  if (m && lines.length === m + 1) {
    lines.shift();

    const matrix = lines.map((line) => line.split(" ").map(Number));

    console.log(getBrandCount(matrix, m, n));
```

```
    lines.length = 0;
  }
});

function getBrandCount(matrix, m, n) {
  const brands = [];

  for (let i = 0; i < m; i++) {
    for (let j = 0; j < n; j++) {
      if (matrix[i][j] === 5) {
        brands.push([i, j]);
      }
    }
  }

  let len = brands.length;
  if (len === 0 || len === n * m) {
    return 0;
  }

  const ufs = new UnionFindSet(len);

  for (let i = 0; i < len; i++) {
    for (let j = i + 1; j < len; j++) {
      let [x1, y1] = brands[i];
      let [x2, y2] = brands[j];

      if (Math.abs(x2 - x1) <= 3 && Math.abs(y2 - y1) <= 3) {
        ufs.union(i, j);
      }
    }
  }

  return ufs.count;
}

class UnionFindSet {
  constructor(n) {
    this.fa = [];
    for (let i = 0; i < n; i++) {
      this.fa.push(i);
    }
    this.count = n;
  }

  find(x) {
    if (x !== this.fa[x]) {
      this.fa[x] = this.find(this.fa[x]);
      return this.fa[x];
    }
    return x;
  }

  union(x, y) {
```

```
    let x_fa = this.fa[x];
    let y_fa = this.fa[y];
    if (x_fa !== y_fa) {
      this.fa[y_fa] = x_fa;
      this.count--;
    }
  }
}
```

Python算法源码

```python
# 输入获取
m, n = map(int, input().split())
matrix = [list(map(int, input().split())) for _ in range(m)]


# 并查集
class UnionFindSet:
    def __init__(self, n):
        self.fa = [idx for idx in range(n)]
        self.count = n

    def find(self, x):
        if x != self.fa[x]:
            self.fa[x] = self.find(self.fa[x])
            return self.fa[x]
        return x

    def union(self, x, y):
        x_fa = self.find(x)
        y_fa = self.find(y)

        if x_fa != y_fa:
            self.fa[y_fa] = x_fa
            self.count -= 1

```

```python

# 算法入口
def getResult():
    brands = []

    for i in range(m):
        for j in range(n):
            if matrix[i][j] == 5:
                brands.append((i, j))

    k = len(brands)
    if k == 0 or k == n * m:
        return 0

    ufs = UnionFindSet(k)

    for i in range(k):
        x1, y1 = brands[i]
        for j in range(i + 1, k):
            x2, y2 = brands[j]

            if abs(x2 - x1) <= 3 and abs(y2 - y1) <= 3:
                ufs.union(i, j)

    return ufs.count


# 算法调用
print(getResult())
```