蛟龙四班

深度优先搜索(题目选讲)

Mas

#1168、连通块

【题目描述】

一个 $n \times m$ 的方格图，一些格子被涂成了黑色，在方格图中被标为 1 ，白色格子标为 0 。问有多少个四连通（上下左右四个方向）的黑色格子连通块。四连通的黑色格子连通块指的是一片由黑色格子组成的区域，其中的每个黑色格子能通过四连通的走法（上下左右），只走黑色格子，到达该联通块中的其它黑色格子。

【输入】

第一行两个整数 $n, m(1 \leq n, m \leq 100)$ ，表示一个 $n \times m$ 的方格图。

接下来 n 行，每行 m 个整数，分别为 0 或 1 ，表示这个格子是白色还是黑色。

【输出】

一行一个整数 ans ，表示图中有 ans 个黑色格子连通块。

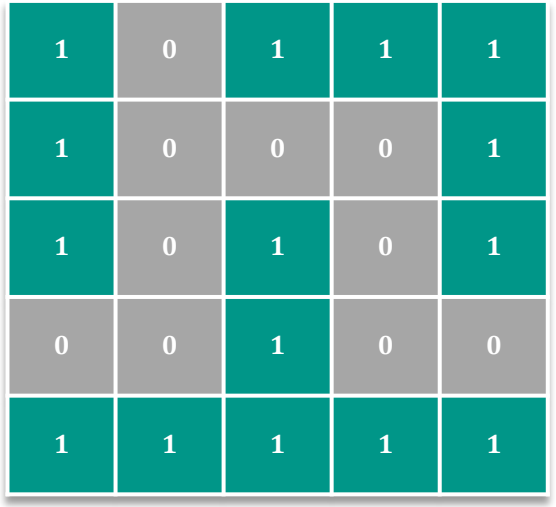
【输入样例】

```
3 3
1 1 1
0 1 0
1 0 1
```

【输出样例】

```
3
```

#1168、连通块



洪水填充($Flood\ fill$):

从一个起始节点开始把附近与其**连通**的节点标记成一颜色

直到封闭区域内的所有节点都被处理过为止

本题仅与上下左右四个方向连通

遍历二维数组,若该位置是黑色,且未被标记过颜色

那么将其作为起点,对周边黑色未染色格子进行染色

并将周边黑色未染色格子作为新的起点

遍历二维数组过程中发现的起点数量即为连通块数量

#1168、连通块


```
int maze[101][101], cnt, row, col;
bool vis[101][101];
struct node
{
  int x, y;
};
int dir[4][2] = {{1, 0}, {-1, 0}, {0, 1}, {0, -1}};
void bfs(int x, int y)
{
  queue<node> q;
  q.push({x, y});
  while (!q.empty())
  {
    node t = q.front();
    q.pop();
    for (int i = 0; i < 4; i++)
    {
      int tx = t.x + dir[i][0], ty = t.y + dir[i][1];
      if (tx >= 0 && tx < row && ty >= 0 && ty < col && !vis[tx][ty] && maze[tx][ty])
      {
        vis[tx][ty] = true;
        q.push({tx, ty});
      }
    }
  }
}
```

几个条件表达式能否调换顺序?

时间复杂度?

能否用DFS实现?

#1432、抠图的烦恼

题目描述

小 Z 老师需要设计很多的宣传材料。所以他经常要用 $Photoshop$ 来对一些素材进行抠图。

每张图片都是由 $w \times h$ 个像素点组成的。每个像素点用 $0 \sim 255$ 的数字表示颜色，其中 0 表示黑色 255 表示白色。

小震老师需要抠出的区域用黑色的线圈包围起来了，请你把黑线以外的所有部分变成黑色,黑色线圈内部的像素点不变。

输入格式

第一行两个正整数 $w, h(1 \leq w, h \leq 500)$

接下来 w 行，每行 h 个 $0 \sim 255$ 的数字

输出格式

输出处理后的图像矩阵

输入样例

```
4 4
1 2 3 4
5 0 0 0
6 0 7 0
8 0 0 0
```

输出样例

```
0 0 0 0
0 0 0 0
0 0 7 0
0 0 0 0
```

闭合区域以外的部分

必然与矩阵的上下左右四个边界连通

闭合区域内的部分必然不与外界连通

枚举边界上的点作为起点

将闭合区域进行染色成0即可

#257、污染侵袭

题目描述

在一个 $n \times m$ 的格子矩阵上，每个格子上的数字是 0 或者 1 。

数字 1 表示格子已经被污染了，数字 0 表示格子是干净的。

干净的格子可以通过上下左右四个方向和周围的干净格子连成一片干净的区域。

如果这个干净的区域的周围**完全**被污染的格子包围，那么这片干净的区域中的所有的格子都会被污染。也就是这片区域所有格子上的数字都会变为 1 。

求最终这 $n \times m$ 的格子矩阵的状态，每个格子是否被污染。

数据输入

第一行两个数字 n 和 m 。表示格子矩阵的大小。

接下来 n 行，每行 m 个用空格隔开的数字，每个数字非 0 即 1 .表示是否被污染。

数据输出

输出 n 行，每行 m 个用空格隔开的数字，行末无空格。每个数字非 0 即 1 ，表示格子相应格子是否被污染。

样例输入1

```
4 4
0 1 1 1
0 1 0 1
1 0 0 1
1 1 1 0
```

样例输出1

```
0 1 1 1
0 1 1 1
1 1 1 1
1 1 1 0
```

范围说明

对于 30% 的数据有： $1 \leq n, m \leq 20$ 。

对于 100% 的数据有： $1 \leq n, m \leq 1000$ 。

#257、污染侵袭

与上题类似

封闭区域以内的0不与外界的0连通

考虑将外界所有0进行染色,染色成−1

再将染色后的矩阵中的0标记为1

输出时,将−1还原成0即可

```
for (int i = 0; i < n; i++)
  for (int j = 0; j < m; j++)
    if ((i == 0 || j == 0 || i == n - 1 || j == m - 1) && !vis[i][j] && !maze[i][j])
      bfs(i, j);
for (int i = 0; i < n; i++)
  for (int j = 0; j < m; j++)
    cout << (maze[i][j] == -1 ? "0" : "1") << " \n"[j == m - 1];
```

#1444、木棍与等边三角形

题目描述

地上有 n 根木棍，每根木棍的长度为 l_i 。好奇的 Mas 想把这堆木棍拼成一个等边三角形，并且要利用上每一根木棍。
比如有长度 $1, 2, 3, 3$ 的木棍，你可以把 $1, 2$ 、 3 和 3 分别拼成三角形的三条边，这样就形成了一个等边三角形。
木棍可能有很多，请你帮 Mas 判断木棍是否能拼成等边三角形。

输入格式

第一个一个整数 T ,代表有 T 组数据
每数据有两行，每组数据第一行一个整数 n ，代表木棍的数量。接下来一行 n 个数表示每根木棍的长度。

输出格式

对于每一组数据，如果能拼成等边三角形输出 `yes` ,否则输出 `no` 。每组数据的输出结果占一行。

输入样例

```
1
4
1 2 3 3
```

输出样例

```
yes
```

数据规模

对于 100% 的数据 $1 \leq T \leq 5, 3 \leq n \leq 20, 1 \leq l_i \leq 100000$

#1444、木棍与等边三角形

记所有木棍长度之和为sum

当sum不是3的倍数显然无法拼凑成功

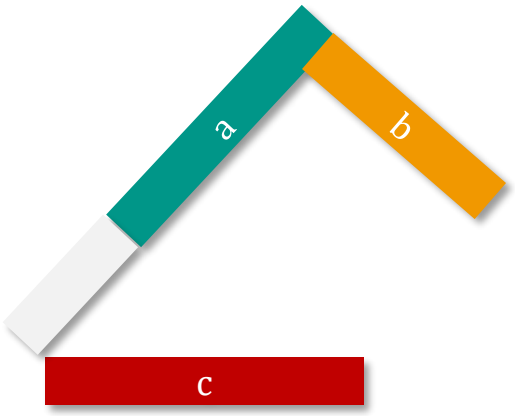
当最长木棍长度超过$\frac{sum}{3}$显然无法拼凑成功

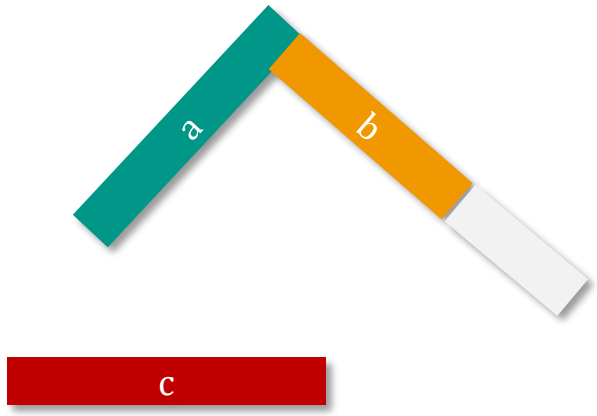
设三条边分别为a、b、c

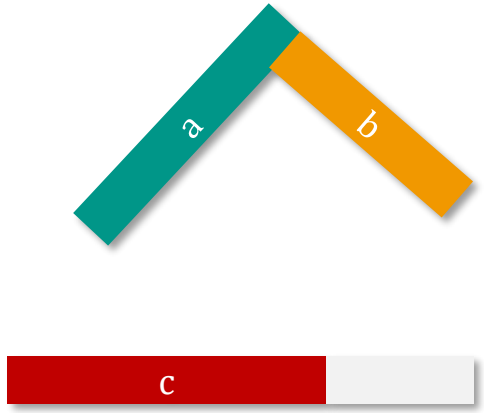
对于每一根木棍要么被加到a上，要么被加到b上，要么被加到c上

对三种情况进行搜索

```
bool judge(int a, int b, int c, int k)
{
    if (a == b && b == c && k == n)
        return true;
    if (a > len || b > len || c > len || k >= n)
        return false;
    return judge(a + l[k], b, c, k + 1)
        || judge(a, b + l[k], c, k + 1)
        || judge(a, b, c + l[k], k + 1);
}
```







#1154、数独

【问题描述】

这个游戏只有一个规则：

将格子填满使得每一行， 每一列，和每一个小的九宫格恰好包含 1 — 9 这 9 个数字正是由于规则简单而又变化多端，数独一时间风靡全球。现在，我们希望你能编写一个程序解决数独问题。

【输入数据】

输入数据一共 9 行，每行有 9 个字符。

输入数据描述了一个待解决的数独，其中， “ ? ” 表示数独中的空缺。

我们的输入数据总保证有唯一解。

【输出数据】

输出一共 9 行，每行 9 个数字，表示你的答案。

【样例输入】

```
5????7??6
?6????5?4
?834?????
???182?4?
??1???9??
?7?369???
?????543?
1?5????9?
7??2????1
```

【样例输出】

```
514927386
967831524
283456179
659182743
321574968
478369215
892615437
135748692
746293851
```

#1154、数独

当前位置为r行c列,那么可以求出如下图所在3×3宫格的编号$\left\lfloor \frac{r}{3} \right\rfloor \times 3 + \left\lfloor \frac{c}{3} \right\rfloor$

从第一行第一列开始对每一个?尝试放置$1 \sim 9$的数

使用三个数组对行、列、宫格进行上的数进行标记,回溯时取消标记

若放置数不冲突,继续向右搜索若已到达一行末尾,从下一行第一列开始继续枚举

0	0	0	1	1	1	2	2	2
0	0	0	1	1	1	2	2	2
0	0	0	1	1	1	2	2	2
3	3	3	4	4	4	5	5	5
3	3	3	4	4	4	5	5	5
3	3	3	4	4	4	5	5	5
6	6	6	7	7	7	8	8	8
6	6	6	7	7	7	8	8	8
6	6	6	7	7	7	8	8	8

```
void dfs(int x, int y)
{
    if (x != 8 && y == 9)
        dfs(x + 1, 0);
    else if (x == 8 && y == 9)
        for (int i = 0; i < 9; i++)
        {
            for (int j = 0; j < 9; j++)
                cout << sudo[i][j];
            cout << endl;
        }
    else if (sudo[x][y] != '?')
        dfs(x, y + 1);
    else
        for (int i = 1; i <= 9; i++)
            if (!row[x][i] && !col[y][i] && !block[getBlockIndex(x, y)][i])
            {
                row[x][i] = col[y][i] = block[getBlockIndex(x, y)][i] = true, sudo[x][y] = i + '0';
                dfs(x, y + 1);
                sudo[x][y] = '?', row[x][i] = col[y][i] = block[getBlockIndex(x, y)][i] = false;
            }
}
```

谢谢观看