P1101 单词方阵

题目描述

给一n×n的字母方阵，内可能蕴含多个“yizhong”单词。单词在方阵中是沿着同一方向连续摆放的。摆放可沿着 8 个方向的任一方向，同一单词摆放时不再改变方向，单词与单词之间可以交叉,因此有可能共用字母。输出时，将不是单词的字母用*代替，以突出显示单词。例如：

输入：

```
8                   输出：
qyizhong            *yizhong
gydthkjy            gy******
nwidghji            n*i*****
orbzsfgz            o**z****
hhgrhwth            h***h***
zzzzzozo            z****o**
iwdfrgng            i*****n*
yyyygggg            y******g
```

输入格式

第一行输入一个数n。(7≤n≤100)。

第二行开始输入n×n的字母矩阵。

输出格式

突出显示单词的n×n矩阵。

输入样例

```
7
aaaaaaa
aaaaaaa
aaaaaaa
aaaaaaa
aaaaaaa
aaaaaaa
aaaaaaa
```

输出样例

****
****
****
****
****
****
****

解析

本题与一般的搜索最大的差异就是需要将所需的数据标记出来，如果按照常规的搜索方式进行搜索，在处理最后的标记会比较麻烦。在这里，我们给出一个全新的思路：

任意选择一个坐标作为起点，然后依次沿着八方向中的路径，径直检测，直到出现的字母不是目标字符串中的一个。如图所示：

	0	1	2	3	4	5	6	7
0	*	y	i	z	h	o	n	g
1								
2								
3								
4								
5								
6								
7								

如上图所示，我们从0,1点开始向右侧遍历，直到将七个数字都判断完，之后，我们就可以一并将这7个位置进行标记，用于最后打印。

注意，在这里我们是沿着一个方向一直测试，而不是常规的搜索方法。

编码

```
#include<bits/stdc++.h>

using namespace std;
//方向数量常量
const int Forward = 8;
//目标文字数量常量
const int CharNum = 7;
//标准目标常量
const string Stand = "yizhong";
//地图行列数
int n;
//原始地图数组
char maps[105][105];
//目标点记录数组，即所寻找的目标出现的坐标
bool book[105][105];
//八向的常量数组
```

```
int dir[8][2] = {{-1, -1},
                 {-1, 0},
                 {-1, 1},
                 {0,  -1},
                 {0,  1},
                 {1,  -1},
                 {1,  0},
                 {1,  1}};

void dfs(int x, int y, int step) {
    int a = x, b = y;
    //测试沿着某个方向一直计算，判断其是否能组成目标文字
    for (int j = 0; j < CharNum; j++) {
        if (Stand[j] != maps[a][b]) {
            return;
        }
        a += dir[step][0];
        b += dir[step][1];
    }
    //能够组成，则将目标文字全部标记
    for (int j = 0; j < CharNum; j++) {
        book[x][y] = true;
        x += dir[step][0];
        y += dir[step][1];
    }
}

int main() {
    cin >> n;
    //读入原始地图信息
    for (int i = 1; i <= n; i++) {
        for (int j = 1; j <= n; j++) {
            cin >> maps[i][j];
        }
    }

    //遍历全部的坐标，尝试作为起点，开始进行搜索
    for (int i = 1; i <= n; i++) {
        for (int j = 1; j <= n; j++) {
            //延八方向作为起点
          for (int k = 0; k < Forward; k++) {
                dfs(i, j, k);
            }
        }
    }

    //打印最终的结果，如果记录表中不存在，则表示该点不是目标点
    for (int i = 1; i <= n; i++) {
        for (int j = 1; j <= n; j++) {
```

```
            if (!book[i][j]) {
                cout << '*';
            } else {
                cout << maps[i][j];
            }
        }
        cout << endl;
    }
    return 0;
}
```

逻辑航线培优教育，信息学奥赛培训专家。

扫码添加作者获取更多内容。