C++方向编程题答案

答案说明：

> 大家如果对本次题目或者答案有问题，可以联系下方的出题老师答疑。
>
> **出题老师：**
>
> 选择题：张文超　qq：3627274478
>
> 代码题：时亮益　qq：569334855

第七周

day38

> 题目ID：790 红与黑
>
> 链接：https://www.nowcoder.com/questionTerminal/5017fd2fc5c84f78bbaed4777996213a

[题目解析]

1. 输入的m和n就是代表输入后续会输入几行几列字符
2. 第二行开始，输入的字符就是我们的“行走矩阵”，其中“.”->黑色的瓷砖，“#”->白色的瓷砖，“@”->黑色的瓷砖，并且你站在这块瓷砖上
3. 这道题的核心问题是，从你站的位置开始，向周边任意位置走，你能直接走过的黑色瓷砖的总数是多少

[解题思路]

循环接收每组用例，对于每组用例进行如下操作：

1. 找到‘@’所在的位置，即起始搜索的点
2. 使用DFS搜索地板中的每块瓷砖，如果是黑色，给计数+1，然后像该黑色的上下左右四个方向继续搜索

注意：在搜索时，如果遇到白色瓷砖，或者该位置已经搜索过了，则停止该位置的搜索

```
/*
    DFS问题
*/
#include <iostream>
using namespace std;

#include <vector>

int direct[4][2] = {{-1, 0},{1, 0},{0,-1},{0,1}};
void dfs(vector<vector<char>>& map, int row, int col, vector<vector<bool>>&
flag, size_t& count)
{
 // 如果该位置已经遍历过了，则不用遍历
 if(flag[row][col])
     return;

 // 遍历该位置
 flag[row][col] = true;
 // 如果该位置是白色瓷砖，则停止搜索
```

```
 if('#' == map[row][col])
     return;

 // 否则为黑色瓷砖
 count++;

 // 然后继续向该位置的其他四个方向进行遍历
 for(int i = 0; i < 4; ++i)
 {
     int x = row + direct[i][0];
     int y = col + direct[i][1];

     if((x >= 0 && x < map.size()) && (y >= 0 && y < map[0].size()))
         dfs(map, x, y, flag, count);
 }
}

int main()
{
 int m, n;
 while(cin>>m>>n)
 {
     if(0 == m*n)
         continue;
     // 接收矩阵，并记录起点的位置
     vector<vector<char>> map(m, vector<char>(n));
     vector<vector<bool>> flag(m, vector<bool>(n, false));
     int row = 0, col = 0;
     for(int i = 0; i < m; ++i)
     {
         for(int j = 0; j < n; ++j)
         {
             cin>>map[i][j];
             if(map[i][j] == '@')
             {
                 row = i;
                 col = j;
             }
         }
     }

     // 开始遍历
     size_t count = 0;
     dfs(map, row, col, flag, count);
     cout<<count<<endl;

 }
 return 0;
}


//================================================================================
==============
// 优化：标记数组可以不用给出，每走一步，直接在地图上进行标记
#include <iostream>
using namespace std;

#include <vector>
```

```
int direct[4][2] = {{-1, 0},{1, 0},{0,-1},{0,1}};
void dfs(vector<vector<char>>& map, int row, int col,size_t& count)
{
 if(map[row][col] == '#')
     return;

 map[row][col] = '#';
 // 否则为黑色瓷砖
 count++;

 // 然后继续向该位置的其他四个方向进行遍历
 for(int i = 0; i < 4; ++i)
 {
     int x = row + direct[i][0];
     int y = col + direct[i][1];

     if((x >= 0 && x < map.size()) && (y >= 0 && y < map[0].size()))
         dfs(map, x, y, count);
 }
}

int main()
{
 int m, n;
 while(cin>>m>>n)
 {
     if(0 == m*n)
         continue;
     // 接收矩阵，并记录起点的位置
     vector<vector<char>> map(m, vector<char>(n));
     int row = 0, col = 0;
     for(int i = 0; i < m; ++i)
     {
         for(int j = 0; j < n; ++j)
         {
             cin>>map[i][j];
             if(map[i][j] == '@')
             {
                 row = i;
                 col = j;
             }
         }
     }

     // 开始遍历
     size_t count = 0;
     dfs(map, row, col, count);
     cout<<count<<endl;

 }
 return 0;
}
```

题目ID：25951 蘑菇阵

链接：https://www.nowcoder.com/questionTerminal/ed9bc679ea1248f9a3d86d0a55c0be10

[题目解析]

该题类似于走迷宫，蘑菇代表不能走通，但不同的是，迷宫可以向前后左右四个方向移动，但该题走的方式只能向右或者向下两个方向移动，注意：右边界处只能向一个方向移动，因此走不通路径的概率是不相等的。比如：M = 2 N = 3

1　2　3

4　5　6

在1时：既可向右走到2，也可向下走到4，因此从1-->2 和从1-->4的概率均为0.5

在2时：即可向右走到3，也可向下走到5，因此从2-->3和从2-->5的概率均为0.5

在3时：只能走到6，因此从3-->6概率为1

在4、5、6时，只能向右走，因此4-->5和5-->6的概率均为1。

通过以上分析，可以得到：

假设**P(i, j)表示从起点到(i, j)不踩到蘑菇的概率**，那么该位置一定是从(i-1, j)或者(i, j-1)出走过来的。

而**从(i-1, j)或者(i, j-1)到达(i, j)的概率是不等**的，比如：**如果i或者j在边界，只能向一个方向移动，此时走到(i, j)位置的概率为1，当i或者j不在边界时，走到(i,j)的概率分别为0.5**，因此可得出：

P(i,j) = P(i-1, j) * (i==M? 1 : 0.5)+ P(i, j-1) * (j==N? 1 : 0.5);

如果(i, j)为蘑菇时，表示不能走到该位置

[解题思路]

1. **循环接受输入**(注意：一般IO类型算法即需要写main的算法，背后可能有多个测试用例，每个用例必须测试到，因此需要循环输入)
2. **按照输入构造蘑菇地图(二维矩阵)，1代表蘑菇，0代表通路**，因起点是从(1,1)开始，矩阵多给一个行和列
3. 构造用来保存走到(i,j)位置不遇到蘑菇的概率容器
4. 按照上述分析结论：遍历蘑菇地图，当遇到蘑菇时，将概率置为0，即不可能到达该位置
5. 按照要求输出：**注意保留两位精度。**

```
#include <iostream>
using namespace std;
#include <vector>

int main()
{
 int m,n,k;
 while(cin>>n>>m>>k)
 {
     // 因为地图大小已经确定好了，map直接设置好大小
     vector<vector<int>> map(n+1,vector<int>(m+1));

     // 向地图中放入蘑菇
     int row, col;
     while(k--)
     {
```

```
            cin>>row>>col;
            map[row][col] = 1;
        }

        vector<vector<double>> dp(n+1, vector<double>(m+1));
        dp[1][1] = 1.0;
        for(int i = 1; i <= n; i++)
        {
            for(int j = 1; j <= m; ++j)
            {
                // 对于每个位置，按照上述转移方程来确定概率
                if(!(i==1 && j==1))
                    dp[i][j] = dp[i-1][j]*(j==m?1:0.5) + dp[i][j-1]*(i==n?
1:0.5);

                // 如果该位置为蘑菇，表示不能到达该位置，则到达该位置的概率一定为0
                if(map[i][j])
                    dp[i][j] = 0;
            }
        }

        printf("%.2f\n", dp[n][m]);
    }
    return 0;
}
```