给定一个包含了一些 0 和 1的非空二维数组 `grid` , 一个 **岛屿** 是由四个方向 (水平或垂直) 的 `1` (代表土地) 构成的组合。你可以假设二维矩阵的四个边缘都被水包围着。

找到给定的二维数组中最大的岛屿面积。(如果没有岛屿，则返回面积为0。)

示例 1:

```
[[0,0,1,0,0,0,0,1,0,0,0,0,0],
 [0,0,0,0,0,0,0,1,1,1,0,0,0],
 [0,1,1,0,1,0,0,0,0,0,0,0,0],
 [0,1,0,0,1,1,0,0,1,0,1,0,0],
 [0,1,0,0,1,1,0,0,1,1,1,0,0],
 [0,0,0,0,0,0,0,0,0,0,1,0,0],
 [0,0,0,0,0,0,0,1,1,1,0,0,0],
 [0,0,0,0,0,0,0,1,1,0,0,0,0]]
```

思路:

dfs遍历，用深度遍历即可，和迷宫遍历相似，遍历过的节点标记为0防止重复遍历。

```
class Solution {
public:
    int maxAreaOfIsland(vector<vector<int>>& grid) {
        int n=grid.size();
        int m=grid[0].size();
        int max=0;
        for(int i=0;i<n;i++){
            for(int j=0;j<m;j++){
                if(grid[i][j]==1){
                    grid[i][j]=0;
                    int num=dfs(grid,i,j);
                    max=max<num?num:max;
                }
            }
        }
        return max;
    }
    int dfs(vector<vector<int>>& grid,int i,int j){
        int num=1;
        if(j>0){
            if(grid[i][j-1]==1){
                grid[i][j-1]=0;
                num+=dfs(grid,i,j-1);
            }
        }
        if(j+1<grid[0].size()){
            if(grid[i][j+1]==1){
                grid[i][j+1]=0;
                num+=dfs(grid,i,j+1);
```

```
            }
        }
        if(i+1<grid.size()){
            if(grid[i+1][j]==1){
                grid[i+1][j]=0;
                num+=dfs(grid,i+1,j);
            }
        }
        if(i>0){
            if(grid[i-1][j]==1){
                grid[i-1][j]=0;
                num+=dfs(grid,i-1,j);
            }
        }
        return num;
    }
};
```