200. Number of Islands

Given an `m x n` 2D binary grid `grid` which represents a map of `'1'`s (land) and `'0'`s (water), return *the number of islands*.

An **island** is surrounded by water and is formed by connecting adjacent lands horizontally or vertically. You may assume all four edges of the grid are all surrounded by water.

Example 1:

```
Input: grid = [
  ["1","1","1","1","0"],
  ["1","1","0","1","0"],
  ["1","1","0","0","0"],
  ["0","0","0","0","0"]
]
Output: 1
```

Example 2:

```
Input: grid = [
  ["1","1","0","0","0"],
  ["1","1","0","0","0"],
  ["0","0","1","0","0"],
  ["0","0","0","1","1"]
]
Output: 3
```

Constraints:

- `m == grid.length`
- `n == grid[i].length`
- `1 <= m, n <= 300`
- `grid[i][j]` is `'0'` or `'1'`.

200. Number of Islands

```
/*
   Time complexity: O( (nm))
   Space complexity: O(2nm)
*/
class Solution {
   public:
      vector<vector<char>> G;
      map<pair<int,int>,bool> visited;
      int n,m;
      int cnt=0;
      pair<int,int> moves[4] = {
         {-1,0},    {1,0},    {0,-1},  {0,1}
      };
   public:
      bool is_in_grid(int x,int y){
         return x>=0 && x<=n-1 && y>=0 && y<=m-1;
      }
      void DFS(int x,int y){
         if(visited[{x,y}]) return;
         visited[{x,y}]=true;
         for(int i=0;i<4;++i){
            int move_x=x+moves[i].first;
            int move_y=y+moves[i].second;
            if(!is_in_grid(move_x,move_y)) continue;
            if(G[move_x][move_y]=='0') continue;
            DFS(move_x,move_y);
         }
      }
      int numIslands(vector<vector<char>>& grid) {
         n=grid.size();
         m=grid[0].size();
         G=grid;
         for(int i=0;i<n;++i){
            for(int j=0;j<m;++j){
               if(G[i][j]=='0' || visited[{i,j}]) continue;
               cnt++;
               DFS(i,j);
            }
         }
         return cnt;
      }
};
```