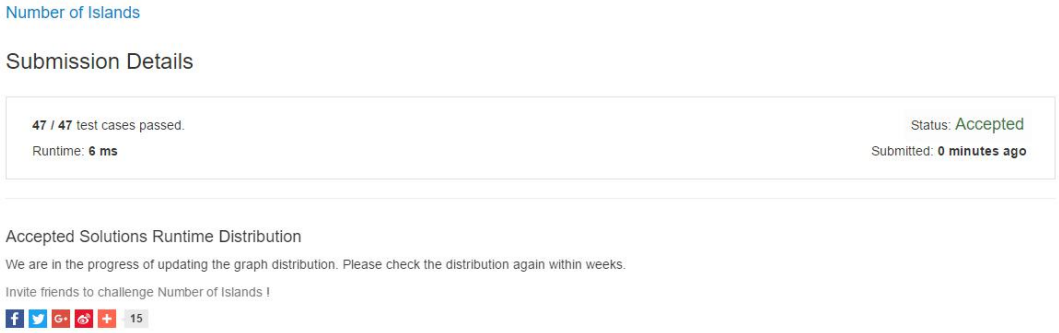
PA1 唐天成 515030910287

Leetcode 200:Number of Island



Code(C++):

```
class Solution {
public:
    int numIslands(vector<vector<char>>& grid) {
        int width = grid.size();
        int number = 0; // number = sequence - minus
        int sequence = 0;
        int minus = 0;

        if (width > 0) {
            int lenth = grid[0].size();
            first_line(grid, sequence, minus, lenth);
            for (int a = 1; a < width; a++) {
                process_line(grid, sequence, minus, lenth,width, a);
            }
            number = sequence - minus;
        }
        return number;
    }
    int first_line(vector<vector<char>>& grid ,int& sequence, int& minus,int lenth) {
        int left = 0;
        for (int i = 0; i < lenth; i++) {
            int thisone = grid[0][i];
            if (thisone != '0') {
                if (left > 0) {
                    grid[0][i] = left;
                }
                else if (left == 0) {
                    sequence = sequence + 1;
                    grid[0][i] = sequence;
                    left = sequence;
                }
            }
```

```
            else if (thisone == '0') {
                left = 0;
                grid[0][i] = 0;
            }
        }
        return 0;
    }
    int process_line(vector<vector<char>>& grid, int& sequence, int& minus, int lenth, int width,int x) {
        int left = 0;
        int up = 0;
        map<int, int> havingcounted;
        int lastup;
        int lastleft;
        if (grid[x][0] == '0') {
            lastleft = 0;
            lastup = 0;
        }
        else {
            lastleft = sequence+1;
            lastup = get_up(grid, x, 0);
            havingcounted[lastup] = lastleft;
        }
    for (int i = 0; i < lenth; i++) {
            int thisone = grid[x][i];
            if (thisone != '0') {

                up = get_up(grid, x, i);
                if (left > 0) {
                    grid[x][i] = left;
                    if (up > 0) {


                        int leftup = get_leftup(grid, x, i);
                        if (leftup == 0 && havingcounted.count(up)==0    ) {
                            minus = minus + 1;
                            havingcounted[up] = left;

                        }
                        else if (leftup == 0 && havingcounted.count(up) != 0 && havingcounted[up] != left) {
                            minus = minus + 1;
                            int tochange = havingcounted[up];
                            havingcounted[up] = left;
```

```
                                for (int a = 0; a < i; a++) {
                                        if (grid[x][a] == tochange) {
                                                grid[x][a] = left;
                                        }
                                }
                        }

                }
        }
                else if (left == 0) {
                sequence = sequence + 1;
                grid[x][i] = sequence;
                left = sequence;
                if (up > 0) {
                        minus = minus + 1;
                }
                if (havingcounted.count(up)!=0 && i != 0 && up !=0) {
                        grid[x][i] = havingcounted[up];
                        left = havingcounted[up];
                }
                else if (i != 0 && up != 0){
                        havingcounted[up] = sequence;
                }
        }
    }
    else if (thisone == '0') {
        if (left != 0) {
                lastleft = left;
                lastup = get_leftup(grid, x, i);
        }

        left = 0;
        grid[x][i] = 0;

    }
  }
  return 0;
}
int get_leftup(vector<vector<char>>& grid, int x, int y) {
  if (x >= 1 && y >= 1) {
        return grid[x - 1][y - 1];
  }
  else {
```

```
                return 0;
            }
        }
        int get_up(vector<vector<char>>& grid, int x, int y ){
            if (x >= 1) {
                return grid[x-1][y];
            }
            else {
                return 0;
            }
        }
};
```