July 30, 2018 题目：388,200,465,266,774,695,836,714,398,833,183,444

388

1. Use a stack to push the length of the file, the number of /t comparing to the size of the stack records the level and determines push/pop operations to the stack.
2. 各种烦，注意最后问的是file path，directory path 不算：

```
class Solution {
public:
    int lengthLongestPath(string input) {
        stack<int> S;
        S.push(-1);
        auto i = input.find_first_not_of('\n');
        int ans = 0, cnt = int(input.find_first_not_of("\t")), cur = 0;
        while(i<input.size() && i!=string::npos){
            auto k = input.find_first_not_of("\t", i);
            if(k==string::npos) k = input.size();
            int tmp = int(k-i);
            if(tmp>cnt){
                assert(tmp == cnt+1);
                S.push(S.top() + cur + 1);
                ++cnt;
            }
            else if(tmp<cnt){
                while(cnt>tmp) {
                    S.pop();
                    --cnt;
```

```
                }
            }
            auto j = input.find('\n', k);
            if(j==string::npos) j=input.size();
            cur = int(j-k);
            if(input.substr(k, j-k).find('.')!=string::npos) a
ns = max(ans, S.top() + cur + 1);
            i = j+1;
        }
        return ans;
    }
};
```

3. 曾经做过的Google OA题，跳过了- -

200

1. BFS, DFS or Union Find
2. Union find, 做过了，这题：

```
class Solution {
    vector<int> P;
    int getRoot(int i){
        if(P[i]==i) return i;
        return P[i]=getRoot(P[i]);
    }
public:
    int numIslands(vector<vector<char>>& grid) {
        if(grid.empty() || grid[0].empty()) return 0;
        int n = grid.size(), m = grid[0].size(), ans = 0;
        P.resize(m*n);
        for(int k=0;k<m*n;++k) if(grid[k/m][k%m] == '1'){
            P[k] = k;
```

```
            if(k/m && grid[k/m-1][k%m]=='1') if(getRoot(k)!=getRoot(k-m)) P[getRoot(k)] = getRoot(k-m);
            if(k%m && grid[k/m][k%m-1]=='1') if(getRoot(k)!=getRoot(k-1)) P[getRoot(k)] = getRoot(k-1);
        }
        for(int i=0;i<n*m;++i) if(P[i]==i && grid[i/m][i%m]=='1') ++ans;
        return ans;
    }
};
```

3. 经典DFS

```
class Solution {
    public int numIslands(char[][] grid) {
        int k = 0;
        for (int i = 0; i < grid.length; i++) {
            for (int j = 0; j < grid[0].length; j++) {
                if (grid[i][j] == '1') {
                    k++;
                    helper(grid, i, j);
                }
            }
        }
        return k;
    }

    public void helper(char[][] grid, int i, int j) {
        if (i >= grid.length || j >= grid[0].length || i < 0 || j < 0 || grid[i][j] == '0') return;
        grid[i][j] = '0';
        helper(grid, i + 1, j);
        helper(grid, i, j + 1);
```

```
        helper(grid, i - 1, j);
        helper(grid, i, j - 1);
    }
}
```

465

1. DFS， 注意别用贪心，因为这是NP hard:

```
class Solution {
    typedef map<int, int, greater<int>> MP;
    int dfs(MP pos, MP neg){
        if(pos.empty()) return 0;
        int q = pos.begin()->first;
        pos[q]--;
        if(!pos[q]) pos.erase(q);
        int ans = 1E9;
        for(auto p: neg){
            MP pp(pos), nn(neg);
            nn[p.first]--;
            if(!nn[p.first]) nn.erase(p.first);
            if(p.first == q) return 1 + dfs(pp, nn);
            else if(p.first > q){
                nn[p.first - q]++;
                ans = min(ans, 1 + dfs(pp, nn));
            }
            else{
                pp[q - p.first]++;
                ans = min(ans, 1 + dfs(pp, nn));
            }
        }
        return ans;
```

```
    }
public:
    int minTransfers(vector<vector<int>>& transactions) {
        unordered_map<int, int> D;
        for(auto v: transactions){
            D[v[0]] += v[2];
            D[v[1]] -= v[2];
        }
        MP pos, neg;
        for(auto p: D){
            if(p.second > 0) pos[p.second]++;
            else if(p.second < 0) neg[-p.second]++;
        }
        return dfs(pos, neg);
    }
};
```

2. backtracking

```
class Solution {
    public int minTransfers(int[][] transactions) {
        Map<Integer, Integer> map = new HashMap<>();

        for (int[] t : transactions) {
            if (!map.containsKey(t[0])) {
                map.put(t[0], 0);
            }
            if (!map.containsKey(t[1])) {
                map.put(t[1], 0);
            }
            map.put(t[0], map.get(t[0]) - t[2]);
            map.put(t[1], map.get(t[1]) + t[2]);
```

```
        }
        List<Integer> candidates = new ArrayList<>();
        for (int i : map.values()) {
            if (i != 0) candidates.add(i);
        }
        Collections.sort(candidates);
        return helper(candidates, 0);
    }
    public int helper(List<Integer> list, int start) {
        while (start < list.size() && list.get(start) == 0) {
            start++;
        }

        if (start == list.size()) return 0;
        int min = Integer.MAX_VALUE;
        for (int i = start + 1; i < list.size(); i++) {
            if (list.get(i) * list.get(start) < 0) {
                list.set(i, list.get(i) + list.get(start));
                min = Math.min(min, 1 + helper(list, start +
1));
                list.set(i, list.get(i) - list.get(start));
            }
        }
        return min;
    }
}
```

266

1. easy
2. 计数就行了：

```
class Solution {
```

```
public:
    bool canPermutePalindrome(string s) {
        map<char, int> cnt;
        for(char c:s) cnt[c]++;
        int odd=0;
        for(auto p:cnt) if(p.second%2) ++odd;
        return odd<2;
    }
};
```

3.

```
class Solution {
    public boolean canPermutePalindrome(String s) {
        int[] chars = new int[256];
        for (char c : s.toCharArray()) {
            chars[(int)c]++;
        }
        boolean flag = false;
        for (int i : chars) {
            if (i % 2 == 1) {
                if (flag) return false;
                flag = true;
            }
        }
        return true;
    }
}
```

774

1. 二分：这种题居然是hard

```
class Solution {
    #define DELTA double(1E-8)
```

```
    #define CNT(x) int((x)-DELTA)
public:
    double minmaxGasDist(vector<int>& stations, int K) {
        sort(stations.begin(), stations.end());
        vector<double> dis;
        for(int i=1; i<stations.size(); ++i) if(stations[i]>st
ations[i-1])
            dis.push_back(double(stations[i]-stations[i-1]));
        double l=0, r=1E8+1.;
        while(l<r-DELTA){
            double c = (l + r)/2.;
            int cnt = 0;
            for(double k: dis) cnt += CNT(k/c);
            if(cnt>K) l = c;
            else r = c;
        }
        return r;
    }
};
```

695

1. One pass Union find:

```
class Solution {
    int n, m, res;
    vector<int> P, area;
    int findRoot(int i){
        if(P[i]==i) return i;
        return P[i] = findRoot(P[i]);
    }
    void connect(int i, int j){
        int ri = findRoot(i);
```

```
        int rj = findRoot(j);
        if(ri != rj){
            P[rj] = ri;
            area[ri] += area[rj];
        }
        res = max(res, area[ri]);
    }
public:
    int maxAreaOfIsland(vector<vector<int>>& grid) {
        if(grid.empty() || grid[0].empty()) return 0;
        n = grid.size();
        m = grid[0].size();
        res = 0;
        P.resize(n*m);
        area.resize(n*m);
        for(int i=0; i<n*m; ++i) P[i]=i;
        for(int i=0; i<n; ++i) for(int j=0; j<m; ++j) if(grid
[i][j]){
            area[i*m + j] = 1;
            res = max(1, res);
            if(j && grid[i][j-1]) connect(i*m +j -1, i*m + j);
            if(i && grid[i-1][j]) connect((i-1)*m + j, i*m +
j);
        }
        return res;
    }
};
```

2. DFS

```
class Solution {
    public int maxAreaOfIsland(int[][] grid) {
```

```
        int max = 0;
        for (int i = 0; i < grid.length; i++) {
            for (int j = 0; j < grid[0].length; j++) {
                if (grid[i][j] == 1) {
                    max = Math.max(max, helper(grid, i, j));
                }
            }
        }
        return max;
    }
    public int helper(int[][] grid, int i, int j) {
        if (i >= grid.length || j >= grid[0].length || i < 0 || j < 0 || grid[i][j] == 0) return 0;
        grid[i][j] = 0;
        return 1 + helper(grid, i + 1, j) + helper(grid, i, j + 1) + helper(grid, i - 1, j) + helper(grid, i, j - 1);
    }
}
```

836

1. x，y 坐标分别比较范围:

```
class Solution {
public:
    bool isRectangleOverlap(vector<int>& rec1, vector<int>& rec2) {
        return !(rec1[0]>=rec2[2] || rec1[2]<=rec2[0] || rec1[1]>=rec2[3] || rec1[3]<=rec2[1]);
    }
};
```

714

1. dp: 标记`position = 0`, 跟 `position = 1` 的状态.

```
class Solution {
public:
    int maxProfit(vector<int>& prices, int fee) {
        if(prices.empty()) return 0;
        long n = prices.size(), zero = 0L, one = INT_MIN;
        for(auto p: prices){
            int zero_ = max(zero, one + p - fee);
            int one_ = max(one, zero - p);
            one = one_;
            zero = zero_;
        }
        return zero;
    }
};
```

398

1. Why only beat `22%`:

```
class Solution {
    unordered_map<int, vector<int>> pos;
public:
    Solution(vector<int> nums) {
        for(int i=0; i<nums.size(); ++i) pos[nums[i]].push_bac
k(i);
    }
    int pick(int target) {
        return pos[target][rand()%pos[target].size()];
    }
};
```

833

1. 注意是同时换，注意该换和不该换两种情况别弄错就行了：

```
class Solution {
public:
    string findReplaceString(string S, vector<int>& indexes, v
ector<string>& sources, vector<string>& targets) {
        map<int, string> rep;
        map<int, int> length;
        for(int i=0; i<indexes.size(); ++i) {
            if(S.substr(indexes[i], sources[i].size()) == sour
ces[i]){
                rep[indexes[i]] = targets[i];
                length[indexes[i]] = int(sources[i].size());
            }
            else {
                rep[indexes[i]] = "";
                length[indexes[i]] = 0;
            }
        }
        string ans;
        int i = 0;
        for(auto p: rep){
            ans += S.substr(i, p.first-i) + p.second;
            i = p.first + length[p.first];
        }
        cout<<i<<endl;
        ans += S.substr(i);
        return ans;
    }
};
```

183

SQL不会

444

1. 就一个拓扑排序花了我这么长时间，各种SB corner case

```
class Solution {
public:
    bool sequenceReconstruction(vector<int>& org, vector<vecto
r<int>>& seqs) {
        int n = org.size(), judge = 0;
        vector<unordered_set<int>> E(n), P(n);
        for(auto vec: seqs) {
            if(vec.empty()) continue;
            judge = 1;
            if(vec[0] > n || vec[0]<0) return false;
            for(int i=1; i<vec.size(); ++i) {
                if(vec[i] > n || vec[i] < 0) return false;
                E[vec[i]-1].insert(vec[i-1]-1);
                P[vec[i-1]-1].insert(vec[i]-1);
            }
        }
        if(!judge) return false;
        vector<int> root;
        for(int i=0; i<n; ++i) if(E[i].empty()) root.push_back
(i);
        for(int i=0; i<n; ++i){
            if(root.size()!=1 || root[0]!=org[i]-1) return fal
se;
            vector<int> new_root;
            for(int j: P[org[i]-1]) if(E[j].count(org[i]-1)) {
                E[j].erase(org[i]-1);
                if(E[j].empty()) new_root.push_back(j);
            }
            swap(root, new_root);
```

```
        }
        return true;
    }
};
```