August 21, 2018 题目：305,732,483,284,659,207,62,528,63,242,9,81,383,13,168

305

1. UnionFind

```
class Solution {
    public List<Integer> numIslands2(int m, int n, int[][] positions) {
        List<Integer> res = new ArrayList<>();
        int[] parents = new int[m * n];
        Arrays.fill(parents, -1);
        int[] dx = {1, -1, 0, 0};
        int[] dy = {0, 0, 1, -1};
        int count = 0;
        for (int[] p : positions) {
            int k = trans(p[0], p[1], n);
            parents[k] = k;
            count++;
            for (int i = 0; i < 4; i++) {
                int ux = p[0] + dx[i];
                int uy = p[1] + dy[i];
                if (ux < 0 || uy < 0 || ux >= m || uy >= n || parents[trans(ux, uy, n)] == -1) continue;
                int a = find(ux, uy, n, parents), b = find(p[0], p[1], n, parents);
                if (a != b) {
                    parents[a] = b;
                    count--;
```

```
                }
            }
            res.add(count);
        }
        return res;
    }

    public int trans(int i, int j, int m) {
        return i * m + j;
    }

    public int find(int i, int j, int m, int[] parents) {
        int k = trans(i, j, m);
        while (k != parents[k]) {
            k = parents[k];
        }
        return k;
    }
}
```

2. 典型Union Find：

```
class Solution {
    vector<int> P;
    int dx[4] = {1, -1, 0, 0};
    int findRoot(int i){
        assert(i>=0);
        if(P[i] == i) return i;
        return P[i] = findRoot(P[i]);
    }
public:
```

```
    vector<int> numIslands2(int n, int m, vector<pair<int, int
>>& positions) {
        P = vector<int>(m*n, -1);
        vector<int> ans;
        int cnt = 0;
        for(auto p: positions){
            ++cnt;
            P[p.first*m + p.second] = p.first*m + p.second;
            for(int k=0; k<4; ++k){
                int x = p.first + dx[k], y = p.second + dx[3-
k];
                if(x>=0 && x<n && y>=0 && y<m && P[x*m + y] >=
0 && findRoot(x*m + y)!=findRoot(p.first*m+p.second)){
                    --cnt;
                    P[findRoot(p.first*m + p.second)] = findRo
ot(x*m + y);
                }
            }
            ans.push_back(cnt);
        }
        return ans;
    }
};
```

732

1. TreeMap to maintain the current number of meetings.

```
class MyCalendarThree {
    TreeMap<Integer, Integer> map;
    public MyCalendarThree() {
        map = new TreeMap<>();
    }
```

```
    public int book(int start, int end) {
        map.put(start, map.getOrDefault(start, 0) + 1);
        map.put(end, map.getOrDefault(end, 0) - 1);
        int res = 0, sum = 0;
        for (int v : map.values()) {
            res = Math.max(res, sum + v);
            sum += v;
        }
        return res;
    }
}
```

2. 同上：

```
class MyCalendarThree {
    map<int, int> range;
public:
    MyCalendarThree() {}

    int book(int start, int end) {
        ++range[start];
        --range[end];
        int tmp = 0, ans = 0;
        for(auto p: range){
            tmp += p.second;
            ans = max(ans, tmp);
        }
        return ans;
    }
};
```

483 昨天的题

284

1. 做过：

```
class PeekingIterator : public Iterator {
    bool peeked;
    int val;
public:
        PeekingIterator(const vector<int>& nums) : Iterator(nu
ms), peeked(false) {}
    int peek(){
        if(!peeked) {
            peeked=true;
            val = this->Iterator::next();
        }
        return val;
    }
    int next() {
        if(peeked){
            peeked = false;
            return val;
        }
        return this->Iterator::next();
        }
        bool hasNext() const {
            return peeked||this->Iterator::hasNext();
        }
};
```

659

1. Maintain lengths:

```
class Solution {
public:
    bool isPossible(vector<int>& nums) {
        int n = nums.size(), i=0, cur=nums[0]-1;
        vector<int> dp;
        while(i<n){
            if(nums[i] > cur+1) {
                for(int k: dp) if(k<3) return false;
                dp.clear();
            }
            cur = nums[i];
            vector<int> tmp;
            int j=1;
            while(i+j<n && nums[i+j] == cur) ++j;
            for(int k=0; k<(int)dp.size()-j; ++k) if(dp[k] <
3) return false;
            for(int k=max(0, int(dp.size())-j); k<(int)dp.size
(); ++k) tmp.push_back(dp[k]+1);
            for(int k=0; k<j-(int)dp.size(); ++k) tmp.push_bac
k(1);
            swap(dp, tmp);
            i += j;
        }
        for(int k: dp) if(k<3) return false;
        return true;
    }
};
```

207

1. 拓扑排序：

```
class Solution {
```

```
public:
    bool canFinish(int n, vector<pair<int, int>>& prerequisites) {
        vector<vector<int>> E(n);
        vector<int> I(n, 0);
        unordered_set<int> rest;
        queue<int> Q;
        for(auto p: prerequisites) {
            E[p.second].push_back(p.first);
            ++I[p.first];
        }
        for(int i=0; i<n; ++i){
            if(!I[i]) Q.push(i);
            else rest.insert(i);
        }
        while(!Q.empty()){
            for(int k: E[Q.front()]){
                --I[k];
                if(!I[k]) {
                    Q.push(k);
                    rest.erase(k);
                    if(rest.empty()) return true;
                }
            }
            Q.pop();
        }
        return rest.empty();
    }
};
```

- [ ] Look into the following one @Zebo L

DFS

For DFS, it will first visit a node, then one neighbor of it, then one neighbor of this neighbor... and so on. If it meets a node which was visited in the current process of DFS visit, a cycle is detected and we will return `false`. Otherwise it will start from another unvisited node and repeat this process till all the nodes have been visited. Note that you should make two records: one is to record all the visited nodes and the other is to record the visited nodes in the current DFS visit.

The code is as follows. We use a `vector<bool> visited` to record all the visited nodes and another `vector<bool> onpath` to record the visited nodes of the current DFS visit. Once the current visit is finished, we reset the `onpath` value of the starting node to `false`.

```
class Solution {
public:
    bool canFinish(int numCourses, vector<pair<int, int>>& prerequisites) {
        vector<unordered_set<int>> graph = make_graph(numCourses, prerequisites);
        vector<bool> onpath(numCourses, false), visited(numCourses, false);
        for (int i = 0; i < numCourses; i++)
            if (!visited[i] && dfs_cycle(graph, i, onpath, visited))
                return false;
        return true;
    }
private:
    vector<unordered_set<int>> make_graph(int numCourses, vector<pair<int, int>>& prerequisites) {
        vector<unordered_set<int>> graph(numCourses);
        for (auto pre : prerequisites)
            graph[pre.second].insert(pre.first);
        return graph;
    }
    bool dfs_cycle(vector<unordered_set<int>>& graph, int node, vector<bool>& onpath, vector<bool>& visited) {
        if (visited[node]) return false;
        onpath[node] = visited[node] = true;
        for (int neigh : graph[node])
            if (onpath[neigh] || dfs_cycle(graph, neigh, onpath, visited))
                return true;
        return onpath[node] = false;
    }
};
```

62

1.

```
class Solution {
    public int uniquePaths(int m, int n) {
        int[][] paths = new int[m][n];
        for (int i = 0; i < m; i++) {
            paths[i][0] = 1;
        }
        for (int j = 0; j < n; j++) {
            paths[0][j] = 1;
        }
        for (int i = 1; i < m; i++) {
            for (int j = 1; j < n; j++) {
                paths[i][j] = paths[i - 1][j] + paths[i][j - 1];
            }
```

```
        }
        return paths[m - 1][n - 1];
    }
}
```

2. 组合数学：C_n^{m+n} 注意别Overflow就行了

```
class Solution {
public:
    int uniquePaths(int m, int n) {
        --n;
        --m;
        long prod = 1;
        for(long i=0, j=min(m, n); i<min(n, m); ++i){
            prod *= (m + n -i);
            while(j>1 && prod % j==0) prod/=j--;
        }
        return prod;
    }
};
```

528

1. binary search

```
class Solution {
    int[] w;
    public Solution(int[] w) {
        for (int i = 1; i < w.length; i++) {
            w[i] += w[i - 1];
        }
        this.w = w;
    }

    public int pickIndex() {
```

```
        Random r = new Random();
        int k = r.nextInt(w[w.length - 1]) + 1;
        int i = 0, j = w.length - 1;
        while (i < j) {
            int mid = i + (j - i) / 2;
            if (k == w[mid]) return mid;
            else if (k > w[mid]) i = mid + 1;
            else j = mid;
        }
        return i;
    }
}
```

2. 求部分和的时候别把index搞错：

```
class Solution {
    vector<int> P;
public:
    Solution(vector<int> w) : P(vector<int>(w.size(), w[0])){
        for(int i=0; i<w.size()-1; ++i) P[i+1] = P[i] + w[i+
1];
    }
    int pickIndex() {
        int k = rand() % P.back();
        int l = -1, r = P.size();
        while(l < r-1){
            int c = (l+r)/2;
            if(P[c] <= k) l = c;
            else r = c;
        }
        return r;
    }
```

```
};
```

63

1. Need dp:

```
class Solution {
public:
    int uniquePathsWithObstacles(vector<vector<int>>& O) {
        if(O.empty() || O[0].empty()) return 0;
        int n = O.size(), m = O[0].size();
        if(O[0][0] || O[n-1][m-1]) return 0;
        vector<int> dp(m, 0);
        dp[0] = 1;
        for(int i=0; i<n; ++i){
            dp[0] *= (1-O[i][0]);
            for(int j=1; j<m; ++j) {
                if(O[i][j]) dp[j] = 0;
                else dp[j] += dp[j-1];
            }
        }
        return dp.back();
    }
};
```

242

1. sort → NlogN 也可以O(N)用char map记appearance

```
class Solution {
    public boolean isAnagram(String s, String t) {
        char[] s1 = s.toCharArray();
        char[] s2 = t.toCharArray();
        Arrays.sort(s1);
        Arrays.sort(s2);
```

```
        return new String(s1).equals(new String(s2));
    }
}
```

2. count:

```
class Solution {
    #define CI(c) int((c) - 'a')
public:
    bool isAnagram(string s, string t) {
        vector<int> cnt(26, 0);
        if(s.size() != t.size()) return false;
        for(char c: s) ++cnt[CI(c)];
        for(char c: t){
            --cnt[CI(c)];
            if(cnt[CI(c)] < 0) return false;
        }
        return true;
    }
};
```

9

1. 如果用数学办法倒过来算x容易overflow， 所以用了string。。

```
class Solution {
    public boolean isPalindrome(int x) {
        if (x < 0) return false;
        String s = String.valueOf(x);
        return new StringBuilder(s).reverse().toString().equal
s(s);
    }
}
```

2. 不会overflow的数学方法：

```
class Solution {
```

```
public:
    bool isPalindrome(int x) {
        if(x < 0) return false;
        int M = 1;
        while(M <= x/10) M *= 10;
        while(M > 1){
            if(x/M != x%10) return false;
            x = x%M/10;
            M /= 100;
        }
        return true;
    }
};
```

81

1. Refer to Leetcode 33

```
class Solution {
    public boolean search(int[] nums, int target) {
        if (nums == null || nums.length == 0) return false;
        int i = 0, j = nums.length - 1;
        while (i <= j) {
            int mid = i + (j - i) / 2;
            if (nums[mid] == target) return true;
            if (nums[i] < nums[mid] || nums[mid] > nums[j]) {
                if (nums[mid] > target && target >= nums[i]) {
                    j = mid - 1;
                } else {
                    i = mid + 1;
                }
            } else if (nums[i] > nums[mid] || nums[mid] < nums
[j]){
```

```
                if (nums[mid] < target && target <= nums[j]) {
                    i = mid + 1;
                } else {
                    j = mid - 1;
                }
            } else {
                i++;
            }
        }
        return false;
    }
}
```

2. 直接找：

```
class Solution(object):
    def search(self, nums, target):
        return target in nums
```

- [ ] Try to understand 1 @Zebo L

383

1.

```
class Solution {
    public boolean canConstruct(String ransomNote, String maga
zine) {
        int[] ch = new int[26];
        for (char c : magazine.toCharArray()) {
            ch[c - 'a']++;
        }

        for (char c : ransomNote.toCharArray()) {
            if (--ch[c - 'a'] < 0) return false;
        }
```

```
        return true;
    }
}
```

2. 数组计数：

```
class Solution {
    #define CI(c) int((c) - 'a')
public:
    bool canConstruct(string ransomNote, string magazine) {
        vector<int> cnt(26, 0);
        for(char c: magazine) ++cnt[CI(c)];
        for(char c: ransomNote){
            --cnt[CI(c)];
            if(cnt[CI(c)] < 0) return false;
        }
        return true;
    }
};
```

13

1. One pass + map

```
class Solution {
    public int romanToInt(String s) {
        Map<Character, Integer> map = new HashMap<>();
        map.put('I', 1);
        map.put('V', 5);
        map.put('X', 10);
        map.put('L', 50);
        map.put('C', 100);
        map.put('D', 500);
        map.put('M', 1000);
        int res = 0;
```

```
        for (int i = 0; i < s.length() - 1; i++) {
            char pre = s.charAt(i);
            char post = s.charAt(i + 1);
            if (map.get(pre) < map.get(post)) {
                res -= map.get(pre);
            } else {
                res += map.get(pre);
            }
        }
        res += map.get(s.charAt(s.length() - 1));
        return res;
    }
}
```

2. 把楼上的map 做全一些，loop 起来更省事：

```
class Solution {
    vector<string> R = {"M", "CM", "D", "CD", "C", "XC", "L",
"XL", "X", "IX", "V", "IV", "I"};
    vector<int> N = {1000, 900, 500, 400, 100, 90, 50, 40, 10,
9, 5, 4, 1};
public:
    int romanToInt(string s) {
        int i = 0, ans = 0, j=0;
        while(i<13 && j<s.size()){
            while(i<13 && R[i]!=s.substr(j, R[i].size())) ++i;
            assert(i<13);
            ans += N[i];
            j += R[i].size();
        }
        return ans;
    }
```

```
};
```

168

1.

```
class Solution {
    public String convertToTitle(int k) {
        StringBuilder sb = new StringBuilder();
        while (k > 0) {
            k = k - 1;
            sb.append((char) (k % 26 + 'A'));
            k = k / 26;
        }
        return sb.reverse().toString();
    }
}
```

2. 同上：

```
class Solution {
public:
    string convertToTitle(int n) {
        string ans;
        while(n){
            ans += char((n-1)%26 + 'A');
            n = (n-1)/26;
        }
        reverse(ans.begin(), ans.end());
        return ans;
    }
};
```