July 19, 2018 题目 812,310,166,665,288,75,629,447,731,712,519,517

812

1. 叉乘 （我就适合做简单题）

```
class Solution {
    int crossProduct(int x1, int y1, int x2, int y2){
        return abs(x1 * y2 - x2 * y1);
    }
public:
    double largestTriangleArea(vector<vector<int>>& points) {
        int n = points.size(), ans = 0;
        if(n<3) return 0;
        for(int i=0;i<n;++i) for(int j=i+1;j<n;++j) for(int k=j+1;k<n;++k){
            int x1 = points[j][0]-points[i][0], y1 = points[j][1]-points[i][1];
            int x2 = points[k][0]-points[i][0], y2 = points[k][1]-points[i][1];
            ans = max(ans, crossProduct(x1, y1, x2, y2));
        }
        return double(ans)/2.;
    }
};
```

2. 公式推的。。没有啥意义的一道题

```
public double largestTriangleArea(int[][] points) {
        double res = 0;
        for (int[] a : points) {
            for (int[] b : points) {
                for (int[] c : points) {
                    res = Math.max(res, 0.5 * (c[0] * b[1] - c
[0] * a[1] - a[0] * b[1] + b[0] * a[1] - b[0] * c[1] + a[0] *
c[1]));
                }
            }
        }
        return res;
    }
```

310

1. 经典 Centroid 算法，不过好像很慢:

```
class Solution {
    typedef vector<int> vi;
    typedef vector<bool> vb;
    vector<vi> E;
    vi path;
    vb mark;
    int longest = 0;
    void dfs(int i, vi cur){
        cur.push_back(i);
        mark[i] = true;
        if(cur.size() > 1 && E[i].size() == 1){
            if(cur.size() > path.size()) path = cur;
            return;
        }
        for(int j: E[i]) if(!mark[j]) dfs(j, cur);
    }
public:
    vector<int> findMinHeightTrees(int n, vector<pair<int, int
>>& edges) {
        if(n==1) return vector<int>{0};
        vi ans;
        E.resize(n);
        for(auto p: edges) {
            E[p.first].push_back(p.second);
            E[p.second].push_back(p.first);
        }
        mark = vb(n, false);
        dfs(0, vi());
        int k = path.back();
        mark= vb(n, false);
        path.clear();
        dfs(k, vi());
        k = path.size();
        ans.push_back(path[(k-1)/2]);
        if(k%2 == 0) ans.push_back(path[k/2]);
        return ans;
    }
};
```

2. topological ?

```
public List<Integer> findMinHeightTrees(int n, int[][] edges)
{
        if (n == 1) return Arrays.asList(0);
        List<Set<Integer>> indegree = new ArrayList<>();

        for (int i = 0; i < n; i++) {
            indegree.add(new HashSet<>());
        }
        for (int[] edge : edges) {
            indegree.get(edge[0]).add(edge[1]);
            indegree.get(edge[1]).add(edge[0]);
        }

        List<Integer> leaves = new ArrayList<>();
        for (int i = 0; i < n; i++) {
            if (indegree.get(i).size() == 1) leaves.add(i);
        }

        while (n > 2) {
            n -= leaves.size();
            List<Integer> newLeaves = new ArrayList<>();
            for (int k : leaves) {
                Set<Integer> cur = indegree.get(k);
                int i = cur.iterator().next();
                indegree.get(i).remove(k);
                if (indegree.get(i).size() == 1) newLeaves.add
(i);
            }
            leaves = newLeaves;
        }
        return leaves;

    }
```

166

1. 题简单，就是陷阱一大堆

```
class Solution {
public:
    string fractionToDecimal(int de, int no) {
        long d = long(de), n = long(no);
        string ans = ((d>0&&n<0L || d<0&&n>0L)?"-":"") + to_st
ring(abs(d)/abs(n));
        d = abs(d);
        n = abs(n);
        d = (d%n)*10;
        if(!d) return ans;
        ans += ".";
        unordered_map<int, int> pos;
        while(d && !pos.count(d)){
            pos[d] = (int)ans.size();
            ans += to_string(d/n);
            d  = (d%n)*10;
        }
        if(d){
            ans = ans.substr(0, pos[d]) + "(" + ans.substr(pos
[d]) + ")";
        }
        return ans;
    }
};
```

2.被Long气到吐血的一道题

```
public String fractionToDecimal(int numerator, int denominato
r) {
        if (numerator == denominator) return "1";
        String sign = (numerator < 0 == denominator < 0 || num
erator == 0) ? "" : "-";
        StringBuilder sb = new StringBuilder();
        long num = Math.abs((long)numerator);
        long den = Math.abs((long)denominator);

        sb.append(sign);
        Map<Long, Integer> map = new HashMap<>();

        long value = num / den;
        sb.append(value);

        long remainder = num % den;
        if (remainder == 0) return sb.toString();
        sb.append(".");

        while (!map.containsKey(remainder)) {
            map.put(remainder, sb.length());
            sb.append(remainder * 10 / den);
            remainder = remainder * 10 % den;
        }

        sb.insert(map.get(remainder), "(");
        sb.append(")");
        return sb.toString().replace("(0)", "");
    }
```

665

1. 其实就是check 跳一个数能否组成 `non-descending`

```
class Solution {
public:
    bool checkPossibility(vector<int>& nums) {
        for(int i=1, head=nums[0], cnt=0; i<nums.size(); ++i){
            if(nums[i]<head){
                ++cnt;
                if(i<2 || nums[i]>=nums[i-2]) head = nums[i];
            }
            else head = nums[i];
            if(cnt>1) return false;
        }
        return true;
    }
};
```
2. 莫名其妙的一道题
```
public boolean checkPossibility(int[] nums) {
        boolean flag = false;

        for (int i = 1; i < nums.length; i++) {
            if (nums[i] < nums[i - 1]) {
                if (flag) return false;

                if (i < 2 || nums[i - 2] <= nums[i]) nums[i - 
1] = nums[i];
                else nums[i] = nums[i - 1];
                flag = true;
            }

        }
        return true;
    }
```

288

1. 难点何在？
```
class ValidWordAbbr {
    unordered_map<string, unordered_set<string>> H;
public:
    ValidWordAbbr(vector<string> dictionary) {
        for(string s: dictionary) {
            if(s.size() <= 2) H[s].insert(s);
            else H[s[0] + to_string(s.size()-2) + s[s.size()-
1]].insert(s);
        }
    }

    bool isUnique(string word) {
        string key;
        if(word.size()<=2) key = word;
        else key = word[0] + to_string(word.size()-2) + word[w
ord.size()-1];
        if(!H.count(key)) return true;
        if(H[key].count(word) && H[key].size()<2) return true;
        return false;
    }
};
```

2.

```
class ValidWordAbbr {

    Map<String, Set<String>> map;
    public ValidWordAbbr(String[] dictionary) {
        map = new HashMap<>();
        for (String s : dictionary) {
            String st = getAbbr(s);
            if (!map.containsKey(st)) {
                map.put(st, new HashSet<>());
            }
            map.get(st).add(s);
        }
    }

    public boolean isUnique(String word) {
        String s = getAbbr(word);
        if (map.containsKey(s)) {
            if (map.get(s).size() == 1 && map.get(s).contains
(word)) return true;
            else return false;
        }
        return true;
    }

    public String getAbbr(String s) {
        if (s.length() <= 2) return s;
        return "" + s.charAt(0) + (s.length() - 2) + s.charAt
(s.length() - 1);
    }
}


/**
 * Your ValidWordAbbr object will be instantiated and called a
s such:
 * ValidWordAbbr obj = new ValidWordAbbr(dictionary);
 * boolean param_1 = obj.isUnique(word);
 */
```

75

1. 严格意义上是1.5 pass

```
class Solution {
public:
    void sortColors(vector<int>& nums) {
        int i = 0, j = nums.size()-1;
        while(i<j){
            if(nums[i]) swap(nums[i], nums[j--]);
            else ++i;
        }
        j = nums.size()-1;
        while(i<j){
            if(nums[i] == 2) swap(nums[i], nums[j--]);
            else ++i;
        }
    }
};
```

2. One pass:

```
class Solution {
public:
    void sortColors(vector<int>& nums) {
        int i = 0, j, k=nums.size()-1;
        while(i<nums.size() && !nums[i]) ++i;
        while(k>=i && nums[k]==2) --k;
        for(int j=i; j<=k; ){
            if(!nums[j]) {
                swap(nums[i++], nums[j]);
                if(j<i) j = i;
            }
            else if(nums[j]==2) swap(nums[j], nums[k--]);
            else ++j;
        }
    }
};
```

1. dp 递归

```
class Solution {
    const int Mod = 1E9 + 7;
    void add(int &x, int y){
        x += y;
        if(x >= Mod) x %= Mod;
    }
public:
    int kInversePairs(int n, int k) {
        vector<int> dp(k+2, 0);
        dp[0] = 1;
        for(int r=2; r<=n; ++r){
            vector<int> tmp(k+2, 0);
            for(int j=0, l=0, sum=0; j<=min(k, r*(r-1)/2); ++
j){
                add(sum, dp[j]);
                if(j>=r) add(sum, Mod-dp[j-r]);
                tmp[j] = sum;
            }
            swap(dp, tmp);
        }
        return dp[k];
    }
};
```

447

1. 矩阵法，好慢

```
class Solution {
public:
    int numberOfBoomerangs(vector<pair<int, int>>& points) {
        int ans = 0;
        for(int i=0;i<points.size();++i){
            unordered_map<int, int> cnt;
            for(int j=0; j<points.size(); ++j) if(j!=i) {
                int dis = (points[j].first - points[i].first)
* (points[j].first - points[i].first);
                dis += (points[j].second - points[i].second)*
(points[j].second - points[i].second);
                cnt[dis]++;
            }
            for(auto p: cnt) ans += p.second * (p.second - 1);
        }
        return ans;
    }
};
```

731

1. 用 `map maintain` 区间

```
class MyCalendarTwo {
    map<int, int> one, two;
public:
    MyCalendarTwo() {
        one[-2] = -1;
        two[-2] = -1;
        one[int(2E9)] = int(2E9+3);
        two[int(2E9)] = int(2E9+3);
    }

    bool book(int start, int end) {
        auto it = two.upper_bound(start);
        if(it->first < end) return false;
        --it;
        if(it->second > start) return false;
        it = --one.upper_bound(start);
        if(it->second > start) {
            if(it->second >= end) two[start] = end;
            else two[start] = it->second;
            start = it->first;
            end = max(end, it->second);
        }
        ++it;
        while(it->first < end){
            two[it->first] = min(end, it->second);
            end = max(end, it->second);
            it = one.erase(it);
        }
        one[start] = end;
        return true;
    }
};
```

2.

```
class MyCalendarTwo {

    List<int[]> list2;

    public MyCalendarTwo() {
        list2 = new ArrayList<>();
    }

    public boolean book(int start, int end) {
        MyCalendar c = new MyCalendar();
        for (int[] l : list2) {
            if (Math.max(l[0], start) < Math.min(l[1], end)) {
                if (!c.book(Math.max(l[0], start), Math.min(l
[1], end))) return false;
            }
        }
        list2.add(new int[]{start, end});
        return true;
    }

    class MyCalendar {

    List<int[]> list;
    public MyCalendar() {
        list = new ArrayList<>();
    }

    public boolean book(int start, int end) {
        for (int[] l : list) {
            if (Math.max(l[0], start) < Math.min(l[1], end)) {
                return false;
            }
        }
        list.add(new int[]{start, end});
        return true;
    }
}
}
```

712

1. 典型的`dp`

```
class Solution {
```

```
public:
    int minimumDeleteSum(string s1, string s2) {
        int n = s1.size(), m = s2.size();
        vector<vector<int>> dp(n+1, vector<int>(m+1, 0));
        for(int i=n; i>=0; --i) for(int j=m; j>=0; --j){
            if(i==n && j==m) dp[i][j] = 0;
            else if(i==n) dp[i][j] = dp[i][j+1] + int(s2[j]);
            else if(j==m) dp[i][j] = dp[i+1][j] + int(s1[i]);
            else if(s1[i] != s2[j]) dp[i][j] = min(int(s1[i])
+ dp[i+1][j], int(s2[j]) + dp[i][j+1]);
            else dp[i][j] = dp[i+1][j+1];
        }
        return dp[0][0];
    }
};
```

519

Empty

517

1. 数学题：切割区间

```
class Solution {
public:
    int findMinMoves(vector<int>& machines) {
        int sum = 0, n = machines.size(), ans = 0;
        for(int k: machines) sum+=k;
        if(sum%n != 0) return -1;
        sum /= n;
        for(int &k: machines) k -= sum;
        sum = 0;
        for(int k: machines){
            sum += k;
            ans = max(max(ans, abs(sum)), max(k, (-k + 1)/2));
        }
        return ans;
    }
};
```