September 7, 2018 题目：358,552,833,494,644,672,842,296,829,590,446,631,45,304,193,199

358

1. 做过，但烦得一B：
   a. 把 个数为$n + k - 1/k$ 的往前排
   b. 把个数为n/k 的往后排
   c. 剩下的按贪心插在中间：

```
class Solution {
    typedef pair<int, char> ic;
public:
    string rearrangeString(string s, int k) {
        if(k<=1) return s;
        int n = s.size(), i = 0, j = 0, m = (int(s.size()) + k
- 1)/k, v = k;
        int l = (n%k? n%k: k);
        unordered_map<char, int> cnt;
        for(char c: s) ++cnt[c];
        set<ic, greater<ic>> reg;
        for(auto p: cnt) reg.insert(ic(p.second, p.first));
        string ans(s);
        for(auto it = reg.begin(); it!=reg.end() && it->first>
=n/k; ){
            if(it->first > m) return "";
            if(it->first == m){
                if(i >= l) return "";
                for(int z=0; z<m; ++z) ans[z*k + i] = it->seco
nd;
                ++i;
                it=reg.erase(it);
            }
            else if(it->first == n/k && v>l){
                --v;
                for(int z=0; z<n/k; ++z) ans[z*k + v] = it->se
cond;
                it=reg.erase(it);
            }
            else ++it;
        }
        for(auto p: reg){
            for(int t=0; t<p.first; ++t){
                ans[j*k + i] = p.second;
                ++j;
                if(i<l && j>= (n+k-1)/k){
                    ++i;
                    j = 0;
                }
                else if(i >= l && j >= n/k){
                    ++i;
                    j = 0;
                }
            }
        }
        return ans;
    }
};
```

552

1. Dp，做过的题懒得再做第二遍了：

```
class Solution {
    int M = 1E9 + 7;
    void add(int &x, int &y){
        x += y;
        if(x >= M) x -= M;
    }
public:
    /*
    i / 3: # of A
    i % 3: # of L
    */
    int checkRecord(int n) {
        vector<int> dp{1, 0, 0, 0, 0, 0};
        int ans = 0;
        for(int j=0; j<n; ++j){
            vector<int> tmp(6, 0);
            for(int i=0; i<6; ++i){
                if(i%3 == 0) for(int k=0; k<i+3; ++k) add(tmp
[i], dp[k]);
                else add(tmp[i], dp[i-1]);
            }
            swap(dp, tmp);
        }
        for(int i=0; i<6; ++i) {
            cout<<dp[i]<<endl;
            add(ans, dp[i]);
        }
        return ans;
    }
};
```

1. 这么蠢的题不明白之前为什么做那么麻烦：

```
class Solution {
    typedef pair<string, string> ss;
public:
    string findReplaceString(string S, vector<int>& indexes, vector<string>& sources, vector<string>& targets) {
        map<int, ss> pos;
        for(int i=0; i<indexes.size(); ++i) pos[indexes[i]] = ss(sources[i], targets[i]);
        string ans;
        int j = 0, n=S.size();
        auto it = pos.begin();
        while(j<n){
            while((it==pos.end()  || j<it->first) && j<n) ans.push_back(S[j++]);
            if(it!=pos.end() && S.substr(it->first, (int)it->second.first.size()) == it->second.first){
                ans += it->second.second;
                j += (int)it->second.first.size();
            }
            ++it;
        }
        return ans;
    }
};
```

494

1. 这种题，我开始居然用了dfs：

```
class Solution {
public:
    int findTargetSumWays(vector<int>& nums, int S) {
        if(nums.empty()) return S==0;
        unordered_map<int, int> cnt;
        cnt[0] = 1;
        for(int n: nums){
            unordered_map<int, int> tmp;
            for(auto p: cnt){
                tmp[p.first + n] += p.second;
                tmp[p.first - n] += p.second;
            }
            swap(tmp, cnt);
        }
        return cnt[S];
    }
};
```

2. 直接数组更快些：

```
class Solution {
public:
    int findTargetSumWays(vector<int>& nums, int S) {
        if(nums.empty()) return S==0;
        if(S > 1000) return 0;
        vector<int> dp(2018, 0);
        dp[1007] = 1;
        for(int n: nums){
            vector<int> tmp(2018, 0);
            for(int i=n; i<2017-n; ++i){
                tmp[i-n] += dp[i];
                tmp[i+n] += dp[i];
            }
            swap(tmp, dp);
        }
        return dp[1007 + S];
    }
};
```

644

1. 做过，典型的二分：

```
class Solution {
    const double delta = 1.E-7;
    bool ok(vector<int> &A, double avg, int k){
        vector<double> P(A.size()+1, 0.);
        for(int i=1; i<=A.size(); ++i) P[i] = P[i-1] + A[i-1]
- avg;
        double m = 0.;
        for(int i=0; i<=A.size()-k; ++i){
            m = min(m, P[i]);
            if(P[i+k] - m >= 0) return true;
        }
        return false;
    }
public:
    double findMaxAverage(vector<int>& nums, int k) {
        double l = -10001, r = 10001;
        while(l < r - delta){
            double c = (l + r)/2.;
            if(ok(nums, c, k)) l = c;
            else r = c;
        }
        return (l+r)/2.;
    }
};
```

672

1. 细节题：

```
class Solution {
public:
    int flipLights(int n, int m) {
        if(!n || !m) return 1;
        if(n==1) return 2;
        if(n<3){
            if(m%2){
                if(m == 1) return 3;
                else return 4;
            }
            else return 4;
        }
        if(m%2){
            if(m == 1) return 4;
            else return 8;
        }
        else{
            if(m==2) return 7;
            else return 8;
        }
    }
};
```

842

1. 单纯一个Brute Force:

```
class Solution {
    bool isF(int l0, int l1, const string&S, vector<int> &ans)
{
        if(l0 + l1 >= S.size()) return false;
        if(l0 > 1 && S[0]=='0') return false;
        if(l1 > 1 && S[l0] == '0') return false;
        long x0 = stol(S.substr(0, l0)), x1 = stol(S.substr(l
0, l1));
        if(x0 > long(INT_MAX) || x1 > long(INT_MAX)) return fa
lse;
        ans = vector<int>{(int)x0, (int)x1};
        int j = l0 + l1;
        while(j < S.size()){
            long cur = x0 + x1;
            if(cur > long(INT_MAX)) return false;
            string tmp = to_string(cur);
            if(S.substr(j, (int)tmp.size()) == tmp){
                x0 = x1;
                x1 = cur;
                j += (int)tmp.size();
                ans.push_back((int)cur);
            }
            else return false;
        }
        return true;
    }
public:
    vector<int> splitIntoFibonacci(string S) {
        int n = S.size();
        vector<int> ans;
        for(int l0=1; l0<=9; ++l0) for(int l1=1; l1<=9; ++ l1)
if(l0 + l1 < n && isF(l0, l1, S, ans)) return ans;
        return vector<int>();
    }
};
```

2. backtracking

```
class Solution {
    public List<Integer> splitIntoFibonacci(String s) {
        List<Integer> res = new ArrayList<>();
        helper(res, 0, s);
        return res;
    }

    private boolean helper(List<Integer> list, int start, Stri
ng s) {
        if (start == s.length() && list.size() >= 3) {
            return true;
        }

        for (int i = start; i < s.length(); i++) {
            if (s.charAt(start) == '0' && i > start) {
                break;
            }
            String cur = s.substring(start, i + 1);
            long num = Long.parseLong(cur);
            if (num > Integer.MAX_VALUE) {
                break;
            }
            int size = list.size();
            if (size <= 1 || list.get(size - 1) + list.get(siz
e - 2) == num) {
                list.add((int) num);
                if (helper(list, i + 1, s)) {
                    return true;
                }
                list.remove(list.size() - 1);
            }
        }
        return false;
    }
}
```

296

1. 取下x 跟 y 的中位数即可：

```
class Solution {
public:
    int minTotalDistance(vector<vector<int>>& G) {
        if(G.empty() || G[0].empty()) return 0;
        int n = G.size(), m = G[0].size();
        vector<int> I, J;
        for(int i=0; i<n; ++i) for(int j=0; j<m; ++j) if(G[i]
[j]){
            I.push_back(i);
        }
        for(int j=0; j<m; ++j) for(int i=0; i<n; ++i) if(G[i]
[j]){
            J.push_back(j);
        }
        int dis = 0, k = I.size();
        for(int i=0; i< k/2; ++i) dis += I[k-1-i] - I[i] + J[k
-1-i] - J[i];
        return dis;
    }
};
```

829

1. Short is Beauty 系列：

```
class Solution(object):
    def consecutiveNumbersSum(self, N):
        print int((2*N)**0.5)
        return sum([int(((2*N)%j == 0) and (2*N/j-j) % 2==1) f
or j in range(1, int((2*N)**0.5+1))])
```

590

2. Preorder + reverse:

```
class Solution {
public:
    vector<int> postorder(Node* root) {
        vector<int> ans;
        stack<Node *> S;
        while(!S.empty() || root){
            while(root){
                ans.push_back(root->val);
                if(root->children.empty()) root = NULL;
                else{
                    for(int j=0; j<(int)root->children.size()-
1; ++j){
                        S.push(root->children[j]);
                    }
                    root = root->children.back();
                }
            }
            if(!S.empty()){
                root = S.top();
                S.pop();
            }
        }
        reverse(ans.begin(), ans.end());
        return ans;
    }
};
```
2.
```
class Solution {
    public List<Integer> postorder(Node root) {
        List<Integer> list = new ArrayList<>();
        helper(root, list);
        return list;
    }

    private void helper(Node root, List<Integer> list) {
        if (root == null) {
            return;
        }

        for (Node child : root.children) {
            helper(child, list);
        }
        list.add(root.val);
    }
}
```

446

✓ ~~这个dp居然超时了： @Zebo L~~

```
class Solution {
public:
    int numberOfArithmeticSlices(vector<int>& A) {
        unordered_map<long, int> one;
        unordered_map<long, unordered_map<long, int>> two, thr
ee;
```

```
        for(int n: A){
            if(!three.count(n)) three[n] = unordered_map<long, int>();
            if(!two.count(n)) two[n] = unordered_map<long, int>();
            for(auto p: three) if(p.second.count(n - p.first)){
                three[n][n-p.first] += p.second[n-p.first];
            }
            for(auto p: two) if(p.second.count(n - p.first)){
                three[n][n-p.first] += p.second[n-p.first];
            }
            for(auto p: one) two[n][n-p.first] += p.second;
            one[n]++;
        }
        int ans = 0;
        for(auto p: three) for(auto q: p.second) ans += q.second;
        return ans;
    }
};
```

2. 调了半天，最后用array居然能过，这题对C++的判定时间也太强了吧：
    - Note: 下面的跟上面的时间复杂度相同，都是 `O(`n^2`)`

```
class Solution {
public:
    int numberOfArithmeticSlices(vector<int>& A) {
        int ans = 0, n = A.size();
        vector<unordered_map<int, int>> dp(n);
        for(int i=0; i<n; ++i){
            for(int j=0; j<i; ++j) {
                if(abs(long(A[i]) - long(A[j])) > long(INT_MA
X)) continue;
                if(dp[j].count(A[i] - A[j])) {
                    dp[i][A[i] - A[j]] += dp[j][A[i] - A[j]];
                    ans += dp[j][A[i] - A[j]];
                }
                dp[i][A[i] - A[j]] += 1;
            }
        }
        return ans;
    }
};
```

631

1. 就是烦：

```
class Excel {
    #define CI(c) int((c) - 'A')
    unordered_map<int, unordered_set<int>> from;
    unordered_map<int, unordered_map<int, int>> to;
    vector<vector<int>> M;
public:
    Excel(int H, char W) { M = vector<vector<int>>(H, vector<i
nt>(CI(W)+1, 0));}

    void dfsAdd(int k, int delta){
        M[k/26][k%26] += delta;
        for(auto p: to[k]) dfsAdd(p.first, p.second * delta);
    }

    void set(int r, char c, int v) {
        int i = r-1, j = CI(c);
        int k = i * 26 + j, delta = v - M[i][j];
        for(auto x: from[k]) to[x].erase(k);
        from[k].clear();
        dfsAdd(k, delta);
    }
    int get(int r, char c) {
        int i = r-1, j = CI(c);
        return M[i][j];
    }
    int sum(int r, char c, vector<string> strs) {
        int i = r-1, j = CI(c);
        int k = i * 26 + j, ans = 0;
        for(auto x: from[k]) to[x].erase(k);
        from[k].clear();
        for(string s: strs){
            auto m = s.find(':');
            if(m==string::npos){
                int j0 = CI(s[0]), i0 = stoi(s.substr(1)) - 1;
                ++to[i0 * 26 + j0][k];
                from[k].insert(i0 * 26 + j0);
                ans += M[i0][j0];
            }
            else{
                int j0 = CI(s[0]), i0 = stoi(s.substr(1, m-1))
- 1;
                int j1 = CI(s[m+1]), i1 = stoi(s.substr(m+2))-
1;
                for(int i_=i0; i_<=i1; ++i_) for(int j_=j0; j_
<=j1; ++j_){
                    ++to[i_ * 26 + j_][k];
                    from[k].insert(i_ * 26 + j_);
                    ans += M[i_][j_];
                }
            }
        }
        int delta = ans - M[i][j];
        dfsAdd(k, delta);
        return ans;
    }
};
```

45

1. 细节题：

```
class Solution {
public:
    int jump(vector<int>& nums) {
        if(nums.size() < 2) return 0;
        int ans = 1, i = 0, j = 1;
        while(i + nums[i] < nums.size() - 1){
            int idx = i;
            while(j<nums.size() && j <= i + nums[i]){
                if(j + nums[j] > idx + nums[idx]) idx = j;
                ++j;
            }
            assert(idx > i);
            i = idx;
            ++ans;
        }
        return ans;
    }
};
```

2.

```
class Solution {
    public int jump(int[] nums) {
        // 2 3 1 1 4
        int max = 0;
        int end = 0;
        int jumps = 0;
        for (int i = 0; i < nums.length - 1; i++) {
            max = Math.max(max, i + nums[i]);
            if (i == end) {
                jumps++;
                end = max;
            }
        }
        return jumps;
    }
}
```

304

1. Index Tree:

```
class NumMatrix {
    typedef vector<int> vi;
    vector<vi> M;
public:
    NumMatrix(vector<vector<int>> matrix) {
        if(!matrix.empty() && !matrix[0].empty()){
            int n = matrix.size(), m = matrix[0].size();
            M = vector<vi>(n+1, vi(m+1, 0));
            for(int i=1; i<=n; ++i) for(int j=1; j<=m; ++j) M[i][j] = M[i][j-1] + M[i-1][j] - M[i-1][j-1] + matrix[i-1][j-1];
        }
    }

    int sumRegion(int row1, int col1, int row2, int col2) {
        if(M.empty()) return 0;
        return M[row2+1][col2+1] - M[row1][col2+1] - M[row2+1]
[col1] + M[row1][col1];
    }
};
```

193

FUCKING BASH

199

1. DFS :

```
class Solution {
    void dfs(TreeNode *root, int l, vector<int> &res){
        if(!root) return;
        if(res.size() <= l) res.push_back(root->val);
        else res[l] = root->val;
        dfs(root->left, l+1, res);
        dfs(root->right, l+1, res);
    }
public:
    vector<int> rightSideView(TreeNode* root) {
        vector<int> ans;
        dfs(root, 0, ans);
        return ans;
    }
};
```