July 22, 2018 题目：387,783,658,224,568,791,585,537,807,771,205,135

387

1. 计数题：

```
class Solution {
public:
    int firstUniqChar(string s) {
        vector<int> M(26, 0);
        for(char c: s) ++M[int(c-'a')];
        for(int i=0; i<s.size(); ++i) if(M[int(s[i]-'a')] < 2) 
return i;
        return -1;
    }
};
```

Python Solution:

```
class Solution(object):
    def firstUniqChar(self, s):
        dic = {}
        for x in range(len(s)):
            if s[x] in dic:
                dic[s[x]] = -1
            else:
                dic[s[x]] = x

        lss = len(s)
        for x in dic:
            if dic[x] != -1 and lss > dic[x]:
                lss = dic[x]

        if lss == len(s):
            return -1
        else:
            return lss
```

783

1. 居然没有出恶心的case

```
class Solution {
public:
    int minDiffInBST(TreeNode* root) {
        int ans = INT_MAX, head = INT_MIN;
        stack<TreeNode *> S;
        while(root || !S.empty()){
            while(root){
                S.push(root);
                root = root->left;
            }
            if(!S.empty()){
                root = S.top()->right;
                if(head!=INT_MIN) ans = min(ans, S.top()->val
- head);
                head = S.top()->val;
                S.pop();
            }
        }
        return ans;
    }
};
```

2. inorder

```
class Solution {
    int min;
    TreeNode pre = null;
    public int minDiffInBST(TreeNode root) {
        if (root == null) return 0;
        min = Integer.MAX_VALUE;
        helper(root);
        return min;
    }
    public void helper(TreeNode node) {
        if (node == null) {
            return;
        }
        helper(node.left);
        if (pre != null) {
            min = Math.min(Math.abs(node.val - pre.val), min);
        }
        pre = node;
        helper(node.right);
    }
}
```

Java Solution:

```
class Solution {
    public int helper(TreeNode root, int[] data){
        if(root == null) return -1;
        helper(root.left, data);

        if(data[2] != -1){
            data[1] = (root.val - data[0] < data[1]) ? root.va
l - data[0] : data[1];
```

```
        }
        data[0] = root.val;
        data[2] = 1;

        helper(root.right,data);
        return data[1];
    }
    public int minDiffInBST(TreeNode root) {
        int[] temp = {0,Integer.MAX_VALUE,-1};
        return helper(root,temp);
    }
}
```

658

1. 奇妙的二分，不小心还是容易出错的

```
class Solution {
public:
    vector<int> findClosestElements(vector<int>& arr, int k, i
nt x) {
        int l = 0, r = arr.size() - k;
        while(l < r-1){
            int c = (l+r)/2;
            if(arr[c+k] - x >= x - arr[c]) r = c;
            else l = c;
        }
        if(l+k<arr.size() && arr[l+k]-x < x-arr[l]) ++l;
        return vector<int>(arr.begin()+l, arr.begin() + l +
k);
    }
};
```

2. PriorityQueue ← 不知道为啥我想了这么一个复杂的办法

```
class Solution {
    class Tuple {
        int num;
        int diff;
        public Tuple (int num, int val) {
            this.num = num;
            this.diff = Math.abs(num - val);
        }
    }
    public List<Integer> findClosestElements(int[] arr, int k, int x) {
        PriorityQueue<Tuple> pq = new PriorityQueue<Tuple>(1, new Comparator<Tuple>(){
            public int compare(Tuple a, Tuple b) {
                return a.diff == b.diff ? b.num - a.num : b.diff - a.diff;
            }
        });
        for (int a : arr) {
            pq.offer(new Tuple(a, x));
            if (pq.size() > k) pq.poll();
        }
        List<Integer> list = new ArrayList<>();
        while (!pq.isEmpty()) {
            list.add(pq.poll().num);
        }
        Collections.sort(list);
        return list;
    }
}
```

3. Binary Search → Find K closest, Binary Search 也是自己写了不知道面试可不可以用 library

```
class Solution {
    public int binary(int[] arr, int target){
        int i = 0;
        int j = arr.length - 1;

        while(i <= j){
            int median = ((j - i) / 2) + i;
            if(arr[median] == target) return median;
            else if(arr[median] > target){
                j = median - 1;
            }
            else{
                i = median + 1;
            }
        }
        i = (i >= arr.length) ? arr.length : i;
        j = (j < 0) ? 0 : j;
        if(Math.abs(arr[i] - target) < Math.abs(arr[j] - targe
t)){
            return i;
        }
        else{
            return j;
        }
    }


    public List<Integer> findClosestElements(int[] arr, int k,
int x) {
        LinkedList<Integer> ll = new LinkedList<Integer>();
        int median = binary(arr,x);
        int left = (median - k >= 0) ? median - k : 0;
        int right = (median + k  < arr.length) ? median + k : 
arr.length - 1;

        ll.add(arr[median]);
        int i = median - 1;
        int j = median + 1;
        while(i >= left && j <= right){
            if(x - arr[i] <= arr[j] - x){
                ll.addFirst(arr[i]);
                i--;
            }
            else{
                ll.add(arr[j]);
                j++;
            }
            if(ll.size() == k){
                return ll;
            }
        }
        while(i >= left && ll.size() < k){
            ll.addFirst(arr[i]);
            i--;
        }
        while(j <= right && ll.size() < k){
            ll.add(arr[j]);
            j++;
        }
        return ll;
```

```
    }
}
```

224

1. Without parentheses, we can only use the sum. Whenever sees the (, store the previous sum and operator on the stack.

```
class Solution {
    public int calculate(String s) {
        if (s == null || s.length() == 0) {
            return 0;
        }

        int sum = 0;
        int op = 1;
        int num = 0;
        int idx = 0;
        int n = s.length();
        Stack<Integer> st = new Stack<>();

        while (idx < n) {
            char c = s.charAt(idx);
            if (c == ' '){
                idx++;
                continue;
            }
            if (c >= '0' && c <= '9') {
                num = num * 10 + c - '0';
            }
            if (c == '(') {
```

```
            st.push(sum);
            st.push(op);
            sum = 0;
            op = 1;
        }
        if (c == '+') {

            if (op == 1) {
                sum += num;
                num = 0;
            }
            if (op == 2) {
                sum -= num;
                num = 0;
            }

                                op = 1;
        }
        if (c == '-') {

            if (op == 1) {
                sum += num;
                num = 0;
            }
            if (op == 2) {
                sum -= num;
                num = 0;
            }
                                op = 2;
        }
```

```
            if (c == ')') {
                if (op == 1) {
                    sum += num;
                    num = 0;
                    op = 1;
                }
                if (op == 2) {
                    sum -= num;
                    num = 0;
                    op = 1;
                }
                int prev_op = st.pop();
                int prev_num = st.pop();
                if (prev_op == 1) {
                    sum = prev_num + sum;
                }
                if (prev_op == 2) {
                    sum = prev_num - sum;
                }
            }
            idx++;
        }
        if (op == 1) {
            sum += num;
            num = 0;
        }
        if (op == 2) {
            sum -= num;
            num = 0;
        }
        return sum;
    }
}
```
2.

```
class Solution {
    public int calculate(String s) {
        Stack<Integer> stack = new Stack<>();
        int res = 0;
        int sign = 1;
        int num = 0;
        for (char c : s.toCharArray()) {
            if (Character.isDigit(c)) {
                num = num * 10 + (c - '0');
            } else if (c == '+') {
                res += sign * num;
                sign = 1;
                num = 0;
            } else if (c == '-') {
                res += sign * num;
                sign = -1;
                num = 0;
            } else if (c == '(') {
                stack.push(res);
                stack.push(sign);
                sign = 1;
                res = 0;
            } else if (c == ')') {
                res += sign * num;
                num = 0;
                res *= stack.pop();
                res += stack.pop();
            }
        }
        if (num != 0) res += sign * num;
        return res;
    }
}
```

3. Beat 1.17% ... 为啥这么慢

```
class Solution {
public:
    int calculate(string s) {
        int ans = 0, sign = 1, i = 0;
        stack<int> A, B;
        while(i<s.size() && i>=0){
            if(s[i] == ' ') ++i;
            else if(s[i]>='0' && s[i]<='9'){
                ans += sign * stoi(s.substr(i));
                i = s.find_first_not_of("0123456789", i);
                sign = 1;
            }
            else if(s[i] == '+') {
                sign = 1;
                ++i;
            }
            else if(s[i] == '-'){
                sign = -1;
                ++i;
            }
            else if(s[i] == '('){
                A.push(ans);
                B.push(sign);
                ans = 0;
                sign = 1;
                ++i;
            }
            else{
                ans = A.top() + B.top() * ans;
                sign = 1;
                A.pop();
                B.pop();
                ++i;
            }
        }
        assert(A.empty() && B.empty());
        return ans;
    }
};
```

568

1. DFS + memo

```
class Solution {
    int N;
    int K;
    public int maxVacationDays(int[][] flights, int[][] days)
{
        N = flights.length;
        K = days[0].length;
        int[][] caches = new int[K][N];
        for (int i = 0; i < K; i++) {
            Arrays.fill(caches[i], -1);
        }
        return DFS(flights, days, 0, 0, caches);
    }

    public int DFS(int[][] flights, int[][] days, int k, int
i, int[][] caches) {
        if (k == K) return 0;
        if (caches[k][i] >= 0) return caches[k][i];
        int max = 0;
        for (int j = 0; j < N; j++) {
            if (i == j || flights[i][j] == 1) {
                max = Math.max(max, DFS(flights, days, k + 1,
j, caches) + days[j][k]);
            }
        }
        caches[k][i] = max;
        return max;
    }
}
```

2. Just 1D-dp:

```
class Solution {
public:
    int maxVacationDays(vector<vector<int>>& flights, vector<v
ector<int>>& days) {
        int N = flights.size(), K = days[0].size(), ans = 0;
        vector<int> dp(N, -1);
        dp[0] = 0;
        for(int i=0; i<N; ++i) flights[i][i] = 1;
        for(int k=0; k<K; ++k){
            vector<int> tmp(N, -1);
            for(int i=0; i<N; ++i) for(int j=0; j<N; ++j) if(flights[j][i] && dp[j]>=0){
                tmp[i] = max(tmp[i], dp[j] + days[i][k]);
            }
            swap(dp, tmp);
        }
        for(int k: dp) ans = max(k, ans);
        return ans;
    }
};
```

3. dfs + memo

```
class Solution {
    public int maxVacationDays(int[][] flights, int[][] days) {
        int len = days[0].length;
        int[][] memo = new int[flights.length][len];
        for (int[] m : memo) {
            Arrays.fill(m, -2);
        }

        return dfs(flights, days, 0, len, 0, 0, memo);
    }
    int dfs(int[][] flights, int[][] days, int week, int len, int sum, int city, int[][] memo) {
        if (week == len) {
            return 0;
        }
        if (memo[city][week] >= 0) {
            return memo[city][week];
        }
        int ans = -1;
        for (int i = 0; i < flights[week].length; i++) {
            if (flights[city][i] == 1 || i == city) {
                int vac = days[i][week] + dfs(flights, days, w
eek + 1, len, sum, i, memo);
                ans = Math.max(ans, vac);
            }
        }
        memo[city][week] = ans;
        return ans;
    }
}
```

791

1. 直接排序就行了: 但有点慢

```
map<char, int> REF;
bool cmp(const char &a, const char &b){
    if(REF.count(a) && REF.count(b)) return REF[a] < REF[b];
    return a < b;
}
class Solution {
public:
    string customSortString(string S, string T) {
        for(int i=0; i<S.size(); ++i) REF[S[i]] = i;
        string s1, s2;
        for(char c: T) {
            if(!REF.count(c)) s1 += c;
            else s2 += c;
        }
        sort(s2.begin(), s2.end(), cmp);
        return s2 + s1;
    }
};
```

2. 自己build整个string, sort 都不用其实

```
class Solution(object):
    def customSortString(self, S, T):
        """
        :type S: str
        :type T: str
        :rtype: str
        """

        string = "";
        for x in S:
            k = T.count(x);
            j = 0;
            while j < k:
                string = string + x
                j += 1

        for y in T:
            if y not in S:
                string = string + y;

        return string;
```

585

SQL

537

1. 难点何在

```
class Solution {
public:
    string complexNumberMultiply(string a, string b) {
        int x1 = stoi(a), y1 = stoi(a.substr(a.find('+')+1));
        int x2 = stoi(b), y2 = stoi(b.substr(b.find('+')+1));
        return to_string(x1*x2 - y1*y2) + "+" + to_string(x1*y2 + y1*x2) + "i";
    }
};
```

807

1. 先确定skyline， 然后在loop 一遍，保证building 增高后不超过对应的skyline 即可

```
class Solution {
public:
    int maxIncreaseKeepingSkyline(vector<vector<int>>& grid) {
        int n = grid.size(), m = grid[0].size(), ans=0;
        vector<int> ub(m, 0), lr(n, 0);
        for(int i=0;i<n;++i) for(int j=0;j<m;++j){
            ub[j] = max(ub[j], grid[i][j]);
            lr[i] = max(lr[i], grid[i][j]);
        }
        for(int i=0; i<n; ++i) for(int j=0; j<m; ++j) ans += min(ub[j], lr[i]) - grid[i][j];
        return ans;
    }
};
```

只有我用的语言跳来跳去。。。

```
class Solution(object):
    def maxIncreaseKeepingSkyline(self, grid):
        hor_arr = []
        for x in grid:
            hor_arr.append(max(x))

        ver_arr = []

        for x in range(len(grid[0])):
            temp = -1
            for y in range(len(grid)):
                if grid[y][x] > temp:
                    temp = grid[y][x]
            ver_arr.append(temp)

        sum = 0
        psum = 0
        for x in range(len(grid[0])):
            for y in range(len(grid)):
                psum += grid[x][y]
                sum += min(hor_arr[x],ver_arr[y])

        return sum-psum
```

771

1. 没什么好说的

```
class Solution {
public:
    int numJewelsInStones(string J, string S) {
        unordered_set<char> jew(J.begin(), J.end());
        int ans = 0;
        for(auto c: S) ans+=jew.count(c);
        return ans;
    }
};
```

205

1. 双向map

```
class Solution {
public:
    bool isIsomorphic(string s, string t) {
        if(s.size()!=t.size()) return false;
        map<char, char> sm, tm;
        for(int i=0; i<s.size(); ++i){
            if(sm.count(s[i]) && sm[s[i]] != t[i]) return fals
e;
            if(tm.count(t[i]) && tm[t[i]] != s[i]) return fals
e;
            sm[s[i]] = t[i];
            tm[t[i]] = s[i];
        }
        return true;
    }
};
```

135

1. 来回各扫一遍即可，难点何在

```
class Solution {
public:
    int candy(vector<int>& ratings) {
        vector<int> ans(ratings.size(), 1);
        for(int i=ratings.size()-2; i>=0; --i) if(ratings[i] >
ratings[i+1]) ans[i] = ans[i+1] + 1;
        int sum = ans[0];
        for(int i=1; i<ratings.size(); ++i){
            if(ratings[i] > ratings[i-1]) ans[i] = max(ans[i],
ans[i-1]+1);
            sum += ans[i];
        }
        return sum;
    }
};
```