August 23, 2018 题目：346,779,522,681,370,509,585,206,211,260,713,724,464,626,230

346

1. 做过：

```
class MovingAverage {
public:
    /** Initialize your data structure here. */
    int cnt, cap, sum;
    queue<int> Q;
    MovingAverage(int size): cap(size), cnt(0), sum(0){}

    double next(int val) {
        Q.push(val);
        sum += val;
        if(cnt<cap) ++cnt;
        else{
            sum -= Q.front();
            Q.pop();
        }
        return double(sum)/cnt;
    }
};
```

779

1. Recursion:

```
class Solution {
    int dfs(int n, int k){
        if(!n){
            assert(!k);
            return 0;
        }
        int pre = dfs(n-1, k/2);
        return (k%2) ^ pre;
    }
public:
    int kthGrammar(int N, int K) {
        return dfs(N-1, K-1);
    }
};
```

2. Stack :

```
class Solution {
public:
    int kthGrammar(int N, int K) {
        --N;
        --K;
        stack<int> Q;
        while(N){
            Q.push(K);
            K/=2;
            --N;
        }
        int ans = 0;
        while(!Q.empty()){
            ans ^= Q.top()%2;
            Q.pop();
        }
        return ans;
    }
};
```

522

1.

```
class Solution {
    public int findLUSlength(String[] strs) {
        Arrays.sort(strs, new Comparator<String>() {
            public int compare(String s1, String s2) {
                return s2.length() - s1.length();
            }
        });

        for (int i = 0; i < strs.length; i++) {
            boolean isValid = true;
            for (int j = 0; j < strs.length; j++) {
                if (j != i && strs[j].length() >= strs[i].leng
th()) {
                    if (checkSub(strs[j], strs[i])) {
                        isValid = false;
                        break;
                    }
                }
            }
            if (isValid) return strs[i].length();
        }
        return -1;
    }

    private boolean checkSub(String s, String sub) {
        int i = 0;
        for (char c : s.toCharArray() ) {
            if (i < sub.length() && c == sub.charAt(i)) {
                i++;
            }
        }
        return i == sub.length();
    }

}
```

2. 标准Brute Force:

```
class Solution {
    bool isSubSeq(string a, string b){
        if(b.size() > a.size()) return false;
        unordered_map<char, set<int>> pos;
        for(int i=0; i<a.size(); ++i) pos[a[i]].insert(i);
        int idx = -1;
        for(char c: b){
            auto it = pos[c].upper_bound(idx);
            if(it == pos[c].end()) return false;
            idx = *it;
        }
        return true;
    }
public:
    int findLUSlength(vector<string>& strs) {
        int L = -1;
        for(int i = 0; i<strs.size(); ++i) if((int)strs[i].size() > L) {
            bool ok = true;
            for(int j=0; j<strs.size() && ok; ++j) if(i!=j && isSubSeq(strs[j], strs[i])) ok = false;
            if(ok) L = strs[i].size();
        }
        return L;
    }
};
```

681

1. 几天前刚做过:

```
class Solution {
    typedef pair<int, int> ii;
    #define CI(c) int((c) - '0')
    #define IC(i) char((i) + '0')
public:
    string nextClosestTime(string time) {
        ii ref = ii(CI(time[0])*10 + CI(time[1]), CI(time[3])*
10 + CI(time[4]));
        set<int> pool{CI(time[0]), CI(time[1]), CI(time[3]), C
I(time[4])};
        vector<int> dig(pool.begin(), pool.end());
        int n = dig.size();
        set<ii> S;
        for(int k=0; k<int(pow(n, 4)); ++k){
            int h = dig[k%n] * 10 + dig[k/n%n];
            int m = dig[k/n/n%n] * 10 + dig[k/n/n/n%n];
            if(h<24 && m < 60) S.insert(ii(h, m));
        }
        for(auto s: S)cout<<s.first<<' '<<s.second<<endl;
        ii ans = *S.begin();
        auto it = S.upper_bound(ref);
        if(it!=S.end()) ans = *it;
        string res;
        return res + IC(ans.first/10) + IC(ans.first%10) + ":" + IC(ans.second/10) + IC(ans.second%10);
    }
};
```

370

1. Index Tree:

```
class Solution {
public:
    vector<int> getModifiedArray(int length, vector<vector<int>>& updates) {
        vector<int> P(length+1, 0);
        for(auto v: updates){
            P[v[0]] += v[2];
            P[v[1] + 1] -= v[2];
        }
        for(int i=0, tmp=0; i<length; ++i){
            tmp += P[i];
            P[i] = tmp;
        }
        P.pop_back();
        return P;
    }
};
```

2. very stupid solution

```
class Solution {
    public int[] getModifiedArray(int length, int[][] updates) {
        int[] res = new int[length];
        for (int[] up : updates) {
            for (int i = up[0]; i <= up[1]; i++) {
                res[i] += up[2];
            }
        }
        return res;
    }
}
```

509

没有这道题

585

SQL

206

1.

```
/**
 * Definition for singly-linked list.
 * public class ListNode {
 *     int val;
 *     ListNode next;
 *     ListNode(int x) { val = x; }
 * }
 */
public class Solution {
    public ListNode reverseList(ListNode head) {
        if (head == null) return head;
        ListNode newhead = null;
        while (head != null) {
            ListNode next = head.next;
            head.next = newhead;
            newhead = head;
            head = next;
        }
        return newhead;

    }
}
```
2. 同上：
```
class Solution {
public:
    ListNode* reverseList(ListNode* head) {
        if(!head || !head->next) return head;
        ListNode *p1=head->next, *p2=head->next->next;
        for(head->next=NULL; p1; head=p1, p1=p2) {
            p2 = p1->next;
            p1->next = head;
        }
        return head;
    }
};
```

211

1. Trie and Recursion
```
class WordDictionary {
    class TrieNode {
        boolean isWord;
        TrieNode[] children;

        public TrieNode () {
            children = new TrieNode[26];
        }
    }
    /** Initialize your data structure here. */

    TrieNode root;
    public WordDictionary() {
        root = new TrieNode();
    }

    /** Adds a word into the data structure. */
    public void addWord(String word) {
        TrieNode node = root;
        for (char c : word.toCharArray()) {
            if (node.children[c - 'a'] == null) {
                node.children[c - 'a'] = new TrieNode();
            }
            node = node.children[c - 'a'];
        }
        node.isWord = true;
    }

    /** Returns if the word is in the data structure. A word c
ould contain the dot character '.' to represent any one lette
r. */
    public boolean search(String word) {
        return match(word, root);
    }

    private boolean match (String s, TrieNode node) {
        if (s.length() < 1) return node.isWord;
        if (node == null) return false;
        TrieNode cur = node;
        for (int i = 0; i < s.length(); i++) {
            char c = s.charAt(i);
            if (c == '.') {
                int j = 0;
                for (TrieNode child : cur.children) {
                    //System.out.println((char) ((j++) +
'a'));
                    if (child != null && match(s.substring(i +
1), child)) {
                        return true;
                    }
                }
                return false;
            } else {
               if (cur.children[c - 'a'] == null) return fals
e;
                cur = cur.children[c - 'a'];
            }
        }
        return cur.isWord;
    }
}

/**
 * Your WordDictionary object will be instantiated and called
as such:
 * WordDictionary obj = new WordDictionary();
 * obj.addWord(word);
 * boolean param_2 = obj.search(word);
 */
```

2. 同上：But slow

```
struct T{
    bool W;
    unordered_map<char, T*> C;
    T(): W(false) {}
};
class WordDictionary {
    T *root;
    bool dfs(T* r, string &s, int i){
        while(i<s.size() && s[i]!='.'){
            if(!r->C.count(s[i])) return false;
            r = r->C[s[i++]];
        }
        if(i==s.size()) return r->W;
        for(auto p: r->C) if(dfs(p.second, s, i+1)) return tru
e;
        return false;
    }
public:
    WordDictionary() { root = new T(); }
    void addWord(string word) {
        T *r = root;
        for(char c: word){
            if(!r->C.count(c)) r->C[c] = new T();
            r = r->C[c];
        }
        r->W = true;
    }
    bool search(string word) {
        return dfs(root, word, 0);
    }
};
```

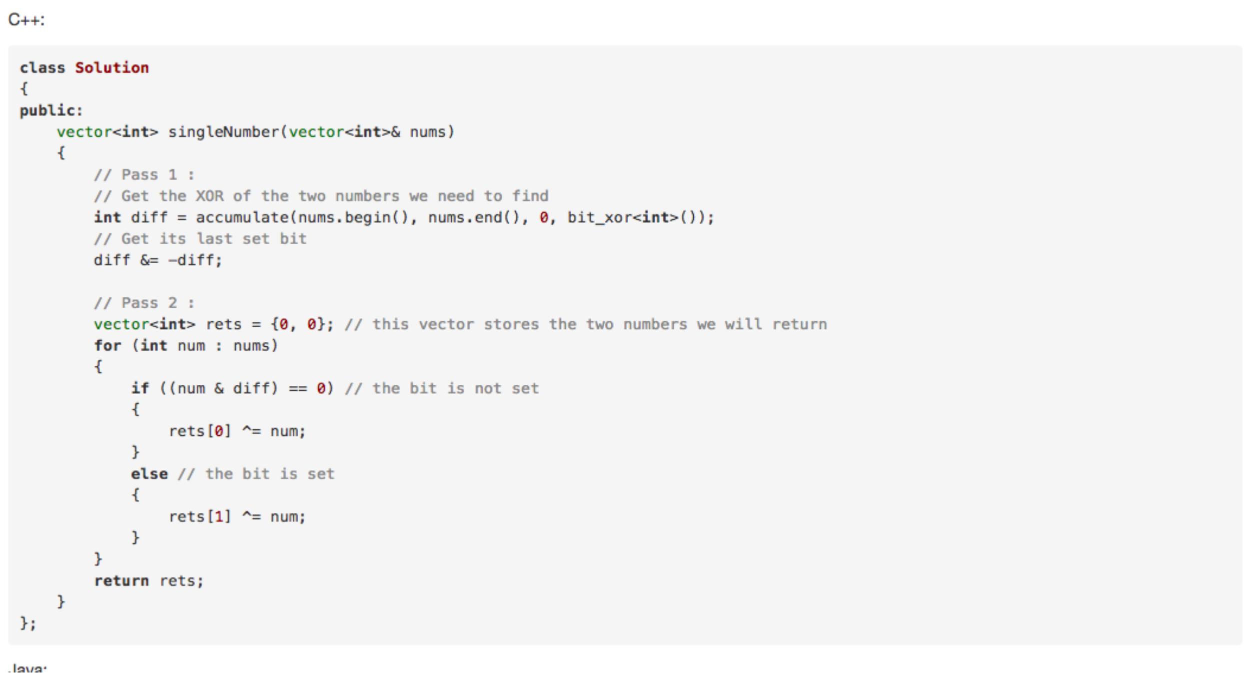
260

1.

```
class Solution {
    public int[] singleNumber(int[] nums) {
        int[] res = new int[2];
        Set<Integer> set = new HashSet<>();
        for (int n : nums) {
            if (!set.add(n)) set.remove(n);
        }
        Iterator<Integer> i = set.iterator();
        res[0] = i.next();
        res[1] = i.next();
        return res;
    }
}
```

2. Linear Time + Constant space：其实可以继续优化：
    - 思路类似 Bucket sort:
    - 根据数组的最大值跟最小值，分成两个bucket

```
class Solution {
    vector<int> dfs(int l, int r, long m, long M, vector<int>&
nums){
        cout<<l<<' '<<r<<' '<<m<< ' '<<M<< endl;
        if(m == M-2) return vector<int>{m, M-1};
        if(l == r-2) return vector<int>{nums[l], nums[r-1]};
        int mid = (m+M) / 2, left = 0, right = 0, i = l, j =
r;
        while(i<r){
            while(nums[i] >= mid && i < j){
                --j;
                swap(nums[i], nums[j]);
            }
            if(nums[i] < mid) left ^= nums[i];
            else right ^= nums[i];
            ++i;
        }
        cout<<left<<' '<<right<<endl;
        if(left && right) return vector<int>{left, right};
        if(left) return dfs(l, j, m, mid, nums);
        return dfs(j, r, mid, M, nums);
    }
public:
    vector<int> singleNumber(vector<int>& nums) {
        int c0 = 0;
        for(int k: nums) if(!k) c0++;
        if(c0==1){
            int ans = 0;
            for(int k:nums) ans ^= k;
            return vector<int>{0, ans};
        }
        long m = INT_MAX, M = INT_MIN;
        for(int k: nums){
            m = min(m, (long)k);
            M = max(M, (long)k+1);
        }
        return dfs(0, (int)nums.size(), m, M, nums);
    }
};
```
3. 看了以下解法，发现自己被秒成渣渣了：（我tm 上面在干什么啊。。）



C++:

```
class Solution
{
public:
    vector<int> singleNumber(vector<int>& nums)
    {
        // Pass 1 :
        // Get the XOR of the two numbers we need to find
        int diff = accumulate(nums.begin(), nums.end(), 0, bit_xor<int>());
        // Get its last set bit
        diff &= -diff;

        // Pass 2 :
        vector<int> rets = {0, 0}; // this vector stores the two numbers we will return
        for (int num : nums)
        {
            if ((num & diff) == 0) // the bit is not set
            {
                rets[0] ^= num;
            }
            else // the bit is set
            {
                rets[1] ^= num;
            }
        }
        return rets;
    }
};
```

Java:

发现自己无比愚蠢

- [x] ~~Use `n & -n` to get last bit，这个真是长见识了 @Zebo L ,之前对负数的bit 表示一直不是很熟。~~

713

1. Sliding window

```
class Solution {
    public int numSubarrayProductLessThanK(int[] nums, int k)
{
        if (nums == null || nums.length == 0) return 0;
        int i = 0, j = 0;
        int res = 0;
        int product = 1;
        while (i < nums.length) {
            product *= nums[i];
            while (product >= k && j <= i) {
                product = product / nums[j];
                j++;
            }
            res += (i - j + 1);
            i++;



        }
        return res;
    }
}
```
2. Conner case 真的好烦啊：
```
class Solution {
public:
    int numSubarrayProductLessThanK(vector<int>& nums, int k)
{
        int i = 0, j = 0, prod = 1, ans = 0;
        while(i<nums.size() && j<nums.size()){
            if(i==j && nums[i]>=k){
                ++i;
                ++j;
            }
            while(j<nums.size() && prod < k) prod *= nums[j+
+];
            if(prod<k) break;
            while(i<j && prod >= k){
                ans += j-i-1;
                prod /= nums[i++];
            }
        }
        while(i<j){
            ans += j-i++;
        }
        return ans;
    }
};
```

724

1. 注意边界

```
class Solution {
    public int pivotIndex(int[] nums) {
        if (nums == null || nums.length == 0) return -1;
        int[] sum = Arrays.copyOf(nums, nums.length);

        for (int i = nums.length - 1; i > 0; i--) {
            sum[i - 1] += sum[i];
        }
        int start = 0;
        if (sum[1] == 0) return 0;
        for (int i = 0; i < nums.length - 2; i++) {
            start += nums[i];
            if (start == sum[i + 2]) return i + 1;
        }
        if (start + nums[nums.length - 2] == 0) return nums.le
ngth - 1;
        return -1;
    }
}
```

2. 同上 其实可以O 1 space

```
class Solution {
public:
    int pivotIndex(vector<int>& nums) {
        int n = nums.size();
        vector<int> P(n+1, 0);
        for(int i=1; i<=n; ++i) P[i] = P[i-1] + nums[i-1];
        for(int i=0; i<n; ++i) if(P[n] - P[i+1] == P[i]) retur
n i;
        return -1;
    }
};
```

464

1. Bit map dp: 不是很快

```
class Solution {
    unordered_map<int, unordered_map<int, bool>> dp;
    int U, n;
    bool dfs(int k, int stat){
        if(k >= U) return false;
        if(dp.count(k) && dp[k].count(stat)) return dp[k][sta
t];
        for(int j=0; j<n; ++j) if(!(1&(stat>>j)) && !dfs(k+j+
1, stat | (1<<j))) return dp[k][stat] = true;
        return dp[k][stat] = false;
    }
public:
    bool canIWin(int maxChoosableInteger, int desiredTotal) {
        n = maxChoosableInteger;
        U = desiredTotal;
        if(!U) return true;
        if(n*(n+1)/2 < U) return false;
        return dfs(0, 0);
    }
};
```

626

SQL

230

1. inorder

```
class Solution {
    int res = -1;
    int k;
    public int kthSmallest(TreeNode root, int k) {
        this.k = k;
        helper(root);
        return res;
    }

    public void helper(TreeNode node) {
        if (node == null) {
            return;
        }
        helper(node.left);
        k--;
        if (k == 0) res = node.val;
        helper(node.right);
    }
}
```

2. 同上：

```
class Solution {
public:
    int kthSmallest(TreeNode* root, int k) {
        stack<TreeNode *> S;
        while(root || !S.empty()){
            while(root){
                S.push(root);
                root=root->left;
            }
            if(!S.empty()){
                --k;
                if(!k) return S.top()->val;
                root = S.top()->right;
                S.pop();
            }
        }
        return -1;
    }
};
```

3. Binary Search

```
class Solution {
    public int kthSmallest(TreeNode root, int k) {
        if (root == null) return 0;
        int ct = count(root.left);
        if (k <= ct) {
            return kthSmallest(root.left, k);
        }
        if (k > ct + 1) {
            return kthSmallest(root.right, k - ct - 1);
        }
        return root.val;
    }

    private int count(TreeNode root) {
        if (root == null) return 0;
        return 1 + count(root.left) + count(root.right);
    }
}
```