August 24, 2018 题目：146,276,139,155,530,104,19,668,179,837,99,180,95,198,233

146

1. 写过好几遍了，还是没法一次写对 （又忘了get也算是use）：

```
class LRUCache {
    typedef pair<int, int> ii;
    unordered_map<int, list<ii>::iterator> pos;
    list<ii> SC;
    int cap;
public:
    LRUCache(int capacity): cap(capacity) {}
    int get(int key) {
        if(!pos.count(key)) return -1;
        SC.push_back(*pos[key]);
        SC.erase(pos[key]);
        pos[key] = --SC.end();
        return SC.back().second;
    }
    void put(int key, int value) {
        if(pos.count(key)) SC.erase(pos[key]);
        SC.push_back(ii(key, value));
        pos[key] = --SC.end();
        if(SC.size()>cap){
            pos.erase(SC.begin()->first);
            SC.erase(SC.begin());
        }
    }
```

```
};
```

276

1. `O(n)` dp: 歪里歪里 stupid

```
class Solution {
public:
    int numWays(int n, int k) {
        if(!n || !k) return 0;
        vector<int> dp(2*k, 0);
        for(int i=0; i<k; ++i) dp[i] = 1;
        int ans = k;
        for(int j=1; j<n; ++j){
            vector<int> tmp(2*k, 0);
            int cnt = 0;
            for(int i=0; i<k; ++i) {
                tmp[i] = ans - dp[i] - dp[k+i];
                cnt += tmp[i];
            }
            for(int i=k; i<2*k; ++i){
                tmp[i] = dp[i-k];
                cnt += tmp[i];
            }
            swap(dp, tmp);
            swap(ans, cnt);
        }
        return ans;
    }
};
```

2. 学到了，`O(n)` time + `O(1)` space:

```
class Solution {
public:
```

```
    int numWays(int n, int k) {
        if(!n || !k) return 0;
        int a = 1, b = 0, ans = k;
        for(int i=1; i<n; ++i){
            int c = a;
            a = ans - a - b;
            b = c;
            ans = k*(a + b);
        }
        return ans;
    }
};
```

139

1. 做过 dfs + memo :

```
class Solution {
    #define CI(c) int((c) - 'a')
    struct T{
        bool isW;
        vector<T *> ch;
        T(): ch(vector<T *>(26, NULL)), isW(false) {}
    };
    void ins(T *r, string w){
        for(char c: w){
            if(!r->ch[CI(c)]) r->ch[CI(c)] = new T();
            r = r->ch[CI(c)];
        }
        r->isW = true;
    }
    unordered_map<int, int> dp;
```

```
    bool dfs(T*root, int k, string &s){
        if(k == s.size()) return true;
        if(dp.count(k)) return dp[k];
        int i = k;
        T *r = root;
        while(i<s.size()){
            if(!r->ch[CI(s[i])]) return dp[k]=false;
            r = r->ch[CI(s[i++])];
            if(r->isW && dfs(root, i, s)) return dp[k]=true;
        }
        return dp[k]=false;
    }
public:
    bool wordBreak(string s, vector<string>& wordDict) {
        T *root = new T();
        for(auto w: wordDict) ins(root, w);
        return dfs(root, 0, s);
    }
};
```

155

1. 经典 Two Stacks:

```
class MinStack {
public:
    /** initialize your data structure here. */
    stack<int> S, mS;
    MinStack() {

    }
```

```
    void push(int x) {
        S.push(x);
        if(mS.empty() || x<=mS.top()) mS.push(x);
    }

    void pop() {
        if(S.empty()) return;
        if(S.top() == mS.top()) mS.pop();
        S.pop();
    }

    int top() {
        return S.top();
    }

    int getMin() {
        return mS.top();
    }
};
```

530

1. 做过，inorder 即可:

```
class Solution {
public:
    int getMinimumDifference(TreeNode* root) {
        long dif = -1, cur = 0;
        stack<TreeNode *> S;
        while(root || !S.empty()){
            while(root){
                S.push(root);
```

```
                root = root->left;
            }
            if(!S.empty()){
                if(dif == -1) dif = long(2) * INT_MAX;
                else dif = min(dif, (long)S.top()->val - cur);
                cur = (long)S.top()->val;
                root = S.top()->right;
                S.pop();
            }
        }
        return dif;
    }
};
```

104

1. Short is Beauty 系列:

```
class Solution {
public:
    int maxDepth(TreeNode* root) { return (root?1+max(maxDepth
(root->left), maxDepth(root->right)): 0); }
};
```

19

1.

```
class Solution {
    public ListNode removeNthFromEnd(ListNode head, int n) {
        if (n == 0) return head;
        ListNode dummy = new ListNode(0);
        dummy.next = head;
        ListNode fast = dummy;
        while (n-- >= 0) {
```

```
            fast = fast.next;
        }
        ListNode slow = dummy;

        while (fast != null) {
            fast = fast.next;
            slow = slow.next;
        }
        slow.next = slow.next.next;
        return dummy.next;
    }
}
```

2. 之前只会Two pass，看到楼上的方法，被折服！！

```
class Solution {
public:
    ListNode* removeNthFromEnd(ListNode* head, int n) {
        if(!head) return NULL;
        ListNode lead(0);
        lead.next = head;
        ListNode *fast=&lead, *slow=&lead;
        while(n--) fast = fast->next;
        while(fast->next){
            fast = fast->next;
            slow = slow->next;
        }
        fast = slow->next->next;
        delete slow->next;
        slow->next = fast;
        return lead.next;
    }
```

```
};
```

668

1. 典型的二分，数数的时候要注意细节：

```
class Solution {
public:
    int findKthNumber(int m, int n, int k) {
        if(n>m) swap(m, n);
        int l = 0, r = n*m+1;
        while(l < r-1){
            int c = (l+r)/2, cnt=0;
            for(int i=1; i<=n && i<=c-1; ++i) cnt += min(m, (c
-1)/i);
            if(cnt<k) l = c;
            else r = c;
        }
        return l;
    }
};
```

179

1. Short is Beauty 系列：

```
class Solution(object):
    def largestNumber(self, nums):
        return str(int(''.join(sorted([str(x) for x in nums],
cmp=lambda x, y: 1 if x+y<y+x else -1))))
```

837

1. 想了好几天，没有想到maintain window sum, 太蠢了:

```
class Solution {
    typedef long double LD;
public:
```

```
    double new21Game(int N, int K, int W) {
        if(!W) return LD(N<=K);
        if(!K) return max(LD(1.), LD(N)/W);
        vector<LD> dp(K+W, 0.);
        LD window_sum = 1., ans = 0.;
        dp[0] = 1.;
        for(int j=1; j<K+W; ++j){
            dp[j] = 1./W * window_sum;
            if(j < K) window_sum += dp[j];
            if(j >= W) window_sum -= dp[j-W];
        }
        for(int i=K; i<=N; ++i) ans += dp[i];
        return ans;
    }
};
```

99

1. inorder

```
class Solution {
    TreeNode pre = null;
    TreeNode first = null;
    TreeNode second = null;
    public void recoverTree(TreeNode root) {
        inorder(root);
        int tmp = first.val;
        first.val = second.val;
        second.val = tmp;
    }

    public void inorder(TreeNode root) {
```

```
        if (root == null) return;
        inorder(root.left);
        if (pre != null && pre.val >= root.val) {
            if (first == null) first = pre;
            second = root;
        }
        pre = root;
        inorder(root.right);
    }
}
```

2. 被楼上方法所折服：

```
class Solution {
    TreeNode *pre=NULL, *first=NULL, *second=NULL;
    void dfs(TreeNode *root){
        if(!root) return;
        dfs(root->left);
        if(pre && pre->val >= root->val){
            if(!first) first = pre;
            second = root;
        }
        pre = root;
        dfs(root->right);
    }
public:
    void recoverTree(TreeNode* root) {
        dfs(root);
        assert(first && second);
        swap(first->val, second->val);
    }
};
```

2. Stack:

```
class Solution {
public:
    void recoverTree(TreeNode* r) {
        stack<TreeNode *> L, R;
        TreeNode *first=NULL, *second=NULL, *root = r;
        while(root||!L.empty()){
            while(root){
                L.push(root);
                root = root->left;
            }
            if(!L.empty()){
                if(first && first->val >= L.top()->val) break;
                first = L.top();
                L.pop();
                root = first->right;
            }
        }
        root = r;
        while(root||!R.empty()){
            while(root){
                R.push(root);
                root = root->right;
            }
            if(!R.empty()){
                if(second && second->val <= R.top()->val) brea
k;
                second = R.top();
                R.pop();
                root = second->left;
```

```
            }
        }
        swap(first->val, second->val);
    }
};
```

180

SQL

95

1. 递归：

```
class Solution {
    vector<TreeNode*> dfs(int l, int r){
        if(l>r) return {NULL};
        vector<TreeNode*> ans;
        for(int v=l;v<=r;++v){
            vector<TreeNode*> left = dfs(l, v-1), right = dfs
(v+1, r);
            for(auto le: left) for(auto ri: right){
                TreeNode *root = new TreeNode(v);
                root->left = le;
                root->right = ri;
                ans.push_back(root);
            }
        }
        return ans;
    }
public:
    vector<TreeNode*> generateTrees(int n) {
        if(!n) return {};
        return dfs(1, n);
```

```
    }
};
```

198

1.

```
class Solution {
    public int rob(int[] nums) {
        int rob = 0;
        int notRob = 0;

        for (int i = 0; i < nums.length; i++) {
            int tmp = notRob;
            notRob = Math.max(rob, notRob);
            rob = nums[i] + tmp;
        }
        return Math.max(rob, notRob);
    }
}
```

2. Short is Beauty 系列: dp

```
class Solution(object):
    def rob(self, nums):
        res  = (0, 0)
        for n in nums: res = (max(res), res[0] + n)
        return max(res)
```

233

1. 细心点就OK：

```
class Solution {
    inline long getAllOne(long l){ return l * long(pow(10, l-1)); }
public:
```

```
    int countDigitOne(int n) {
        if(n<10) return (n>0);
        long M=1, l=0;
        while(M*10 <= n){
            l++;
            M*=10;
        }
        long y = n/M, x = n%M;
        if(y==1) return getAllOne(l) + x + countDigitOne(x) +
1;
        return y*getAllOne(l) + M + countDigitOne(x);
    }
};
```