July 21, 2018 题目：369,520,315,535,545,826,189,716,375,86,781,628

369

1. 2* reverseList:

```
class Solution {
    ListNode* reverseList(ListNode * head){
        if(!head || !head->next) return head;
        ListNode *p1=head,*p2=head->next;
        for( ;p2;){
            ListNode *tmp = p2->next;
            p2->next = p1;
            p1 = p2;
            p2 = tmp;
        }
        head->next = NULL;
        return p1;
    }
public:
    ListNode* plusOne(ListNode* head) {
        head = reverseList(head);
        int carrier = 1;
        ListNode *p=head;
        while(p || carrier){
            carrier += p->val;
            p->val = carrier%10;
            carrier /= 10;
            if(!p->next && carrier) p->next = new ListNode(0);
```

```
            p = p->next;
        }
        return reverseList(head);
    }
};
```

2. Recursion:

```
class Solution {
    int PlusOne(ListNode *head){
        if(!head) return 1;
        int val = PlusOne(head->next);
        val += head->val;
        head->val = val%10;
        return val/10;
    }
public:
    ListNode* plusOne(ListNode* head) {
        int val = PlusOne(head);
        if(val) {
            ListNode *ans = new ListNode (1);
            ans->next = head;
            head = ans;
        }
        return head;

    }
};
```

520

1. 毕竟 Easy:

```
class Solution {
public:
```

```
    bool detectCapitalUse(string word) {
        if(word.size()<=1) return true;
        if(word[0]>='A' && word[0]<='Z'){
            if(word[1] >= 'A' && word[1] <= 'Z'){
                for(int j=2; j<word.size(); ++j) if(word[j]
>'Z' || word[j]<'A') return false;
                return true;
            }
            else{
                for(int j=2; j<word.size(); ++j) if(word[j]
>'z' || word[j]<'a') return false;
                return true;
            }
        }
        for(auto c: word) if(c>'z' || c<'a') return false;
        return true;
    }
};
```

315

1. Counting BST:

```
struct T{
    int val, cnt, less;
    T *left, *right;
    T(int x): val(x), cnt(1), less(0), left(NULL), right(NULL)
{}
};
T *insertNode(T *root, int val, int&ans){
    if(!root) return new T(val);
    if(val == root->val) {
        ans += root->less;
```

```
        root->cnt++;
    }
    else if(val > root->val){
        root->right = insertNode(root->right, val, ans);
        ans += root->less + root->cnt;
    }
    else{
        root->less++;
        root->left = insertNode(root->left, val, ans);
    }
    return root;
}
class Solution {
public:
    vector<int> countSmaller(vector<int>& nums) {
        T *root = NULL;
        vector<int> ans(nums.size(), 0);
        for(int i=nums.size()-1; i>=0; --i) root = insertNode
(root, nums[i], ans[i]);
        return ans;
    }
};
```

- [ ] 想想有没其他方法，每次都建个树好麻烦 @Zebo L

535

1. 这就过了，`O(1)` space, `O(1)` time

```
class Solution {
public:

    // Encodes a URL to a shortened URL.
    string encode(string longUrl) {
```

```
        return longUrl;
    }

    // Decodes a shortened URL to its original URL.
    string decode(string shortUrl) {
        return shortUrl;
    }
};
```

545

1. 不难，但容易错

```
class Solution {
public:
    vector<int> boundaryOfBinaryTree(TreeNode* root) {
        vector<int> ans;
        if(!root) return ans;
        stack<TreeNode *>S;
        if(root->left || root->right) ans.push_back(root->va
l);
        TreeNode *p = root->left;
        while(p) {
            if(p->left || p->right) ans.push_back(p->val);
            if(p->left) p = p->left;
            else p=p->right;
        }
        p = root;
        while(p || !S.empty()){
            while(p){
                if(!p->left && !p->right) ans.push_back(p->va
l);
                if(p->right) S.push(p->right);
```

```
                p = p->left;
            }
            if(!S.empty()) {
                p = S.top();
                S.pop();
            }
        }
        int l = ans.size();
        p = root->right;
        while(p){
            if(p->left || p->right) ans.push_back(p->val);
            if(p->right) p = p->right;
            else p = p->left;
        }
        reverse(ans.begin() + l, ans.end());
        return ans;
    }
};
```

826

1. 排序+贪心

```
class Solution {
public:
    int maxProfitAssignment(vector<int>& difficulty, vector<in
t>& profit, vector<int>& worker) {
        map<int, int, greater<int>> pro;
        for(int i=0; i<profit.size(); ++i){
            if(!pro.count(profit[i]) || pro[profit[i]] > diffi
culty[i]) pro[profit[i]] = difficulty[i];
        }
        sort(worker.begin() ,worker.end(), greater<int>());
```

```
        int ans = 0, j = 0;
        auto it = pro.begin();
        while(j < worker.size() && it!=pro.end()){
            if(it->second > worker[j]) ++it;
            else{
                ans += it->first;
                ++j;
            }
        }
        return ans;
    }
};
```

189

1. Inplace + `O(n)` + `O(1)`

```
class Solution {
    int gcd(int x, int y){
        if(x < y) swap(x, y);
        if(!y) return x;
        return gcd(y, x%y);
    }
public:
    void rotate(vector<int>& nums, int k) {
        int n = nums.size();
        int m = gcd(k, n);
        for(int i=0; i<m; ++i){
            for(int j=0, idx=i, head=nums[i]; j<n/m; ++j){
                int tmp = nums[(idx+k)%n];
                nums[(idx+k)%n] = head;
                head = tmp;
```

```
                idx = (idx + k)%n;
            }
        }
    }
};
```

2. Only using swap:

```
class Solution {
    int gcd(int x, int y){
        if(x < y) swap(x, y);
        if(!y) return x;
        return gcd(y, x%y);
    }
public:
    void rotate(vector<int>& nums, int k) {
        int n = nums.size();
        int m = gcd(k, n);
        for(int i=0; i<m; ++i){
            for(int j=1, idx=(i+k)%n; j<n/m; ++j){
                swap(nums[i], nums[idx]);
                idx = (idx + k)%n;
            }
        }
    }
};
```

716

1. 凭什么是Hard：

```
class MaxStack {
public:
    /** initialize your data structure here. */
    typedef pair<int, int> ii;
```

```
    int cnt;
    set<ii, greater<ii>> tstack, vstack;
    MaxStack() {
        cnt = 0;
    }

    void push(int x) {
        tstack.insert(ii(cnt, x));
        vstack.insert(ii(x, cnt));
        ++cnt;
    }

    int pop() {
        int ans = tstack.begin()->second;
        vstack.erase(ii(tstack.begin()->second, tstack.begin()
->first));
        tstack.erase(tstack.begin());
        return ans;
    }

    int top() {
        return tstack.begin()->second;
    }

    int peekMax() {
        return vstack.begin()->first;
    }

    int popMax() {
        int ans = vstack.begin()->first;
```

```
        tstack.erase(ii(vstack.begin()->second, vstack.begin()
->first));
        vstack.erase(vstack.begin());
        return ans;
    }
};
```

375

1. 别想当然，直接 `O(n^3)` dp

```
class Solution {
public:
    int getMoneyAmount(int n) {
        vector<vector<int>> dp(n+2, vector<int>(n+2, 0));
        for(int l=1; l<n; ++l){
            for(int i=1; i<=n-l; ++i){
                int j = i+l;
                dp[i][j] = n * n;
                for(int k=i; k<=j; ++k) {
                    dp[i][j] = min(dp[i][j], k + max(dp[i][k-
1], dp[k+1][j]));
                }
            }
        }
        return dp[1][n];
    }
};
```

- [ ] Look at this one https://leetcode.com/problems/guess-number-higher-or-lower-ii/discuss/84826/An-O(n2)-DP-Solution-Quite-Hard. @Zebo L

86

1. 额外造一个List (可以依次插入，不过麻烦一些)

```
class Solution {
```

```
public:
    ListNode* partition(ListNode* head, int x) {
        ListNode lead1(0), lead2(0);
        ListNode *pre1 = &lead1, *pre2 = &lead2;
        while(head){
            if(head->val < x) {
                pre1->next = head;
                pre1 = head;
            }
            else{
                pre2->next = head;
                pre2 = head;
            }
            head = head->next;
        }
        pre2->next = NULL;
        pre1->next = lead2.next;
        return lead1.next;
    }
};
```

781

1. 计数问题

```
class Solution {
public:
    int numRabbits(vector<int>& answers) {
        unordered_map<int, int> cnt;
        for(auto n: answers) ++cnt[n];
        int ans = 0;
        for(auto p: cnt){
```

```
            int m = (p.second + p.first) / (p.first + 1);
            ans += m * (p.first + 1);
        }
        return ans;
    }
};
```

628

1. 真是烦啊 `O(n) & O(1)`

```
class Solution {
public:
    int maximumProduct(vector<int>& nums) {
        if(nums.size() == 3) return nums[0]*nums[1]*nums[2];
        int np = 0;
        for(auto k: nums) np += (k>=0);
        if(!np){
            int n1 = INT_MIN, n2 = INT_MIN, n3 = INT_MIN;
            for(int n: nums){
                if(n >= n1){
                    n3 = n2;
                    n2 = n1;
                    n1 = n;
                }
                else if(n>=n2){
                    n3 = n2;
                    n2 = n;
                }
                else if(n >= n3){
                    n3 = n;
                }
```

```
            return n1 * n2 * n3;
        }
    }
    priority_queue<int> pq, nq;
    for(int n: nums){
        if(n>=0){
            pq.push(-n);
            if(pq.size() > 3) pq.pop();
        }
        else{
            nq.push(n);
            if(nq.size() > 2) nq.pop();
        }
    }
    if(nq.size() < 2){
        int ans = 1;
        while(!pq.empty()) {
            ans *= -pq.top();
            pq.pop();
        }
        return ans;
    }
    if(pq.size()<3){
        int ans = 1;
        while(!pq.empty()) {
            ans = - pq.top();
            pq.pop();
        }
        while(!nq.empty()){
            ans *= nq.top();
```

```
                nq.pop();
            }
            return ans;
        }
        int p1 = -pq.top(); pq.pop();
        int p2 = -pq.top(); pq.pop();
        int p3 = -pq.top();
        int n1 = nq.top(); nq.pop();
        int n2 = nq.top();
        return max(p1 * p2 * p3, n1 * n2 * p3);
    }
};
```