August 31, 2018 题目：230,327,832,173,541,122,16,21,811,245,736,441,149,238,828,814,60

230

1. Inorder:

```
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;
    }
};
```

- [ ] Stackless 遍历的算法 @Zebo L

2. inorder recursion

```
class Solution:
    curr = 0
```

```
    def kthSmallest(self, root, k):
        def inorder(root):
            if root == None:
                return None
            ans = inorder(root.left)
            if ans != None:
                return ans
            self.curr += 1
            if self.curr == k:
                return root.val
            ans = inorder(root.right)
            if ans != None:
                return ans
            return None
        return inorder(root)
```

327

1. Counting BST:

```
struct CntBst{
    long val;
    int cnt;
    int cnt_left;
    int cnt_right;
    CntBst *left, *right;
    CntBst(long val_): val(val_), cnt(1), cnt_left(0), cnt_rig
ht(0), left(NULL), right(NULL) {}
};

CntBst *insertCntBst(CntBst *root, long val){
    if(!root) return new CntBst(val);
    if(root->val == val) ++root->cnt;
```

```
    else if(root->val > val){
        ++root->cnt_left;
        root->left = insertCntBst(root->left, val);
    }
    else{
        ++root->cnt_right;
        root->right = insertCntBst(root->right, val);
    }
    return root;
}

int cntLess(CntBst *root, long val){
    int ans = 0;
    while(root && root->val != val){
        if(root->val > val) root = root->left;
        else{
            ans += root->cnt + root->cnt_left;
            root = root->right;
        }
    }
    if(root) ans += root->cnt_left;
    return ans;
}

int cntGreater(CntBst *root, long val){
    int ans = 0;
    while(root && root->val != val){
        if(root->val > val){
            ans += root->cnt + root->cnt_right;
            root = root->left;
```

```
        }
        else root = root->right;
    }
    if(root) ans += root->cnt_right;
    return ans;
}

class Solution {
public:
    int countRangeSum(vector<int>& nums, int lower, int upper)
{
        CntBst *root = new CntBst(0);
        int ans = 0;
        for(long i=0, m=0; i<nums.size(); ++i){
            m += nums[i];
            int ng = cntGreater(root, m-lower);
            int nl = cntLess(root, m-upper);
            ans += i + 1 - ng - nl;
            root = insertCntBst(root, m);
        }
        return ans;
    }
};
```

- [ ] Use merge sort to do counting @Zebo L

832

1. 做过：

```
class Solution {
public:
    vector<vector<int>> flipAndInvertImage(vector<vector<int>>
& A) {
```

```
        for(auto &vec: A) {
            for(int i=0; i<(vec.size()+1)/2; ++i){
                int tmp = vec[i];
                vec[i] = 1 - vec[vec.size()-i-1];
                vec[vec.size()-i-1] = 1 - tmp;
            }
        }
        return A;
    }
};
```

2.swap

```
class Solution:
    def flipAndInvertImage(self, A):
        if not A: return [[]]
        w = len(A[0])
        for row in A:
            for i in range(0, int((w+1)/2)):
                row[i], row[w-i-1] = 1 - row[w-i-1], 1 - row
[i]
        return A
```

173

1. Stack:

```
class BSTIterator {
    stack<TreeNode *> S;
public:
    BSTIterator(TreeNode *root) {
        while(root) {
            S.push(root);
            root = root->left;
        }
```

```
    }

    /** @return whether we have a next smallest number */
    bool hasNext() {
        return !S.empty();
    }

    /** @return the next smallest number */
    int next() {
        auto ans = S.top(), root=S.top()->right;
        S.pop();
        while(root){
            S.push(root);
            root = root->left;
        }
        return ans->val;
    }
};
```

2. stack

```
class BSTIterator(object):
    def __init__(self, root):
        """
        :type root: TreeNode
        """
        self.stack = []
        while root:
            self.stack.append(root)
            root = root.left

    def hasNext(self):
```

```
        """
        :rtype: bool
        """
        return len(self.stack) != 0

    def next(self):
        """
        :rtype: int
        """
        node = self.stack.pop()
        ans = node.val
        if node.right:
            node = node.right
            while node:
                self.stack.append(node)
                node = node.left

        return ans
```

541

1. Short is Beauty 系列 :

```
class Solution(object):
    def reverseStr(self, s, k):
        return ''.join([(s[i: i+k])[::-1] if (i/k)%2==0 else s
[i: i+k] for i in range(0, len(s), k)])
```

122

2. Short is Beauty 系列 :

```
class Solution(object):
    def maxProfit(self, prices):
```

```
        return sum([max(0, prices[i+1]-prices[i]) for i in ran
ge(len(prices)-1)])
```

16

1. Two sum:

```
class Solution {
public:
    int threeSumClosest(vector<int>& nums, int target) {
        int ans = nums[0] + nums[1] + nums[2], n = nums.size
();
        sort(nums.begin(), nums.end());
        for(int i=0; i<n-2; ++i) {
            for(int j=i+1, k=n-1; j<k; ){
                int tmp=nums[i]+nums[j]+nums[k];
                if(abs(tmp - target) < abs(ans - target)) ans
= tmp;
                if(tmp >= target) --k;
                else ++j;
            }
        }
        return ans;
    }
};
```

2. Three sum:

```
from sys import maxsize
class Solution:
    def threeSumClosest(self, nums, target):
        ans = []
        nums.sort()
        minDiff = maxsize
```

```
        for i in range(len(nums) - 2):
            if i > 0 and nums[i] == nums[i-1]:
                continue

            l = i + 1
            r = len(nums) - 1

            while r > l:
                currSum = nums[i] + nums[l] + nums[r]

                if abs(currSum - target) < minDiff:
                    minDiff = abs(currSum - target)
                    ans = currSum
                    if minDiff == 0: break

                if currSum - target > 0:
                    r -= 1
                else:
                    l += 1
        return ans
```

21

1. Easy:

```
class Solution {
public:
    ListNode* mergeTwoLists(ListNode* l1, ListNode* l2) {
        ListNode lead(0);
        for(ListNode *p=&lead; l1 || l2; p=p->next){
            if(!l2 || (l1 && l1->val<l2->val)) {
                p->next = l1;
```

```
                l1 = l1->next;
            }
            else{
                p->next = l2;
                l2 = l2->next;
            }
        }
        return lead.next;
    }
};
```

811

1. Use hash map to count number:

```
class Solution {
public:
    vector<string> subdomainVisits(vector<string>& cpdomains)
{
        unordered_map<string, int> cnt;
        for(string s: cpdomains){
            int tmp = stoi(s);
            auto i = s.find(' ') + 1;
            while(true){
                cnt[s.substr(i)] += tmp;
                i = s.find('.', i);
                if(i!=string::npos) ++i;
                else break;
            }
        }
        vector<string> ans;
        for(auto p: cnt) ans.push_back(to_string(p.second) + "
" + p.first);
```

```
        return ans;
    }
};
```

245

1. 依次比较：

```
class Solution {
public:
    int shortestWordDistance(vector<string>& words, string word1, string word2) {
        int pos1=-1, pos2=-1, ans=words.size();
        if(word1 == word2){
            for(int i=0; i<words.size(); ++i) if(words[i] == word1){
                if(pos1 >= 0) ans = min(ans, i-pos1);
                pos1 = i;
            }
        }
        else{
            for(int i=0; i<words.size(); ++i) if(words[i] == word1 || words[i] == word2) {
                if(words[i] == word1) pos1 = i;
                else pos2 = i;
                if(pos1>=0 && pos2>=0) ans = min(ans, abs(pos1 - pos2));
            }
        }
        return ans;
    }
};
```

736

1. 细节题： Use recursion

```
class Solution {
    typedef unordered_map<string, int> usi;
    vector<string> getTokens(string s){
        if(s[0] == '(') s = s.substr(1, s.size()-2);
        vector<string> tokens;
        auto i = s.find_first_not_of(" ");
        while(i<s.size()){
            if(s[i]=='('){
                int j = i+1;
                for(int cnt=1; j<s.size() && cnt; ++j){
                    if(s[j]=='(') ++cnt;
                    else if(s[j] == ')') --cnt;
                }
                tokens.push_back(s.substr(i, j-i));
                i = j+1;
            }
            else{
                auto j = s.find(' ', i);
                if(j==string::npos) j = s.size();
                tokens.push_back(s.substr(i, j-i));
                i = j+1;
            }
        }
        return tokens;
    }
    int evalu(string s, usi eval){
        auto tokens = getTokens(s);
        if(tokens.empty()) return 0;
        if(tokens.size() == 1){
```

```
            if((tokens[0][0]>='0' && tokens[0][0]<='9') || tok
ens[0][0]=='-') return stoi(tokens[0]);
            else return eval[tokens[0]];
        }
        if(tokens[0] == "add"){
            assert(tokens.size() == 3);
            return evalu(tokens[1], eval) + evalu(tokens[2], e
val);
        }
        if(tokens[0]=="mult"){
            assert(tokens.size() == 3);
            return evalu(tokens[1], eval) * evalu(tokens[2], e
val);
        }
        assert(tokens[0] == "let" && tokens.size()>3);
        for(int i=1; i+1<tokens.size(); i+=2) {
            eval[tokens[i]] = evalu(tokens[i+1], eval);
        }
        return evalu(tokens.back(), eval);
    }
public:
    int evaluate(string s){
        return evalu(s, usi());
    }
};
```

441

1. Short is Beauty 系列：

```
class Solution {
public:
```

```
    int arrangeCoins(int n) {
        return int((-1. + sqrt(1. + 8.*double(n)))/2.);
    }
};
```

149

1. 用`pair<int, int>`来hash lines:

```
class Solution {
    typedef vector<bool> vb;
    typedef pair<int, int> ii;
    int gcd(int x, int y){
        if(x < y) swap(x, y);
        if(!y) return x;
        return gcd(y, x%y);
    }
    ii getLineProp(ii p0, ii p1){
        assert(p0 != p1);
        int x = p1.first - p0.first, y = p1.second - p0.secon
d;
        if(!x) return ii(0, 1);
        if(!y) return ii(1, 0);
        int m = gcd(abs(x), abs(y));
        x/=m;
        y/=m;
        if(x < 0){
            x *= -1;
            y *= -1;
        }
        return ii(x, y);
    }
public:
```

```
    int maxPoints(vector<Point>& points) {
        int n = points.size();
        vector<set<ii>> L(n);
        map<ii, int> cnt;
        for(auto p: points) cnt[ii(p.x, p.y)]++;
        if(cnt.empty()) return 0;
        if(cnt.size() == 1) return cnt.begin()->second;
        vector<vb> LN(n, vb(n, false));
        int i=0, j=0, ans = 0;
        for(auto it=cnt.begin(); it!=cnt.end(); ++it){
            int current = it->second;
            map<ii, int> CP;
            auto it1 = it;
            ++it1;
            j = i+1;
            for(j=i+1; it1!=cnt.end(); ++it1){
                ii line = getLineProp(it->first, it1->first);
                if(!L[i].count(line)){
                    CP[line] += it1->second;
                    L[j].insert(line);
                }
                ++j;
            }
            for(auto p: CP) ans = max(ans, current + p.secon
d);
            ++i;
        }
        return ans;
    }
};
```

238

1. 讨论下单零多零的情况即可：

```
class Solution {
public:
    vector<int> productExceptSelf(vector<int>& nums) {
        int cnt = 0, prod=1, idx=-1, n = nums.size();
        vector<int> ans(n, 0);
        for(int i=0; i<n; ++i){
            if(nums[i]) prod *= nums[i];
            else {
                idx = i;
                ++cnt;
            }
        }
        if(cnt > 1) return ans;
        if(cnt == 1){
            ans[idx] = prod;
            return ans;
        }
        for(int i=0; i<n; ++i) ans[i] = prod/nums[i];
        return ans;
    }
};
```

2. left to right, then right to left

```
class Solution:
    def productExceptSelf(self, nums):
        n = len(nums)
        ans = [1] * n
        for i in range(1, n):
            ans[i] = ans[i - 1] * nums[i - 1]
```

```
        prodFromRight = nums[n-1]

        for i in range(n-2, -1, -1):
            ans[i] *= prodFromRight
            prodFromRight *= nums[i]
        return ans
```

828

1. Maintain 一个 sum 就行了 (不难，但无缘无故错了好几次):

```
class Solution {
    const int M = 1000000007;
    typedef pair<int, int> ii;
    #define CI(c) int((c)-'A')
public:
    int uniqueLetterString(string S) {
        long sum = 0L, ans = 0L, n = S.size();
        vector<ii> pos(26, ii(-1, -1));
        for(int i=0; i<n; ++i){
            if(pos[CI(S[i])] == ii(-1, -1)){
                sum += i + 1;
                pos[CI(S[i])] = ii(i, -1);
            }
            else{
                sum += M + i - 2*pos[CI(S[i])].first + pos[CI
(S[i])].second;
                pos[CI(S[i])] = ii(i, pos[CI(S[i])].first);
            }
            sum %= M;
            ans = (ans + sum) % M;
        }
```

```
        return ans;
    }
};
```

814

1. Easy DFS:

```
class Solution {
    int dfs(TreeNode *root){
        if(!root) return 0;
        int l = dfs(root->left), r = dfs(root->right);
        if(root->left && !l){
            delete root->left;
            root->left = NULL;
        }
        if(root->right && !r){
            delete root->right;
            root->right = NULL;
        }
        return root->val + r + l;
    }
public:
    TreeNode* pruneTree(TreeNode* root) {
        if(!root) return NULL;
        int res = dfs(root);
        if(!res) {
            delete root;
            return NULL;
        }
        return root;
    }
```

```
};
```

60

1. 非常Naive的一道题：

```
class Solution {
public:
    string getPermutation(int n, int k) {
        --k;
        string ans, pool;
        int prod = 1;
        for(int i=1; i<=n; ++i){
            pool.push_back(char(i + '0'));
            if(i<n) prod *= i;
        }
        for(int i=n-1; i>=0; --i){
            int j = k/prod;
            k = k%prod;
            if(i) prod /= i;
            ans.push_back(pool[j]);
            pool = pool.substr(0, j) + pool.substr(j+1);
        }
        return ans;
    }
};
```