August 29, 2018 题目：560,756,401,583,163,290,504,701,427,426,36,473,704,663,609

560

1. One Pass:

```
class Solution {
public:
    int subarraySum(vector<int>& nums, int k) {
        unordered_map<int, int> cnt;
        cnt[0] = 1;
        int sum = 0, ans = 0;
        for(int n: nums){
            sum += n;
            if(cnt.count(sum-k)) ans += cnt[sum-k];
            ++cnt[sum];
        }
        return ans;
    }
};
```

2. same python version

```
class Solution:
    def subarraySum(self, nums, k):
        dic = { 0:1 }
        currSum = 0
        ans = 0

        for n in nums:
            currSum += n
            if dic.get(currSum - k, 0):
                ans += dic[currSum - k]
            dic[currSum] = dic.get(currSum, 0) + 1

        return ans
```

756

1. DFS + MEMO:

```
class Solution {
    unordered_map<string, string> next;
    unordered_map<string, bool> dp;
    bool dfs(string s){
        int n = s.size(), prod = 1;
        if(n <= 1) return true;
        if(dp.count(s)) return dp[s];
        vector<string> pool;
        for(int i=0; i<n-1; ++i){
            if(!next.count(s.substr(i, 2))) return dp[s] = fal
se;
            string tmp = next[s.substr(i, 2)];
            pool.push_back(tmp);
            prod *= (int)tmp.size();
        }
        for(int k=0; k<prod; ++k){
            int m = k;
            string cur;
            for(int i=0; i<n-1; ++i){
                int l = pool[i].size();
                cur.push_back(pool[i][m%l]);
                m /= l;
            }
            if(dfs(cur)) return dp[s] = true;
        }
        return dp[s] = false;
    }
public:
    bool pyramidTransition(string bottom, vector<string>& allo
wed) {
        for(string s: allowed) next[s.substr(0, 2)].push_back
(s[2]);
        return dfs(bottom);
    }
};
```

401

1. 做过，Brute Force：

```
class Solution {
public:
    vector<string> readBinaryWatch(int num) {
        vector<string> ans;
        for(int k=0; k<64 * 16; ++k){
            int n = k, cnt = 0;
            while(n){
                ++cnt;
                n &= (n-1);
            }
            if(cnt == num && k%64 < 60 && k/64 < 12){
                ans.push_back(to_string(k/64) + ":" + (k%64<1
0?"0"+to_string(k%64):to_string(k%64)));
            }
        }
        return ans;
    }
};
```
2. 同上
```
class Solution(object):
    def readBinaryWatch(self, num):
        pairs = []
        for h in range(12):
            for m in range(60):
                if (bin(h) + bin(m)).count('1') == num:
                    pairs.append("%d:%02d" % (h, m))
        return pairs
```

583

1. 做过，标准DP：
```
class Solution {
public:
    int minDistance(string word1, string word2) {
        int n = word1.size(), m = word2.size();
        vector<int> dp(m+1, 0);
        for(int j=1; j<=m; ++j) dp[j] = j;
        for(int i=1; i<=n; ++i) {
            vector<int> tmp(dp);
            dp[0] = i;
            for(int j=1; j<=m; ++j){
                if(word1[i-1] == word2[j-1]) dp[j] = tmp[j-1];
                else dp[j] = 1 + min(tmp[j], dp[j-1]);
            }
        }
        return dp[m];
    }
};
```

2. memo DP

```
class Solution(object):
    def minDistance(self, word1, word2):
        memo = {}

        def dp(i, j):
            if (i, j) not in memo:
                if i == len(word1) or j == len(word2):
                    ans = len(word1) - i + len(word2) - j
                elif word1[i] == word2[j]:
                    ans = dp(i+1, j+1)
                else:
                    ans = 1 + min(dp(i+1, j), dp(i, j+1))
                memo[i, j] = ans

            return memo[i, j]

        return dp(0, 0)
```

163

1. LeetCode就喜欢出些无聊的overflow cases：

```
class Solution {
public:
    vector<string> findMissingRanges(vector<int>& nums, int lo
wer, int upper) {
        vector<string> ans;
        long p = (long)lower-1;
        for(int k: nums){
            if(k>p+1){
                if(k==p+2) ans.push_back(to_string(p+1));
                else ans.push_back(to_string(p+1) + "->" + to_
string(k-1));
            }
            p = k;
        }
        if(long(upper) > p){
            if(long(upper) == p+1) ans.push_back(to_string(upp
er));
            else ans.push_back(to_string(p+1) + "->" + to_stri
ng(upper));
        }
        return ans;
    }
};
```

2. python

```
class Solution:
    def findMissingRanges(self, nums, lower, upper):
        ans = []
        nums = [lower-1] + nums + [upper+1]
        for i in range(1, len(nums)):
            if nums[i-1] + 2 == nums[i]:
                ans.append(str(nums[i] - 1))
            elif nums[i-1] + 2 < nums[i]:
                ans.append(str(nums[i-1]+1) + "->" + str(nums
[i]-1))
        return ans
```

290

1. 双向map：

```
class Solution(object):
    def wordPattern(self, pattern, s):
        tokens = s.split(' ')
        if len(pattern) != len(tokens):
            return False
        f, r = {}, {}
        for c, p in zip(pattern, tokens):
            if (c in f) or (p in r):
                if (c not in f) or (p not in r) or (f[c] != p)
or (r[p] != c):
                    return False
            else:
                f[c] = p
                r[p] = c
        return True
```

504

1. Straightforward:

```
class Solution {
public:
    string convertToBase7(int n) {
        if(!n) return "0";
        int sign = (n>=0?1: -1);
        n *= sign;
        string ans;
        while(n){
            ans.push_back(char(n%7 + '0'));
            n /= 7;
        }
        if(sign<0) ans.push_back('-');
        reverse(ans.begin(), ans.end());
        return ans;
    }
};
```

2. recursion

```
class Solution(object):
    def convertToBase7(self, num):
        if num < 0:
            return '-' + self.convertToBase7(-num)
        elif num < 7:
            return str(num)
        else:
            return self.convertToBase7(num // 7) + str(num %
7)
```

701

1. Recursion

```
class Solution {
public:
    TreeNode* insertIntoBST(TreeNode* root, int val) {
        if(!root) return new TreeNode(val);
        if(root->val > val) root->left = insertIntoBST(root->l
eft, val);
        else if(root->val < val) root->right = insertIntoBST(r
oot->right, val);
        return root;
    }
};
```

427

1. Index Tree 2D:

```
class Solution {
    Node *dfs(vector<vector<int>>&G, int u, int d, int l, int
r){
        int sum = G[d][r] - G[d][l] - G[u][r] + G[u][l], tar =
(d-u) * (d-u);
        if(sum == tar) return new Node(true, true, NULL, NULL,
NULL, NULL);
        if(!sum) return new Node(false, true, NULL, NULL, NUL
L, NULL);
        return new Node(false, false, dfs(G, u, (u+d)/2, l, (l
+r)/2),
                       dfs(G, u, (u+d)/2, (l+r)/2, r),
                       dfs(G, (u+d)/2, d, l, (l+r)/2),
                        dfs(G, (u+d)/2, d, (l+r)/2, r));
    }
public:
    Node* construct(vector<vector<int>>& grid) {
        int n = grid.size();
        vector<vector<int>> G(n+1, vector<int>(n+1, 0));
        for(int i=1; i<=n; ++i) for(int j=1; j<=n; ++j) {
            G[i][j] = grid[i-1][j-1] + G[i-1][j] + G[i][j-1] -
G[i-1][j-1];
        }
        return dfs(G, 0, n, 0, n);
    }
};
```

426

1. Stuped dfs:

```
class Solution {
    void build(Node *root, Node* &l, Node* &r){
        l = r = root;
        Node *t;
        if(root->left){
            build(root->left, l, t);
            root->left = t;
            t->right = root;
        }
        if(root->right){
            build(root->right, t, r);
            root->right = t;
            t->left = root;
        }
    }
public:
    Node* treeToDoublyList(Node* root) {
        if(!root) return NULL;
        Node *l, *r;
        build(root, l, r);
        l->left = r;
        r->right = l;
        return l;
    }
};
```
2. Stack, 也不怎么快：
```
class Solution {
public:
    Node* treeToDoublyList(Node* root) {
        Node *left=NULL, *lead=NULL;
        stack<Node *> S;
        while(root || !S.empty()){
            while(root){
                S.push(root);
                root = root->left;
            }
            if(!S.empty()){
                if(lead){
                    lead->right = S.top();
                    root = S.top()->right;
                    S.top()->left = lead;
                    lead = S.top();
                    S.pop();
                }
                else{
                    left = lead = S.top();
                    root = S.top()->right;
                    S.pop();
                }
            }
        }
        if(left){
            left->left = lead;
            lead->right = left;
        }
        return left;
    }
};
```

36

1. 有段时间天天玩数独，所以发现这题好简单：

```
class Solution {
    typedef vector<bool> vb;
public:
    bool isValidSudoku(vector<vector<char>>& board) {
        assert(board.size() == 9 && board[0].size() == 9);
        vector<vb> dp(27, vb(9, false));
        for(int i=0; i<9; ++i) for(int j=0; j<9; ++j) if(board[i][j] != '.'){
            int x = i, y = j + 9, z = (i/3)*3 + (j/3) + 18, v = int(board[i][j] - '1');
            if(dp[x][v] || dp[y][v] || dp[z][v]) return false;
            dp[x][v] = dp[y][v] = dp[z][v] = true;
        }
        return true;
    }
};
```

473

1. 迭代背包：

```
class Solution {
    void dfs(vector<int> &res, int i, int stat, int tar, vector<int>&A){
        if(!tar || i==A.size()){
            if(!tar) res.push_back(stat);
            return;
        }
        for(int j=i; j<A.size() && A[j] <= tar; ++j) if(!(1&(stat>>j))){
            dfs(res, j, stat | 1<<j, tar-A[j], A);
        }
    }
    bool dfs2(int stat, int tar, int k, vector<int>&A){
        if(!k) return true;
        vector<int> pool;
        dfs(pool, 0, stat, tar, A);
        for(int s: pool) if(dfs2(s, tar, k-1, A)) return true;
        return false;
    }
public:
    bool makesquare(vector<int>& nums) {
        sort(nums.begin(), nums.end());
        int sum = 0;
        for(int k: nums) sum+=k;
        if(!sum || sum % 4) return false;
        return dfs2(0, sum/4, 3, nums);
    }
};
```

704

1. 。。。

```
class Solution {
public:
    int search(vector<int>& nums, int target) {
        int l = -1, r = nums.size();
        while(l < r-1){
            int c = (l+r)/2;
            if(nums[c] <= target) l = c;
            else r = c;
        }
        return (l==-1 || nums[l]!=target? -1: l);
    }
};
```

663

1. 需要用`unordered_map`做记忆：

```
class Solution {
    unordered_map<int, int> cnt;
    int dfs(TreeNode *root){
        if(!root) return 0;
        int ans = root->val + dfs(root->left) + dfs(root->righ
t);
        ++cnt[ans];
        return ans;
    }
public:
    bool checkEqualTree(TreeNode* root) {
        int res = dfs(root);
        if(res%2) return false;
        for(auto p: cnt) cout<<p.first<<' '<<p.second<<endl;
        --cnt[res];
        return cnt[res/2] > 0;
    }
};
```

609

1. Python 处理string是容易：

```
class Solution(object):
    def findDuplicate(self, paths):
        content2path = {}
        for path in paths:
            tokens = path.split(' ')
            for f in tokens[1:]:
                fn, cont = f.split('(')[0], f.split('(')[-1]
                if cont not in content2path:
                    content2path[cont] = []
                content2path[cont].append(tokens[0] + "/" + f
n)
```

```
        return [flist for flist in content2path.values() if len(flist) > 1]
```