August 16, 2018 题目：297,531,281,788,816,176,610,127,437,650,649,653,616,106

297

1. 自己encode连corner case都不用写，不过好想很慢：

```
class Codec {
public:
    string serialize(TreeNode* root) {
        if(!root) return "#";
        return to_string(root->val) + "(" + serialize(root->left) + ")(" + serialize(root->right) + ")";
    }
    TreeNode* deserialize(string data) {
        if(data == "#") return NULL;
        int val = stoi(data);
        TreeNode *root = new TreeNode(val);
        auto k = (int)data.find('(')+1;
        int j = int(k), cnt = 1;
        while(cnt){
            if(data[j]=='(') ++cnt;
            if(data[j]==')') --cnt;
            ++j;
        }
        root->left = deserialize(data.substr(k, j-1-k));
        k = ++j;
        cnt = 1;
        while(cnt){
            if(data[j]=='(') ++cnt;
```

```
            if(data[j]==')') --cnt;
            ++j;
        }
        root->right = deserialize(data.substr(k, j-1-k));
        return root;
    }
};
```

531

1.

```
class Solution {
    public int findLonelyPixel(char[][] picture) {
        int[] row = new int[picture.length];
        int[] col = new int[picture[0].length];
        for (int i = 0; i < picture.length; i++) {
            for (int j = 0; j < picture[0].length; j++) {
                if (picture[i][j] == 'B') {
                    row[i]++;
                    col[j]++;
                }
            }
        }
        int res = 0;
        for (int i = 0; i < picture.length; i++) {
            for (int j = 0; j < picture[0].length; j++) {
                if (picture[i][j] == 'B' && row[i] == 1 && col
[j] == 1) {
                    res++;
                }
            }
        }
```

```
        return res;
    }
}
```

2. 同上：

```
class Solution {
public:
    int findLonelyPixel(vector<vector<char>>& P) {
        if(P.empty() || P[0].empty()) return 0;
        int n = P.size(), m = P[0].size(), ans=0;
        vector<int> row(n, 0);
        vector<vector<int>> col(m);
        for(int i=0; i<n; ++i) for(int j=0; j<m; ++j) if(P[i]
[j]=='B') {
            ++row[i];
            col[j].push_back(i);
        }
        for(int j=0; j<m; ++j) if(col[j].size()==1 && row[col
[j][0]]==1) ++ans;
        return ans;
    }
};
```

281

1. Pure Easy:

```
class ZigzagIterator {
    queue<int> Q;
public:
    ZigzagIterator(vector<int>& v1, vector<int>& v2) {
        for(int i=0; i<max(v1.size(), v2.size()); ++i) {
            if(i<v1.size()) Q.push(v1[i]);
            if(i<v2.size()) Q.push(v2[i]);
```

```
        }
    }
    int next() {
        int ans = Q.front();
        Q.pop();
        return ans;
    }
    bool hasNext() { return !Q.empty(); }
};
```

788

1.

```
class Solution {
    public int rotatedDigits(int N) {
        Map<Integer, Integer> map = new HashMap<>();
        map.put(0, 0);
        map.put(1, 1);
        map.put(2, 5);
        map.put(5, 2);
        map.put(6, 9);
        map.put(8, 8);
        map.put(9, 6);

        int res = 0;
        for (int i = 1; i <= N; i++) {
            String s = String.valueOf(i);
            StringBuilder sb = new StringBuilder();
            boolean flag = true;
            for (char c : s.toCharArray()) {
                int n = c - '0';
```

```
                if (map.containsKey(n)) {
                    sb.append(map.get(n));
                } else {
                    flag = false;
                    break;
                }
            }
            if (flag && Integer.parseInt(sb.toString()) != i)
res++;
        }
        return res;
    }
}
```

2. dp很新颖：

```
class Solution {
    vector<int> ref = {0, 0, 1, -1, -1, 1, 1, -1, 0, 1};
public:
    int rotatedDigits(int N) {
        int ans = 0;
        for(int i=1; i<=min(9, N); ++i) ans += (ref[i]==1);
        for(int i=10; i<=N; ++i){
            int tmp = 0;
            if(ref[i%10]==-1 || ref[i/10]==-1) tmp=-1;
            tmp = ref[i%10] | ref[i/10];
            ans += (tmp==1);
            ref.push_back(tmp);
        }
        return ans;
    }
};
```

- [ ] 可以用排列组合试试 @Zebo L

816

1. 完全brute force：

```
class Solution {
    int T[6][4] = {
        {-1, 1, 2, -1},
        {3, -1, -1, 6},
        {3, 2, 2, 6},
        {-1, 5, 4, -1},
        {-1, 5, 4, 6},
        {-1, 5, 4, -1}
    };
    int getIdx(char c){
        if(c=='.') return 0;
        if(c=='0') return 1;
        if(c>='1' && c<='9') return 2;
        return 3;
    }
    bool valid(string s){
        s += '#';
        int stat = 0;
        for(int c: s) {
            stat = T[stat][getIdx(c)];
            if(stat==-1) return false;
        }
        return true;
    }
public:
    vector<string> ambiguousCoordinates(string S) {
        vector<string> ans;
```

```
        S = S.substr(1, S.size()-2);
        for(int i=1; i<S.size(); ++i){
            string s1 = S.substr(0, i), s2 = S.substr(i);
            for(int j=1; j<=s1.size(); ++j) {
                string t1 = s1.substr(0, j) + (j<s1.size
()?".":"") + s1.substr(j);
                if(!valid(t1)) continue;
                for(int k=1; k<=s2.size(); ++k){
                    string t2 = s2.substr(0, k) + (k<s2.size
()?".":"") + s2.substr(k);
                    if(valid(t2)) ans.push_back("(" + t1 + ",
" + t2 + ")");
                }
            }
        }
        return ans;
    }
};
```

2. From LeetCode discussion:

```
class Solution {
    vector<string> addDot(string s){
        if(s.size() == 1) return {s};
        vector<string> res;
        if(s[0] != '0') res.push_back(s);
        for(int i = 1; i < s.size(); ++i){
            string l = s.substr(0, i); // Seperate the string using "."
            string r = s.substr(i);
            if(r.back() == '0') continue; // X.X0
            if(l.size() > 1 && l[0] == '0') continue; // 0XX
            res.push_back(l + "." + r);
        }
        return res;
    }
public:
    vector<string> ambiguousCoordinates(string S) {
        vector<string> res;
        S = S.substr(1, S.size() - 2); // Remove "()"
        for(int i = 1; i < S.size(); ++i){
            string left = S.substr(0, i); // Seperate the string using ","
            string right = S.substr(i);
            for(string& a : addDot(left)){
                for(string& b : addDot(right)){
                    res.push_back("(" + a + ", " + b + ")"); // There is a space after ","
                }
            }
        }
        return res;
    }
};
```

3. 做了半天超时了。。放弃了

176

SQL

610

又是SQL

127

1. BFS 会超时。。改用Bi-BFS ----这道题貌似被改过了test case和input，原来对的答案现在过不了了

```
class Solution {
    public int ladderLength(String beginWord, String endWord,
List<String> words) {
        Set<String> wordList = new HashSet<>(words);
        if (!wordList.contains(endWord)) return 0;
```

```
        Set<String> begin = new HashSet<>();
        Set<String> end = new HashSet<>();
        begin.add(beginWord);
        end.add(endWord);
        int level = 1;
        while (!begin.isEmpty() && !end.isEmpty()) {
            if (begin.size() > end.size()) {
                Set<String> tmp = begin;
                begin = end;
                end = tmp;
            }
            Set<String> set = new HashSet<>();
            for (String cur : begin) {
                char[] ch = cur.toCharArray();
                for (int i = 0; i < ch.length; i++) {
                    char tmp = ch[i];
                    for (char j = 'a'; j <= 'z'; j++) {
                        if (tmp != j) {
                            ch[i] = j;
                            String s = new String(ch);
                            if (end.contains(s)) return level
+ 1;
                            if (wordList.contains(s)) {
                                wordList.remove(s);
                                set.add(s);
                            }
                        }
                    }
                    ch[i] = tmp;
                }
```

```
            }
            if (set.size() == 0) return 0;
            begin = set;
            level++;
        }
        return 0;
    }
}
```

2. BFS 过了，但DFS一直报错,见鬼了

```
class Solution {
    typedef pair<string, int> si;
public:
    int ladderLength(string beginWord, string endWord, vector<
string>& wordList) {
        queue<si> Q;
        unordered_set<string> S{beginWord}, P(wordList.begin
(), wordList.end());
        Q.push(si(beginWord, 1));
        while(!Q.empty()){
            string s = Q.front().first;
            int k = Q.front().second;
            Q.pop();
            if(s == endWord) return k;
            for(int i=0; i<s.size(); ++i){
                char x = s[i];
                for(char c='a';c<='z';++c){
                    s[i] = c;
                    if(P.count(s) && !S.count(s)){
                        S.insert(s);
                        Q.push(si(s, k+1));
                    }
```

```
                }
                s[i] = x;
            }
        }
        return 0;
    }
};
```

3. 这个dfs有什么问题么？

- [ ] Figure out why @Zebo L

```
class Solution {
    unordered_map<string, int> dp;
    unordered_set<string> Pool;
    const int inf = 1E8;
    int dfs(string &s){
        if(dp.count(s)) return dp[s];
        dp[s] = inf;
        int ans = inf;
        for(int i=0; i<s.size(); ++i){
            string s1 = s;
            for(char c='a'; c<='z'; ++c) if(c!=s[i]) {
                s1[i] = c;
                if(Pool.count(s1)) ans = min(ans, 1 + dfs(s
1));
            }
        }
        return dp[s]=ans;
    }
public:
    int ladderLength(string beginWord, string endWord, vector<
string>& wordList) {
        for(string &s: wordList) Pool.insert(s);
```

```
        dp[endWord] = 1;
        return (dfs(beginWord)>=inf?0:dfs(beginWord));
    }
};
```

437

1. dfs :

```
class Solution {
    int res;
    void dfs(TreeNode *root, stack<int>&S, unordered_map<int, int>& cnt, int& target){
        if(!root) return;
        int sum = S.top() + root->val;
        if(cnt.count(sum - target)) res += cnt[sum - target];
        cnt[sum]++;
        S.push(sum);
        dfs(root->left, S, cnt, target);
        dfs(root->right, S, cnt, target);
        cnt[sum]--;
        if(!cnt[sum]) cnt.erase(sum);
        S.pop();
    }
public:
    int pathSum(TreeNode* root, int sum) {
```

```
        res = 0;
        stack<int> S;
        unordered_map<int, int> cnt;
        cnt[0]=1;
        S.push(0);
        dfs(root, S, cnt, sum);
        return res;
    }
};
```

650

1. BFS:

```
class Solution {
    typedef vector<int> vi;
    typedef pair<int, int> ii;
public:
    int minSteps(int n) {
        queue<vi> Q;
        set<ii> S;
        Q.push(vi{1, 0, 0});
        S.insert(ii(1, 0));
        while(!Q.empty()){
            vi tmp = Q.front();
            int x = tmp[0], y = tmp[1], z = tmp[2];
            if(x == n) return z;
            Q.pop();
            if(n%x==0 && !S.count(ii(x, x))){
                Q.push(vi{x, x, z+1});
                S.insert(ii(x, x));
            }
```

```
            if(x+y<=n && !S.count(ii(x+y, y))){
                Q.push(vi{x+y, y, z+1});
                S.insert(ii(x+y, y));
            }
        }
        return -1;
    }
};
```

- [ ] 这个： https://leetcode.com/problems/2-keys-keyboard/discuss/154409/C++-DP-0-ms @Zebo L

649

1. 以前用两个Queue做过，感觉太low，于是换了种方法：

```
class Solution {
public:
    string predictPartyVictory(string senate) {
        int R=0, D=0;
        for(char c: senate){
            if(c == 'R') ++R;
            else ++D;
        }
        for(int i=0, cnt=0; i<senate.size() && R && D; ++i){
            if(senate[i]=='R'){
                if(cnt>=0) senate += 'R';
                else --R;
                ++cnt;
            }
            else{
                if(cnt<=0) senate += 'D';
                else --D;
                --cnt;
```

```
            }
        }
        return (R?"Radiant":"Dire");
    }
};
```

653

1. 做了半天才发现只要取两个数。。。还有一种办法是inorder以后做binary search

```
class Solution {
    public boolean findTarget(TreeNode root, int k) {
        Set<Integer> set = new HashSet<>();
        return helper(root, set, k);
    }

    public boolean helper(TreeNode node, Set<Integer> set, int
k) {
        if (node == null) {
            return false;
        }
        if (set.contains(k - node.val)) return true;
        set.add(node.val);
        return helper(node.left, set, k) || helper(node.right,
set, k);

    }
}
```

2. 跟two pointer 思路一样：

```
class Solution {
public:
    bool findTarget(TreeNode* root, int k) {
        stack<TreeNode *> L, R;
```

```
        for(auto p=root; p; p=p->left) L.push(p);
        for(auto p=root; p; p=p->right) R.push(p);
        while(!L.empty() && !R.empty() && L.top()->val<R.top()
->val){
            if(L.top()->val + R.top()->val == k) return true;
            else if(L.top()->val + R.top()->val < k){
                root = L.top()->right;
                L.pop();
                for(; root; root=root->left) L.push(root);
            }
            else{
                root = R.top()->left;
                R.pop();
                for(; root; root=root->right) R.push(root);
            }
        }
        return false;
    }
};
```

616

1. 跟Insert Intervals 一模一样：(这个题直接fill 更快些)

```
class Solution {
    void insertIntervals(map<int, int> &I, int start, int end)
{
        auto it = --I.upper_bound(start);
        if(it->second < start) ++it;
        start = min(start, it->first);
        while(it!=I.end() && it->first <= end){
            end = max(end, it->second);
            it = I.erase(it);
```

```
        }
        I[start] = end;
    }
public:
    string addBoldTag(string s, vector<string>& dict) {
        map<int, int> range;
        range[-20] = -20;
        range[10000] = 10000;
        for(string t: dict){
            auto i = s.find(t);
            int start = i, end = start + t.size();
            while(i!=string::npos && i<s.size()){
                i = s.find(t, i+1);
                if(i==string::npos || i > end){
                    insertIntervals(range, start, end);
                    start = i;
                    end = start + t.size();
                }
                else{
                    end = max(end, int(i + t.size()));
                }
            }
        }
        int i = 0;
        string ans;
        for(auto it=++range.begin(); it->second < 10000; ++it)
{
            ans += s.substr(i, it->first-i);
            ans += "<b>" + s.substr(it->first, it->second-it->
first) + "</b>";
```

```
            i = it->second;
        }
        ans += s.substr(i);
        return ans;
    }
};
```

106

1. recursion

```
class Solution {
    public TreeNode buildTree(int[] inorder, int[] postorder)
{
        int n = inorder.length;

        return helper(inorder, postorder, 0, n - 1, n - 1);

    }

    public TreeNode helper(int[] inorder, int[] postorder, int
i, int j, int k) {
        if (i > j || k < 0) return null;
        int cur = j;
        TreeNode root = new TreeNode(postorder[k]);
        for (int index = j; index >= i; index--) {
            if (inorder[index] == postorder[k]) {
                cur = index;
                break;
            }
        }
        root.left = helper(inorder, postorder, i, cur - 1, k -
1 - (j - cur));
```

```
        root.right = helper(inorder, postorder, cur + 1, j, k
- 1);
        return root;
    }
}
```

2. 同上：

```
class Solution {
    TreeNode* dfs(vector<int>& I, vector<int>& P, int l1, int
r1, int l2, int r2){
        if(l1>r1) return NULL;
        assert(r1-l1 == r2-l2);
        auto root = new TreeNode (P[r2]);
        int k = 0;
        while(k<=r1-l1 && I[l1+k]!=P[r2]) ++k;
        root->left = dfs(I, P, l1, l1 + k-1, l2, l2 + k-1);
        root->right = dfs(I, P, l1 + k+1, r1, l2 + k, r2-1);
        return root;
    }
public:
    TreeNode* buildTree(vector<int>& I, vector<int>& P) {
        return dfs(I, P, 0, I.size()-1, 0, I.size()-1);
    }
};
```