August 1, 2018 题目：57,261,409,399,357,117,51,186,250,64,666,487,362

57

1. straightforward left to right scan
2. 同上

```
class Solution {
public:
    vector<Interval> insert(vector<Interval>& intervals, Interval newInterval) {
        vector<Interval> ans;
        int l = newInterval.start, r = newInterval.end, i = 0, n = intervals.size();
        while(i<n && intervals[i].end < l) ans.push_back(intervals[i++]);
        while(i<n && intervals[i].start <= r){
            l = min(l, intervals[i].start);
            r = max(r, intervals[i].end);
            ++i;
        }
        ans.push_back(Interval(l, r));
        while(i<n) ans.push_back(intervals[i++]);
        return ans;
    }
};
```

261

1. Union Find

```
class Solution {
```

```
    int[] parents;
    public int find(int i) {
        while (parents[i] != i) {
            i = parents[i];
        }
        return i;
    }
    public boolean validTree(int n, int[][] edges) {
        parents = new int[n];
        for (int i = 0; i < n; i++) {
            parents[i] = i;
        }
        for (int[] edge : edges) {
            int p1 = find(edge[0]);
            int p2 = find(edge[1]);
            if (p1 != p2) parents[p1] = p2;
            else return false;
        }
        return edges.length == n - 1;
    }
}
```

2. 同上：

```
class Solution {
    vector<int> P;
    int findRoot(int i){
        if(i==P[i]) return i;
        return P[i] = findRoot(P[i]);
    }
public:
    bool validTree(int n, vector<pair<int, int>>& edges) {
```

```
        if(edges.size() != n-1) return false;
        P.resize(n);
        for(int i=0; i<n; ++i) P[i] = i;
        for(auto p: edges){
            if(findRoot(p.first) == findRoot(p.second)) return
false;
            P[findRoot(p.second)] = findRoot(p.first);
        }
        return true;
    }
};
```

409

1.

```
class Solution {
    public int longestPalindrome(String s) {
        Set<Character> set = new HashSet<>();
        int count = 0;
        for (char c : s.toCharArray()) {
            if (set.contains(c)) {
                set.remove(c);
                count++;
            } else {
                set.add(c);
            }
        }

        return set.size() == 0 ? count * 2 : count * 2 + 1;
    }
}
```

1. 数数：

```
class Solution {
public:
    int longestPalindrome(string s) {
        map<char, int> cnt;
        for(char c: s) ++cnt[c];
        int ans = 0, odd = 0;
        for(auto p: cnt){
            ans += p.second/2*2;
            if(p.second%2) odd = 1;
        }
        return ans + odd;
    }
};
```

399

1. 前几天做过，很有意思的题，做法是稍微变形的Union find：

```
class Solution {
    unordered_map<string, double> coef;
    unordered_map<string, string> unit;
    string findUnit(string s){
        if(unit[s] == s) return s;
        string p = unit[s];
        unit[s] = findUnit(p);
        coef[s] *= coef[p];
        return unit[s];
    }
public:
    vector<double> calcEquation(vector<pair<string, string>> e
quations, vector<double>& values, vector<pair<string, string>>
queries) {
        vector<double> ans;
```

```
        for(int i=0; i<equations.size(); ++i){
            string x = equations[i].first, y = equations[i].se
cond;
            double v = values[i];
            if(!unit.count(x) && !unit.count(y)){
                unit[x] = unit[y] = y;
                coef[y] = 1.;
                coef[x] = v;
            }
            else if(!unit.count(x) && unit.count(y)){
                string d = findUnit(y);
                unit[x] = d;
                coef[x] = coef[y] * v;
            }
            else if(unit.count(x) && !unit.count(y)){
                string c = findUnit(x);
                unit[y] = c;
                coef[y] = coef[x] / v;
            }
            else{
                string c = findUnit(x), d = findUnit(y);
                unit[c] = d;
                coef[c] = coef[y] * v / coef[x];
            }
        }
        for(auto p: queries){
            string x = p.first, y = p.second;
            if(!unit.count(x) || !unit.count(y)) ans.push_back
(-1.);
            else if(findUnit(x) != findUnit(y)) ans.push_back
(-1.);
```

```
            else ans.push_back(coef[x] / coef[y]);
        }
        return ans;
    }
};
```

357

1. 前几天也做过，别忘了0就行：

```
class Solution {
public:
    int countNumbersWithUniqueDigits(int n) {
        int ans = 0, cnt = 9;
        for(int i=0; i<n; ++i){
            ans += cnt;
            cnt *= (9-i);
        }
        return ans + 1;
    }
};
```

117

1. BFS

```
/**
 * Definition for binary tree with next pointer.
 * public class TreeLinkNode {
 *     int val;
 *     TreeLinkNode left, right, next;
 *     TreeLinkNode(int x) { val = x; }
 * }
 */
public class Solution {
```

```
    public void connect(TreeLinkNode root) {
        if (root == null) return;
        Queue<TreeLinkNode> q = new LinkedList<>();
        q.offer(root);

        while (!q.isEmpty()) {
            int size = q.size();
            TreeLinkNode node = q.poll();
            for (int i = 0; i < size; i++) {
                if (node.left != null) q.offer(node.left);
                if (node.right != null) q.offer(node.right);
                if (i == size - 1) {
                    node.next = null;
                    break;
                }
                TreeLinkNode next = q.poll();
                node.next = next;
                node = node.next;
            }
        }

    }
}
```

2. constant space 就意味着很慢：

```
class Solution {
    void dfs(TreeLinkNode *l, TreeLinkNode *r){
        if(!l || !r) return;
        l->next = r;
        dfs(l->left, r->right);
        dfs(l->left, r->left);
```

```
        dfs(l->right, r->right);
        dfs(l->right, r->left);
    }
public:
    void connect(TreeLinkNode *root) {
        if(!root) return;
        dfs(root->left, root->right);
        connect(root->left);
        connect(root->right);
    }
};
```

51

1. 经典题:

```
class Solution {
    vector<bool> xy, yx, x;
    void dfs(vector<vector<int>> &ans, vector<int> cur, int j,
int n){
        if(j==n){
            ans.push_back(cur);
            return;
        }
        for(int i=0; i<n; ++i) if(!xy[i + j] && !yx[i+n-j] &&
!x[i]){
            cur.push_back(i);
            xy[i + j] = yx[i+n-j] = x[i] = true;
            dfs(ans, cur, j+1, n);
            cur.pop_back();
            xy[i + j] = yx[i+n-j] = x[i] = false;
        }
    }
```

```
public:
    vector<vector<string>> solveNQueens(int n) {
        vector<vector<int>> res;
        xy = yx = vector<bool>(2*n, false);
        x = vector<bool>(n, false);
        dfs(res, vector<int>(), 0, n);
        vector<vector<string>> ans(res.size(), vector<string>
(n, string(n, '.')));
        for(int i=0; i<ans.size(); ++i) for(int j=0; j<n; ++j)
ans[i][j][res[i][j]] = 'Q';
        return ans;
    }
};
```

186

1. array

```
class Solution {
    public void reverseWords(char[] str) {
        reverse(str, 0, str.length - 1);
        int start = 0;
        for (int i = start + 1; i < str.length; i++) {
            if (str[i] == ' ') {
                reverse(str, start, i - 1);
                start = i + 1;
            }
        }
        if (start != str.length - 1) reverse(str, start, str.l
ength - 1);
    }
    public void reverse(char[] str, int i, int j) {
        while (i < j) {
```

```
            char tmp = str[i];
            str[i] = str[j];
            str[j] = tmp;
            i++;
            j--;
        }
    }
}
```

2. 先 `reverse` 每个词，然后再 `reverse` 整个 `string`:

```
class Solution {
public:
    void reverseWords(vector<char>& str) {
        int i = 0, n = str.size();
        while(i<n) {
            int j = i+1;
            while(j<n && str[j] != ' ') ++j;
            reverse(str.begin() + i, str.begin() + j);
            i = j + 1;
        }
        reverse(str.begin(), str.end());
    }
};
```

250

1. 简单的recursion，但在判定 `if(!unique(root->left) || (root->left && root->val != root->left->val))` 的时候，不能直接返回 `false`，否则下一个 `unique(root→right)` 就不执行了。

```
class Solution {
    int res;
    bool unique(TreeNode *root){
        if(!root) return true;
```

```
        bool ok = true;
        if(!unique(root->left) || (root->left && root->val != root->left->val)) ok = false;
        if(!unique(root->right) || (root->right && root->val != root->right->val)) ok = false;
        if(ok) ++res;
        return ok;
    }
public:
    int countUnivalSubtrees(TreeNode* root) {
        res = 0;
        unique(root);
        return res;
    }
};
```

64

1. Easy dp:

```
class Solution {
public:
    int minPathSum(vector<vector<int>>& grid) {
        if(grid.empty() || grid[0].empty()) return 0;
        vector<int> dp(grid[0]);
        for(int j=1; j<grid[0].size(); ++j) dp[j] += dp[j-1];
        for(int i=1; i<grid.size(); ++i){
            dp[0] += grid[i][0];
            for(int j=1; j<grid[0].size(); ++j) dp[j] = grid[i][j] + min(dp[j], dp[j-1]);
        }
        return dp[grid[0].size()-1];
    }
```

```
};
```

666

1. 计算一下每个节点在 `path sum` 中应该出现的次数即可

```
class Solution {
public:
    int pathSum(vector<int>& nums) {
        int i = 0, ans = 0, n=nums.size();
        sort(nums.begin(), nums.end(), greater<int>());
        vector<int> coef(8, 0);
        for(int depth=4, width=8; depth; --depth, width /= 2){
            while(i<n && nums[i]/100==depth){
                int k = (nums[i]/10)%10-1, v = nums[i]%10;
                if(!coef[k]) coef[k]=1;
                ans += v*coef[k];
                ++i;
            }
            vector<int> tmp(width, 0);
            for(int j=0; j<width; j+=2) tmp[j/2] = coef[j] + c
oef[j+1];
            swap(coef, tmp);
        }
        return ans;
    }
};
```

487

1. 维护第一个零，跟第二个零的位置即可。

```
class Solution {
public:
    int findMaxConsecutiveOnes(vector<int>& nums) {
```

```
        int i = -1, j = -1, k = 0, n = nums.size(), ans = 0;
        while(k<n && nums[k]) ++k;
        if(k >= n) return n;
        while(i<n && k<n){
            i = j+1;
            j = k;
            k = j+1;
            while(k<n && nums[k]) ++k;
            ans = max(ans, k-i);
        }
        return ans;
    }
};
```

362

1. 今天没 `hard` 啊:

```
class HitCounter {
    queue<int> Q;
public:
    HitCounter() {}
    void hit(int timestamp) {
        Q.push(timestamp);
        while(Q.front() <= timestamp-300) Q.pop();
    }
    int getHits(int timestamp) {
        while(!Q.empty() && Q.front() <= timestamp-300) Q.pop
();
        return Q.size();
    }
};
```