August 12, 2018 题目: 549,503,394,159,257,594,462,113,306,410,733

549

1.

```
/**
 * Definition for a binary tree node.
 * public class TreeNode {
 *     int val;
 *     TreeNode left;
 *     TreeNode right;
 *     TreeNode(int x) { val = x; }
 * }
 */
class Solution {
    int res = 0;
    public int longestConsecutive(TreeNode root) {
        helper(root);
        return res;
    }
    public int[] helper(TreeNode node) {
        int[] cur = new int[2];
        if (node == null) return cur;
        int[] left = helper(node.left);
        int[] right = helper(node.right);

        if (node.left != null) {
            if (node.left.val != node.val - 1) left[0] = 0;
```

```
            if (node.left.val != node.val + 1) left[1] = 0;
        }
        if (node.right != null) {
            if (node.right.val != node.val - 1) right[0] = 0;
            if (node.right.val != node.val + 1) right[1] = 0;
        }
        cur[0] = Math.max(right[0], left[0]) + 1;
        cur[1] = Math.max(right[1], left[1]) + 1;
        res = Math.max(res, cur[0] + cur[1] - 1);
        return cur;
    }
}
```

2. 常规 dfs :

```
class Solution {
    int res;
    void dfs(TreeNode *root, int &i, int &d){
        if(!root){
            i = d = 0;
            return;
        }
        int i1, d1, i2, d2;
        dfs(root->left, i1, d1);
        dfs(root->right, i2, d2);
        i = d = 1;
        if(root->left){
            if(root->val + 1 == root->left->val) i = max(i, i1
+ 1);
            if(root->val - 1 == root->left->val) d = max(d, d1
+ 1);
        }
        if(root->right){
```

```
            if(root->val + 1 == root->right->val) i = max(i, i
2 + 1);
            if(root->val - 1 == root->right->val) d = max(d, d
2 + 1);
        }
        res = max(res, i + d - 1);
    }
public:
    int longestConsecutive(TreeNode* root) {
        res = 0;
        int i, d;
        dfs(root, i, d);
        return res;
    }
};
```

503

1.

```
class Solution {
    public int[] nextGreaterElements(int[] nums) {
        int[] res = new int[nums.length];
        Arrays.fill(res, -1);
        for (int i = 0; i < res.length; i++) {
            for (int k = 1; k <= res.length; k++) {
                if (nums[(i + k) % res.length] > nums[i]) {
                    res[i] = nums[(i + k) % res.length];
                    //System.out.println(res[i]);
                    break;
                }
            }
        }
```

```
        return res;
    }
}
```

2. Two pass:

```
class Solution {
public:
    vector<int> nextGreaterElements(vector<int>& nums) {
        int n = nums.size();
        vector<int> ans(n, -1);
        nums.resize(2 * n);
        copy(nums.begin(), nums.begin() + n, nums.begin() +
n);
        stack<int> S;
        for(int i=2*n-1; i>=0; --i){
            while(!S.empty() && nums[S.top()] <= nums[i]) S.po
p();
            if(i<n && !S.empty()) ans[i] = nums[S.top()];
            S.push(i);
        }
        return ans;
    }
};
```

394

1. Stack 轻松搞定

```
class Solution {
    const string res = "0123456789[]";
public:
    string decodeString(string s) {
        string ans;
        stack<string> S;
```

```
        stack<int> I;
        int i = 0;
        while(i<s.size()){
            if(isalpha(s[i])){
                auto j = s.find_first_of(res, i+1);
                if(j == string::npos) j = s.size();
                ans += s.substr(i, j-i);
                i = j;
            }
            else if(isdigit(s[i])){
                int cnt = stoi(s.substr(i));
                i += to_string(cnt).size() + 1;
                S.push(ans);
                I.push(cnt);
                ans = "";
            }
            else{
                string s = S.top();
                S.pop();
                for(int i=0; i<I.top(); ++i) s+=ans;
                I.pop();
                swap(s, ans);
                ++i;
            }
        }
        return ans;
    }
};
```

1. 标记连个字母出现的最后位置，比 sliding window 快

```
class Solution {
public:
    int lengthOfLongestSubstringTwoDistinct(string s) {
        unordered_map<char, int> cnt;
        int idx1 = -1, idx2 = -1, ans = 0, start = 0;
        for(int i=0; i<s.size(); ++i){
            if(idx2<0 || s[i]==s[idx2]) idx2 = i;
            else if(idx1<0 || s[i]==s[idx1]){
                idx1 = idx2;
                idx2 = i;
            }
            else {
                start = idx1 + 1;
                idx1 = idx2;
                idx2 = i;
            }
            ans = max(ans, i-start+1);
        }
        return ans;
    }
};
```

257

1. tree traversal

```
class Solution {
    public List<String> binaryTreePaths(TreeNode root) {
        List<String> list = new ArrayList<>();
        if (root == null) return list;
        helper(root, list, "");
        return list;
```

```
    }
    public void helper(TreeNode root, List<String> list, Strin
g cur) {
        if (root.left == null && root.right == null) {
            cur = cur + root.val;
            list.add(cur);
            return;
        }
        if (root.left != null) helper(root.left, list, cur + r
oot.val + "->");
        if (root.right != null) helper(root.right, list, cur +
root.val + "->");
    }
}
```

2. 典型dfs：

```
class Solution {
    inline bool isLeaf(TreeNode *root){ return root&&!root->le
ft&&!root->right; }
    void dfs(vector<string> &ans, string cur, TreeNode* root){
        if(!root) return;
        if(isLeaf(root)){
            ans.push_back(cur + to_string(root->val));
            return;
        }
        cur += to_string(root->val) + "->";
        dfs(ans, cur, root->left);
        dfs(ans, cur, root->right);
    }
public:
    vector<string> binaryTreePaths(TreeNode* root) {
        vector<string> ans;
```

```
        dfs(ans, "", root);
        return ans;
    }
};
```

571

SQL

594

1. 看对题就行:

```
class Solution {
public:
    int findLHS(vector<int>& nums) {
        unordered_map<int, int> cnt;
        for(int k: nums) ++cnt[k];
        int ans = 0;
        for(auto p: cnt) if(cnt.count(p.first+1)) ans = max(an
s, p.second + cnt[p.first+1]);
        return ans;
    }
};
```

462

昨天的题

1. 还没凉：

```
class Solution {
public:
    int minMoves2(vector<int>& nums) {
        sort(nums.begin(), nums.end());
        int n = nums.size(), ans = 0;
        for(int i=0, j=n-1; i<j; ++i,--j) ans+=nums[j]-nums
[i];
```

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

113

1. 典型dfs :

```
class Solution {
    #define ISLEAF(r) ((r)&&(!r->left)&&(!r->right))
    int target;
    void dfs(vector<vector<int>> &ans, vector<int> cur, int su
m, TreeNode *root){
        if(!root) return;
        cur.push_back(root->val);
        sum += root->val;
        if(ISLEAF(root)){
            if(sum == target) ans.push_back(cur);
            return;
        }
        dfs(ans, cur, sum, root->left);
        dfs(ans, cur, sum, root->right);
    }
public:
    vector<vector<int>> pathSum(TreeNode* root, int sum) {
        target = sum;
        vector<vector<int>> ans;
        dfs(ans, vector<int>(), 0, root);
        return ans;
    }
};
```

614

SQL

306

1. 注意一下Leading zero 跟 overflow 即可：

```
class Solution {
    #define CI(c) int((c) - '0')
    string ad(string a, string b){
        string c;
        for(int i=a.size()-1, j=b.size()-1, cur=0; i>=0 || j>=
0 || cur; ){
            cur += (i>=0?CI(a[i--]):0) + (j>=0?CI(b[j--]):0);
            c += char(cur%10 + '0');
            cur /= 10;
        }
        while(c.size()>1 && c.back()=='0') c.pop_back();
        reverse(c.begin(), c.end());
        return c;
    }
    bool isAdd(string s, int i, int j){
        string a = s.substr(0, i), b = s.substr(i, j-i);
        if((b.size()>1 && b[0]=='0') || (a.size()>1 && a[0]=
='0')) return false;
        while(j < s.size()){
            string c = ad(a, b);
            if(j+c.size()>s.size()||s.substr(j, c.size())!= c)
return false;
            j += c.size();
            a = b;
            b = c;
        }
        return true;
```

```
    }
public:
    bool isAdditiveNumber(string num) {
        if(num.empty()) return false;
        for(int i=1; i<num.size(); ++i) for(int j=i+1; j<num.s
ize(); ++j) if(isAdd(num, i, j)) return true;
        return false;
    }
};
```

410

1. 二分：

```
class Solution {
    bool ok(vector<int>& A, int m, long val){
        int i = 0;
        while(i<A.size() && m){
            if(val < A[i]) return false;
            long tmp = (long)A[i];
            while(i<A.size() && tmp <= val) tmp += (long)A[++
i];
            --m;
        }
        return i==A.size();
    }
public:
    int splitArray(vector<int>& nums, int m) {
        long sum = 0;
        for(int k: nums) sum+=long(k);
        long l = sum/m - 1, r = sum;
        while(l<r-1){
            long c = (l+r)/2;
```

```
            if(ok(nums, m, c)) r = c;
            else l = c;
        }
        return r;
    }
};
```

733

1. 典型dfs

```
class Solution {
    int d[4] = {1, -1, 0, 0};
public:
    vector<vector<int>> floodFill(vector<vector<int>>& image, int sr, int sc, int newColor) {
        int n = image.size(), m = image[0].size(), o = image[sr][sc];
        queue<int> Q;
        Q.push(sr*m + sc);
        image[sr][sc] = newColor;
        while(!Q.empty()){
            int i = Q.front()/m, j = Q.front()%m;
            Q.pop();
            for(int k=0; k<4; ++k){
                int x = i+d[k], y = j + d[3-k];
                if(x>=0 && x<n && y>=0 && y<m && image[x][y]==o && image[x][y]!=newColor){
                    Q.push(x*m + y);
                    image[x][y] = newColor;
                }
            }
        }
```

```
        return image;
    }
};
```