September 5, 2018 题目：280,353,208,809,551,572,20,557,767,810,742,436,72,419,499,439

280

1. 依次交换即可：

```
class Solution {
public:
    void wiggleSort(vector<int>& nums) {
        for(int i=0; i<(int)nums.size()-1; ++i){
            if(i%2 && nums[i] < nums[i+1]) swap(nums[i], nums[i+1]);
            else if(i%2==0 && nums[i] > nums[i+1]) swap(nums[i], nums[i+1]);
        }
    }
};
```

353

1. 需要注意的是 x, y 别弄混了：

```
class SnakeGame {
    typedef pair<int, int> ii;
    int n, m, x, y;
    queue<ii> Q;
    set<ii> C;
    stack<ii> S;
public:
    SnakeGame(int width, int height, vector<pair<int, int>> food): n(height), m(width), x(0), y(0) {
```

```
        for(int i=food.size()-1; i>=0; --i) S.push(ii(food[i].
second, food[i].first));
        Q.push(ii(0, 0));
        C.insert(ii(0, 0));
    }
    int move(string d) {
        if(d=="R") ++x;
        else if(d=="L") --x;
        else if(d=="D") ++y;
        else --y;
        if(x<0 || x>=m || y<0 || y>=n) return -1;
        if(!S.empty() && S.top() == ii(x, y)){
            assert(!C.count(ii(x, y)));
            Q.push(ii(x, y));
            C.insert(ii(x, y));
            S.pop();
            return Q.size() - 1;
        }
        C.erase(Q.front());
        Q.pop();
        if(C.count(ii(x, y))) return -1;
        Q.push(ii(x, y));
        C.insert(ii(x, y));
        return Q.size() - 1;
    }
};
```

208

1. 做过，就是Tier的实现：

```
class Trie {
    struct T{
```

```
        bool W;
        unordered_map<char, T*> C;
        T(): W(false){}
    };
    T * root;
public:
    /** Initialize your data structure here. */
    Trie(): root(new T()){}
    void insert(string word) {
        T* r = root;
        for(char c: word){
            if(!r->C.count(c)) r->C[c] = new T();
            r = r->C[c];
        }
        r->W = true;
    }
    bool search(string word) {
        T *r = root;
        for(char c: word){
            if(!r->C.count(c)) return false;
            r = r->C[c];
        }
        return r->W;
    }
    bool startsWith(string prefix) {
        T *r = root;
        for(char c: prefix){
            if(!r->C.count(c)) return false;
            r = r->C[c];
        }
```

```
        return true;
    }
};
```

809

1. 按题意，依次check就行了，做过：

```
class Solution {
    bool exT(string s, string t){
        int i = 0, j = 0;
        while(i<s.size() && j<t.size()){
            char c = s[i], d = t[j];
            if(c != d) return false;
            int cnt1 = 0, cnt2 = 0;
            while(i<s.size() && s[i]==c){
                ++i;
                ++cnt1;
            }
            while(j<t.size() && t[j]==d){
                ++j;
                ++cnt2;
            }
            if(cnt1<cnt2 || (cnt1!=cnt2 && cnt1 < 3)) return f
alse;
        }
        return i==s.size() && j==t.size();
    }
public:
    int expressiveWords(string S, vector<string>& words) {
        int ans = 0;
        for(string s: words) ans += exT(S, s);
        return ans;
```

```
    }
};
```

551

1. Counting:

```
class Solution {
public:
    bool checkRecord(string s) {
        int cntA = 0, cntL = 0;
        for(char c: s){
            if(c == 'L'){
                ++cntL;
                if(cntL > 2) return false;
            }
            else{
                cntL = 0;
                if(c == 'A'){
                    ++cntA;
                    if(cntA > 1) return false;
                }
            }
        }
        return true;
    }
};
```

572

1.

```
class Solution {
    public boolean isSubtree(TreeNode s, TreeNode t) {
        if (s == null && t == null) return true;
```

```
        if (s == null || t == null) return false;

        return isSame(s, t) || isSubtree(s.left, t) || isSubtr
ee(s.right, t);
    }

    public boolean isSame(TreeNode s, TreeNode t) {
        if (s == null && t == null) return true;
        if (s == null || t == null) return false;
        if (s.val != t.val) return false;
        return isSame(s.left, t.left) && isSame(s.right, t.rig
ht);
    }
}
```

2. 看来树的躯干就是不是 subtree了：

```
class Solution {
    bool eq(TreeNode *s, TreeNode *t){
        if(!t || !s) return !t && !s;
        return eq(s->left, t->left) && eq(s->right, t->right)
&& s->val == t->val;
    }
public:
    bool isSubtree(TreeNode* s, TreeNode* t) {
        if(!s || !t) return !t && !s;
        return eq(s, t) || isSubtree(s->left, t) || isSubtree
(s->right, t);
    }
};
```

20

1. Stack:

```
class Solution {
    char getPair(char c){
        if(c == ')') return '(';
        else if(c == ']') return '[';
        else return '{';
    }
public:
    bool isValid(string s) {
        stack<char> S;
        for(char c: s){
            if(c=='(' || c=='[' || c=='{') S.push(c);
            else{
                if(S.empty() || S.top() != getPair(c)) return
false;
                S.pop();
            }
        }
        return S.empty();
    }
};
```

557

1. array

```
class Solution {
    public String reverseWords(String s) {
        char[] ch = s.toCharArray();
        int i = 0;
        int j = 0;
        while (i < ch.length) {
            if (ch[i] == ' ') {
                reverse(ch, j, i - 1);
```

```
                j = i + 1;
            }
            if (i == ch.length - 1) {
                reverse(ch, j, i);
            }
            i++;
        }
        return new String(ch);
    }

    private void reverse(char[] ch, int i, int j) {
        while (i < j) {
            char t = ch[i];
            ch[i] = ch[j];
            ch[j] = t;
            i++;
            j--;
        }
    }
}
```

2. Short is Beauty 系列：

```
class Solution(object):
    def reverseWords(self, s):
        return ' '.join([t[::-1] for t in s.split(' ')])
```

767

1. 贪心，跟那个 连续k个无相同字母的题一样：

```
class Solution {
    #define CI(c) int((c) - 'a')
    #define IC(i) char((i) + 'a')
```

```
    typedef pair<int, int> ii;
public:
    string reorganizeString(string S) {
        vector<int> cnt(26, 0);
        for(char c: S) cnt[CI(c)]++;
        set<ii, greater<ii>> pool;
        for(int i=0; i<26; ++i) if(cnt[i]) pool.insert(ii(cnt
[i], i));
        if(pool.begin()->first > ((int)S.size() + 1)/2) return
"";
        int i = 0;
        for(auto p: pool) for(int j=0; j<p.first; ++j){
            S[i] = IC(p.second);
            i += 2;
            if(i >= S.size()) i = 1;
        }
        return S;
    }
};
```

810

1. 不停的减掉相同的对数，最后看剩下数的个数的奇偶性。
    - 注意：`Also, if any player starts their turn with the bitwise XOR of all the elements of the chalkboard equal to 0, then that player wins.`

```
class Solution {
public:
    bool xorGame(vector<int>& nums) {
        unordered_set<int> res;
        int sum = 0;
        for(int k: nums){
```

```
            if(res.count(k)) res.erase(k);
            else res.insert(k);
            sum ^= k;
        }
        return !(int(res.size())%2) || !sum;
    }
};
```

2. 哎，太sb了，发现根本不用减，因为奇偶性一直不变：一行就够了

```
bool xorGame(vector<int>& n) {
    return n.size() % 2 == 0 || !accumulate(n.begin(), n.end(), 0, bit_xor<int>());
}
```

3. Short is Beauty 系列：

```
class Solution(object):
    def xorGame(self, nums):
        return (not len(nums)%2) or (not reduce(lambda x, y: x^y, nums))
```

742

1. 比较笨的方法，先构建图，然后再BFS：

```
class Solution {
    unordered_map<int, vector<int>> E;
    unordered_set<int> LF;
    void dfs(TreeNode *r){
        if(!r->left && !r->right){
            LF.insert(r->val);
            return;
        }
        if(r->left) {
            E[r->val].push_back(r->left->val);
            E[r->left->val].push_back(r->val);
```

```
            dfs(r->left);
        }
        if(r->right) {
            E[r->val].push_back(r->right->val);
            E[r->right->val].push_back(r->val);
            dfs(r->right);
        }
    }
public:
    int findClosestLeaf(TreeNode* root, int k) {
        dfs(root);
        unordered_set<int> S{k};
        queue<int> Q;
        Q.push(k);
        while(!Q.empty()){
            if(LF.count(Q.front())) return Q.front();
            for(int j: E[Q.front()]) if(!S.count(j)){
                S.insert(j);
                Q.push(j);
            }
            Q.pop();
        }
        return -1;
    }
};
```

- [ ] 可以依次dfs，思路不难，但判定很多，可以研究一下 https://leetcode.com/problems/closest-leaf-in-a-binary-tree/discuss/139334/JAVA-+-DFS-+-ONETIME-TRAVERSE-+-24ms @Zebo L

436

1. 典型二分：

```
class Solution {
    typedef pair<int, int> ii;
public:
    vector<int> findRightInterval(vector<Interval>& Is) {
        map<int, int> pos;
        vector<ii> res;
        vector<int> ans(Is.size(), -1);
        for(int i=0; i<Is.size(); ++i) {
            pos[Is[i].start] = i;
            res.push_back(ii(Is[i].start, Is[i].end));
        }
        sort(res.begin(), res.end());
        for(int i=0; i<res.size(); ++i){
            int st = res[i].first, ed = res[i].second;
            int l = i, r = res.size();
            while(l<r-1){
                int c = (l + r)/2;
                if(res[c].first >= ed) r = c;
                else l = c;
            }
            if(r < res.size()) ans[pos[st]] = pos[res[r].firs
t];
        }
        return ans;
    }
};
```

2. 之前的Array跟本就是多余的：

```
class Solution {
public:
    vector<int> findRightInterval(vector<Interval>& Is) {
```

```
        map<int, int> pos;
        vector<int> ans(Is.size(), -1);
        for(int i=0; i<Is.size(); ++i) pos[Is[i].start] = i;
        for(auto p: pos){
            int end = Is[p.second].end;
            auto it = pos.lower_bound(end);
            if(it!=pos.end()) ans[p.second] = it->second;
        }
        return ans;
    }
};
```

72

1. 典型dp：

```
class Solution {
    const int inf = 1E8;
public:
    int minDistance(string word1, string word2) {
        int n = word1.size(), m = word2.size();
        vector<vector<int>> dp(n+1, vector<int>(m+1, inf));
        dp[n][m] = 0;
        for(int i=0; i<n; ++i) dp[i][m] = n-i;
        for(int j=0; j<m; ++j) dp[n][j] = m-j;
        for(int i=n-1; i>=0; --i) for(int j=m-1; j>=0; --j){
            if(word1[i] == word2[j]) dp[i][j] = dp[i+1][j+1];
            else dp[i][j] = 1 + min(dp[i][j+1], min(dp[i+1]
[j], dp[i+1][j+1]));
        }
        return dp[0][0];
    }
};
```

419

1. 条件很强，直接One Pass即可：

```
class Solution {
public:
    int countBattleships(vector<vector<char>>& B) {
        if(B.empty() || B[0].empty()) return 0;
        int ans = 0, n = B.size(), m = B[0].size();
        for(int i=0; i<n; ++i){
            for(int j=0, cnt=0; j<=m; ++j){
                if(j<m && B[i][j]=='X') ++cnt;
                else if(cnt){
                    if(cnt > 1) ++ans;
                    else if(!i || B[i-1][j-1] == '.') ++ans;
                    cnt = 0;
                }
            }
        }
        return ans;
    }
};
```

499

1. BFS

```
class Solution {
    class Point {
        int x;
        int y;
        int dis;
        String dir;
        public Point(int x, int y, int dis, String dir) {
```

```
            this.x = x;
            this.y = y;
            this.dis = dis;
            this.dir = dir;
        }
    }
    public String findShortestWay(int[][] maze, int[] ball, in
t[] hole) {
        PriorityQueue<Point> pq = new PriorityQueue<>((a, b) -
> a.dis == b.dis ? a.dir.compareTo(b.dir) : a.dis - b.dis);
        int m = maze.length, n = maze[0].length;
        Point[][] points = new Point[m][n];
        pq.offer(new Point(ball[0], ball[1], 0, ""));
        for (int i = 0; i < m; i++) {
            for (int j = 0; j < n; j++) {
                points[i][j] = new Point(i, j, Integer.MAX_VAL
UE, "");
            }
        }
        int[] dx = {1, -1, 0, 0};
        int[] dy = {0, 0, -1, 1};
        String[] directions = {"d", "u", "l", "r"};
        while (!pq.isEmpty()) {
            Point cur = pq.poll();
            if (points[cur.x][cur.y].dis <= cur.dis) continue;
            points[cur.x][cur.y] = cur;
            int x = cur.x;
            int y = cur.y;
            for (int i = 0; i < 4; i++) {
                int ux = x;
                int uy = y;
```

```
                int dis = cur.dis;
                while (ux >= 0 && uy >= 0 && ux < m && uy < n
&& maze[ux][uy] == 0 && !(ux == hole[0] && uy == hole[1])) {
                    ux += dx[i];
                    uy += dy[i];
                    dis++;
                }
                if (!(ux == hole[0] && uy == hole[1])) {
                    ux -= dx[i];
                    uy -= dy[i];
                    dis--;
                }
                pq.offer(new Point(ux, uy, dis, cur.dir + dire
ctions[i]));
            }
        }
        return points[hole[0]][hole[1]].dis == Integer.MAX_VAL
UE ? "impossible" : points[hole[0]][hole[1]].dir;
    }
}
```

2. 就是非常单纯的BFS，但容易出错：楼上的方法更有效一些

```
class Solution {
    typedef vector<int> vi;
    typedef pair<int, string> is;
    typedef pair<int, char> ic;
    string ref = "dlru";
    vi getDire(char c, int i, int j){
        if(c == 'd') return {i+1, j};
        if(c == 'l') return {i, j-1};
        if(c == 'r') return {i, j+1};
        return {i-1, j};
```

```
    }
public:
    string findShortestWay(vector<vector<int>>& M, vector<int>
& ball, vector<int>& hole) {
        assert(!M.empty() && !M[0].empty());
        int n = M.size(), m = M[0].size();
        map<char, string> redirect;
        redirect['l'] = "du";
        redirect['r'] = "du";
        redirect['d'] = "lr";
        redirect['u'] = "lr";
        set<ic> S;
        queue<is> Q;
        for(int k=0; k<4; ++k){
            char direct = ref[k];
            vi next = getDire(direct, ball[0], ball[1]);
            if(next[0]>=0 && next[1]>=0 && next[0]<n && next
[1]<m && !M[next[0]][next[1]]){
                Q.push(is(next[0]*m + next[1], string(1, direc
t)));
                S.insert(ic(next[0]*m + next[1], direct));
            }
        }
        while(!Q.empty()){
            int pos = Q.front().first;
            string path = Q.front().second;
            Q.pop();
            int i = pos/m, j = pos%m;
            if(i==hole[0] && j==hole[1]) return path;
            char direct = path.back();
            vi next = getDire(direct, i, j);
```

```
            if(next[0]>=0 && next[1]>=0 && next[0]<n && next
[1]<m && !M[next[0]][next[1]]){
                if(!S.count(ic(next[0]*m + next[1], direct))){
                    Q.push(is(next[0]*m + next[1], path));
                    S.insert(ic(next[0]*m + next[1], direct));
                }
            }
            else{
                for(char c: redirect[direct]){
                    vi z = getDire(c, i, j);
                    if(z[0]>=0 && z[0]<n && z[1]>=0 && z[1]<m
&& !M[z[0]][z[1]] && !S.count(ic(z[0]*m +z[1], c))){
                        Q.push(is(z[0]*m + z[1], path + c));
                        S.insert(ic(z[0]*m + z[1], c));
                    }
                }
            }
        }
        return "impossible";
    }
};
```

439

1. recursion会超时（大概是爆栈）改成iterative就好了

```
class Solution {
    public String parseTernary(String expression) {
        System.out.println(expression);
        if (expression.length() == 1) return expression;

        while (expression.length() != 1) {
            int index = expression.lastIndexOf("?");
```

```
            expression = expression.substring(0, index - 1) +
(expression.charAt(index - 1) == 'T' ? expression.substring(in
dex + 1, index + 2) : expression.substring(index + 3, index +
4)) + (index + 4 < expression.length() ? expression.substring
(index + 4) : "");
        }
        return expression;
        //int index = expression.lastIndexOf("?");
        //return parseTernary(expression.substring(0, index -
1) + (expression.charAt(index - 1) == 'T' ? expression.substri
ng(index + 1, index + 2) : expression.substring(index + 3, ind
ex + 4)) + (index + 4 < expression.length() ? expression.subst
ring(index + 4) : ""));
    }
}
```

2. 直接recursion 过了 @Tongtong X

```
class Solution {
public:
    string parseTernary(string s) {
        auto i = s.find('?');
        if(i==string::npos) return s;
        int j = i+1, cnt = 0;
        while(j<s.size()){
            if(s[j]=='?') ++cnt;
            if(s[j]==':') --cnt;
            ++j;
            if(cnt==-1) break;
        }
        if(s.substr(0, i)=="T") return parseTernary(s.substr(i
+1, j-i-2));
        else return parseTernary(s.substr(j));
    }
```

```
};
```

2. Iterative, 基本一样：

```
class Solution {
public:
    string parseTernary(string s) {
        while(true){
            auto i = s.find('?');
            if(i==string::npos) return s;
            int j = i+1, cnt = 0;
            while(j<s.size()){
                if(s[j]=='?') ++cnt;
                if(s[j]==':') --cnt;
                ++j;
                if(cnt==-1) break;
            }
            if(s.substr(0, i)=="T") s = s.substr(i+1, j-i-2);
            else s = s.substr(j);
        }
    }
};
```