September 12, 2018 题目：354,493,417,667,360,819,796,763,512,757,466,669

354

1. LIS 完全相同的思路：

```
class Solution {
    typedef pair<int, int> ii;
public:
    int maxEnvelopes(vector<pair<int, int>>& E) {
        if(E.empty()) return 0;
        sort(E.begin(), E.end(), [](const ii&x, const ii&y){
            if(x.first == y.first) return x.second > y.second;
            return x.first < y.first;
        });
        vector<int> res;
        for(auto p: E){
            if(res.empty() || p.second > res.back()){
                res.push_back(p.second);
            }
            else{
                int l = -1, r = res.size();
                while(l < r-1){
                    int c = (l+r)/2;
                    if(res[c] < p.second) l = c;
                    else r = c;
                }
                res[r] = p.second;
            }
        }
        return res.size();
    }
};
```

493

1. Segment Tree 注意细节，over flow的情况很烦的:

```
class Solution {
    vector<long> SEG;
    int N;
    int dfs(int l, int r, int i, int j, int k){
        if(j<=l || i>=r) return 0;
        if(i<=l && j>=r) return SEG[k];
        return dfs(l, (l+r)/2, i, j, 2*k+1) + dfs((l+r)/2, r,
i, j, 2*k+2);
    }
    void updateSeg(int i, long val){
        int k = N - 1 + i;
        SEG[k] += val;
        do{
            k = (k-1)/2;
            SEG[k] += val;
        }while(k);
    }
    typedef pair<long, int> ii;
public:
    int reversePairs(vector<int>& nums) {
        int n = nums.size(), ans = 0;
        if(n<=1) return 0;
        vector<ii> reorder;
        for(int i=0; i<n; ++i) reorder.push_back(ii(long(nums
[i]), i));
        sort(reorder.begin(), reorder.end());
        // for(auto p: reorder) cout<<p.first<<' '<<p.second<<
endl;
        N = 1;
        while(N<n) N *= 2;
        SEG = vector<long>(2*N, 0);
        for(int i=n-1, j=n-1; i>=0; --i){
            while(j>=0 && reorder[j].first > 2*reorder[i].firs
t) {
                updateSeg(reorder[j].second, 1);
                --j;
            }
            // for(int k: SEG) cout<<k <<' ';
            // cout<<endl;
            ans += dfs(0, N, 0, reorder[i].second, 0);
        }
        return ans;
    }
};
```
2. MergeSort 确实过了：
```
class Solution {
    int res;
    vector<long> A;
    void mergeSort(int l, int r){
        if(l >= r-1) return;
        int m = (l+r)/2;
        mergeSort(l, m);
        mergeSort(m, r);
        vector<long> B;
        int i=l, j=m;
        while(i<m || j<r){
            if(j==r || (i<m && A[i] <= A[j])) B.push_back(A[i+
+]);
            else{
                int left = l-1, right = m;
                while(left < right-1){
                    int mid = (left + right)/2;
                    if(A[mid] > 2*A[j]) right = mid;
                    else left = mid;
                }
                res += m - right;
                B.push_back(A[j++]);
            }
        }
        copy(B.begin(), B.end(), A.begin() + l);
    }
public:
    int reversePairs(vector<int>& nums) {
        for(int k: nums) A.push_back(long(k));
        res = 0;
        mergeSort(0, (int)A.size());
        return res;
    }
};
```

417

1. 有一点逆向思维

```
class Solution {
    final int[] dx = {1, -1, 0, 0};
    final int[] dy = {0, 0, -1, 1};
    public List<int[]> pacificAtlantic(int[][] matrix) {
        List<int[]> res = new ArrayList<>();
        if (matrix == null || matrix.length == 0) {
            return res;
        }
        Queue<int[]> leftTop = new LinkedList<>();
        Queue<int[]> rightBottom = new LinkedList<>();
        int m = matrix.length, n = matrix[0].length;
        boolean[][] pVisited = new boolean[m][n];
        boolean[][] aVisited = new boolean[m][n];

        for (int i = 0; i < n; i++) {
            leftTop.offer(new int[]{0, i});
            rightBottom.offer(new int[]{m - 1, i});
            pVisited[0][i] = true;
            aVisited[m - 1][i] = true;
        }

        for (int i = 0; i < m; i++) {
            leftTop.offer(new int[]{i, 0});
            rightBottom.offer(new int[]{i, n - 1});
            pVisited[i][0] = true;
            aVisited[i][n - 1] = true;
        }

        helper(leftTop, pVisited, matrix, m, n);
        helper(rightBottom, aVisited, matrix, m, n);

        for (int i = 0; i < m; i++) {
            for (int j = 0; j < n; j++) {
                if (pVisited[i][j] && aVisited[i][j]) {
                    res.add(new int[]{i, j});
                }
            }
        }
        return res;
    }

    public void helper(Queue<int[]> q, boolean[][] visited, in
t[][] matrix, int m, int n) {
        while (!q.isEmpty()) {
            int[] cur = q.poll();
            for (int k = 0; k < 4; k++) {
                int ux = cur[0] + dx[k];
                int uy = cur[1] + dy[k];
                if (ux < 0 || uy < 0 || ux >= m || uy >= n ||
visited[ux][uy] || matrix[ux][uy] < matrix[cur[0]][cur[1]]) {
                    continue;
                }
                visited[ux][uy] = true;
                q.offer(new int[]{ux, uy});
            }
        }
    }
}
```

2. BFS记Frontier：

```
class Solution {
    typedef pair<int, int> ii;
    int d[4] = {1, -1, 0, 0}, n, m;
    void bfs(unordered_set<int> &S, vector<vector<int>>& M){
        queue<int> Q;
        for(int k: S) Q.push(k);
        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 && !S.count(x*m+
y) && M[x][y]>=M[i][j]){
                    S.insert(x*m + y);
                    Q.push(x*m + y);
                }
            }
        }
    }
public:
    vector<pair<int, int>> pacificAtlantic(vector<vector<int>>
& matrix) {
        vector<pair<int, int>> ans;
        if(matrix.empty() || matrix[0].empty()) return ans;
        n = matrix.size();
        m = matrix[0].size();
        unordered_set<int> P, A;
        for(int i=0; i<n; ++i){
            P.insert(i*m + 0);
            A.insert(i*m + m - 1);
        }
        for(int j=0; j<m; ++j){
            P.insert(0 + j);
            A.insert((n-1)*m + j);
        }
        bfs(P, matrix);
        bfs(A, matrix);
        for(int k: P) if(A.count(k)) ans.push_back(ii(k/m, k%
m));
        return ans;
    }
};
```

667

1.

```
class Solution {
    public int[] constructArray(int n, int k) {
        // 1, n - 1, 2, n - 2, 3, n - 3, ...
        int[] res = new int[n];
        int lo = 1, high = n;
        int i = 0;
        while (i < k) {
            res[i++] = lo++;
            if (i < k) {
                res[i++] = high--;
            }
        }
        if (k % 2 == 0) {
            while (i < n) {
                res[i++] = high--;
            }
        } else {
            while (i < n) {
                res[i++] = lo++;
            }
        }
        return res;
    }
}
```
2. 做过，折叠：
```
class Solution {
public:
    vector<int> constructArray(int n, int k) {
        vector<int> ans;
        int l = 1, r = n, f = 0;
        while(k){
            if(f) ans.push_back(r--);
            else ans.push_back(l++);
            f = 1-f;
            --k;
        }
        if(ans.back() == l-1) for(;l<=r;++l) ans.push_back(l);
        else for(; r>=l;--r) ans.push_back(r);
        return ans;
    }
};
```

360

1. 先找最小值点，然后向两边延伸：
```
class Solution {
public:
    vector<int> sortTransformedArray(vector<int>& nums, int a,
int b, int c) {
        if(!a){
            bool rev = (b < 0);
            for(auto &n: nums) n = b*n + c;
            if(rev) reverse(nums.begin(), nums.end());
            return nums;
        }
        bool rev = (a < 0);
        int l=-1, r=nums.size(), n=nums.size();
        double center = -double(b)/2./a;
        while(l < r-1){
            int c = (l+r)/2;
            if(double(nums[c]) < center) l = c;
            else r = c;
        }
        vector<int> ans;
        for(int i=0; i<n; ++i){
            if(l<0) {
                ans.push_back(a * nums[r] * nums[r] + b * nums
[r] + c);
                ++r;
            }
            else if(r>=n) {
                ans.push_back(a * nums[l] * nums[l] + b * nums
[l] + c);
                --l;
            }
            else if(nums[r]-center < center-nums[l]){
                ans.push_back(a * nums[r] * nums[r] + b * nums
[r] + c);
                ++r;
            }
            else{
                ans.push_back(a * nums[l] * nums[l] + b * nums
[l] + c);
                --l;
            }
        }
        if(rev) reverse(ans.begin(), ans.end());
        return ans;
    }
};
```

819

1. 最后一个case应该是这几天新加上去的，**"a, a, a, a, b,b,b,c, c" => ["a"] 不知道咋改**

```
class Solution {
    public String mostCommonWord(String paragraph, String[] ba
nned) {
        String[] strs = paragraph.toLowerCase().split("\\W+");
        //System.out.println(Arrays.toString(strs));
        Map<String, Integer> map = new HashMap<>();
        for (String str : strs) {
            map.put(str, map.getOrDefault(str, 0) + 1);
        }

        for (String s : banned) {
            map.remove(s);
        }

        int max = 0;
        String res = "";
        for (String s : map.keySet()) {
            if (map.get(s) > max) {
                max = map.get(s);
                res = s;
            }
        }
        return res;
    }
}
```

2. 一道Easy 费了我九牛二虎之力：

```
class Solution:
    def _process(self, s):
        return s.strip('!?\',;.\n').lower()
    def mostCommonWord(self, s, banned):
        banned = [w.lower() for w in banned]
        pool = [self._process(w) for w in s.split(' ')]
        pool = [w for w in pool if w and w not in banned]
        print(pool, banned)
        cnt = {}
        res = pool[0]
        for w in pool:
            if w not in cnt:
                cnt[w] = 1
            else:
                cnt[w] += 1
        for w in cnt:
            if cnt[w] > cnt[res]:
                res = w
        return res
```

796

1. 厉害了。。好像还有一道题也是类似的解法，想不起来了

```
class Solution {
    public boolean rotateString(String A, String B) {
        return A.length() == B.length() && (A + A).contains
(B);
    }
}
```

2. 楼上解法神了！！！ @Tongtong X：

```
class Solution {
public:
    bool rotateString(string A, string B) {
        if(A.size() != B.size()) return false;
        if(A.empty() && B.empty()) return true;
        for(int i=0, n=A.size(); i<n; ++i){
            bool ok = true;
            for(int j=0; j<n && ok; ++j) if(A[(i+j)%n] != B
[j]) ok = false;
            if(ok) return true;
        }
        return false;
    }
};
```

763

1.

```
class Solution {
    public List<Integer> partitionLabels(String S) {
        int[] chs = new int[26];
        List<Integer> list = new ArrayList<>();
        int start = 0;
        int end = 0;
        for (int i = 0; i < S.length(); i++) {
            chs[S.charAt(i) - 'a'] = i;
        }

        for (int i = 0; i < S.length(); i++) {
            end = Math.max(end, chs[S.charAt(i) - 'a']);
            if (end == i) {
                list.add(end - start + 1);
                start = end + 1;
            }
        }
        return list;
    }
}
```
2. n 方solution：
```
class Solution {
    bool inter(unordered_map<char, int> &m1, unordered_map<cha
r, int> &m2){
        for(auto p: m1) if(m2.count(p.first)) return true;
        return false;
    }
public:
    vector<int> partitionLabels(string S) {
        int head = -1;
        vector<int> ans;
        unordered_map<char, int> m1, m2;
        for(char c: S) ++m2[c];
        for(int i=0; i<S.size(); ++i){
            m2[S[i]]--;
            if(!m2[S[i]]) m2.erase(S[i]);
            m1[S[i]]++;
            if(!inter(m1, m2)){
                ans.push_back(i-head);
                head = i;
            }
        }
        return ans;
    }
};
```
3. 好方法：
```
class Solution {
    #define CI(c) int((c) - 'a')
public:
    vector<int> partitionLabels(string S) {
        vector<int> pos(26, 0), ans;
        for(int i=0; i<S.size(); ++i) pos[CI(S[i])] = i;
        int head = -1;
        for(int i=0, tmp=pos[CI(S[0])]; i<S.size(); ++i){
            tmp = max(tmp, pos[CI(S[i])]);
            if(i == tmp){
                ans.push_back(i-head);
                head = i;
            }
        }
        return ans;
    }
};
```

512没有题

757

1. n方非常慢的solution：
```
class Solution {
    typedef pair<int, int> ii;
public:
    int intersectionSizeTwo(vector<vector<int>>& intervals) {
        map<int, int> I, R;
        for(auto p: intervals){
            if(!I.count(p[0])) I[p[0]] = p[1];
            else I[p[0]] = min(I[p[0]], p[1]);
            R[p[0]] = 2;
        }
        int ans = 0;
        while(!I.empty()){
            int start = INT_MAX, l = 0;
            for(auto p: R){
                if(I[p.first] - p.second + 1 < start){
                    start = I[p.first] - p.second + 1;
                    l = p.second;
                }
                else if(I[p.first] - p.second + 1 == start &&
p.second > l){
                    l = p.second;
                }
            }
            int end = start + l - 1;
            ans += l;
            for(auto it=I.begin(); it!=I.end() && it->first <=
end; ){
                int x = max(it->first, start), y = min(it->sec
ond, end);
                int d = y - x + 1;
                R[it->first] -= d;
                if(R[it->first] <= 0) {
                    R.erase(it->first);
                    it = I.erase(it);
                }
                else ++it;
            }
        }
        return ans;
    }
};
```
2. 改进 n log n：
```
class Solution {
public:
    int intersectionSizeTwo(vector<vector<int>>& intervals) {
        map<int, int> LR, RL, RT;
        for(auto p: intervals){
            if(!RL.count(p[1])) RL[p[1]] = p[0];
            else RL[p[1]] = max(RL[p[1]], p[0]);
        }
        for(auto p: RL){
            if(!LR.count(p.second)) LR[p.second] = p.first;
            else LR[p.second] = min(LR[p.second], p.first);
        }
        for(auto p: LR){
            RT[p.second] = 2;
        }
        int ans = 0;
        while(!LR.empty()){
            int end = RT.begin()->first, l = RT.begin()->secon
d;
            int start = end - l + 1;
            ans += l;
            for(auto it=LR.begin(); it!=LR.end() && it->first<
=end;){
                int x = max(it->first, start), y = min(it->sec
ond, end);
```

```
                int d = y-x+1;
                RT[it->second] -= d;
                if(RT[it->second] <= 0) {
                    RT.erase(it->second);
                    it = LR.erase(it);
                }
                else ++it;
            }
        }
        return ans;
    }
};
```
3. 精妙的方法：
```
class Solution {
public:
    int intersectionSizeTwo(vector<vector<int>>& intervals) {
        int ans = 0, x=-1, y=-1;
        sort(intervals.begin(), intervals.end(), [](vector<int
>&a, vector<int>&b){
            return (a[1] < b[1]) || (a[1] == b[1] && a[0] > b
[0]);
        });
        for(auto v: intervals) if(v[0] > x){
            if(v[0] <= y){
                ++ans;
                x = y;
                y = v[1];
            }
            else{
                ans += 2;
                y = v[1];
                x = y-1;
            }
        }
        return ans;
    }
};
```

466

1. 找循环节，非常Ugly地做对了：

```
class Solution {
    typedef pair<int, int> ii;
    unordered_map<char, set<int>> pos;
    vector<ii> trans, transM;
public:
    int getMaxRepetitions(string s1, int n1, string s2, int n
2) {
        int l1 = s1.size(), l2 = s2.size();
        for(int i=0; i<l1; ++i) pos[s1[i]].insert(i);
        for(char c: s2) if(!pos.count(c)) return 0;
        if(!n1) return 0;
        trans.resize(l1);
        transM.resize(l1);
        for(int i=l1-1; i>=0; --i){
            if(i<l1-1 && s1[i] != s2[0]) trans[i] = trans[i+
1];
            else{
                int cnt = 0;
                int j = i;
                for(char c: s2){
                    auto it = pos[c].lower_bound(j);
                    if(it == pos[c].end()){
```

```
                    ++cnt;
                    it = pos[c].lower_bound(0);
                }
                j = (*it) + 1;
                if(j==l1){
                    ++cnt;
                    j = 0;
                }
            }
            trans[i] = ii(j, cnt);
        }
    }
    for(int i=l1-1; i>=0; --i){
        if(i<l1-1 && s1[i] != s2[0]) transM[i] = transM[i+
1];
        else{
            int j = i, cnt = 0, k=0;
            unordered_map<int, int> cnts, T;
            T[j] = 0;
            cnts[0] = 0;
            while(k < n2){
                cnt += trans[j].second;
                j = trans[j].first;
                ++k;
                if(T.count(j)) break;
                T[j] = k;
                cnts[k] = cnt;
            }
            if(k < n2) {
                int start = T[j], end = k;
                int delta_cnt = cnt - cnts[start];
                int delta_k = end - start;
                int loop = (n2 - k)/ delta_k;
                k += loop * delta_k;
                cnt += loop * delta_cnt;
                while(k < n2){
                    cnt += trans[j].second;
                    j = trans[j].first;
                    ++k;
                }
            }
            transM[i] = ii(j, cnt);
        }
        cout<<i<<"shaocong"<<endl;
    }
    int j = 0, cnt = 0, k = 0;
    unordered_map<int, int> cnts, T;
    for(auto p: transM) cout<<p.first<<' '<<p.second<<end
l;
    cnts[0] = 0;
    T[0] = 0;
    while(cnt<n1) {
        cnt += transM[j].second;
        j = transM[j].first;
        if(cnt >= n1){
            if(cnt==n1 && j==0) return k+1;
            else return k;
        }
        ++k;
        if(T.count(j)) break;
```

```
            T[j] = k;
            cnts[k] = cnt;
        }
        int delta_k = k - T[j];
        int delta_cnt = cnt - cnts[T[j]];
        int loop = (n1 - 1 - cnt)/delta_cnt;
        cout<<loop<<endl;
        cnt += loop * delta_cnt;
        k += delta_k * loop;
        while(cnt<n1) {
            cnt += transM[j].second;
            j = transM[j].first;
            if(cnt >= n1){
                if(cnt==n1 && j==0) return k+1;
                else return k;
            }
            ++k;
        }
        return k;
    }
};
```

669

1.

```
class Solution {
    public TreeNode trimBST(TreeNode root, int L, int R) {
        if (root == null) {
            return null;
        }
        if (root.val < L) {
            return trimBST(root.right, L, R);
        }
        if (root.val > R) {
            return trimBST(root.left, L, R);
        }

        root.left = trimBST(root.left, L, R);
        root.right = trimBST(root.right, L, R);
        return root;
    }
}
```

2. 单纯recursion：

```
class Solution {
public:
    TreeNode* trimBST(TreeNode* root, int L, int R) {
        if(!root) return NULL;
        if(root->val > R) return trimBST(root->left, L, R);
        if(root->val < L) return trimBST(root->right, L, R);
        root->left = trimBST(root->left, L, R);
        root->right = trimBST(root->right, L, R);
        return root;
    }
};
```