July 28, 2018 题目：331,308,382,231,498,347,143,527,316,307,173,741,801

331

1. 利用树的一些性质：
    - number of `#` = number of `数字` + 1
    - 除了最后一个 `#`, 其余的与数字一一对应。

```
class Solution {
public:
    bool isValidSerialization(string preorder) {
        int depth = 0;
        auto i = preorder.find_first_not_of(" ");
        while(i<preorder.size() && depth>=0){
            if(preorder[i]!='#'){
                ++depth;
                i = preorder.find(',', i);
                if(i==string::npos) i = preorder.size();
                ++i;
            }
            else{
                --depth;
                i+=2;
            }
        }

        return i>=preorder.size() && depth<0;
    }
};
```

2. 奇技淫巧，degree = indegree - outdegree. When a new node comes, indegree ++, when the current node is not null, its out degree increase by 2. During the whole process, the degree shall not be less than 0. And finally the degree shall be zero.

```java
class Solution {
    public boolean isValidSerialization(String preorder) {
        String[] pc = preorder.split(",");
        int d = 1;
        for (String p : pc) {
            if (--d < 0) {
                return false;
            }
            if (!p.equals("#")) {
                d += 2;
            }
        }
        return d == 0;
    }
}
```

3. 同上

```java
class Solution {
    public boolean isValidSerialization(String preorder) {
        int diff = 1;
        for (String s : preorder.split(",")) {
            diff--;
            if (diff < 0) return false;
            if (!s.equals("#")) diff += 2;
        }
        return diff == 0;
    }
}
```

308

1. Do the same thing using 1d binary indexed tree technique.

```
class NumMatrix {
    int[][] num;
    int[][] tree;

    public NumMatrix(int[][] matrix) {
        if (matrix.length == 0 || matrix[0].length == 0) retur
n;
        tree = new int[matrix.length + 1][matrix[0].length +
1];
        num = new int[matrix.length][matrix[0].length];
        for (int i = 0; i < matrix.length; i++) {
            for (int j = 0; j < matrix[0].length; j++) {
                update(i, j, matrix[i][j]);
            }
        }
    }

    public void update(int row, int col, int val) {
        int dif = val - num[row][col];
        num[row][col] = val;
        for (int i = row + 1; i < tree.length; i += i & (-i))
{
            for (int j = col + 1; j < tree[0].length; j += j &
(-j)) {
                tree[i][j] += dif;
            }
        }
    }

    public int sumRegion(int row1, int col1, int row2, int col
2) {
        return sum(row2 + 1, col2 + 1) - sum(row1, col2 + 1) -
sum(row2 + 1, col1) + sum(row1, col1);
    }
    int sum(int row1, int col1) {
        int sum = 0;
        for (int i = row1; i > 0; i -= i & (-i)) {
            for (int j = col1; j > 0; j -= j & (-j)) {
                sum += tree[i][j];
            }
        }
        return sum;
    }
}
```

2. 二维Segment Tree：

```
class NumMatrix {
    typedef vector<int> vi;
    int n, m, N, M;
    vector<vi> Seg;
public:
    NumMatrix(vector<vector<int>> matrix) {
        if(!matrix.empty() && !matrix[0].empty()){
            n = matrix.size();
            m = matrix[0].size();
            for(N=1; N<n; N*=2);
            for(M=1; M<m; M*=2);
            Seg = vector<vi>(2*N, vi(2*M, 0));
            for(int i=N-1; i<N+n-1; ++i){
                for(int j=M-1; j<M+m-1; ++j) Seg[i][j] = matri
x[i-N+1][j-M+1];
```

```
                for(int k=M-1; k; k = (k-1)/2){
                    for(int j=k; j<2*k+1; j+=2) Seg[i][(j-1)/2] = Seg[i][j] + Seg[i][j+1];
                }
            }
            for(int j=0; j<M+m-1; ++j){
                for(int k=N-1; k; k = (k-1)/2){
                    for(int i=k; i<2*k+1; i+=2) Seg[(i-1)/2][j] = Seg[i][j] + Seg[i+1][j];
                }
            }
        }
    }

    void update(int row, int col, int val) {
        int delta = val - Seg[N-1+row][M-1+col];
        cout<<"zebo"<<endl;
        for(int i=(N-1+row)*2+1; i; i=(i-1)/2){
            for(int j=(M-1+col)*2+1; j; j=(j-1)/2) Seg[(i-1)/2][(j-1)/2] += delta;
        }
    }
    int query1D(int i0, int l, int r, int k, int j1, int j2){
        if(j2<=l || j1>=r) return 0;
        if(j1<=l && j2>=r) return Seg[i0][k];
        return query1D(i0, l, (l+r)/2, 2*k+1, j1, j2) + query1D(i0, (l+r)/2, r, 2*k+2, j1, j2);
    }
    int query2D(int u, int d, int t, int i1, int i2, int l, int r, int k, int j1, int j2){
        if(i2<=u || i1>=d) return 0;
        if(i1<=u && i2>=d) return query1D(t, l, r, k, j1, j2);
        return query2D(u, (u+d)/2, 2*t+1, i1, i2, l, r, k, j1, j2) + query2D((u+d)/2, d, 2*t+2, i1, i2, l, r, k, j1, j2);
    }

    int sumRegion(int row1, int col1, int row2, int col2) {
        return query2D(0, N, 0, row1, row2+1, 0, M, 0, col1, col2+1);
    }
};
```

3. Square root decomposition from Leetcode discussion, 如果不嫌麻烦，可以看看

```
class NumMatrix {
    vector<int> s; // block. visit by s[idx(x, y)]
    int m, n;
    int d1, d2; // block size
    int xlen, ylen; // how many block in x/y axis
    int idx(int x, int y) {
        return x * ylen + y;
    }
    vector<vector<int>> mat;
    int idx2(int x, int y) {
        return x * n + y;
    }
    vector<int> scol, srow; // srow[idx2(i, j)] = sum_{k from 0 to j} mat[i][k]; scol[idx2(i, j)] = sum_{k from 0 to i} mat[k][j]
    int rowsum(int row, int col1, int col2) { // sum[row, col1...col2]
        if (col1 > col2) return 0;
        if (row < 0 || row >= m) return 0;
        return srow[idx2(row, col2)] - (col1 ? srow[idx2(row,
col1 - 1)] : 0);
    }
    int colsum(int col, int row1, int row2) { // sum[row1...ro
w2, col]
        if (row1 > row2) return 0;
        if (col < 0 || col >= n) return 0;
        return scol[idx2(row2, col)] - (row1 ? scol[idx2(row1
-1, col)] : 0);
    }
public:
    NumMatrix(vector<vector<int>> matrix): mat(move(matrix)) {

        if (mat.empty()) goto FINISH;
        m = mat.size(); n = mat[0].size();
        d1 = sqrt(m); d2 = sqrt(n);
        xlen = (m + d1 - 1) / d1; ylen = (n + d2 - 1) / d2;
        s.resize(xlen * ylen);
        scol.resize(m * n);
        srow.resize(m * n);
        for (int i=0; i<m; ++i)
            for (int j=0; j<n; ++j)
            {
                s[idx(i / d1, j / d2)] += mat[i][j];
                srow[idx2(i, j)] = (j ? srow[idx2(i, j-1)] :
0) + mat[i][j];
            }
        for (int j=0; j<n; ++j)
            for (int i=0; i<m; ++i)
                scol[idx2(i, j)] = (i ? scol[idx2(i-1, j)] :
0) + mat[i][j];
```

```
        FINISH: (void)0;
    }

    void update(int row, int col, int val) {
        s[idx(row / d1, col / d2)] += val - mat[row][col];
        for (int j=col; j<n; ++j)
            srow[idx2(row, j)] += val - mat[row][col];
        for (int i=row; i<m; ++i)
            scol[idx2(i, col)] += val - mat[row][col];
        mat[row][col] = val;
    }

    int sumRegion(int row1, int col1, int row2, int col2) {
        int bx1 = (row1 + d1 - 1) / d1, bx2 = (row2 + 1) / d1
- 1;
        int by1 = (col1 + d2 - 1) / d2, by2 = (col2 + 1) / d2
- 1;
        int ans = 0;
        // special case where there is no block in given range
        if (row2 - row1 < 2 * d1)
        {
            for (int i=row1; i<=row2; ++i) // row2 - row1 < 2
* d1 = 2 * sqrt(m)
                ans += rowsum(i, col1, col2);
            goto FINISH;
        }
        if (col2 - col2 < 2 * d1) {
            for (int j=col1; j <= col2; ++j)
                ans += colsum(j, row1, row2);
            goto FINISH;
        }
        {
            for (int i=bx1; i<=bx2; ++i)
                for (int j=by1; j<=by2; ++j)
                    ans += s[idx(i, j)];
            int rowup = max(row1, bx1 * d1 - 1), rowbottom = min(row2, (bx2 + 1) * d1); // [row1, rowup], [rowbottom, row2]
            int colup = max(col1, by1 * d2 - 1), colbottom = min(col2, (by2 + 1) * d2); // [col1, colup], [colbottom, col2]
            // ^ paddings
            /*cout << row1 << " " << col1 << " " << row2 << " " << col2 << ";"
                << bx1 << " " << bx2 << "  " << by1 << " " << by2 << ";"
                << rowup << " " << rowbottom << " | " << colup << " " << colbottom << endl; */
            // just draw this on paper!
            for (int i=row1; i<=rowup; ++i)
                ans += rowsum(i, colup + 1, col2);
            for (int i=rowbottom; i<=row2; ++i)
                ans += rowsum(i, col1, colbottom - 1);
            for (int j=col1; j<=colup; ++j)
                ans += colsum(j, row1, rowbottom - 1);
            for (int j=colbottom; j<=col2; ++j)
                ans += colsum(j, rowup + 1, row2);
        }
        FINISH:
        return ans;
    }
};
```

1. Reservoir sampling

```
class Solution {
    ListNode head;
    Random rand;

    public Solution(ListNode head) {
        this.head = head;
        rand = new Random();
    }

    public int getRandom() {
        ListNode res = null;
        int count = 0;
        ListNode trav = head;
        while (trav != null) {
            if (rand.nextInt(count + 1) == 0) {
                res = trav;
            }
            count++;
            trav = trav.next;
        }
        return res.val;
    }
}
```

2. 同上

```
/**
 * Definition for singly-linked list.
 * public class ListNode {
 *     int val;
 *     ListNode next;
 *     ListNode(int x) { val = x; }
 * }
 */
class Solution {

    /** @param head The linked list's head.
        Note that the head is guaranteed to be not null, so it contains at least one node. */
    ListNode head;
    Random random;
    public Solution(ListNode head) {
        this.head = head;
        random = new Random();
    }

    /** Returns a random node's value. */
    public int getRandom() {
        int size = 1;
        ListNode node = head;
        int res = node.val;
        while (node.next != null) {
            node = node.next;
            if (random.nextInt(size + 1) == size++) {
                res = node.val;
            }

        }
        return res;
    }
}
```

```

/**
 * Your Solution object will be instantiated and called as suc
h:
 * Solution obj = new Solution(head);
 * int param_1 = obj.getRandom();
 */
```
3. 扔array里面:
```
class Solution {
public:
    /** @param head The linked list's head.
        Note that the head is guaranteed to be not null, so it
contains at least one node. */
    vector<int> cont;
    Solution(ListNode* head) {
        while(head) {
            cont.push_back(head->val);
            head = head->next;
        }
    }

    /** Returns a random node's value. */
    int getRandom() {
        assert(!cont.empty());
        return cont[rand()%cont.size()];
    }
};
```
- [ ] Investigating Reservoir sampling @Zebo L

231

1. bits manipulation

2.以前研究出来一个适用于各种power的解法

```
class Solution {
    public boolean isPowerOfTwo(int n) {
        int max = (int) Math.pow(2, (int) (Math.log(Integer.MA
X_VALUE) / Math.log(2)));
        return n > 0 && max % n == 0;
    }
}
```

3. 同1:

```
class Solution {
public:
    bool isPowerOfTwo(int n) {
        if(!n) return false;
        return (long(n) & (long(n)-1L)) == 0;
    }
};
```

498

1. 观察规律，而不是bfs，bfs太麻烦
2. 找规律

```
class Solution {
    public int[] findDiagonalOrder(int[][] matrix) {
        if (matrix == null || matrix.length == 0) return new i
nt[]{};
        int M = matrix.length;
        int N = matrix[0].length;
        int[] res = new int[M * N];
        int i = 0, j = 0;
        int k = 0;

        while (k < res.length) {
            res[k++] = matrix[i][j];
            // UP
            if ((i + j) % 2 == 0) {
                if (j == N - 1) i++;
                else if (i == 0) j++;
                else {
                    i--;
                    j++;
                }
            }
            // DOWN
            else {
                if (i == M - 1) j++;
                else if (j == 0) i++;
                else {
                    i++;
                    j--;
                }
            }
        }
        return res;
    }
}
```

3. Simulation:

```
class Solution {
public:
    vector<int> findDiagonalOrder(vector<vector<int>>& matrix)
{
        vector<int> ans;
        if(matrix.empty() || matrix[0].empty()) return ans;
        int n = matrix.size(), m = matrix[0].size();
        for(int i=0, j=0, k=0; k<n*m; ++k){
            ans.push_back(matrix[i][j]);
            if((i+j)%2){
                if(!j && i<n-1) ++i;
                else if(i==n-1) ++j;
                else {
                    ++i;
                    --j;
                }
            }
            else{
                if(!i && j<m-1) ++j;
                else if(j==m-1) ++i;
                else{
                    ++j;
                    --i;
                }
            }
        }
        return ans;
    }
};
```

347

1. top k frequent, min heap
2. bucket sort

```
class Solution {
    public List<Integer> topKFrequent(int[] nums, int k) {
        Map<Integer, Integer> map = new HashMap<>();
        List<Integer>[] bucket = new ArrayList[nums.length +
1];
        for (int n : nums) {
            map.put(n, map.getOrDefault(n, 0) + 1);
        }

        for (int key : map.keySet()) {
            int freq = map.get(key);
            if (bucket[freq] == null) {
                bucket[freq] = new ArrayList<>();
            }
            bucket[freq].add(key);
        }

        List<Integer> res = new ArrayList<>();
        for (int i = bucket.length - 1; i >= 0 && res.size() <
k; i--) {
            if (bucket[i] != null) {
                res.addAll(bucket[i]);
            }
        }
        return res;
    }
}
```
3. 一个set就行了吧：
```
class Solution {
    typedef pair<int, int> ii;
public:
    vector<int> topKFrequent(vector<int>& nums, int k) {
        vector<int> ans;
        if(!k) return ans;
        set<ii> S;
        unordered_map<int, int> cnt;
        for(int n: nums) cnt[n]++;
        for(auto p: cnt){
            if(S.size()<k) S.insert(ii(p.second, p.first));
            else{
                if(p.second > S.begin()->first){
                    S.erase(S.begin());
                    S.insert(ii(p.second, p.first));
                }
            }
        }
        for(auto p: S) ans.push_back(p.second);
        return ans;
    }
};
```

- [ ] Investigate approach 2 @Zebo L

143

1. three steps, find mid point, reverse second half, merge 2 linked list
2. 同上

```
/**
 * Definition for singly-linked list.
 * public class ListNode {
 *     int val;
 *     ListNode next;
 *     ListNode(int x) { val = x; }
 * }
 */
class Solution {
    public void reorderList(ListNode head) {
        if (head == null) return;
        ListNode fast = head;
        ListNode slow = head;
        while (fast != null && fast.next != null) {
            fast = fast.next.next;
            slow = slow.next;
        }

        ListNode reverse = reverse(slow.next);
        slow.next = null;
        while (head != null && reverse != null) {
            ListNode n1 = head.next;
            ListNode n2 = reverse.next;
            reverse.next = head.next;
            head.next = reverse;
            head = n1;
            reverse = n2;
        }
    }
    public ListNode reverse(ListNode head) {
        ListNode pre = null;
        ListNode node = head;
        while (node != null) {
            ListNode tmp = node.next;
            node.next = pre;
            pre = node;
            node = tmp;
        }
        return pre;
    }
```

```
}
```

3. 同上，c++ version：

```
class Solution {
    ListNode *revertList(ListNode *head){
        if(!head || !head->next) return head;
        ListNode *p0=head, *p1=head->next;
        while(p1){
            ListNode *tmp=p1->next;
            p1->next = p0;
            p0 = p1;
            p1 = tmp;
        }
        head->next = NULL;
        return p0;
    }
public:
    void reorderList(ListNode* head) {
        if(!head || !head->next) return;
        ListNode *slow=head, *fast=head->next;
        while(fast && fast->next){
            slow = slow->next;
            fast = fast->next->next;
        }
        fast = revertList(slow->next);
        slow->next = NULL;
        ListNode lead(0);
        slow = &lead;
        while(head){
            slow->next = head;
            head = head->next;
            swap(head, fast);
            slow = slow->next;
        }
        head = lead.next;
    }
};
```

527

1. Annoying greedy
2. 非常无聊的一道题

```
class Solution {
    public List<String> wordsAbbreviation(List<String> dict) {
        String[] res = new String[dict.size()];
        int[] prefix = new int[dict.size()];
        Map<String, List<Integer>> map = new HashMap<>();
        for (int i = 0; i < dict.size(); i++) {
            prefix[i] = 1;
            res[i] = getAbbr(dict.get(i), 1);
            List<Integer> l = map.getOrDefault(res[i], new Arr
ayList<>());
            l.add(i);
            map.put(res[i], l);
        }

        for (int i = 0; i < dict.size(); i++) {
            if (map.get(res[i]).size() > 1) {
                List<Integer> list = map.get(res[i]);
                map.remove(res[i]);
                for (int j : list) {
                    prefix[j]++;
                    res[j] = getAbbr(dict.get(j), prefix[j]);
                    List<Integer> l = map.getOrDefault(res[j], 
new ArrayList<>());
                    l.add(j);
                    map.put(res[j], l);
                }
                i--;
            }
        }
        return Arrays.asList(res);
    }

    public String getAbbr(String s, int k) {
        if (k >= s.length() - 2) return s;
        StringBuilder sb = new StringBuilder();
        sb.append(s.substring(0, k));
        sb.append(s.length() - k - 1);
        sb.append(s.charAt(s.length() - 1));
        return sb.length() < s.length() ? sb.toString() : s;
    }
}
```

- [ ] 目前还没做出来 @Zebo L

316

1. 奇怪的描述
2. 每次做这道题都感觉莫名其妙

```
class Solution {
    public String removeDuplicateLetters(String s) {
        int pos = 0;
        int[] chars = new int[26];
        for (char c : s.toCharArray()) {
            chars[c - 'a']++;
        }
        for (int i = 0; i < s.length(); i++) {
            if (s.charAt(i) < s.charAt(pos)) pos = i;
            if (--chars[s.charAt(i) - 'a'] == 0) break;
        }
        return s.length() == 0 ? "" : s.charAt(pos) + removeDu
plicateLetters(s.substring(pos + 1).replaceAll("" + s.charAt(p
os), ""));
    }
}
```

3. 类似stack：

```
class Solution {
    typedef pair<int, char> ic;
public:
    string removeDuplicateLetters(string s) {
        string ans;
        unordered_map<char, int> cnt;
        unordered_set<char> bui;
        for(char c: s) cnt[c]++;
        for(char c: s) {
            cnt[c]--;
            if(!cnt[c]) cnt.erase(c);
            if(bui.count(c)) continue;
            while(!ans.empty() && ans.back() > c && cnt.count
(ans.back())){
                bui.erase(ans.back());
                ans.pop_back();
            }
            ans += c;
            bui.insert(c);
        }
        return ans;
    }
};
```

- [ ] 好好研究上面的recursion 法。

307

1. Stack的最好理解，单调增，如果后面还有就踢掉
2. Binary indexed tree

```
class NumArray {

    int[] tree;
    int m;
    int[] numArr;

    public NumArray(int[] nums) {
        if (nums != null && nums.length != 0) {
            m = nums.length;
            tree = new int[m + 1];
            numArr = new int[m];
            for (int i = 0; i < m; i++) {
                update(i, nums[i]);
            }
        }
    }

    public void update(int i, int val) {
        if (m == 0) return;
        int diff = val - numArr[i];
        numArr[i] = val;
        for (int k = i + 1; k <= m; k += k & (-k)) {
            tree[k] += diff;
        }
    }

    public int sumRange(int i, int j) {
        if (m == 0) return 0;
        return sum(j) - sum(i - 1);
    }

    public int sum(int i) {
        int sum = 0;
        for (int k = i + 1; k > 0; k -= k & (-k)) {
            sum += tree[k];
        }
        return sum;
    }
}

/**
 * Your NumArray object will be instantiated and called as suc
h:
 * NumArray obj = new NumArray(nums);
 * obj.update(i,val);
 * int param_2 = obj.sumRange(i,j);
 */
```

2. 一天之内写了两次 Segment Tree：

```
class NumArray {
    int n, N;
    vector<int> Seg;
public:
    NumArray(vector<int> nums) {
        n = nums.size();
        for(N=1; N<n; N*=2);
        Seg = vector<int>(2*N, 0);
        for(int i=N-1; i<N-1+n; ++i) Seg[i] = nums[i-N+1];
        for(int k=N-1; k; k=(k-1)/2){
            for(int j=k; j<2*k+1; j+=2) Seg[(j-1)/2] = Seg[j]
+ Seg[j+1];
        }
    }

    void update(int i, int val) {
        int delta = val - Seg[i+N-1];
        for(int k=2*(i+N-1)+1; k; k=(k-1)/2) Seg[(k-1)/2] += d
elta;
    }

    int query(int l, int r, int k, int i, int j){
        if(l>=j || i>=r) return 0;
        if(i<=l && j>=r) return Seg[k];
        return query(l, (l+r)/2, 2*k+1, i, j) + query((l+r)/2,
r, 2*k+2, i, j);
    }
    int sumRange(int i, int j) {
        return query(0, N, 0, i, j+1);
    }
};
```

173

1. Stack , push all
2. Exactly the same as preorder traversal:

```
class BSTIterator {
    stack<TreeNode *> S;
public:
    BSTIterator(TreeNode *root) {
        while(root) {
            S.push(root);
            root = root->left;
        }
    }

    /** @return whether we have a next smallest number */
    bool hasNext() {
        return !S.empty();
    }

    /** @return the next smallest number */
    int next() {
        auto ans = S.top(), root=S.top()->right;
        S.pop();
        while(root){
            S.push(root);
            root = root->left;
        }
        return ans->val;
    }
};
```

741

1. 难 3d dp, convert the problem to 2 guys starting from beginning to the end
2. 很有意思的一道题：3D dp，未经优化：

```
class Solution {
    typedef vector<int> vi;
```

```
public:
    int cherryPickup(vector<vector<int>>& grid) {
        int n = grid.size();
        vector<vector<vi>> dp(2*n, vector<vi>(2*n, vi(2*n, -1)));
        dp[0][n][n] = grid[0][0];
        for(int d=1; d<2*n-1; ++d){
            for(int idx1=1; idx1<2*n; ++idx1) if((idx1+d-n)%2==0) {
                int i1 = (idx1+d-n)/2, j1 = (n+d-idx1)/2;
                if(i1<0 || i1>=n || j1<0 || j1>=n || grid[i1][j1]==-1) continue;
                for(int idx2=1; idx2<2*n; ++idx2) if((idx2+d-n)%2==0) {
                    int i2 = (idx2+d-n)/2, j2 = (n+d-idx2)/2;
                    if(i2<0 || i2>=n || j2<0 || j2>=n || grid[i2][j2]==-1) continue;
                    if(i1 && i2 && grid[i1-1][j1]!=-1 && grid[i2-1][j2]!=-1) {
                        dp[d][idx1][idx2] = max(dp[d][idx1][idx2], dp[d-1][i1-1+n-j1][i2-1+n-j2]);
                    }
                    if(i1 && j2 && grid[i1-1][j1]!=-1 && grid[i2][j2-1]!=-1){
                        dp[d][idx1][idx2] = max(dp[d][idx1][idx2], dp[d-1][i1-1+n-j1][i2+1+n-j2]);
                    }
                    if(j1 && i2 && grid[i1][j1-1]!=-1 && grid[i2-1][j2]!=-1){
                        dp[d][idx1][idx2] = max(dp[d][idx1][idx2], dp[d-1][i1+1+n-j1][i2-1+n-j2]);
                    }
                    if(j1 && j2 && grid[i1][j1-1]!=-1 && grid[i2][j2-1]!=-1){
                        dp[d][idx1][idx2] = max(dp[d][idx1][idx2], dp[d-1][i1+1+n-j1][i2+1+n-j2]);
                    }
                    if(dp[d][idx1][idx2] >= 0){
                        dp[d][idx1][idx2] += grid[i1][j1];
                        if(i1!=i2 || j1!=j2) dp[d][idx1][idx2] += grid[i2][j2];
                    }
                }
            }
        }
        return int(dp[2*n-2][n][n]>0) * dp[2*n-2][n][n];
    }
};
```

- [ ] 优化上述算法，(iteration 上下界，每次跳跃2，而不是1), Check Leetcode Discussion @Zebo L

801

1. 注意细节，主要思路是把原来的数组截成一段一段的，不同的段之间互不相干：
    a. 写的时候容易出错，要注意

```
class Solution {
public:
    int minSwap(vector<int>& A, vector<int>& B) {
        int ans = 0, n = A.size();
        int i=0, j=0;
        while(i<n-1){
            while(i<n-1 && max(A[i], B[i]) < min(A[i+1], B[i+1])) ++i;
            int j = i+1, cnt = 1, bias = int(A[i]>B[i]);
            while(j<n && min(A[j], B[j]) <= max(A[j-1], B[j-
1])){
                if(A[j] > B[j]) ++bias;
                ++cnt;
                ++j;
            }
            i = j;
            ans += min(bias, cnt-bias);
        }
        return ans;
    }
};
```

- [ ] Review this @Zebo L and investigate the following :

2. dp: (Leetcode discussion 真是互相抄，全是一种方法)

```
class Solution {
public:
    int minSwap(vector<int>& A, vector<int>& B) {
        int s = 1, ns = 0; // s -> swap, ns -> not swap
        int ps, pns;       // ps -> previous swap, pns -> prev
ious not swap

        for (int i = 1; i < A.size(); ++i) {
            ps = s; pns = ns;
            s = INT_MAX; ns = INT_MAX;

            if (A[i] > A[i-1] && B[i] > B[i-1]) {
                // We must swap both or don't swap both
                s = ps + 1;
                ns = pns;
            }

            if (A[i] > B[i-1] && B[i] > A[i-1]) {
                // We must swap only one
                s = min(s, pns + 1);
                ns = min(ns, ps);
            }
            //cout << "s:" << s << " ns:" << ns << endl;
        }
        return (min(s, ns));
    }
};
```