August 27, 2018 题目：587,838,815,270,839,416,103,321,449,97,654,153,646,764,723

587

1. 传说中烦得1B的题：

```
typedef pair<int, int> ii;
double delta = 1.E-7;
bool eq(double x, double y) {
    return abs(x-y) < delta;
}
bool lt(double x, double y) {
    return x + delta < y;
}
struct Le{
    int x, y;
    Le(int x_, int y_): x(x_), y(y_) {}
    void set(int x_, int y_){
        x = x_;
        y = y_;
    }
    bool cmp(const ii&a, const ii&b){
        double n1 = (a.first*x + a.second*y);
        double n2 = (b.first*x + b.second*y);
        double d1 = sqrt(a.first*a.first + a.second * a.secon
d);
        double d2 = sqrt(b.first*b.first + b.second * b.secon
d);
        if(eq(n1 * d2, n2 * d1)) return lt(d1, d2);
```

```
        return lt(n2 * d1, n1 * d2);
    }
};
class Solution {
    static bool key(const Point&a, const Point&b){
        if(a.y == b.y) return a.x < b.x;
        return a.y < b.y;
    }
public:
    vector<Point> outerTrees(vector<Point>& pts) {
        if(pts.size()<=3) return pts;
        sort(pts.begin(), pts.end(), key);
        vector<Point> ans;
        set<ii> res;
        ii current = ii(pts[0].x, pts[0].y);
        Le le(1, 0);
        while(!res.count(current)){
            res.insert(current);
            ans.push_back(Point(current.first, current.secon
d));
            ii tmp = ii(pts[0].x, pts[0].y);
            if(tmp == current) tmp = ii(pts[1].x, pts[1].y);
            for(auto p: pts) if(p.x != current.first || p.y !=
current.second) {
                if(le.cmp(ii(p.x-current.first, p.y-current.se
cond), ii(tmp.first-current.first, tmp.second - current.secon
d))){
                    tmp = ii(p.x, p.y);
                }
            }
```

```
            le.set(tmp.first - current.first, tmp.second - cur
rent.second);
            current = tmp;
        }
        return ans;
    }
};
```

2. 太难啦，带名字的Algorithm。。。

838

1. 前两天刚做过，标记每个 `L` `R` 的位置：然后process intervals

```
class Solution {
public:
    string pushDominoes(string D) {
        D = "L" + D + "R";
        string ref;
        vector<int> pos;
        for(int i=0; i<D.size(); ++i) if(D[i] != '.'){
            pos.push_back(i);
            ref += D[i];
        }
        for(int i=1; i<pos.size(); ++i){
            if(ref[i]==ref[i-1]) for(int j=pos[i-1]+1; j<pos
[i]; ++j) D[j] = ref[i];
            else if(ref[i]=='L' && ref[i-1]=='R'){
                for(int l=pos[i-1]+1, r=pos[i]-1; l<r; ++l,--
r){
                    D[l] = ref[i-1];
                    D[r] = ref[i];
                }
            }
```

```
        }
        return D.substr(1, D.size()-2);
    }
};
```

815

1. 第二次做还错了，两层filtering，否则会超时：

```
class Solution {
    typedef pair<int, int> ii;
public:
    int numBusesToDestination(vector<vector<int>>& routes, int s, int t) {
        unordered_map<int, vector<int>> B;
        for(int i=0; i<routes.size(); ++i) for(int k: routes[i]) B[k].push_back(i);
        unordered_set<int> S{s}, SB;
        queue<ii> Q;
        Q.push(ii(s, 0));
        while(!Q.empty()){
            int j = Q.front().first, k = Q.front().second;
            if(j == t) return k;
            Q.pop();
            for(int b: B[j]) if(!SB.count(b)) {
                SB.insert(b);
                for(int x: routes[b]) if(!S.count(x)){
                    S.insert(x);
                    Q.push(ii(x, k+1));
                }
            }
        }
        return -1;
```

```
    }
};
```

270

1. recrusion

```
class Solution {
    public int closestValue(TreeNode root, double target) {
        int val = root.val;
        TreeNode next = val > target ? root.left : root.right;
        if (next == null) return val;
        int n = closestValue(next, target);
        return Math.abs(target - val) > Math.abs(target - n) ?
n : val;
    }
}
```

2. Leetcode 很喜欢出这种overflow的无聊case：

```
class Solution {
public:
    int closestValue(TreeNode* root, double target) {
        long l = 4*long(INT_MIN), r = 4*long(INT_MAX);
        while(root){
            if(root->val > target) {
                r = root->val;
                root = root->left;
            }
            else{
                l = root->val;
                root = root->right;
            }
        }
        return (target-l < r-target? l: r);
```

```
    }
};
```

839

1. 如此蹩脚的BFS居然能过：(并查集似乎快一点)

```
class Solution {
    bool connect(string a, string b){
        for(int i=0, cnt=0; i<a.size(); ++i) if(a[i] != b[i])
{
            ++cnt;
            if(cnt>2) return false;
        }
        return true;
    }
public:
    int numSimilarGroups(vector<string>& A) {
        unordered_set<string> S(A.begin(), A.end());
        int ans = 0;
        while(!S.empty()){
            queue<string> Q;
            Q.push(*S.begin());
            S.erase(S.begin());
            while(!Q.empty()){
                for(auto it = S.begin(); it!=S.end();){
                    if(connect(Q.front(), *it)){
                        Q.push(*it);
                        it = S.erase(it);
                    }
                    else ++it;
                }
                Q.pop();
```

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

416

1. dp

```
class Solution {
    public boolean canPartition(int[] nums) {
        int sum = 0;
        for (int i = 0; i < nums.length; i++) {
            sum += nums[i];
        }
        if (sum % 2 != 0) return false;
        sum /= 2;
        boolean[] dp = new boolean[sum + 1];
        dp[0] = true;
        for (int i = 0; i < nums.length; i++) {
            for (int j = sum; j >= nums[i]; j--) {
                dp[j] = dp[j] | dp[j - nums[i]];
            }
        }
        return dp[sum];
    }
}
```

2. very slow method：

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

```
        if(nums.size() <= 1) return false;
        unordered_set<int> dp;
        int sum = 0;
        for(int k: nums) sum += k;
        if(sum % 2) return false;
        for(int k: nums){
            unordered_set<int> tmp;
            tmp.insert(k);
            for(int j: dp) tmp.insert(k+j);
            for(int j: tmp) if(!dp.count(j)) dp.insert(j);
            if(dp.count(sum/2)) return true;
        }
        return false;
    }
};
```

3. 受楼上启发：

```
class Solution {
public:
    bool canPartition(vector<int>& nums) {
        if(nums.size() <= 1) return false;
        int sum = 0, tmp = 0;
        for(int k: nums) sum += k;
        if(sum % 2) return false;
        vector<bool> dp(sum+1, false);
        dp[0] = true;
        for(int k: nums){
            tmp += k;
            for(int i=tmp; i>=k; --i) if(dp[i-k]) dp[i]=true;
            if(dp[sum/2]) return true;
        }
```

```
        return false;
    }
};
```

4. 慢的主要原因是用 `map`:

```
class Solution {
public:
    bool canPartition(vector<int>& nums) {
        if(nums.size() <= 1) return false;
        set<int, greater<int>> dp;
        int sum = 0;
        for(int k: nums) sum += k;
        if(sum % 2) return false;
        for(int k: nums){
            for(int j: dp) dp.insert(k+j);
            dp.insert(k);
            if(dp.count(sum/2)) return true;
        }
        return false;
    }
};
```

103

1. backtracking

```
class Solution {
    public List<List<Integer>> zigzagLevelOrder(TreeNode root)
{
        List<List<Integer>> res = new ArrayList<>();
        helper(res, root, 0);
        return res;
    }
```

```
    public void helper(List<List<Integer>> res, TreeNode root,
int level) {
        if (root == null) return;
        if (res.size() <= level) res.add(new ArrayList<>());
        if (level % 2 == 0) {
            res.get(level).add(root.val);
        } else {
            res.get(level).add(0, root.val);
        }
        helper(res, root.left, level + 1);
        helper(res, root.right, level + 1);
    }
}
```

2. 加一步额外的reversion：

```
class Solution {
    void dfs(vector<vector<int>> &ans, TreeNode *root, int lev
el){
        if(!root) return;
        if(level >= ans.size()) ans.resize(level+1);
        ans[level].push_back(root->val);
        dfs(ans, root->left, level+1);
        dfs(ans, root->right, level+1);
    }
public:
    vector<vector<int>> zigzagLevelOrder(TreeNode* root) {
        vector<vector<int>> ans;
        dfs(ans, root, 0);
        for(int i=1; i<ans.size(); i+=2) reverse(ans[i].begin
(), ans[i].end());
        return ans;
    }
```

```
};
```

321

1. 一个非常蠢的brute force：

```
class Solution {
    #define CI(c) int((c) - '0')
    #define IC(i) char((i) + '0')
    string getMax(string s, int l){
        while(s.size() > l){
            int i = 0;
            while(i<s.size()-1 && s[i] >= s[i+1]) ++i;
            s = s.substr(0, i) + s.substr(i+1);
        }
        return s;
    }
    string maxComb(string a, string b){
        string c;
        while(!a.empty() || !b.empty()){
            if(a > b) {
                c.push_back(a[0]);
                a = a.substr(1);
            }
            else{
                c.push_back(b[0]);
                b = b.substr(1);
            }
        }
        return c;
    }
public:
```

```
    vector<int> maxNumber(vector<int>& nums1, vector<int>& nums2, int k) {
        int n = nums1.size(), m = nums2.size();
        string A, B, res;
        vector<int> ans;
        for(int k: nums1) A.push_back(IC(k));
        for(int k: nums2) B.push_back(IC(k));
        for(int l=max(0, k-m); l<=min(n, k); ++l) res = max(res, maxComb(getMax(A, l), getMax(B, k-l)));
        for(char c: res) ans.push_back(CI(c));
        return ans;
    }
};
```

2. 思路不变，改进了以下：

```
class Solution {
    typedef vector<int> vi;
    vi getMax(vi A, int l){
        int n = A.size();
        vi ans;
        for(int k: A){
            while(!ans.empty() && k>ans.back() && n>l){
                ans.pop_back();
                --n;
            }
            ans.push_back(k);
        }
        while(ans.size()>l) ans.pop_back();
        return ans;
    }
    bool cmp(vi &A, int i, vi&B, int j){
        while(i<A.size() && j<B.size()){
```

```
            if(A[i] > B[j]) return true;
            if(A[i] < B[j]) return false;
            ++i;
            ++j;
        }
        return j==B.size();
    }
    vi maxComb(vi A, vi B){
        vi C;
        for(int i=0, j=0; i<A.size()||j<B.size(); ){
            if(cmp(A, i, B, j)) C.push_back(A[i++]);
            else C.push_back(B[j++]);
        }
        return C;
    }
public:
    vector<int> maxNumber(vector<int>& A, vector<int>& B, int
k) {
        int n = A.size(), m = B.size();
        vector<int> ans;
        for(int l=max(0, k-m); l<=min(n, k); ++l) ans=max(ans,
maxComb(getMax(A, l), getMax(B, k-l)));
        return ans;
    }
};
```

449

1. 跟297的区别是啥？

```
public class Codec {

    // Encodes a tree to a single string.
```

```
    public String serialize(TreeNode root) {
        if (root == null) return "#,";
        return root.val + "," + serialize(root.left) + serialize(root.right);
    }

    // Decodes your encoded data to tree.
    public TreeNode deserialize(String data) {
        String[] datas = data.split(",");
        Queue<String> q = new LinkedList<>();
        q.addAll(Arrays.asList(datas));
        return helper(q);
    }
    private TreeNode helper(Queue<String> q) {
        String cur = q.poll();
        if (cur.equals("#")) return null;
        else {
            int val = Integer.parseInt(cur);
            TreeNode node = new TreeNode(val);
            node.left = helper(q);
            node.right = helper(q);
            return node;
        }
    }
}
```

2. 同上:

```
class Codec {
public:
    string serialize(TreeNode* root) {
        if(!root) return "#";
```

```
        return to_string(root->val) + "(" + serialize(root->le
ft) + ")(" + serialize(root->right) + ")";
    }
    TreeNode* deserialize(string data) {
        if(data == "#") return NULL;
        int val = stoi(data);
        TreeNode *root = new TreeNode(val);
        auto k = (int)data.find('(')+1;
        int j = int(k), cnt = 1;
        while(cnt){
            if(data[j]=='(') ++cnt;
            if(data[j]==')') --cnt;
            ++j;
        }
        root->left = deserialize(data.substr(k, j-1-k));
        k = ++j;
        cnt = 1;
        while(cnt){
            if(data[j]=='(') ++cnt;
            if(data[j]==')') --cnt;
            ++j;
        }
        root->right = deserialize(data.substr(k, j-1-k));
        return root;
    }
};
```

97

1. dp

```
class Solution {
```

```
    public boolean isInterleave(String s1, String s2, String s3) {
        if (s1.length() + s2.length() != s3.length()) return false;
        boolean[][] dp = new boolean[s1.length() + 1][s2.length() + 1];
        for (int i = 0; i <= s1.length(); i++) {
            for (int j = 0; j <= s2.length(); j++) {
                if (i == 0 && j == 0) {
                    dp[i][j] = true;
                } else if (i == 0) {
                    dp[i][j] = s2.charAt(j - 1) == s3.charAt(i + j - 1) && dp[i][j - 1];
                } else if (j == 0) {
                    dp[i][j] = s1.charAt(i - 1) == s3.charAt(i + j - 1) && dp[i - 1][j];
                } else {
                    dp[i][j] = s1.charAt(i - 1) == s3.charAt(i + j - 1) && dp[i - 1][j] || s2.charAt(j - 1) == s3.charAt(i + j - 1) && dp[i][j - 1];
                }
            }
        }
        return dp[s1.length()][s2.length()];
    }
}
```

2. 试了下贪心，发现不对，只能老老实实dp：注意长度不等的情况

```
class Solution {
public:
    bool isInterleave(string s1, string s2, string s3) {
        int n = s1.size(), m = s2.size();
        if(s3.size() != n+m) return false;
```

```
        vector<vector<int>> dp(n+1, vector<int>(m+1, false));
        dp[0][0] = true;
        for(int i=1; i<=n && s1[i-1]==s3[i-1]; ++i) dp[i][0] =
true;
        for(int j=1; j<=m && s2[j-1]==s3[j-1]; ++j) dp[0][j] =
true;
        for(int i=1; i<=n; ++i) for(int j=1; j<=m; ++j){
            dp[i][j] = (s3[i+j-1] == s1[i-1] && dp[i-1][j]) ||
(s3[i+j-1]==s2[j-1] && dp[i][j-1]);
        }
        return dp[n][m];
    }
};
```

654

1. recursion

```
class Solution {
    public TreeNode constructMaximumBinaryTree(int[] nums) {
        return helper(nums, 0, nums.length - 1);
    }

    public TreeNode helper(int[] nums, int start, int end) {
        if (start > end) return null;
        int max = nums[start];
        int index = start;
        for (int i = start + 1; i <= end; i++) {
            if (max < nums[i]) {
                max = nums[i];
                index = i;
            }
        }
```

```
        TreeNode root = new TreeNode(max);
        root.left = helper(nums, start, index - 1);
        root.right = helper(nums, index + 1, end);
        return root;
    }
}
```

2. `O(n)` Solution beats `12%` !!!!!

```
class Solution {
public:
    TreeNode* constructMaximumBinaryTree(vector<int>& nums) {
        if(nums.empty()) return NULL;
        TreeNode *root = new TreeNode(nums[0]);
        for(int i=1; i<nums.size(); ++i){
            TreeNode *p = new TreeNode (nums[i]);
            if(nums[i]>root->val){
                p->left = root;
                root = p;
            }
            else{
                TreeNode *q = root;
                while(q->right && q->right->val > nums[i]) q =
q->right;
                p->left = q->right;
                q->right = p;
            }
        }
        return root;
    }
};
```

153

1. 一年前相同的做法只beat了6%。。这次100%。。不知道这一年发生了什么

```
class Solution {
    public int findMin(int[] nums) {
        int i = 0, j = nums.length - 1;
        while (i < j) {
            if (nums[i] < nums[j]) return nums[i];
            int mid = i + (j - i) / 2;
            if (nums[mid] < nums[j]) {
                j = mid;
            } else if (nums[mid] >= nums[i]) {
                i = mid + 1;
            }
        }
        return nums[i];
    }
}
```

2. 这题除了二分还有其他方法么，为什么只 `Beat 12%`:
    - (重run了一遍beat `100%` 🤣)

```
class Solution {
public:
    int findMin(vector<int>& nums) {
        int n = nums.size();
        if(n==1 || nums[0] < nums[n-1]) return nums[0];
        int l = 0, r = n;
        while(l<r-1){
            int c = (l+r)/2;
            if(nums[c] > nums[0]) l = c;
            else r = c;
        }
```

```
        return nums[r];
    }
};
```

646

1. 贪心，难度何在？

```
class Solution {
    static bool cmp(vector<int> &a, vector<int> &b){
        if(a[1] == b[1]) return a[0]<b[0];
        return a[1] < b[1];
    }
public:
    int findLongestChain(vector<vector<int>>& pairs) {
        sort(pairs.begin(), pairs.end(), cmp);
        int ans = 0, head = INT_MIN;
        for(auto v: pairs) if(v[0] > head){
            ++ans;
            head = v[1];
        }
        return ans;
    }
};
```

764

1. $O(n^2)$ Solution: 可以更快，这题蛮无聊的。上下左右做4个dp

```
class Solution {
public:
    int orderOfLargestPlusSign(int N, vector<vector<int>>& min
es) {
        vector<set<int>> V(N, set<int>{-1, N}), H(N, set<int>
{-1, N});
```

```
        for(auto p: mines) {
            H[p[0]].insert(p[1]);
            V[p[1]].insert(p[0]);
        }
        int ans = 0;
        for(int i=0; i<N; ++i){
            auto l = H[i].begin(), r = ++H[i].begin();
            while(r!=H[i].end()){
                for(int m=*l+1; m<*r; ++m){
                    auto u = --V[m].lower_bound(i), d = V[m].l
ower_bound(i);
                    ans = max(ans, min(min(i-*u, *d-i), min(m-
*l, *r-m)));
                }
                ++l;
                ++r;
            }
        }
        return ans;
    }
};
```

723

1. 非常直接的逻辑：

```
class Solution {
    const int M = 40000;
    bool process(vector<vector<int>>& B){
        bool ok = false;
        int n = B.size(), m = B[0].size();
        for(int i=0; i<n; ++i) for(int j=1, cnt=1; j<=m; ++j){
            if(j<m && B[i][j]%M == B[i][j-1]%M) ++cnt;
```

```
            else{
                if(cnt >= 3 && B[i][j-1]){
                    ok = true;
                    int k = j-1, x = B[i][j-1]%M;
                    while(k>=0 && B[i][k]%M==x) {
                        B[i][k] += M;
                        --k;
                    }
                }
                cnt = 1;
            }
        }
        for(int j=0; j<m; ++j) for(int i=1, cnt=1; i<=n; ++i){
            if(i<n && B[i][j]%M == B[i-1][j]%M) ++cnt;
            else{
                if(cnt>=3 && B[i-1][j]){
                    ok = true;
                    int k = i-1, x = B[i-1][j]%M;
                    while(k>=0 && B[k][j]%M==x) {
                        B[k][j] += M;
                        --k;
                    }
                }
                cnt = 1;
            }
        }
        for(int i=0; i<n; ++i) for(int j=0; j<m; ++j) if(B[i]
[j] >= M) B[i][j] = 0;
        if(ok){
            for(int j=0; j<m; ++j){
```

```
                for(int i=n-1, k=n-1; i>=0; --i){
                    if(B[i][j]){
                        if(i!=k) swap(B[i][j], B[k][j]);
                        --k;
                    }
                }
            }
        }
        return ok;
    }
public:
    vector<vector<int>> candyCrush(vector<vector<int>>& B) {
        if(B.empty() || B[0].empty()) return B;
        do{}while(process(B));
        return B;
    }
};
```

670

无难度：

```
class Solution {
public:
    int maximumSwap(int num) {
        if(num<10) return num;
        vector<int> dig, max_dig;
        while(num){
            dig.push_back(num%10);
            num /= 10;
        }
        for(int i=0, m=dig[0]; i<dig.size(); ++i){
```

```
            m = max(dig[i], m);
            max_dig.push_back(m);
        }
        int i = dig.size()-1, j=0, ans=0;
        while(i>=0 && dig[i]==max_dig[i]) ans = ans*10 + dig[i
--];
        if(i<0) return ans;
        while(j<i && dig[j] != max_dig[i]) ++j;
        swap(dig[i], dig[j]);
        while(i>=0) ans = ans*10 + dig[i--];
        return ans;
    }
};
```