September 13, 2018 853,66,377,687,737,640,699,88,662,88,136,194

853

1.

```
class Solution {
    public int carFleet(int target, int[] position, int[] speed) {
        TreeMap<Integer, Double> map = new TreeMap<>();
        for (int i = 0; i < position.length; i++) {
            map.put(-position[i], (double) (target - position[i]) / speed[i]);
        }

        double cur = 0;
        int res = 0;
        for (double d : map.values()) {
            if (d > cur) {
                cur = d;
                res++;
            }
        }
        return res;
    }
}
```

2. Check 谁先到站即可：

```
class Solution {
    typedef pair<int, int> ii;
```

```
public:
    int carFleet(int target, vector<int>& position, vector<int
>& speed) {
        vector<ii> cars;
        for(int i=0; i<position.size(); ++i) cars.push_back(ii
(position[i], speed[i]));
        sort(cars.begin(), cars.end());
        double current = 0.;
        int ans = 0;
        for(int j=position.size()-1; j>=0; --j){
            if(double(target-cars[j].first)/cars[j].second > c
urrent){
                ++ans;
                current = double(target-cars[j].first)/cars
[j].second;
            }
        }
        return ans;
    }
};
```

66

1. Easy:

```
class Solution {
public:
    vector<int> plusOne(vector<int>& digits) {
        vector<int> ans;
        for(int i=digits.size()-1, cur=1; i>=0||cur; --i){
            cur += (i>=0? digits[i]:0);
            ans.push_back(cur%10);
            cur /= 10;
```

```
        }
        reverse(ans.begin(), ans.end());
        return ans;
    }
};
```

377

1. DP

```
class Solution {
    public int combinationSum4(int[] nums, int target) {
        int[] dp = new int[target + 1];
        dp[0] = 1;
        for (int i = 1; i <= target; i++) {
            for (int n : nums) {
                if (i - n >= 0) {
                    dp[i] += dp[i - n];
                }
            }
        }
        return dp[target];
    }
}6
```

2. 注意跟次序有关：

```
class Solution {
public:
    int combinationSum4(vector<int>& nums, int target) {
        vector<int> dp(target+1, 0);
        dp[0] = 1;
        for(int i=0; i<target; ++i) for(int n: nums) if(i+n<=t
arget) dp[i+n] += dp[i];
        return dp[target];
```

```
    }
};
```

687

1. 简答dfs：

```
class Solution {
    int res;
    int dfs(TreeNode *root, int val){
        if(!root) return 0;
        int l = dfs(root->left, root->val);
        int r = dfs(root->right, root->val);
        res = max(res, l+r);
        if(root->val == val) return 1 + max(l, r);
        else return 0;
    }
public:
    int longestUnivaluePath(TreeNode* root) {
        res = 0;
        dfs(root, 0);
        return res;
    }
};
```

737

1. 之前这方法怎么过的：

```
class Solution {
public:
    bool areSentencesSimilarTwo(vector<string>& words1, vector
<string>& words2, vector<pair<string, string>> pairs) {
        if(words1.size() != words2.size()) return false;
        unordered_map<string, int> W1, W2;
```

```
        for(string s: words1) ++W1[s];
        for(string s: words2) ++W2[s];
        unordered_map<string, vector<string>> E;
        for(auto p: pairs){
            E[p.first].push_back(p.second);
            E[p.second].push_back(p.first);
        }
        while(!W1.empty()){
            unordered_set<string> RM{W1.begin()->first};
            queue<string> Q;
            Q.push(W1.begin()->first);
            while(!Q.empty()){
                string s = Q.front();
                Q.pop();
                for(string k: E[s]) if(!RM.count(k)){
                    RM.insert(k);
                    Q.push(k);
                }
            }
            int cnt1 = 0, cnt2 = 0;
            for(auto it=W1.begin(); it!=W1.end(); ){
                if(RM.count(it->first)) {
                    cnt1 += it->second;
                    it = W1.erase(it);
                }
                else ++it;
            }
            for(auto it=W2.begin(); it!=W2.end(); ){
                if(RM.count(it->first)) {
                    cnt2 += it->second;
```

```
                    it = W2.erase(it);
                }
                else ++it;
            }
            if(cnt1 != cnt2) return false;
        }
        return W2.empty();
    }
};
```

2. 上面解法真是过的莫名其妙：

```
class Solution {
    unordered_map<string, string> parent;
    string findRoot(string s){
        if(s == parent[s]) return s;
        return parent[s] = findRoot(parent[s]);
    }
public:
    bool areSentencesSimilarTwo(vector<string>& words1, vector
<string>& words2, vector<pair<string, string>> pairs) {
        if(words1.size() != words2.size()) return false;
        for(auto p: pairs) if(!parent.count(p.first) || !paren
t.count(p.second) || findRoot(p.first)!=findRoot(p.second)) {
            if(!parent.count(p.first)) parent[p.first] = p.fir
st;
            if(!parent.count(p.second)) parent[p.second] = p.s
econd;
            parent[findRoot(p.first)] = findRoot(p.second);
        }
        for(int i=0; i<words1.size(); ++i){
           if(words1[i]!=words2[i]&&(
```

```
                !parent.count(words1[i]) || !parent.count(words2
[i]) || findRoot(words1[i])!=findRoot(words2[i]))) return fals
e;
        }
        return true;
    }
};
```

640

1. Python is easier:

```
import re
class Solution(object):
    def solveEquation(self, equation):
        left, right = equation.split('=')
        if left[0] not in '+-':
            left = '+' + left
        if right[0] not in '+-':
            right = '+' + right
        lval = [1 if c=='+' else -1 for c in left if c in '+-
']
        lqty = [str(s) for s in re.split('[+-]', left) if s]
        rval = [1 if c=='+' else -1 for c in right if c in '+-
']
        rqty = [str(s) for s in re.split('[+-]', right) if s]
        getCoeff = lambda sx: int(sx[:-1]) if len(sx) > 1 else
1
        lcof = sum([x*getCoeff(y) for x, y in zip(lval, lqty)
if 'x' in y])
        rcof = sum([x*getCoeff(y) for x, y in zip(rval, rqty)
if 'x' in y])
        lnum = sum([x*int(y) for x, y in zip(lval, lqty) if
'x' not in y])
```

```
        rnum = sum([x*int(y) for x, y in zip(rval, rqty) if
'x' not in y])
        if lcof == rcof:
            if lnum == rnum:
                return 'Infinite solutions'
            else:
                return 'No solution'
        return 'x=' + str((rnum - lnum)/(lcof - rcof))
```

699

1. Maintain 区间 :

```
class Solution {
public:
    vector<int> fallingSquares(vector<pair<int, int>>& positio
ns) {
        map<int, int> H;
        H[-1] = 0;
        H[1E9] = 0;
        vector<int> ans;
        int res = 0;
        for(auto p: positions){
            int start = p.first, end = p.first + p.second, edg
e = p.second, mH = 0;
            auto it1 = --H.upper_bound(start), it2 = H.lower_b
ound(end);
            for(auto it=it1; it!=it2; ++it) {
                mH = max(mH, it->second);
            }
            auto it3 = H.lower_bound(start), it4 = --H.lower_b
ound(end);
            auto y = it4->second;
```

```
            for(auto it=it3; it!=it2; it=H.erase(it));
            H[start] = mH + edge;
            if(!H.count(end)) H[end] = y;
            res = max(res, mH + edge);
            ans.push_back(res);
        }
        return ans;
    }
};
```

2. interval check overlap，然后往上叠加

```
class Solution {
    class Interval {
        int height, start, end;
        public Interval(int start, int end, int height) {
            this.height = height;
            this.end = end;
            this.start = start;
        }
    }
    public List<Integer> fallingSquares(int[][] positions) {
        List<Interval> intervals = new ArrayList<>();
        List<Integer> res = new ArrayList<>();
        int height = 0;
        for (int[] p : positions) {
            Interval cur = new Interval(p[0], p[1] + p[0] - 1,
p[1]);
            height = Math.max(height, getHeight(intervals, cu
r));
            res.add(height);
        }
        return res;
```

```
    }

    public int getHeight(List<Interval> list, Interval cur) {
        int pre = 0;
        for (Interval i : list) {
            if (i.start > cur.end) continue;
            if (cur.start > i.end) continue;
            pre = Math.max(pre, i.height);
        }
        cur.height += pre;
        list.add(cur);
        return cur.height;
    }
}
```

88

1. trick: 从后往前

```
class Solution {
    public void merge(int[] nums1, int m, int[] nums2, int n)
{
        int i = m - 1, j = n - 1;
        int k = m + n - 1;
        while (i >= 0 && j >= 0) {
            if (nums1[i] > nums2[j]) {
                nums1[k--] = nums1[i--];
            } else {
                nums1[k--] = nums2[j--];
            }
        }
        while (j >= 0) {
            nums1[k--] = nums2[j--];
```

```
        }
    }
}
```

2. 同上：

```
class Solution {
public:
    void merge(vector<int>& nums1, int m, vector<int>& nums2, int n) {
        for(int i=m-1, j=n-1, k=n+m-1; k>=0; --k){
            if(i<0 || (j>=0 && nums2[j] > nums1[i])) nums1[k] = nums2[j--];
            else nums1[k] = nums1[i--];
        }
    }
};
```

662

1. 这题面试一定要问清楚：

```
class Solution {
    vector<int> L, R;
    void dfs(TreeNode *root, int lvl, int pos){
        if(!root) return;
        if(L.size() <= lvl) {
            L.push_back(INT_MAX);
            R.push_back(INT_MIN);
        }
        L[lvl] = min(pos, L[lvl]);
        R[lvl] = max(pos+1, L[lvl]);
        dfs(root->left, lvl+1, pos*2);
        dfs(root->right, lvl+1, pos*2+1);
    }
```

```
    int dep(TreeNode *root){
        if(!root) return 0;
        return 1+max(dep(root->left), dep(root->right));
    }
public:
    int widthOfBinaryTree(TreeNode* root) {
        int depth = dep(root);
        if(depth <= 1) return depth;
        dfs(root, 0, 0);
        int ans = 0;
        for(int i=0; i<L.size(); ++i) ans = max(ans, R[i] - L
[i]);
        for(int i=0; i<L.size(); ++i) cout<<L[i]<< ' ' << R[i]
<<endl;
        return ans;
    }
};
```

2. recursion, 记最左边的node的id

```
class Solution {
    public int widthOfBinaryTree(TreeNode root) {
        return helper(root, 1, 0, new ArrayList<>());
    }

    public int helper(TreeNode root, int id, int depth, List<I
nteger> nodes) {
        if (root == null) {
            return 0;
        }
        if (nodes.size() <= depth) {
            nodes.add(id);
        }
```

```
        return Math.max(id + 1 - nodes.get(depth), Math.max(he
lper(root.left, id * 2, depth + 1, nodes), helper(root.right,
id * 2 + 1, depth + 1, nodes)));
    }
}
```

136

1. Short is Beauty 系列：

```
class Solution(object):
    def singleNumber(self, nums):
        return reduce(lambda x, y: x^y, nums)
```

194

FUCKING BASH