August 20, 2018 题目：379,840,471,501,374,798,324,323,40,681,483,70,626,585,253

379

1. 刚做过：

```
    int cap, n;
    unordered_set<int> re;
public:
    PhoneDirectory(int maxNumbers): cap(maxNumbers), n(0) {}
    int get() {
        if(n==cap && re.empty()) return -1;
        if(!re.empty()){
            int k = *re.begin();
            re.erase(re.begin());
            return k;
        }
        ++n;
        return n-1;
    }
    bool check(int number) { return number>=n || re.count(number); }
    void release(int number) { if(!check(number)) re.insert(number); }
};
```

2. Set

```
class PhoneDirectory {


    /** Initialize your data structure here
```

```
        @param maxNumbers - The maximum numbers that can be st
ored in the phone directory. */
    Set<Integer> set;
    public PhoneDirectory(int maxNumbers) {
        set = new HashSet<>();
        for (int i = 0; i < maxNumbers; i++) {
            set.add(i);
        }
    }

    /** Provide a number which is not assigned to anyone.
        @return - Return an available number. Return -1 if non
e is available. */
    public int get() {
        if (set.isEmpty()) return -1;
        int k = set.iterator().next();
        set.remove(k);
        return k;
    }

    /** Check if a number is available or not. */
    public boolean check(int number) {
        return set.contains(number);
    }

    /** Recycle or release a number. */
    public void release(int number) {
        set.add(number);
    }
}
```

```
/**
 * Your PhoneDirectory object will be instantiated and called
as such:
 * PhoneDirectory obj = new PhoneDirectory(maxNumbers);
 * int param_1 = obj.get();
 * boolean param_2 = obj.check(number);
 * obj.release(number);
 */
```

840

1. Pure Brute Force :

```
class Solution {
public:
    int numMagicSquaresInside(vector<vector<int>>& G) {
        int cnt = 0, n = G.size(), m = G[0].size();
        for(int i=0; i<n-2; ++i) for(int j=0; j<m-2; ++j){
            bool ok = true;
            vector<int> ref(9, 0);
            for(int k=0; k<3&&ok; ++k) for(int l=0;l<3&&ok;++
l) {
                if(G[i+k][j+l]>9 || G[i+k][j+l]<1) ok = false;
                else{
                    ref[G[i+k][j+l]-1]++;
                    if(ref[G[i+k][j+l]-1] > 1) ok = false;
                }
            }
            for(int k=0; k<3 && ok; ++k) if(G[i+k][j]+G[i+k][j
+1]+G[i+k][j+2]!=15 || G[i][j+k]+G[i+1][j+k]+G[i+2][j+k]!=15)
ok=false;
```

```
            if(G[i][j]+G[i+1][j+1]+G[i+2][j+2]!=15 || G[i][j+
2]+G[i+1][j+1]+G[i+2][j]!=15) ok = false;
            cnt += ok;
        }
        return cnt;
    }
};
```

2. brute force

```
class Solution {
    public int numMagicSquaresInside(int[][] grid) {
        int res = 0;
        for (int i = 0; i < grid.length - 2; i++) {
            for (int j = 0; j < grid[0].length - 2; j++) {
                if (helper(grid, i, j)) {
                    res++;
                }
            }
        }
        return res;
    }

    public boolean helper(int[][] grid, int x, int y) {
        int[] valid = new int[10];
        for (int i = x; i < x + 3; i++) {
            for (int j = y; j < y + 3; j++) {
                if (grid[i][j] < 1 || grid[i][j] > 9 || valid
[grid[i][j]] > 0) {
                    return false;
                }
                valid[grid[i][j]] = 1;
            }
```

```
        }
        int rowSum = grid[x][y] + grid[x + 1][y + 1] + grid[x
+ 2][y + 2];
        int colSum = grid[x + 2][y] + grid[x + 1][y + 1] + gri
d[x][y + 2];
        if (rowSum != colSum) return false;
        for (int i = 0; i < 3; i++) {
            if (grid[x][y + i] + grid[x + 1][y + i] + grid[x +
2][y + i] != rowSum) return false;
            if (grid[x + i][y] + grid[x + i][y + 1] + grid[x +
i][y + 2] != colSum) return false;
        }
        return true;
    }
}
```

471

1. 用`map<string, string>`做`hash`明显比用`vector<vector<string>>`要快：

```
class Solution {
    unordered_map<string, string> dp;
    string dfs(string &ref, int i, int j){
        string s = ref.substr(i, j-i);
        if(j-i < 5) return s;
        if(dp.count(s)) return dp[s];
        dp[s] = s;
        for(int l=1; l<=(j-i)/2; ++l){
            string s1 = dfs(ref, i, i+l) + dfs(ref, i+l, j);
            if(s1.size() < dp[s].size()) dp[s] = s1;
            for(int k=l; k<j-i && s[k] == s[k%l]; ++k) if((k+
1)%l == 0){
                string s2 = to_string((k+1)/l) + "[" + dfs(re
f, i, i+l) + "]" + dfs(ref, i + k + 1, j);
```

```
                if(s2.size() < dp[s].size()) dp[s] = s2;
            }
        }
        return dp[s];
    }
public:
    string encode(string s) {
        return dfs(s, 0, s.size());
    }
};
```

2. Python 居然beat `100%`:

```
class Solution(object):
    _dp = {}
    def dfs(self, ref, i, j):
        s = ref[i: j]
        if j-i < 5:
            return s
        if s in self._dp:
            return self._dp[s]
        self._dp[s] = s;
        for l in range(1, int(len(s)/2) + 1):
            t = self.dfs(ref, i, i+l) + self.dfs(ref, i+l, j)
            if len(t) < len(self._dp[s]):
                self._dp[s] = t
            for k in range(l, len(s), l):
                if s[k: k+l] == s[: l]:
                    t = str(int(k/l)+1) +"[" + self.dfs(ref,
i, i+l) + "]" + self.dfs(ref, i+k+l, j)
                    if len(t) < len(self._dp[s]):
                        self._dp[s] = t
```

```
                else:
                    break
        return self._dp[s]
    def encode(self, s):
        return self.dfs(s, 0, len(s))
```

501

1. In order, faster than recursion:

```
class Solution {
public:
    vector<int> findMode(TreeNode* root) {
        vector<int> ans;
        if(!root) return ans;
        stack<TreeNode *> S;
        int pre = INT_MAX, cnt = 0, cur = 0;
        while(root || !S.empty()){
            while(root){
                S.push(root);
                root = root->left;
            }
            if(!S.empty()){
                if(S.top()->val==pre) ++cur;
                else{
                    if(cur>cnt){
                        ans= vector<int>{pre};
                        cnt = cur;
                    }
                    else if(cur == cnt){
                        ans.push_back(pre);
                    }
```

```
                    cur = 1;
                    pre = S.top()->val;
                }
                root = S.top()->right;
                S.pop();
            }
        }
        if(cur > cnt) ans=vector<int>{pre};
        else if(cur == cnt) ans.push_back(pre);
        return ans;
    }
};
```

2. same

```
/**
 * Definition for a binary tree node.
 * public class TreeNode {
 *     int val;
 *     TreeNode left;
 *     TreeNode right;
 *     TreeNode(int x) { val = x; }
 * }
 */
class Solution {
    Integer pre = null;
    int count = 1;
    int max = 0;
    public int[] findMode(TreeNode root) {
        List<Integer> res = new ArrayList<>();

        helper(root, res);
```

```
        int[] arr = new int[res.size()];
        for (int i = 0; i < arr.length; i++) {
            arr[i] = res.get(i);
        }
        return arr;
    }
    public void helper(TreeNode node, List<Integer> res) {
        if (node == null) return;
        helper(node.left, res);
        if (pre != null) {
            if (pre == node.val) {
                count++;
            } else {
                count = 1;
            }
        }
        if (count > max) {
            max = count;
            res.clear();
            res.add(node.val);
        } else if (count == max) {
            res.add(node.val);
        }
        pre = node.val;
        helper(node.right, res);
    }
}
```

374

1. 注意Overflow即可：

```
class Solution {
    long guessRange(long i, long j){
        if(i==j) return i;
        int res = guess(long(i+j)/2);
        if(!res) return long(i+j)/2;
        if(res==1) return guessRange(long(i+j)/2+1, j);
        else return guessRange(i, long(i+j)/2-1);
    }
public:
    int guessNumber(int n) {
        return guessRange(1, n);
    }
};
```

798

1. 这么简单的题半天没做对：利用区间覆盖

```
class Solution {
    typedef pair<int, int> ii;
    static bool cmp(const ii&a, const ii&b){
        if(a.second == b.second) return a.first < b.first;
        return a.second < b.second;
    }
public:
    int bestRotation(vector<int>& A) {
        int n = A.size(), ans = 0, res = 0;
        vector<ii> B;
        for(int i=0; i<n; ++i){
            if(A[i] <= 0) B.push_back(ii(0, n));
            else if(A[i] > i) B.push_back(ii(i+1, n-A[i]+i));
            else{
```

```
                B.push_back(ii(0, i-A[i]));
                if(i<n-1) B.push_back(ii(i+1, n-1));
            }
        }
        sort(B.begin(), B.end(), cmp);
        priority_queue<int> Q;
        for(int i=B.size()-1; i>=0; --i) {
            while(!Q.empty() && Q.top() > B[i].second) Q.pop
();
            Q.push(B[i].first);
            if(Q.size() >= res){
                res = Q.size();
                ans = Q.top();
            }
        }
        return ans;
    }
};
```

2. `O(n)`: index Tree:

```
class Solution {
public:
    int bestRotation(vector<int>& A) {
        int n = A.size(), ans = 0;
        vector<int> B(n+1, 0);
        for(int i=0; i<n; ++i){
            if(A[i] <= 0) {
                ++B[0];
                --B[n];
            }
            else if(A[i] > i) {
```

```
                ++B[i+1];
                --B[n-A[i]+i+1];
            }
            else{
                ++B[0];
                --B[i+1-A[i]];
                ++B[i+1];
                --B[n];
            }
        }
        for(int i=0,tmp=0,res=0; i<n; ++i){
            tmp += B[i];
            if(tmp > res){
                res = tmp;
                ans = i;
            }
        }
        return ans;
    }
};
```

324

1. Quick sort

```
class Solution {
    public void wiggleSort(int[] nums) {
        int mid = quickSort(nums, 0, nums.length - 1, nums.len
gth / 2);
        int n = nums.length;
        //1 5 1 1 6 4 =>       1 1 1 4 5 6 => 1 4 1 5 1 6
        //6 => 0110 | 1 = 7    1 3 5 0 2 4
        int i = 0, j = 0, k = nums.length - 1;
```

```
        while (i <= k) {
            if (nums[index(i, n)] > mid) {
                swap(index(i++, n), index(j++, n), nums);
            } else if (nums[index(i, n)] < mid) {
                swap(index(k--, n), index(i, n), nums);
            } else {
                i++;
            }
        }
    }
    public int index(int k, int n) {
        return (k * 2 + 1) % (n | 1);
    }
    private void swap(int i, int j, int[] nums) {
        int k = nums[i];
        nums[i] = nums[j];
        nums[j] = k;
    }

    public int quickSort(int[] nums, int left, int right, int
k) {
        if (left == right) return nums[left];
        int index = partition(left, right, nums);
        if (index > k) return quickSort(nums, left, index - 1,
k);
        else if (index < k) return quickSort(nums, index + 1,
right, k);
        return nums[k];
    }

    public int partition(int left, int right, int[] nums) {
```

```
        int i = left, j = right;
        int pivot = nums[i];
        while (i < j) {
            while (i < j && nums[j] >= pivot) {
                j--;
            }
            nums[i] = nums[j];
            while (i < j && nums[i] <= pivot) {
                i++;
            }
            nums[j] = nums[i];
        }
        nums[i] = pivot;
        return i;
    }
}
```

- [ ] SOLVE IT @Zebo L

323

1. unionfind

```
class Solution {
    class UnionFind {
        int[] parents;
        int number;
        UnionFind(int n) {
            number = n;
            parents = new int[n];
            for (int i = 0; i < n; i++) {
                parents[i] = i;
            }
        }
```

```
        void union(int i, int j) {
            int a = find(i);
            int b = find(j);
            if (a != b) {
                parents[a] = b;
                number--;
            }
        }

        int find(int i) {
            while (i != parents[i]) {
                i = parents[i];
            }
            return i;
        }
    }
    public int countComponents(int n, int[][] edges) {
        UnionFind uf = new UnionFind(n);
        for (int[] edge : edges) {
            uf.union(edge[0], edge[1]);
        }
        return uf.number;
    }
}
```

2. 同上：

```
class Solution {
    vector<int> P;
    int findRoot(int i){
        if(P[i] == i) return i;
        return P[i] = findRoot(P[i]);
```

```
    }
public:
    int countComponents(int n, vector<pair<int, int>>& edges)
{
        P.resize(n);
        for(int i=0; i<n; ++i) P[i]=i;
        for(auto p: edges) if(findRoot(p.first) != findRoot(p.
second)) P[findRoot(p.second)] = findRoot(p.first);
        int ans = 0;
        for(int i=0; i<n; ++i) ans += P[i] == i;
        return ans;
    }
};
```

40

1. Backtrack dfs:

```
class Solution {
    typedef vector<int> vi;
    void dfs(vector<vi>&ans, vi cur, map<int, int>::iterator i
t, int target, map<int, int> &cnt){
        if(!target || it==cnt.end()){
            if(!target) ans.push_back(cur);
            return;
        }
        do{
            int x = it->first, y = it->second;
            vi tmp(cur);
            ++it;
            for(int i=1; i<=y && i*x<=target; ++i){
                tmp.push_back(x);
                dfs(ans, tmp, it, target-i*x, cnt);
```

```
            }
        }while(it!=cnt.end() && it->first<=target);
    }
public:
    vector<vector<int>> combinationSum2(vector<int>& candidate
s, int target) {
        map<int, int> cnt;
        for(int k: candidates) ++cnt[k];
        vector<vi> ans;
        dfs(ans, vi(), cnt.begin(), target, cnt);
        return ans;
    }
};
```

681

1.

```
class Solution {
    public String nextClosestTime(String time) {
        int hour = Integer.valueOf(time.substring(0, 2));
        int minute = Integer.valueOf(time.substring(3));

        while (true) {
            if (++minute == 60) {
                minute = 0;
                ++hour;
                hour = hour % 24;
            }
            String cur = String.format("%02d:%02d", hour, minu
te);
            boolean flag = true;
            for (int i = 0; i < cur.length(); i++) {
```

```
                if (time.indexOf(cur.charAt(i)) < 0) {
                    flag = false;
                    break;
                }
            }
            if (flag) return cur;
        }
    }
}
```

2. BRUTE FORCE:

```
class Solution {
    typedef pair<int, int> ii;
    #define CI(c) int((c) - '0')
    #define IC(i) char((i) + '0')
public:
    string nextClosestTime(string time) {
        ii ref = ii(CI(time[0])*10 + CI(time[1]), CI(time[3])*
10 + CI(time[4]));
        set<int> pool{CI(time[0]), CI(time[1]), CI(time[3]), C
I(time[4])};
        vector<int> dig(pool.begin(), pool.end());
        int n = dig.size();
        set<ii> S;
        for(int k=0; k<int(pow(n, 4)); ++k){
            int h = dig[k%n] * 10 + dig[k/n%n];
            int m = dig[k/n/n%n] * 10 + dig[k/n/n/n%n];
            if(h<24 && m < 60) S.insert(ii(h, m));
        }
        for(auto s: S)cout<<s.first<<' '<<s.second<<endl;
        ii ans = *S.begin();
        auto it = S.upper_bound(ref);
```

```
        if(it!=S.end()) ans = *it;
        string res;
        return res + IC(ans.first/10) + IC(ans.first%10) + ":"
+ IC(ans.second/10) + IC(ans.second%10);
    }
};
```

483

1. 几次二分，整个题里面全是细节:

```
class Solution {
    int cmp(long x, long k, long n){
        long prod = 1L;
        while(k && n>=prod){
            n -= prod;
            --k;
            if(prod > n/x) break;
            prod *= x;
        }
        if(!n && !k) return 0;
        if(!k) return -1;
        return 1;
    }
public:
    string smallestGoodBase(string n) {
        long N = stol(n);
        long k = int(log(N)/log(2)) + 3;
        while(k>1){
            long l = max(1, int(pow(N, 1./k))-1), r = long(pow
(N, 1./(k-1))) + 1L;
            while(l<r-1){
                long c = (l+r)/2;
```

```
                if(cmp(c, k, N) < 0) l = c;
                else r = c;
            }
            if(!cmp(r, k, N)) return to_string(r);
            --k;
        }
        return to_string(N-1);
    }
};
```

2. 这个解法更清晰一些：而且为什么这个比我的快！！

```
// Return 0 if k not fit
size_t try_k(size_t n, int k) {
        double ord = 1.0 / (k-1);
        double root = pow(n, ord); //O(log n) for required flo
ating point precission
        size_t a = floor(root);
        if (a < 2) return 0;
        size_t sum = 1;
        for (int i = 0; i < k-1; i++) sum = sum * a + 1;  // k
-1 time, which is O(log n)
        if (sum != n) return 0;
        return a;
}

string smallestGoodBase(string nstr) {
        size_t n = stoull(nstr);
        for (int k = 63; k >= 3; k--) {
                size_t result = try_k(n, k);
                if (result > 0) return to_string(result);
        }
```

```
        // only trivial solution left
        return to_string(n-1);
}
```

70

1. DP. Recursion会超时

```
class Solution {
    public int climbStairs(int n) {
        if (n <= 2) return n;
        int[] dp = new int[n + 1];
        dp[0] = 0;
        dp[1] = 1;
        dp[2] = 2;
        for (int i = 3; i <= n; i++) {
            dp[i] = dp[i - 1] + dp[i - 2];
        }
        return dp[n];
    }
}
```

2. Fibonacci 序列，以下是黑技术，利用矩阵，时间复杂度 `O(log(n))`:

```
class Solution {
    typedef vector<int> vi;
    vector<vi> matmul(vector<vi>A, vector<vi>B, int n){
        vector<vi> C(n, vi(n, 0));
        for(int i=0; i<n; ++i) for(int j=0; j<n; ++j) for(int
k=0; k<n; ++k) C[i][j] += A[i][k] * B[k][j];
        return C;
    }
    vector<vi> REF = {{1, 1}, {1, 0}};
    vector<vi> pwr(vector<vi> A, int n, int k){
        vector<vi> ans = {{1, 0}, {0, 1}};
```

```
        while(k){
            if(k%2) ans = matmul(ans, A, n);
            A = matmul(A, A, n);
            k /= 2;
        }
        return ans;
    }
public:
    int climbStairs(int n) {
        if(n <= 1) return 1;
        vector<vi> res = pwr(REF, 2, n-1);
        return res[0][0] + res[0][1];
    }
};
```

626

SQL

585

SQL

253

1. Index tree 解这种题就是简单啊：

```
class Solution {
public:
    int minMeetingRooms(vector<Interval>& intervals) {
        map<int, int> range;
        for(auto i: intervals) {
            ++range[i.start];
            --range[i.end];
        }
```

```
        int ans = 0, tmp = 0;
        for(auto p: range){
            tmp += p.second;
            ans = max(ans, tmp);
        }
        return ans;
    }
};
```