August 15, 2018 题目：800,524,683,505,802,450,850,299,466,58,284,809,780,175

800

1. 做过了：

```
class Solution {
    const string ref = "0123456789abcdef";
    map<char, int> xo;
    set<int> po;
    void init(){
        for(int i=0;i<ref.size();++i) xo[ref[i]] = i;
        for(int i=0;i<16;++i) po.insert(i*16 + i);
        po.insert(1000000);
        po.insert(-10000000);
    }
    int hex2int(string s){
        return xo[s[0]] * 16 + xo[s[1]];
    }
    string int2hex(int n){
        string ans;
        ans += ref[n/16];
        ans += ref[n%16];
        return ans;
    }
    string closest(string s){
        int k = hex2int(s), dif, ans;
        auto it = po.lower_bound(k);
        dif = *it - k;
```

```
        ans = *it;
        --it;
        if(k - *it <= dif){
            ans = *it;
        }
        return int2hex(ans);
    }
public:
    string similarRGB(string color) {
        init();
        return "#" + closest(color.substr(1, 2)) + closest(col
or.substr(3, 2)) + closest(color.substr(5, 2));
    }
};
```

524

1.

```
class Solution {
    public String findLongestWord(String s, List<String> d) {
        Collections.sort(d, (a, b) -> a.length() == b.length()
? a.compareTo(b) : b.length() - a.length());
        System.out.println(d);
        for (String str : d) {
            int i = 0;
            for (char c : s.toCharArray()) {
                if (i < str.length() && c == str.charAt(i)) i+
+;
            }
            if (i == str.length()) return str;
        }
        return "";
```

```
    }
}
```

2. Find subsequence:

```
class Solution {
    #define CI(c) int((c) - 'a')
    typedef vector<int> vi;
public:
    string findLongestWord(string str, vector<string>& d) {
        int n = str.size();
        vector<vi> next(n+1, vi(26, n));
        for(int j=n-1; j>=0; --j){
            copy(next[j+1].begin(), next[j+1].end(), next[j].b
egin());
            next[j][CI(str[j])] = j;
        }
        string ans;
        for(string s: d) if(s.size()>ans.size() || (s.size()==
ans.size() && s<ans)){
            bool ok = true;
            for(int i=0, j=-1; i<s.size()&&ok; ++i){
                j = next[j+1][CI(s[i])];
                if(j == n) ok = false;
            }
            if(ok) ans = s;
        }
        return ans;
    }
};
```

683

1. 用 set 来维持区间

```
class Solution {
public:
    int kEmptySlots(vector<int>& flowers, int k) {
        set<int> R{INT_MIN, INT_MAX};
        for(int i=1; i<=flowers.size(); ++i){
            int lower = *(--R.lower_bound(flowers[i-1])), uppe
r = *R.lower_bound(flowers[i-1]);
            if(flowers[i-1]-lower==k+1 || upper - flowers[i-1]
==k+1) return i;
            R.insert(flowers[i-1]);
        }
        return -1;
    }
};
```

2. Bucket Sort : Much Faster

```
class Solution {
    typedef pair<int, int> ii;
public:
    int kEmptySlots(vector<int>& flowers, int k) {
        int n = flowers.size(), chunk=k+1, n_chunk = (n+k)/(k+
1);
        vector<ii> B(n_chunk, ii(50000, -50000));
        for(int i=0; i<n; ++i){
            int f = flowers[i] - 1, idx = (flowers[i]-1)/chun
k;
            B[idx].first = min(B[idx].first, f);
            B[idx].second = max(B[idx].second, f);
            if(idx && B[idx].first - B[idx-1].second == chunk)
return i+1;
            if(idx < n_chunk-1 && B[idx+1].first - B[idx].seco
nd == chunk) return i+1;
```

```
        }
        return -1;
    }
};
```

505

1. Dijkstra

```
class Solution {
    public int shortestDistance(int[][] maze, int[] start, int
[] destination) {
        // BFS
        PriorityQueue<int[]> q = new PriorityQueue<>((a, b) ->
a[2] - b[2]);
        int[][] dis = new int[maze.length][maze[0].length];
        q.offer(new int[]{start[0], start[1], 0});
        boolean[][] visited = new boolean[maze.length][maze
[0].length];
        for (int[] d : dis) {
            Arrays.fill(d, Integer.MAX_VALUE);
        }
        dis[start[0]][start[1]] = 0;

        int[] dx = {1, -1, 0, 0};
        int[] dy = {0, 0, -1, 1};
        while (!q.isEmpty()) {
            int[] cur = q.poll();
            if (visited[cur[0]][cur[1]]) continue;
            visited[cur[0]][cur[1]] = true;
```

```
            if (cur[0] == destination[0] && cur[1] == destinat
ion[1]) return cur[2];
            for (int k = 0; k < 4; k++) {
                int ux = cur[0];
                int uy = cur[1];
                int d = cur[2];
                while (ux >= 0 && uy >= 0 && ux < maze.length
&& uy < maze[0].length && maze[ux][uy] == 0) {
                    ux += dx[k];
                    uy += dy[k];
                    d++;
                }
                ux -= dx[k];
                uy -= dy[k];
                d--;
                if (d < dis[ux][uy] && !visited[ux][uy]) {
                    q.offer(new int[]{ux, uy, d});
                    dis[ux][uy] = d;
                }
            }

        }
        return -1;
    }
}
```

2. 纯粹的BFS：

```
class Solution {
    typedef vector<int> vi;
    typedef pair<int, int> ii;
    int dx[4] = {1, -1, 0, 0}, dy[4] = {0, 0, 1, -1};
    int turn[4][2] = {
```

```
        {2, 3},
        {2, 3},
        {0, 1},
        {0, 1},
    };
public:
    int shortestDistance(vector<vector<int>>& M, vector<int>&
st, vector<int>& de) {
        int n = M.size(), m = M[0].size();
        queue<vi> Q;
        set<vi> S;
        for(int k=0; k<4; ++k){
            Q.push(vi{st[0], st[1], k, 0});
            S.insert(vi{st[0], st[1], k});
        }
        while(!Q.empty()){
            vi tmp = Q.front();
            int x = tmp[0], y = tmp[1], d = tmp[2], step = tmp
[3];
            Q.pop();
            int x1 = x + dx[d], y1 = y + dy[d];
            if(x1>=0 && x1<n && y1>=0 && y1<m && !M[x1][y1]){
                if(!S.count(vi{x1, y1, d})){
                    Q.push(vi{x1, y1, d, step+1});
                    S.insert(vi{x1, y1, d});
                }
            }
            else{
                if(x==de[0] && y==de[1]) return step;
                for(int k=0; k<2; ++k){
```

```
                    int d1 = turn[d][k];
                    x1 = x + dx[d1];
                    y1 = y + dy[d1];
                    if(x1>=0 && x1<n && y1>=0 && y1<m && !M[x
1][y1] && !S.count(vi{x1, y1, d1})){
                        Q.push(vi{x1, y1, d1, step+1});
                        S.insert(vi{x1, y1, d1});
                    }
                }
            }
        }
        return -1;
    }
};
```

- [ ] 研究一下别人解法，顺便学习一下 `Dijkstra` @Zebo L

802

1. 这也能过？

```
class Solution {
    typedef vector<int> vi;
    vector<vi> E;
public:
    vector<int> eventualSafeNodes(vector<vector<int>>& graph)
{
        set<int> S;
        queue<int> Q;
        E.resize(graph.size());
        for(int i=0; i<graph.size(); ++i) {
            if(graph[i].empty()) {
                S.insert(i);
                Q.push(i);
```

```
            }
            for(int j: graph[i]) E[j].push_back(i);
        }
        while(!Q.empty()){
            for(int k: E[Q.front()]) if(!S.count(k)) {
                bool ok = true;
                for(int l: graph[k]) if(!S.count(l)) {
                    ok = false;
                    break;
                }
                if(ok){
                    Q.push(k);
                    S.insert(k);
                }
            }
            Q.pop();
        }
        return vi(S.begin(), S.end());
    }
};
```

2. 之前怎么想到的dp，如下：

```
class Solution {
    int dp[10005], inf = 6000000;
    int dfs(int i, vector<vector<int>>& G){
        if(dp[i] >= 0) return dp[i];
        if(G[i].empty()) return dp[i] = 0;
        dp[i] = inf;
        int ans = 0;
        for(auto j: G[i]) ans = max(ans, dfs(j, G));
        return dp[i] = ans;
```

```
    }
public:
    vector<int> eventualSafeNodes(vector<vector<int>>& graph)
{
        memset(dp, -1, sizeof(dp));
        vector<int> ans;
        for(int i=0; i<graph.size(); ++i) if(dfs(i, graph) < i
nf) ans.push_back(i);
        return ans;
    }
};
```

450

1. recursion

```
/**
 * Definition for a binary tree node.
 * public class TreeNode {
 *     int val;
 *     TreeNode left;
 *     TreeNode right;
 *     TreeNode(int x) { val = x; }
 * }
 */
class Solution {
    public TreeNode deleteNode(TreeNode root, int key) {
        // find the key
        if (root == null) return root;
        if (key > root.val) {
            root.right = deleteNode(root.right, key);
        } else if (key < root.val) {
            root.left = deleteNode(root.left, key);
```

```
        } else {
            if (root.right == null) {
                return root.left;
            } else if (root.left == null) {
                return root.right;
            }
            TreeNode min = findMin(root.right);
            root.val = min.val;
            root.right = deleteNode(root.right, root.val);
        }
        return root;
    }

    public TreeNode findMin(TreeNode node) {
        while (node.left != null) {
            node = node.left;
        }
        return node;
    }
}
```

2. 思路同上，小细节要注意：

```
class Solution {
public:
    TreeNode* deleteNode(TreeNode* root, int key) {
        if(!root) return root;
        if(root->val > key) {
            root->left = deleteNode(root->left, key);
            return root;
        }
        else if(root->val < key) {
```

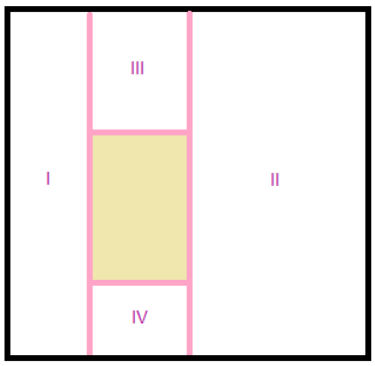
```
            root->right = deleteNode(root->right, key);
            return root;
        }
        else{
            if(!root->left) {
                auto p = root->right;
                delete root;
                return p;
            }
            else if(!root->left->right){
                auto p = root->left;
                p->right = root->right;
                delete root;
                return p;
            }
            else {
                auto p = root->left;
                while(p->right->right) p = p->right;
                auto q = p->right;
                p->right = q->left;
                q->right = root->right;
                q->left = root->left;
                delete root;
                return q;
            }
        }
    }
};
```

1. 用map 维持区间，跟index tree 很类似的东西:

```
class Solution {
    const long M = 1000000007;
public:
    int rectangleArea(vector<vector<int>>& recs) {
        map<long, map<long, long>> cnt;
        for(auto v: recs) {
            cnt[v[0]][v[1]]++;
            cnt[v[0]][v[3]]--;
            cnt[v[2]][v[1]]--;
            cnt[v[2]][v[3]]++;
        }
        long ans = 0L;
        map<long, long> current;
        auto x0 = cnt.begin(), x1 = ++cnt.begin();
        while(x1!=cnt.end()){
            map<long, long> par;
            long cur = 0L, tmp = 0L;
            for(auto p: x0->second) current[p.first] += p.seco
nd;
            for(auto p: current) {
                cur += p.second;
                par[p.first] = cur;
            }
            auto y0 = par.begin(), y1 = ++par.begin();
            while(y1 != par.end()){
                if(y0->second) tmp += y1->first - y0->first;
                y0 = y1;
                ++y1;
            }
```

```
                ans += ((tmp%M) * (x1->first-x0->first)) %M;
                x0 = x1;
                ++x1;
            }
            return ans%M;
        }
};
```

- [ ] The following solution @Zebo L




```
class Solution(object):
    def getIntersectingArea(self, r1, c1, r2, c2, x1, y1, x2, y2):
        return max(0, min(r2, x2)-max(r1, x1)) * max(0, min(c2, y2)-max(c1, y1))
    def cutRectangle(self, now, rectangles, r1, c1, r2, c2):
        if now >= len(rectangles) or r1 >= r2 or c1 >= c2: return 0
        x1, y1, x2, y2 = rectangles[now]
        if x1 >= r2 or y1 >= c2 or x2 <= r1 or y2 <= c1: return self.cutRectangle(now+1, rectangles, r1, c1, r2, c2)
        s1 = self.cutRectangle(now+1, rectangles, r1, c1, min(x1, r2), c2) if x1 > r1 else 0
        s2 = self.cutRectangle(now+1, rectangles, max(x2, r1), c1, r2, c2) if x2 < r2 else 0
        s3 = self.cutRectangle(now+1, rectangles, max(x1, r1), c1, min(x2, r2), y1) if y1 > c1 else 0
        s4 = self.cutRectangle(now+1, rectangles, max(x1, r1), y2, min(x2, r2), c2) if y2 < c2 else 0
        return (s1+s2+s3+s4+self.getIntersectingArea(r1, c1, r2, c2, x1, y1, x2, y2))
    def rectangleArea(self, rectangles):
        trp = zip(*rectangles)
        return self.cutRectangle(0, rectangles, min(trp[0]), min(trp[1]), max(trp[2]), max(trp[3]))%(10**9+7)
```

299

1. Counting numbers:

```
class Solution {
public:
    string getHint(string secret, string guess) {
        int a=0, b=0;
        vector<int> A(10, 0), B(10, 0);
        for(int i=0; i<secret.size(); ++i){
            if(secret[i] == guess[i]) ++a;
```

```
            else{
                ++A[int(secret[i]-'0')];
                ++B[int(guess[i]-'0')];
            }
        }
        for(int i=0; i<10; ++i) b += min(A[i], B[i]);
        return to_string(a) + "A" + to_string(b) + "B";
    }
};
```

466

58

1. 上一道太难，写道Short is Beauty 系列题：

```
class Solution:
    def lengthOfLastWord(self, s):
        return len(s.strip().split(" ")[-1])
```

284

1. Use queue

```
// Java Iterator interface reference:
// https://docs.oracle.com/javase/8/docs/api/java/util/Iterator.html
class PeekingIterator implements Iterator<Integer> {


    Queue<Integer> q;
        public PeekingIterator(Iterator<Integer> iterator) {
            // initialize any member here.
            q = new LinkedList<>();
        while (iterator.hasNext()) {
            q.offer(iterator.next());
```

```
        }
        }

    // Returns the next element in the iteration without advan
cing the iterator.
        public Integer peek() {
        return q.peek();
        }

        // hasNext() and next() should behave the same as in t
he Iterator interface.
        // Override them if needed.
        @Override
        public Integer next() {
            return q.poll();
        }

        @Override
        public boolean hasNext() {
            return !q.isEmpty();
        }
}
```

2. 只需要一个数：

```
class PeekingIterator : public Iterator {
    bool peeked;
    int val;
public:
        PeekingIterator(const vector<int>& nums) : Iterator(nu
ms), peeked(false) {}
    int peek(){
        if(!peeked) {
```

```
            peeked=true;
            val = this->Iterator::next();
        }
        return val;
    }
    int next() {
        if(peeked){
            peeked = false;
            return val;
        }
        return this->Iterator::next();
        }
        bool hasNext() const {
            return peeked||this->Iterator::hasNext();
        }
};
```

809

1. 就是烦了些：

```
class Solution {
    bool exT(string s, string t){
        int i = 0, j = 0;
        while(i<s.size() && j<t.size()){
            char c = s[i], d = t[j];
            if(c != d) return false;
            int cnt1 = 0, cnt2 = 0;
            while(i<s.size() && s[i]==c){
                ++i;
                ++cnt1;
            }
            while(j<t.size() && t[j]==d){
```

```
                ++j;
                ++cnt2;
            }
            if(cnt1<cnt2 || (cnt1!=cnt2 && cnt1 < 3)) return f
alse;
        }
        return i==s.size() && j==t.size();
    }
public:
    int expressiveWords(string S, vector<string>& words) {
        int ans = 0;
        for(string s: words) ans += exT(S, s);
        return ans;
    }
};
```

780

1. 逆推，完全一数学题：

```
class Solution {
public:
    bool reachingPoints(int sx, int sy, int tx, int ty) {
        while(tx>=sx && ty>=sy){
            if(tx > ty){
                if(ty == sy && (tx-sx)%sy == 0) return true;
                tx %= ty;
            }
            else{
                if(tx == sx && (ty-sy)%sx == 0) return true;
                ty %= tx;
            }
        }
```

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

175

SQL