August 18, 2018 题目：380,514,332,320,214,513,182,227,195,642,793,734,576,696

380

1. 做过：hashing + array：

```
class RandomizedSet {
public:
    vector<int> A;
    unordered_map<int, int> B;
    RandomizedSet() {}
    bool insert(int val) {
        if(B.count(val)) return false;
        B[val] = A.size();
        A.push_back(val);
        return true;
    }
    bool remove(int val) {
        if(!B.count(val)) return false;
        if(A.back() != val) A[B[A.back()]=B[val]] = A.back();
        B.erase(val);
        A.pop_back();
        return true;
    }
    int getRandom() { return A[rand()%(int)A.size()]; }
};
```

514

1. 不知道难点何在：

```
class Solution {
public:
    int findRotateSteps(string ring, string key) {
        if(key.empty()) return 0;
        int n = ring.size(), ans = 1E8;
        unordered_map<char, vector<int>> pos;
        for(int i=0; i<n; ++i) pos[ring[i]].push_back(i);
        vector<int> dp(n, 1E8);
        for(int i: pos[key[0]]) dp[i] = 1 + min(i-0, n-i);
        for(int i=1; i<key.size(); ++i){
            vector<int> tmp(n, 1E8);
            for(int k: pos[key[i]]) for(int j:pos[key[i-1]]){
                tmp[k] = min(tmp[k], 1 + dp[j] + min((j+n-k)%
n, (k+n-j)%n));
            }
            swap(dp, tmp);
        }
        for(int j: pos[key.back()]) ans = min(ans, dp[j]);
        return ans;
    }
};
```

2. DP

332

1. 本想用贪心，结果以下case贪心是不对的-_-:

AUA: ADL ANU AXA
TIA: ADL AUA
ADL: ANU EZE EZE
AXA: AUA EZE TIA
JFK: AXA AXA
EZE: ADL ANU HBA TIA
ANU: AUA EZE JFK

贪心counter

于是用dfs：

```
class Solution {
    int N;
    unordered_map<string, map<string, int>> E;
    bool dfs(vector<string>& ans){
        if(ans.size() == N) return true;
        for(auto p: E[ans.back()]) if(p.second){
            --E[ans.back()][p.first];
            ans.push_back(p.first);
            if(dfs(ans)) return true;
            ans.pop_back();
            ++E[ans.back()][p.first];
        }
        return false;
    }
public:
    vector<string> findItinerary(vector<pair<string, string>>
tickets) {
        vector<string> ans{"JFK"};
        for(auto p: tickets) E[p.first][p.second]++;
        N = tickets.size() + 1;
        assert(dfs(ans));
        return ans;
    }
```

```
};
```

- [ ] how to iteratively solve this kind of problem @Zebo L

2. .....我以为是topological，审题错了。。。两种解法，iterate和recursion

```
class Solution {
    public List<String> findItinerary(String[][] tickets) {
        Map<String, PriorityQueue<String>> map = new HashMap<>
();

        for (String[] s : tickets) {
            if (!map.containsKey(s[0])) {
                map.put(s[0], new PriorityQueue<String>());
            }
            map.get(s[0]).offer(s[1]);
        }
        Stack<String> stack = new Stack<>();
        stack.push("JFK");
        List<String> res = new ArrayList<>();
        while (!stack.isEmpty()) {
            while (map.containsKey(stack.peek()) && !map.get(s
tack.peek()).isEmpty()) {
                stack.push(map.get(stack.peek()).poll());
            }
            res.add(0, stack.pop());
        }

        //helper(res, map, "JFK");
        return res;
    }
    /**
    public void helper(List<String> res, Map<String, PriorityQ
ueue<String>> map, String cur) {
```

```
        while (map.containsKey(cur) && !map.get(cur).isEmpty
()) {
            helper(res, map, map.get(cur).poll());
        }
        res.add(0, cur);
    }*/
}
```

320

1. 很直接的方法，用bit map：

```
class Solution {
public:
    vector<string> generateAbbreviations(string word) {
        vector<string> ans(1<<(int(word.size())));
        for(int k=0; k<(1<<(int(word.size()))); ++k){
            int cnt = 0;
            for(int j=0; j<word.size(); ++j){
                if(1&(k>>j)){
                    if(cnt) ans[k] += to_string(cnt);
                    ans[k] += word[j];
                    cnt=0;
                }
                else ++cnt;
            }
            if(cnt) ans[k] += to_string(cnt);
        }
        return ans;
    }
};
```

2. 很难想象，一个月前的我居然想出了以下这个奇葩方法：（因为自己的习惯是遇到类似的题，第一时间就用bit了）

```
class Solution {
public:
    vector<string> generateAbbreviations(string word) {
        vector<string> ans{""};
        for(int j=word.size()-1; j>=0; --j){
            int n = ans.size();
            for(int i=0; i<n; ++i){
                ans.push_back(word[j] + ans[i]);
                if(!ans[i].empty() && isdigit(ans[i][0])){
                    int cur = stoi(ans[i]);
                    ans[i] = to_string(cur+1) + ans[i].substr
(to_string(cur).size());
                }
                else ans[i] = '1' + ans[i];
            }
        }
        return ans;
    }
};
```

3. 就。。。backtracking

```
class Solution {
    public List<String> generateAbbreviations(String word) {
        List<String> list = new ArrayList<>();
        helper(list, "", word, 0, 0);
        return list;
    }
    private void helper(List<String> list, String cur, String
word, int k, int index) {
        if (index == word.length()) {
            if (k > 0) cur += (k);
```

```
            list.add(cur);
        } else {
            helper(list, cur, word, k + 1, index + 1);
            helper(list, cur + (k > 0 ? k : "") + word.charAt
(index), word, 0, index + 1);
        }
    }
}
```

214

1. 标准KMP:

```
class Solution {
public:
    string shortestPalindrome(string s) {
        string t = s;
        reverse(t.begin(), t.end());
        vector<int> kmp(t.size(), -1);
        for(int i=1; i<t.size(); ++i){
            int j = kmp[i-1];
            while(j>=0 && s[i] != s[j+1]) j = kmp[j];
            kmp[i] = (s[i]==s[j+1]?j+1:-1);
        }
        int i=0, j=0;
        while(i<t.size()){
            if(t[i] == s[j]){
                ++i;
                ++j;
            }
            else{
                if(j>0) j = kmp[j-1]+1;
                else{
```

```
                    j = 0;
                    ++i;
                }
            }
        }
        return t + s.substr(j);
    }
};
```

2. 一个月前用过的hash 大法：这个就比上面那道题更像我的手法

```
class Solution {
    const int Mod = 1E9 + 7;
    const int p = 37;
    int mul(int x, int y){
        return long(x) * long(y) % Mod;
    }
    int add(int x, int y){
        return (x + y) % Mod;
    }
public:
    string shortestPalindrome(string s) {
        int n = s.size();
        string tmp(s);
        reverse(tmp.begin(), tmp.end());
        vector<int> dp(n, 0), re(n, 0);
        for(int i=0; i<n; ++i){
            if(!i) dp[i] = (int)(s[i] - 'a');
            else dp[i] = add((int)(s[i] - 'a'), mul(p, dp[i-
1]));
        }
        for(int j=n-1, m = p; j>=0; --j){
```

```
            if(j == n-1) re[j] = (int)(tmp[j] - 'a');
            else{
                re[j] = add(re[j+1], mul((int)(tmp[j] - 'a'),
m));
                m = mul(m, p);
            }
        }
        for(int i=0; i<n; ++i) if(dp[n-i-1] == re[i]) return t
mp.substr(0, i) + s;
        return "";
    }
};
```

3. KMP

```
class Solution {
    public String shortestPalindrome(String s) {
        int[] kmp = build(s + "#" + new StringBuilder(s).rever
se().toString());
        return new StringBuilder(s.substring(kmp[kmp.length -
1])).reverse().toString() + s;
    }

    public int[] build(String s) {
        int i = 0, j = 1;
        int[] kmp = new int[s.length()];
        while (i < s.length() && j < s.length()) {
            if (s.charAt(i) == s.charAt(j)) {
                kmp[j] = kmp[j - 1] + 1;
                i++;
                j++;
            } else {
```

```
                while (i - 1 >= 0 && s.charAt(i) != s.charAt
(j)) {
                    i = kmp[i - 1];
                }
                if (s.charAt(i) == s.charAt(j)) i++;
                kmp[j] = i;
                j++;
            }
        }
        return kmp;
    }
}r
```

513

1. 典型dfs : 也可以BFS的

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

```
        return res;
    }
};
```

2. DFS

```
class Solution {
    int res;
    int max;
    public int findBottomLeftValue(TreeNode root) {
        res = 0;
        max = -1;
        helper(root, 0);
        return res;
    }

    public void helper(TreeNode node, int level) {
        if (node == null) return;
        if (level > max) {
            max = level;
            res = node.val;
        }
        helper(node.left, level + 1);
        helper(node.right, level + 1);
    }
}
```

182

SQL

227

1. `to_string` and `stoi` are slow, try to avoid them.

```
class Solution {
```

```
public:
    int calculate(string s) {
        int ans = 0, cur = 0;
        char pend = '+';
        int i = 0;
        for(; i<s.size() && s[i]==' '; ++i);
        for(; i<s.size() && isdigit(s[i]); ++i) cur = cur*10 +
int(s[i]-'0');
        while(i<s.size()){
            if(s[i] == ' '){
                ++i;
                continue;
            }
            int tmp = 0, j=i+1;
            for(; s[j]==' '; ++j);
            for(; j<s.size()&&isdigit(s[j]); ++j) tmp = tmp*10
+ int(s[j]-'0');
            if(s[i]=='*') cur = cur * tmp;
            else if(s[i]=='/') cur = cur / tmp;
            else{
                ans = (pend=='+'? ans+cur: ans-cur);
                cur = tmp;
                pend = s[i];
            }
            i = j;
        }
        return ans = (pend=='+'? ans+cur: ans-cur);
    }
};
```

BASH:

1. 唯一看懂的一个解法：

```
awk 'NR==10' file.txt
```

642

1. Tier，并且在每个节点maintain 这三个 输出的string:

```
class AutocompleteSystem {
    typedef pair<int, string> is;
    struct T{
        unordered_map<char, T*> C;
        set<is> X;
        T() {}
    };
    T *root, *current;
    string sc;
    unordered_map<string, int> P;
    void insertX(T *r, string s, int cnt){
        for(char c: s){
            if(!r->C.count(c)) r->C[c] = new T();
            r = r->C[c];
            if(r->X.count(is(-cnt+1, s))) r->X.erase(is(-cnt+
1, s));;
            r->X.insert(is(-cnt, s));
            if(r->X.size()>3) r->X.erase(--(r->X.end()));
        }
    }
public:
    AutocompleteSystem(vector<string> sentences, vector<int> t
imes) {
        for(int i=0; i<sentences.size(); ++i) P[sentences[i]]
+= times[i];
```

```
        root = new T();
        current = root;
        sc = "";
        for(auto p: P) insertX(root, p.first, p.second);
    }

    vector<string> input(char c) {
        vector<string> ans;
        if(c=='#' && !sc.empty()){
            ++P[sc];
            insertX(root, sc, P[sc]);
            sc = "";
            current = root;
        }
        else{
            if(!current->C.count(c)) current->C[c] = new T();
            current = current->C[c];
            sc += c;
            for(auto p: current->X) ans.push_back(p.second);
        }
        return ans;
    }
};
```

2. 写的略复杂

```
class AutocompleteSystem {

    class TrieNode {
        TrieNode[] children;
        Map<String, Integer> map;
        public TrieNode() {
```

```
            children = new TrieNode[256];
            map = new HashMap<>();
        }
    }
    class Trie {
        TrieNode root;
        public Trie() {
            root = new TrieNode();
        }

        public void build(String s, int k) {
            TrieNode node = root;
            for (char c : s.toCharArray()) {
                if (node.children[c - ' '] == null) {
                    node.children[c - ' '] = new TrieNode();
                }
                node = node.children[c - ' '];
                node.map.put(s, node.map.getOrDefault(s, 0) +
k);
            }
        }

        public List<String> find(char c, TrieNode node) {
            List<String> list = new ArrayList<>();

            PriorityQueue<Map.Entry<String, Integer>> pq = new
PriorityQueue<>((a, b) -> (a.getValue() == b.getValue()) ? a.g
etKey().compareTo(b.getKey()) : b.getValue() - a.getValue());
            pq.addAll(node.map.entrySet());
            int k = 0;
            while (!pq.isEmpty() && k < 3) {
```

```
                list.add(pq.poll().getKey());
                k++;
            }

            return list;
        }
    }
    Trie trie;
    TrieNode node;
    String update;
    public AutocompleteSystem(String[] sentences, int[] times)
{
        trie = new Trie();
        for (int i = 0; i < times.length; i++) {
            trie.build(sentences[i], times[i]);
        }
        node = trie.root;
        update = "";
    }

    public List<String> input(char c) {
        List<String> res = new ArrayList<>();
        if (c == '#' && !update.isEmpty()) {
            node = trie.root;
            trie.build(update, 1);
            update = "";
        } else {
            update += c;
            if (node != null && node.children[c - ' '] != nul
l) {
```

```
                node = node.children[c - ' '];
                res = trie.find(c, node);
            } else {
                node = null;
            }
        }

        return res;
    }
}
```

793

1. 直接二分：
    - 结果只有 `5` 跟 `0` 两种可能性，关键是判定是否存在 `x` $s.t. f(x) = K$

```
class Solution {
    long f(long x){
        long ans = 0;
        x /= 5;
        while(x){
            ans +=  x;
            x /= 5;
        }
        return ans;
    }
public:
    int preimageSizeFZF(int K) {
        long l = -1, r = 2E18;
        while(r>l+1){
            long c = (r+l)/2;
            if(f(c)>=K) r=c;
            else l=c;
```

```
        }
        if(f(r) == K) return 5;
        return 0;
    }
};
```

- [ ] 这俩post 值得一看 @Zebo L
  - [ ] https://leetcode.com/problems/preimage-size-of-factorial-zeroes-function/discuss/117821/Four-binary-search-solutions-based-on-different-ideas
  - [ ] https://leetcode.com/problems/preimage-size-of-factorial-zeroes-function/discuss/146026/C++-solution.-Beats-100-and-a-safe-lead!

734

1. 注意重复：

```
class Solution {
public:
    bool areSentencesSimilar(vector<string>& words1, vector<string>& words2, vector<pair<string, string>> pairs) {
        if(words1.size() != words2.size()) return false;
        unordered_map<string, unordered_set<string>> sim;
        for(auto p: pairs) sim[p.first].insert(p.second);
        for(int i=0; i<words1.size(); ++i) if(words1[i] != words2[i]){
            if((!sim.count(words1[i]) || !sim[words1[i]].count(words2[i])) && (!sim.count(words2[i]) || !sim[words2[i]].count(words1[i]))) return false;
        }
        return true;
    }
};
```

2.

```
class Solution {
```

```
    public boolean areSentencesSimilar(String[] words1, String
[] words2, String[][] pairs) {
        if (words1.length != words2.length) return false;
        Map<String, Set<String>> map = new HashMap<>();
        for (String[] pair : pairs) {
            if (!map.containsKey(pair[0])) map.put(pair[0], ne
w HashSet<>());
            if (!map.containsKey(pair[1])) map.put(pair[1], ne
w HashSet<>());

            map.get(pair[0]).add(pair[1]);
            map.get(pair[1]).add(pair[0]);
        }

        for (int i = 0; i < words1.length; i++) {
            if (!words1[i].equals(words2[i])) {
                if (!map.containsKey(words1[i]) || !map.get(wo
rds1[i]).contains(words2[i])) return false;
            }
        }
        return true;
    }
}
```

576

1. dp, don’t know why slow.

```
class Solution {
    const long M = long(1E9) + 7;
    long dp[55][55][55], dx[4]={1,-1,0,0};
    long dfs(int i, int j, int N, int &m, int &n){
        if(i<0 || i>=m || j<0 || j>=n) return 1;
```

```
        if(!N) return 0;
        if(dp[i][j][N] >= 0) return dp[i][j][N];
        dp[i][j][N] = 0L;
        for(int k=0; k<4; ++k) dp[i][j][N] += dfs(i+dx[k], j+dx[3-k], N-1, m, n);
        dp[i][j][N] %= M;
        return dp[i][j][N];
    }
public:
    int findPaths(int m, int n, int N, int i, int j) {
        memset(dp, -1, sizeof(dp));
        return dfs(i, j, N, m, n);
    }
};
```

- [ ] A fancy solution from LeetCode discussion: @Zebo L

```
int findPaths(int m, int n, int N, int i, int j) {
    unsigned int g[2][50][50] = {};
    while (N-- > 0)
        for (auto k = 0; k < m; ++k)
            for (auto l = 0, nc = (N + 1) % 2, np = N % 2; l < n; ++l)
                g[nc][k][l] = ((k == 0 ? 1 : g[np][k - 1][l]) + (k == m - 1 ? 1 : g[np][k + 1][l])
                    + (l == 0 ? 1 : g[np][k][l - 1]) + (l == n - 1 ? 1 : g[np][k][l + 1])) % 1000000007;
    return g[1][i][j];
}
```

696

1. One pass

```
class Solution {
public:
    int countBinarySubstrings(string s) {
        int ans = 0, pre = 0, cur = 1;
        for(int i=1; i<s.size(); ++i){
            if(s[i]==s[i-1]) ++cur;
            else{
                ans += min(pre, cur);
```

```
                pre = cur;
                cur = 1;
            }
        }
        return ans + min(pre, cur);
    }
};
```