【第四十五课】状态机模型与语言解释器(一)

1.trie_match

```cpp
/*************************************************************************
  > File Name: 1.trie_match.cpp
  > Author: huguang
  > Mail: hug@haizeix.com
  > Created Time:
 ************************************************************************/

#include <iostream>
#include <cstdio>
#include <cstdlib>
#include <queue>
#include <stack>
#include <algorithm>
#include <string>
#include <map>
#include <set>
#include <vector>
#include <unordered_set>
using namespace std;

#define BASE 26
struct Node {
    Node() : flag(false) {
        for (int i = 0; i < BASE; i++) next[i] = nullptr;
        return ;
    }
    bool flag;
    Node *next[BASE];
};

struct Trie {
public :
    Trie() = default;
    void insert(string s) {
        Node *p = &root;
        for (auto x : s) {
            int ind = x - 'a';
            if (p->next[ind] == nullptr) p->next[ind] = new Node();
            p = p->next[ind];
        }
        p->flag = true;
        return ;
    }
```

```
    unordered_set<string> match(string &s) {
        int cnt = 0;
        unordered_set<string> ret;
        for (int i = 0, n = s.size(); i < n; i++) {
            Node *p = &root;
            cnt += 1;
            for (int j = i; s[j]; j++) {
                int ind = s[j] - 'a';
                if (p->next[ind] == nullptr) break;
                p = p->next[ind];
                cnt += 1;
                if (p->flag) ret.insert(s.substr(i, j - i + 1));
            }
        }
        cout << "Total operator : " << cnt << endl;
        return ret;
    }
private:
    Node root;
};

int main() {
    int n;
    Trie tree;
    string s;
    cin >> n;
    for (int i = 0; i < n; i++) {
        cin >> s;
        tree.insert(s);
    }
    cin >> s;
    auto ans = tree.match(s);
    for (auto x : ans) cout << x << endl;
    cout << "find : " << ans.size() << " item(s)" << endl;
    return 0;
}
```

2.ac

```
/*************************************************************************
  > File Name: 1.trie_match.cpp
  > Author: huguang
  > Mail: hug@haizeix.com
  > Created Time:
 ************************************************************************/

#include <iostream>
#include <cstdio>
```

```cpp
#include <cstdlib>
#include <queue>
#include <stack>
#include <algorithm>
#include <string>
#include <map>
#include <set>
#include <vector>
#include <unordered_set>
using namespace std;

#define BASE 26
struct Node {
    Node() : flag(false), fail(nullptr) {
        for (int i = 0; i < BASE; i++) next[i] = nullptr;
        return ;
    }
    string *s;
    bool flag;
    Node *fail;
    Node *next[BASE];
};

struct Automaton {
public :
    Automaton() = default;
    void insert(string s) {
        Node *p = &root;
        for (auto x : s) {
            int ind = x - 'a';
            if (p->next[ind] == nullptr) p->next[ind] = new Node();
            p = p->next[ind];
        }
        if (p->flag == false) {
            p->flag = true;
            p->s = new string(s);
        }
        return ;
    }
    void build_ac() {
        queue<Node *> q;
        for (int i = 0; i < BASE; i++) {
            if (root.next[i] == nullptr) continue;
            root.next[i]->fail = &root;
            q.push(root.next[i]);
        }
        while (!q.empty()) {
            Node *now = q.front(), *p;
            q.pop();
```

```cpp
            for (int i = 0; i < BASE; i++) {
                if (now->next[i] == nullptr) continue;
                p = now->fail;
                while (p && p->next[i] == nullptr) p = p->fail;
                if (p) p = p->next[i];
                else p = &root;
                now->next[i]->fail = p;
                q.push(now->next[i]);
            }
        }
        return ;
    }
    unordered_set<string> match(string &s) {
        int cnt = 0;
        unordered_set<string> ret;
        Node *p = &root, *k;
        for (auto x : s) {
            // 状态转移
            int ind = x - 'a';
            while (p && p->next[ind] == nullptr) p = p->fail, cnt += 1;
            if (p) p = p->next[ind], cnt += 1;
            else p = &root, cnt += 1;
            // 提取结果
            k = p;
            while (k) {
                if (k->flag) ret.insert(*(k->s));
                k = k->fail;
            }
        }
        cout << "Total operator : " << cnt << endl;
        return ret;
    }
private:
    Node root;
};

int main() {
    int n;
    Automaton tree;
    string s;
    cin >> n;
    for (int i = 0; i < n; i++) {
        cin >> s;
        tree.insert(s);
    }
    tree.build_ac();
    cin >> s;
    auto ans = tree.match(s);
    for (auto x : ans) cout << x << endl;
```

```
    cout << "find : " << ans.size() << " item(s)" << endl;
    return 0;
}
```

3.ac_op

```
/*************************************************************************
  > File Name: 1.trie_match.cpp
  > Author: huguang
  > Mail: hug@haizeix.com
  > Created Time:
 ************************************************************************/

#include <iostream>
#include <cstdio>
#include <cstdlib>
#include <queue>
#include <stack>
#include <algorithm>
#include <string>
#include <map>
#include <set>
#include <vector>
#include <unordered_set>
using namespace std;

#define BASE 26
struct Node {
    Node() : flag(false), fail(nullptr) {
        for (int i = 0; i < BASE; i++) next[i] = nullptr;
        return ;
    }
    string *s;
    bool flag;
    Node *fail;
    Node *next[BASE];
};

struct Automaton {
public :
    Automaton() = default;
    void insert(string s) {
        Node *p = &root;
        for (auto x : s) {
            int ind = x - 'a';
            if (p->next[ind] == nullptr) p->next[ind] = new Node();
            p = p->next[ind];
        }
```

```
        if (p->flag == false) {
            p->flag = true;
            p->s = new string(s);
        }
        return ;
    }
    void build_ac() {
        queue<Node *> q;
        for (int i = 0; i < BASE; i++) {
            if (root.next[i] == nullptr) {
                root.next[i] = &root;
                continue;
            }
            root.next[i]->fail = &root;
            q.push(root.next[i]);
        }
        while (!q.empty()) {
            Node *now = q.front(), *p;
            q.pop();
            for (int i = 0; i < BASE; i++) {
                if (now->next[i] == nullptr) {
                    now->next[i] = now->fail->next[i];
                    continue;
                }
                now->next[i]->fail = now->fail->next[i];
                q.push(now->next[i]);
            }
        }
        return ;
    }
    unordered_set<string> match(string &s) {
        int cnt = 0;
        unordered_set<string> ret;
        Node *p = &root, *k;
        for (auto x : s) {
            // 状态转移
            int ind = x - 'a';
            p = p->next[ind], cnt += 1;
            // 提取结果
            k = p;
            while (k) {
                if (k->flag) ret.insert(*(k->s));
                k = k->fail;
            }
        }
        cout << "Total operator : " << cnt << endl;
        return ret;
    }
private:
```

```
    Node root;
};

int main() {
    int n;
    Automaton tree;
    string s;
    cin >> n;
    for (int i = 0; i < n; i++) {
        cin >> s;
        tree.insert(s);
    }
    tree.build_ac();
    cin >> s;
    auto ans = tree.match(s);
    for (auto x : ans) cout << x << endl;
    cout << "find : " << ans.size() << " item(s)" << endl;
    return 0;
}
```

4.P3808

```
/*************************************************************************
  > File Name: 1.trie_match.cpp
  > Author: huguang
  > Mail: hug@haizeix.com
  > Created Time:
 ************************************************************************/

#include <iostream>
#include <cstdio>
#include <cstdlib>
#include <queue>
#include <stack>
#include <algorithm>
#include <string>
#include <map>
#include <set>
#include <vector>
#include <unordered_set>
using namespace std;

#define BASE 26
#define MAX_N 1000000
struct Node {
    Node() : flag(0), fail(0) {
        for (int i = 0; i < BASE; i++) next[i] = 0;
        return ;
```

```
    }
    int flag;
    int fail;
    int next[BASE];
};

Node tree[MAX_N + 5];
int node_buff_cnt = 1;
int getNewNode() {
    return ++node_buff_cnt;
}

struct Automaton {
public :
    Automaton() : root(1) {}
    void insert(char *s) {
        int p = root;
        for (int i = 0; s[i]; i++) {
            int ind = s[i] - 'a';
            if (tree[p].next[ind] == 0) tree[p].next[ind] = getNewNode();
            p = tree[p].next[ind];
        }
        tree[p].flag += 1;
        return ;
    }
    void build_ac() {
        queue<int> q;
        for (int i = 0; i < BASE; i++) {
            if (tree[root].next[i] == 0) {
                tree[root].next[i] = root;
                continue;
            }
            tree[tree[root].next[i]].fail = root;
            q.push(tree[root].next[i]);
        }
        while (!q.empty()) {
            int now = q.front(), p;
            q.pop();
            for (int i = 0; i < BASE; i++) {
                if (tree[now].next[i] == 0) {
                    tree[now].next[i] = tree[tree[now].fail].next[i];
                    continue;
                }
                tree[tree[now].next[i]].fail = tree[tree[now].fail].next[i];
                q.push(tree[now].next[i]);
            }
        }
        return ;
    }
```

```
    int match(char *s) {
        int cnt = 0;
        int p = root, k;
        for (int i = 0; s[i]; i++) {
            // 状态转移
            int ind = s[i] - 'a';
            p = tree[p].next[ind];
            // 提取结果
            k = p;
            while (k) {
                cnt += tree[k].flag;
                tree[k].flag = 0;
                k = tree[k].fail;
            }
        }
        return cnt;
    }
private:
    int root;
};

char s[1000005];

int main() {
    int n;
    Automaton solve;
    scanf("%d", &n);
    for (int i = 0; i < n; i++) {
        scanf("%s", s);
        solve.insert(s);
    }
    solve.build_ac();
    scanf("%s", s);
    auto ans = solve.match(s);
    printf("%d\n", ans);
    return 0;
}
```

2216. 美化数组的最少删除数

给你一个下标从 **0** 开始的整数数组 `nums` ，如果满足下述条件，则认为数组 `nums` 是一个 **美丽数组** ：

- `nums.length` 为偶数
- 对所有满足 `i % 2 == 0` 的下标 `i` ， `nums[i] != nums[i + 1]` 均成立

注意，空数组同样认为是美丽数组。

你可以从 `nums` 中删除任意数量的元素。当你删除一个元素时，被删除元素右侧的所有元素将会向左移动一个单位以填补空缺，而左侧的元素将会保持 **不变** 。

返回使 `nums` 变为美丽数组所需删除的 **最少** 元素数目。

示例 1：

```
输入：nums = [1,1,2,3,5]
输出：1
解释：可以删除nums[0] 或nums[1] ，这样得到的nums = [1,2,3,5] 是一个美丽数组。可以证明，要想使 nums 变为美丽数组，至少需要删除 1 个元素。
```

```cpp
class Solution {
public:
    int minDeletion(vector<int>& nums) {
        int n = nums.size(), cnt = 0, a = nums[0];
        for (int i = 1; i < n; i++) {
            if (nums[i] == a) continue;
            cnt += 1;
            if (i + 1 == n) break;
            a = nums[i + 1];
        }
        return n - cnt * 2;
    }
};
```

1562. 查找大小为 M 的最新分组

给你一个数组 `arr` ，该数组表示一个从 `1` 到 `n` 的数字排列。有一个长度为 `n` 的二进制字符串，该字符串上的所有位最初都设置为 `0` 。

在从 `1` 到 `n` 的每个步骤 `i` 中（假设二进制字符串和 `arr` 都是从 `1` 开始索引的情况下），二进制字符串上位于位置 `arr[i]` 的位将会设为 `1` 。

给你一个整数 `m` ，请你找出二进制字符串上存在长度为 `m` 的一组 `1` 的最后步骤。一组 `1` 是一个连续的、由 `1` 组成的子串，且左右两边不再有可以延伸的 `1` 。

返回存在长度 **恰好** 为 `m` 的 **一组** `1` 的最后步骤。如果不存在这样的步骤，请返回 `-1` 。

示例 1：

```
输入：arr = [3,5,1,2,4], m = 1
输出：4
解释：
步骤 1："00100"，由 1 构成的组：["1"]
步骤 2："00101"，由 1 构成的组：["1", "1"]
步骤 3："10101"，由 1 构成的组：["1", "1", "1"]
步骤 4："11101"，由 1 构成的组：["111", "1"]
步骤 5："11111"，由 1 构成的组：["11111"]
存在长度为 1 的一组 1 的最后步骤是步骤 4 。
```

```cpp
class UnionSet {
```

```
public :
    UnionSet(int n) : fa(n + 1), size(n + 1), cnt(n + 2, 0) {
        for (int i = 0; i <= n; i++) {
            fa[i] = i;
            size[i] = 1;
        }
        cnt[1] = n + 1;
        return ;
    }
    int get(int x) {
        if (fa[x] == x) return x;
        return (fa[x] = get(fa[x]));
    }
    void merge(int a, int b) {
        int aa = get(a), bb = get(b);
        if (aa == bb) return ;
        fa[aa] = bb;
        cnt[size[bb]] -= 1;
        cnt[size[aa]] -= 1;
        size[bb] += size[aa];
        cnt[size[bb]] += 1;
        return ;
    }
    vector<int> fa, size, cnt;
};

class Solution {
public:
    int findLatestStep(vector<int>& arr, int m) {
        int n = arr.size(), ans = -1;
        UnionSet u(n);
        for (int i = 0; i < n; i++) {
            u.merge(arr[i], arr[i] - 1);
            if (u.cnt[m + 1]) ans = i + 1;
        }
        return ans;
    }
};
```

1574. 删除最短的子数组使剩余数组有序

给你一个整数数组 `arr` ，请你删除一个子数组（可以为空），使得 `arr` 中剩下的元素是 **非递减** 的。

一个子数组指的是原数组中连续的一个子序列。

请你返回满足题目要求的最短子数组的长度。

示例 1：

```
输入：arr = [1,2,3,10,4,2,3,5]
输出：3
解释：我们需要删除的最短子数组是 [10,4,2] ，长度为 3 。剩余元素形成非递减数组 [1,2,3,3,5] 。
另一个正确的解为删除子数组 [3,10,4] 。
```

```cpp
class Solution {
public:
    int findLengthOfShortestSubarray(vector<int>& arr) {
        int n = arr.size(), q = n - 1, ans = n + 1;
        while (q && arr[q - 1] <= arr[q]) --q;
        if (q == 0) return 0;
        ans = q;
        for (int i = 0; i < n; i++) {
            if (i && arr[i] < arr[i - 1]) break;
            while (q <= i || (q < n && arr[q] < arr[i])) q += 1;
            ans = min(q - i - 1, ans);
        }
        return ans;
    }
};
```

2226. 每个小孩最多能分到多少糖果

给你一个 **下标从 0 开始** 的整数数组 `candies` 。数组中的每个元素表示大小为 `candies[i]` 的一堆糖果。你可以将每堆糖果分成任意数量的 **子堆** ，但 **无法** 再将两堆合并到一起。

另给你一个整数 `k` 。你需要将这些糖果分配给 `k` 个小孩，使每个小孩分到 **相同** 数量的糖果。每个小孩可以拿走 **至多一堆** 糖果，有些糖果可能会不被分配。

返回每个小孩可以拿走的 **最大糖果数目** **。

示例 1：

```
输入：candies = [5,8,6], k = 3
输出：5
解释：可以将 candies[1] 分成大小分别为 5 和 3 的两堆，然后把 candies[2] 分成大小分别为 5 和 1 的两堆。现在就有五堆大小分别为 5、5、3、5 和 1 的糖果。可以把 3 堆大小为 5 的糖果分给 3 个小孩。可以证明无法让每个小孩得到超过 5 颗糖果。
```

```cpp
class Solution {
public:
    long long check(vector<int> &arr, long long k) {
        if (k == 0) return INT64_MAX;
        long long cnt = 0;
        for (auto x : arr) cnt += x / k;
        return cnt;
    }
    int maximumCandies(vector<int>& candies, long long k) {
```

```
        long long mid, l = 0, r = 0;
        for (auto x : candies) r = max(r, 1LL * x);
        r += 1;
        while (l < r) {
            mid = (l + r) >> 1;
            if (check(candies, mid) >= k) l = mid + 1;
            else r = mid;
        }
        return l - 1;
    }
};
```

1575. 统计所有可行路径

给你一个 **互不相同** 的整数数组，其中 `locations[i]` 表示第 `i` 个城市的位置。同时给你 `start`，`finish` 和 `fuel` 分别表示出发城市、目的地城市和你初始拥有的汽油总量

每一步中，如果你在城市 `i`，你可以选择任意一个城市 `j`，满足 `j != i` 且 `0 <= j < locations.length`，并移动到城市 `j`。从城市 `i` 移动到 `j` 消耗的汽油量为 `|locations[i] - locations[j]|`，`|x|` 表示 `x` 的绝对值。

请注意，`fuel` 任何时刻都 **不能** 为负，且你 **可以** 经过任意城市超过一次（包括 `start` 和 `finish`）。

请你返回从 ** `start` 到 `finish` 所有可能路径的数目。

由于答案可能很大，请将它对 `10^9 + 7` 取余后返回。

示例 1：

```
输入：locations = [2,3,6,8,4], start = 1, finish = 3, fuel = 5
输出：4
解释：以下为所有可能路径，每一条都用了 5 单位的汽油：
1 -> 3
1 -> 2 -> 3
1 -> 4 -> 3
1 -> 4 -> 2 -> 3
```

```
class Solution {
public:
    int f[105][205], mod_num = (int)(1e9+7);
    int getResult(int p, int d, int r, int n, vector<int> &c) {
        if (f[p][r] != -1) return f[p][r];
        if (p == d) f[p][r] = 1;
        else f[p][r] = 0;
        for (int i = 0; i < n; i++) {
            if (p == i) continue;
            if (abs(c[p] - c[i]) > r) continue;
            f[p][r] += getResult(i, d, r - abs(c[p] - c[i]), n, c);
            f[p][r] %= mod_num;
```

```
        }
        return f[p][r];
    }
    int countRoutes(vector<int>& locations, int start, int finish, int fuel) {
        memset(f, -1, sizeof(f));
        return getResult(start, finish, fuel, locations.size(), locations);
    }
};
```

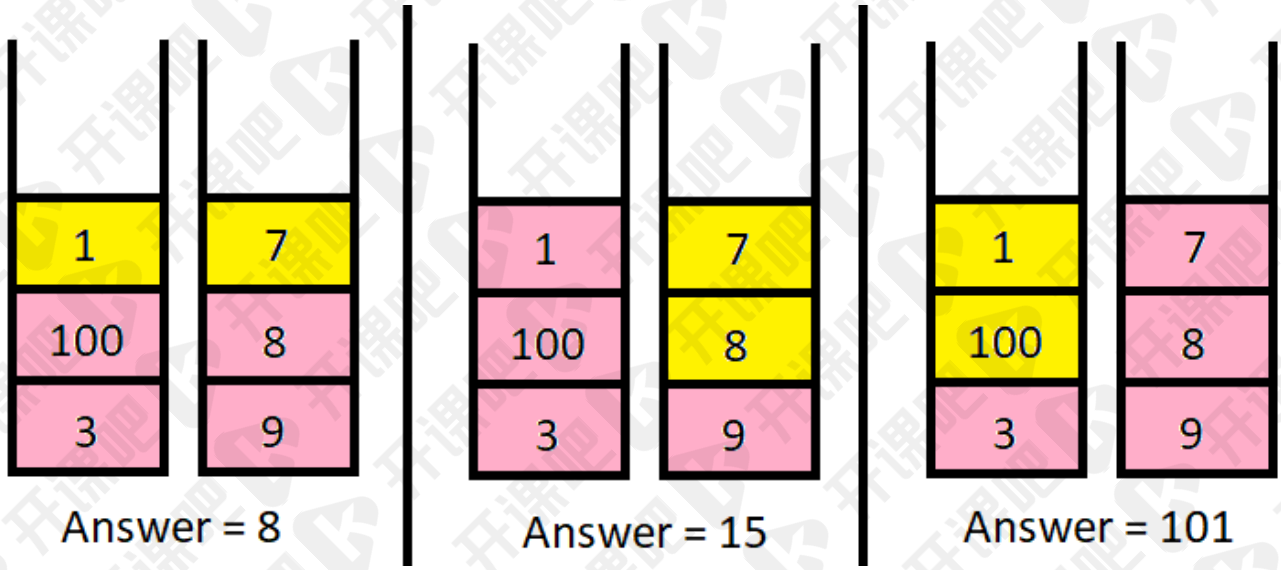
2218. 从栈中取出 K 个硬币的最大面值和

一张桌子上总共有 `n` 个硬币 **栈** 。每个栈有 **正整数** 个带面值的硬币。

每一次操作中，你可以从任意一个栈的 **顶部** 取出 1 个硬币，从栈中移除它，并放入你的钱包里。

给你一个列表 `piles` ，其中 `piles[i]` 是一个整数数组，分别表示第 `i` 个栈里 **从顶到底** 的硬币面值。同时给你一个正整数 `k` ，请你返回在 **恰好** 进行 `k` 次操作的前提下，你钱包里硬币面值之和 **最大为多少** 。

示例 1：




```
输入：piles = [[1,100,3],[7,8,9]], k = 2
输出：101
解释：
上图展示了几种选择 k 个硬币的不同方法。
我们可以得到的最大面值为 101 。
```

```
class Solution {
public:
    int maxValueOfCoins(vector<vector<int>>& piles, int k) {
        int n = piles.size();
        vector<vector<int>> dp(n + 1, vector<int>(k + 1, 0));
        for (int i = 1; i <= n; i++) {
            for (int j = 1; j <= k; j++) {
                dp[i][j] = dp[i - 1][j];
                int x = 0, y = 0;
```

```
                for (auto t : piles[i - 1]) {
                    x += 1, y += t;
                    if (x > j) break;
                    dp[i][j] = max(dp[i][j], dp[i - 1][j - x] + y);
                }
            }
        }
        return dp[n][k];
    }
};
```

1583. 统计不开心的朋友

给你一份 `n` 位朋友的亲近程度列表，其中 `n` 总是 **偶数** 。

对每位朋友 `i`，`preferences[i]` 包含一份 **按亲近程度从高到低排列** 的朋友列表。换句话说，排在列表前面的朋友与 `i` 的亲近程度比排在列表后面的朋友更高。每个列表中的朋友均以 `0` 到 `n-1` 之间的整数表示。

所有的朋友被分成几对，配对情况以列表 `pairs` 给出，其中 `pairs[i] = [xi, yi]` 表示 `xi` 与 `yi` 配对，且 `yi` 与 `xi` 配对。

但是，这样的配对情况可能会使其中部分朋友感到不开心。在 `x` 与 `y` 配对且 `u` 与 `v` 配对的情况下，如果同时满足下述两个条件，`x` 就会不开心：

- `x` 与 `u` 的亲近程度胜过 `x` 与 `y`，且
- `u` 与 `x` 的亲近程度胜过 `u` 与 `v`

返回 **不开心的朋友的数目** 。

示例 1：

```
输入：n = 4, preferences = [[1, 2, 3], [3, 2, 0], [3, 1, 0], [1, 2, 0]], pairs = [[0, 1], [2, 3]]
输出：2
解释：
朋友 1 不开心，因为：
-1 与 0配对，但1 与 3 的亲近程度比1 与 0 高，且
-3 与 1 的亲近程度比3 与 2 高。
朋友 3 不开心，因为：
- 3 与 2 配对，但3 与 1 的亲近程度比3 与 2 高，且
-1 与 3 的亲近程度比1 与 0 高。
朋友 0 和 2 都是开心的。
```

```
class Solution {
public:
    int unhappyFriends(int n, vector<vector<int>>& preferences, vector<vector<int>>& pairs) {
        vector<vector<int>> g(n, vector<int>(n, 0));
        vector<int> like(n);
        for (int i = 0; i < n; i++) {
```

```
            for (int j = 0; j < n - 1; j++) {
                g[i][preferences[i][j]] = j;
            }
        }
        for (auto x : pairs) {
            like[x[0]] = g[x[0]][x[1]];
            like[x[1]] = g[x[1]][x[0]];
        }
        int ans = 0;
        for (int i = 0; i < n; i++) {
            for (int j = 0, J = like[i]; j < J; j++) {
                int t = preferences[i][j];
                if (g[t][i] >= like[t]) continue;
                ans += 1;
                break;
            }
        }
        return ans;
    }
};
```

1585. 检查字符串是否可以通过排序子字符串得到另一个字符串

给你两个字符串 `s` 和 `t` ，请你通过若干次以下操作将字符串 `s` 转化成字符串 `t` ：

- 选择 `s` 中一个 **非空** 子字符串并将它包含的字符就地 **升序** 排序。

比方说，对下划线所示的子字符串进行操作可以由 `"1**4234**"` 得到 `"1**2344**"` 。

如果可以将字符串 `s` 变成 `t` ，返回 `true` 。否则，返回 `false` 。

一个 **子字符串** 定义为一个字符串中连续的若干字符。

示例 1：

```
输入：s = "84532", t = "34852"
输出：true
解释：你可以按以下操作将 s 转变为 t ：
"84532" （从下标 2 到下标 3）-> "84352"
"84352" （从下标 0 到下标 2） -> "34852"
```

```
class Solution {
public:
    bool isTransformable(string s, string t) {
        vector<queue<int>> pos(10);
        for (int i = 0; s[i]; i++) pos[s[i] - '0'].push(i);
        for (int i = 0; t[i]; i++) {
            if (pos[t[i] - '0'].empty()) return false;
            int p = pos[t[i] - '0'].front();
            for (int j = 0, J = t[i] - '0'; j < J; j++) {
```

```
                if (!pos[j].empty() && pos[j].front() < p) return false;
            }
            pos[t[i] - '0'].pop();
        }
        return true;
    }
};
```