

Assalamualaikum wr wb.

Mohammad Hanif Furqan Aufa Putra

5025221161

TUGAS 8 PBO C

Rare Order

🕒 Time limit **1** second

🧠 Memory limit **128** MiB

A rare book collector recently discovered a book written in an unfamiliar language that used the same characters as the English language. The book contained a short index, but the ordering of the items in the index was different from what one would expect if the characters were ordered the same way as in the English alphabet. The collector tried to use the index to determine the ordering of characters (i.e., the collating sequence) of the strange alphabet, then gave up with frustration at the tedium of the task.

You are to write a program to complete the collector's work. In particular, your program will take a set of strings that has been sorted according to a particular collating sequence and determine what that sequence is.

Input data

Consists of multiple test cases.

Each test case consists of an ordered list of strings of uppercase letters, one string per line. Each string contains at most 2020 characters. The end of the list is signalled by a line with the single character '#'. Not all letters are necessarily used, but the list will imply a complete ordering among those letters that are used.

Output data

For each test case print a single line containing uppercase letters in the order that specifies the collating sequence used to produce the input data.

Examples

Input example #1

```
XWY
ZX
ZXY
ZXW
YWWX
#
```

Output example #1

```
XZYW
```

Masalah yang dihadapi dalam soal ini dapat dijelaskan sebagai berikut. Seorang kolektor buku langka menemukan sebuah buku yang ditulis dalam bahasa yang tidak dikenal, namun menggunakan karakter yang sama dengan alfabet bahasa Inggris. Buku ini berisi indeks singkat, tetapi urutan item dalam indeks tersebut berbeda dari yang diharapkan jika karakter diurutkan sesuai dengan alfabet bahasa Inggris. Tugasnya adalah menentukan urutan karakter atau alfabet dari bahasa asing ini berdasarkan urutan indeks yang diberikan. Dengan kata lain, kita perlu mengembangkan program untuk menentukan urutan alfabet dari karakter-karakter yang digunakan dalam bahasa asing tersebut berdasarkan urutan indeks yang terlihat pada buku.

Untuk menyelesaikan tantangan ini, dapat diambil pendekatan dengan membentuk representasi graf yang mencerminkan hubungan urutan antar karakter, yang diambil dari indeks-indeks yang tersedia. Graf ini kemudian dapat dieksplorasi menggunakan algoritma seperti Topological Sort untuk menentukan urutan alfabet karakter-karakter tersebut. Proses ini melibatkan konstruksi matriks jarak antar karakter dan penerapan algoritma Floyd-Warshall untuk mengidentifikasi jarak terpendek antar karakter. Setelahnya, kita dapat menemukan urutan alfabet dengan mengekstrak urutan Topological dari graf yang terbentuk, membentuk suatu metode yang sistematis untuk menangani urutan alfabet dari bahasa asing yang terdapat dalam buku tersebut.

Desain solusi berbasis OOP dapat melibatkan pembuatan kelas-kelas seperti ``Graph`` untuk merepresentasikan graf hubungan antar karakter, ``AlphabetOrderFinder`` untuk mencari urutan alfabet, dan kelas-kelas lainnya yang diperlukan untuk memisahkan tanggung jawab dan memudahkan pemeliharaan kode.

Implementasi yang efisien dapat dicapai dengan menggunakan algoritma dengan kompleksitas waktu yang rendah, seperti Floyd-Warshall untuk mencari jarak terpendek. Selain itu, pemahaman yang baik tentang struktur data seperti matriks dan graf akan membantu meningkatkan efisiensi dan keterbacaan kode.

Kelas-kelas yang dibuat dapat mencakup ``CollatingSequenceSolver`` sebagai kelas utama yang berkoordinasi antara kelas-kelas lainnya, ``Graph`` untuk merepresentasikan graf hubungan karakter, dan ``AlphabetOrderFinder`` sebagai kelas yang bertanggung jawab mencari urutan alfabet.

Langkah-langkah desain kode dimulai dengan analisis struktur masalah, pembuatan representasi graf hubungan antar karakter, dan pengorganisasian kelas-kelas. Langkah selanjutnya melibatkan implementasi algoritma untuk mencari jarak terpendek dan algoritma Topological Sort. Kode kemudian diuji dengan menggunakan contoh masukan dan diperbaiki jika diperlukan.


Kode bekerja dengan cara membaca masukan test case, membangun graf dari hubungan antar karakter, menggunakan algoritma Floyd-Warshall untuk mencari jarak terpendek, dan kemudian menggunakan algoritma Topological Sort untuk menemukan urutan alfabet. Hasil akhirnya adalah urutan alfabet yang dicetak sebagai output untuk setiap test case.

Dibawah merupakan keseluruhan code


```
1 #include <iostream>
2 #include <vector>
3 #include <string>
4 #include <map>
5 #include <set>
6
7 using namespace std;
8
9 const int INF = 1e9;
10
11 void buildCollatingSequence(vector<string>& strings) {
12     set<char> uniqueChars;
13     for (const string &s : strings)
14         for (char c : s)
15             uniqueChars.insert(c);
16
17     map<char, int> charOrder;
18     int order = 0;
19     for (char c : uniqueChars)
20         charOrder[c] = order++;
21
22     int n = uniqueChars.size();
23     vector<vector<int>> dist(n, vector<int>(n, INF));
24
25     for (int i = 0; i < n; i++)
26         dist[i][i] = 0;
27
28     for (int i = 0; i < strings.size() - 1; i++) {
29         const string &s1 = strings[i];
30         const string &s2 = strings[i + 1];
31         int len = min(s1.length(), s2.length());
32
33         for (int j = 0; j < len; j++) {
34             char c1 = s1[j];
35             char c2 = s2[j];
36             if (c1 != c2) {
37                 dist[charOrder[c1]][charOrder[c2]] = 1;
38                 break;
39             }
40         }
41     }
42
43     for (int k = 0; k < n; k++)
44         for (int i = 0; i < n; i++)
45             for (int j = 0; j < n; j++)
46                 if (dist[i][k] + dist[k][j] < dist[i][j])
47                     dist[i][j] = dist[i][k] + dist[k][j];
48
49     string collatingSequence;
50     for (int i = 0; i < n; i++) {
51         char currentChar = ' ';
52         int minIncomingEdges = INF;
53         for (const pair<char, int> &p : charOrder) {
54             char c = p.first;
55             int charIndex = p.second;
56             int incomingEdges = 0;
57             for (int j = 0; j < n; j++)
58                 if (j != charIndex && dist[j][charIndex] != INF)
59                     incomingEdges++;
60
61             if (incomingEdges < minIncomingEdges) {
62                 minIncomingEdges = incomingEdges;
63                 currentChar = c;
64             }
65         }
66         collatingSequence += currentChar;
67         charOrder.erase(currentChar);
68     }
69
70     cout << collatingSequence << endl;
71 }
72
73 int main() {
74     char line[22];
75     while (true) {
76         vector<string> strings;
77         while (true) {
78             scanf("%s", line);
79             if (line[0] == '#') break;
80             strings.push_back(string(line));
81         }
82
83         if (strings.empty()) break;
84
85         buildCollatingSequence(strings);
86     }
87
88     return 0;
89 }
90
```

Dengan menjalankan program tersebut diperoleh hasil seperti dibawah ini

Success #stdin #stdout 0.01s 5284KB [comments \(0\)](#)

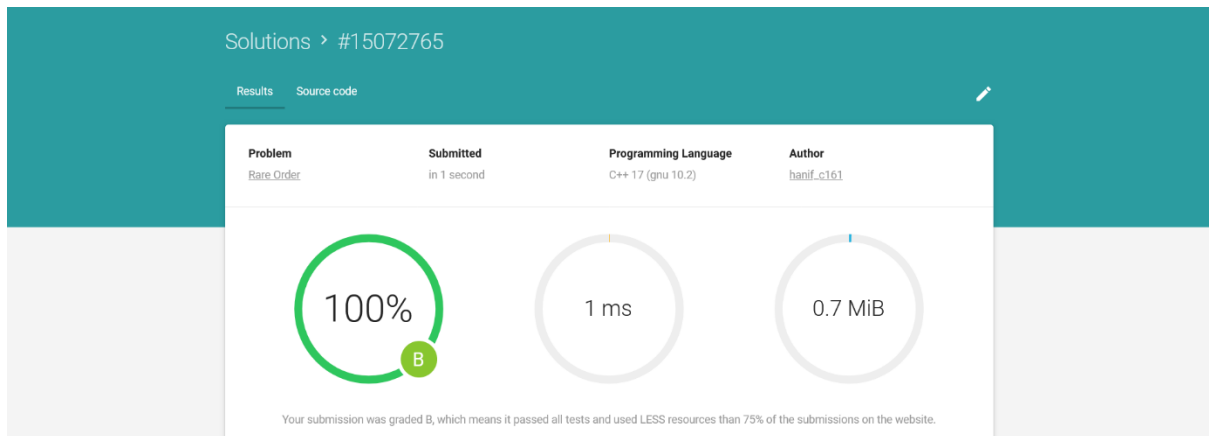
 stdin [copy](#)

XWY
ZX
ZXY
ZXW
YWWX
#

 stdout [copy](#)

XZYW

Dan dibawah bukti Verdict Accept



Terima Kasih telah membaca, Waalaikumsalam wr wb.