Báo cáo bài 054: Thứ tự từ điển.

1. **Đề bài:**

#### Một bảng danh mục gồm các từ đã được sắp xếp theo một trật tự từ điển nào đấy (không nhất thiết là từ điển thông thường). Yêu cầu từ bảng danh mục, hãy khôi phục lại trật tự từ điển đã dùng.

**Dữ liệu** vào được cho bởi file văn bản NOTE.INP. Dòng đầu là số lượng từ, các dòng tiếp, (theo thứ tự) mỗi dòng là một từ trong bảng danh mục. Giả thiết rằng mỗi từ đều không quá 20 ký tự được lấy trong bảng chữ cái nhỏ tiếng Anh (từ 'a' đến 'z'). Số lượng từ trong bảng danh mục không quá 10000.

* + **Kết quả** đưa ra file văn bản NOTE.OUT gồm một dòng là xâu gồm các chữ cái đã xuất hiện trong bảng danh mục. Các chữ cái trong xâu viết liền nhau và theo thứ tự phù hợp với trật tự từ điển đã dùng.

|  |  |  |
| --- | --- | --- |
| **NOTE.INP** |  | **NOTE.OUT** |
| **10**  **svxngqqnsnvqv snngg qsqsqvgsqq qqns qnvq nsxnxnvsqsvvs nqg nn**  **xsgvsgggqvsqqsxgv xxgxxggsvnxsnxsnqq** |  | **gsvqnx** |

1. **Ý tưỡng:**

* Thuật toán: dựa trên bài toán Alien Dictionary và thuật toán [Topological Sorting](https://www.geeksforgeeks.org/topological-sorting/)
* Ý tưỡng: Ý tưởng là tạo một đồ thị các ký tự và sau đó tìm cách [Topological Sorting](https://www.geeksforgeeks.org/topological-sorting/) của đồ thị đã tạo.
* Ví dụ:
  + Input: [ wrt , wrf , er, ett , rftt ]
  + Output: wertf
  + Ta xét các cặp từ trong Input để tìm ra thứ tự từ điển như sau:

|  |  |  |
| --- | --- | --- |
| STT | Xét cặp từ | Kết quả |
| 1 | **wrt , wrf**  Ta thấy:   * - 2 kí tự đầu wr trùng nhau. * - kí tự cuối t và f khác nhau. * - và t đứng trước f |  |
| 2 | **wrf , er**  Ta thấy:   * - w khác e và w đứng trước e |  |
| 3 | **er, ett**  **Ta thấy:**   * **- kí tự đầu e trùng nhau.** * **- kí tự tiếp theo r và t khác nhau.** * **- và r đứng trước t** |  |
| 4 | **ett , rftt**  Ta thấy:  - e khác r và e đứng trước r  **- t** khác f và t đứng trước f(đã xét)   * **- kí tự cuối t trùng nhau.** |  |
|  | **Từ 1, 2, 3, 4, 5** |  |

1. **Mã giả**

* Chuẩn bị

+ Một đồ thị degree

+ Xác định một bản đồ được gọi là đồ thị

+ n = kích thước của từ

* For khởi tạo i= 0, i<size of words , tăng i lên 1 , do

For khởi tạo j = 0, khi j < size of words[i], tăng j lên 1), do

degree[words[i, j]] = 0

* For i=0, khi  i < n – 1 , tăng i lên 1, do

i := minimum of size of words[i] and size of words[i + 1]

For j: = 0, khi j < i, tăng j lên 1, do

x := words[i, j]

y: = words [i + 1, j]

if x không bằng y, then

* + - 1. chèn y vào cuối graph[x]
      2. (tăng degree[y] lên 1)
      3. Thoát vòng lặp
* ret: = chuỗi trống
* Define queue q
* For mỗi cặp key-value trong, hãy thực hiện -

If giá trị của nó bằng 0, then–

insert key vào q

* while (not q is empty), do

x = phần tử đầu tiên của q

xóa phần tử khỏi q

ret: = ret + x

for each sit trong graph do

decrease degree[sit] by 1

if degree[sit] = 0, then –

insert sit into q

(increase sit by 1)

* return (if size of ret = size of degree thì return ret, nếu không return chuỗi trống).

1. **Code**

#include <bits/stdc++.h>

using namespace std;

int n;

vector<string> v;

void docFile(){

ifstream ifs("NOTE.INP");

if(!ifs){

cerr << "Error: file not opened." << endl;

return ;

}

string str;

ifs >> str;

while(ifs >> str){

// cout << str << endl;

v.push\_back(str);

}

}

void ghiFile(string s)

{

ofstream f2("NOTE.OUT");

f2<<s;

f2.close();

}

string alienOrder(vector<string>& words)

{

map<char, int> degree;

map<char, vector<char> > graph;

int n = words.size();

for (int i = 0; i < words.size(); i++) {

for (int j = 0; j < words[i].size(); j++) {

degree[words[i][j]] = 0;

}

}

for (int i = 0; i < n - 1; i++) {

int l = min((int)words[i].size(), (int)words[i + 1].size());

for (int j = 0; j < l; j++) {

char x = words[i][j];

char y = words[i + 1][j];

if (x != y) {

graph[x].push\_back(y);

degree[y]++;

break;

}

}

}

string ret = "";

queue<char> q;

map<char, int>::iterator it = degree.begin();

while (it != degree.end()) {

if (it->second == 0) {

q.push(it->first);

}

it++;

}

while (!q.empty()) {

char x = q.front();

q.pop();

ret += x;

vector<char>::iterator sit = graph[x].begin();

while (sit != graph[x].end()) {

degree[\*sit]--;

if (degree[\*sit] == 0) {

q.push(\*sit);

}

sit++;

}

}

return ret.size() == degree.size() ? ret : "";

}

int main()

{

//vector<string> v = { "svxngqqnsnvqv", "svxngqqnsnvqv", "svxngqqnsnvqv", "qqns", "qnvq", "nsxnxnvsqsvvs", "nqg", "nn", "xsgvsgggqvsqqsxgv", "xxgxxggsvnxsnxsnqq" };

docFile();

cout<<alienOrder(v);

ghiFile(alienOrder(v));

}