Programming Technology: Xử lý xâu với phép tách từ

Tran Viet Khoa
tvkhoa.husc@gmail.com

Hue University— Ngày 30 tháng 3 năm 2020

Mục lục

1 Giới thiệu

Bài toán tách từ (Tokenizing a string) là một bài toán cơ bản và được ứng dụng nhiều trong các bài toán thực tế. Ví dụ như:

1. Xử lý ngôn ngữ tự nhiên.
2. Xử lý văn bản, tìm kiếm và nhận dạng từ.
3. xử lý cú pháp lệnh của trình biên dịch.

Những ngôn ngữ tiên tiến như Python, Java cung cấp các thư viện rất mạnh mẽ cho xử lý trên. Đối với ngôn ngữ C các hàm strtok() và strtok_r() của thư viện <string.h> xử lý cho công việc này. Riêng đối với lớp chứa string của STL C++ thì ta cũng có thể sử dụng các hàm trên bằng cách chuyển qua chuỗi của ngôn ngữ C và gọi hàm trên.

Bài hướng dẫn này trình bày cách xây dựng hàm xử lý bài tách từ của xâu thuộc lớp chứa string bằng cách dùng các hàm của thư viện <algorithm> và thư việc <regex> để cài đặt. Các cài đặt này rất hiệu quả và có tính công nghệ, tức là giúp bạn có thể tiếp cận hàm tương tự như vậy trên các ngôn ngữ lập trình khác.

Nội dung bao gồm:

- Cơ bản về lớp string.
- Cài đặt các bài toán tách từ bằng thư viện algorithm.
- Cài đặt các bài toán tách từ bằng thư viện regex.

Thông tin

+ Một số bài toán thực hành có thể dễ dàng cài đặt với mảng ký tự và với thuật toán đơn giản, tuy nhiên tính tổng quát là không cao.

2 Cơ bản về lớp string

2.1 Khai báo

- Khai báo thư viện:

```
#include <string>
using namespace std;
```

Hoặc thay thế bằng #include <bits/stdc++.h>

- Khai báo đối tượng.

```
// Khai báo đối tượng str.
string str;
// Khai báo đối tượng str bằng hàm dựng
string str("abc");
```

2.2 Nhập dữ liệu từ stdin, stream

- Sử dụng toán tử tải bội « với đối tượng **cin**, Ví dụ:

```
string firstName;
cin >> firstName; // lấy dữ liệu từ bộ đệm bàn phím stdin.
cout << firstName;
```

Nhược điểm của toán tử này nó coi ký tự khoảng như (tab \t, space ' ') là ký tự kết thúc cho nên ta không quét được hết dữ liệu.

- Sử dùng hàm **getline()**, ví dụ:

```
string fullName;
getline (cin, fullName);
cout << fullName;
```

2.3 Độ dài của chuỗi

- Sử dụng hàm **length()**, Ví dụ:

```
string str = "ABCDEFGHIJKLMNOPQRSTUVWXYZ";
cout << str.length();
```

Thông tin

+ Vì lớp string là vector các ký tự cho nên hàm **size()** được xem như là bí danh (alias) của **length()**. Bạn có thể dùng hai hàm trên.

2.4 Phép ghép xâu

- Sử dụng toán tử tải bội +, ví dụ:

```
string firstName = "Khoa";
string lastName = "Tran Viet";
string fullName = firstName + " " + lastName;
cout << fullName;
```

- Sử dụng hàm **append()**, ví dụ:

```
string firstName = "Khoa";
string lastName = "Tran Viet";
string fullName = firstName.append(lastName);
cout << fullName;
```

2.5 Phép truy cập và duyệt

- Sử dụng toán tử tải bội [] hoặc hàm **at()**, ví dụ:

```
   //Khai báo và dựng (hàm dựng) đối tượng.
string str ("Programming is Fun");
//Duyệt in bằng chỉ mục với toán tử [].
//tương tự mảng chỉ số i=0..str.length()-1.
for (unsigned i=0; i<str.length(); ++i)
   cout << str[i];
//Duyệt in bằng chỉ mục với hàm at().
for (unsigned i=0; i<str.length(); ++i)
   cout << str.at(i);
```

- Sử dụng con trỏ **iterator** để duyệt xâu với các hàm **begin(), end(), rbegin(), rend()**, ví dụ:

```
// Khai báo và khởi tạo chuỗi.
string str = "Dogma I am God";
// Khai báo biến lặp.
string::iterator it;
// Khai báo biến lặp ngược.
string::reverse_iterator it1;
// Duyệt in các ký tự của chuỗi nhờ biến lặp.
cout << "The string using forward iterators is : ";
for (it=str.begin(); it!=str.end(); it++)
   cout << *it;
cout << endl;
// Duyệt chiều ngược lại bằng con trỏ ngược.
```

```
13 cout << "The reverse string using reverse iterators is : ";
14 for (it1=str.rbegin(); it1!=str.rend(); it1++)
15    cout << *it1;
16 cout << endl;
```

3 Bài toán tách từ

Tách từ từ chuỗi (string tokenization) là phép tách một chuỗi thành các phần gọi là token, ví dụ str="Programming is fun" sẽ được tách thành "Programming", "is" và "fun". Phép toán này được dùng rất nhiều trong chương trình dịch nhằm phân tích một câu cú pháp lệnh để biết được đâu là biến, hàm, hằng số, phép toán.

Nếu một xâu được tiền xử lý và đẹp như xâu ví dụ trên thì bài toán này rất đơn giản. Trong thực tế, ta gặp nhiều bài toán xử lý chia xâu thành các token phức tạp hơn, ví dụ xét xâu: str=" *Good friends, good books, and a sleepy conscience: this is the ideal life. — Mark Twain*". Nhận thấy xâu này có các ký tự dấu phẩy, chấm, hai chấm, gạch ngang để phân tách từ ngoài ký tự trắng thông dụng. Nếu câu viết không đúng cú pháp thì việc tách càng khó, ví dụ: str=" hello again..... sea you late...! ".

Thông thường đối với những xâu như trên, người ta sẽ mất một công sức cho công việc gọi là tiền xử lý làm đẹp xâu, ví dụ xóa ký tự trắng thừa, xóa ký tự sai cú pháp. Và như vậy bài toán trở nên rất phức tạp.

Bài toán đặt ra cho các bạn là: Cho một xâu *str*, việc phân cách từ trong xâu bằng cách sử dụng các ký tự đặc biệt (người ta gọi là delimiter) có thể được liệt kê trước như dấu chấm, phẩy, hai chấm, chấm phẩy, than, hoa thị,.. và cả dấu cách (space). Hãy tách các từ trên.

3.1 Xử lý xâu bằng các hàm algorithm

Bài tập 1: Tìm từ dài nhất

Cho một xâu ký tự gồm các ký tự Alphabet ở dạng chữ hoa và chữ thường, ngoài ra còn có các ký tự đặc biệt sau: apostrophe,full stop,comma,semicolon,colon, ký tự space và newline. Trong các ký tự trên chỉ có ký tự apostrophe là dùng để nối từ, các ký tự còn lại nếu đúng cú pháp là để phân biệt từ. Hãy lập trình tìm từ có chiều dài lớn nhất trong xâu cho trước.

Input

+ Gồm nhiều dòng, mỗi dòng là một xâu ký tự có tối đa 80 ký tự.

Output

+ Ứng với mỗi dòng in chiều dài lớn nhất của các từ tương ứng trong các xâu.

Samples

input	output
If you're smart you will solve this right.	6

Tham khảo:

Bài tập PT074, http://oj.hueuni.edu.vn/practice/problem/174/details

Bài toán này được giải bởi hai bước chính sau:

1. Đọc xâu *str* và tách ra từng từ dựa vào việc nhận diện ký tự phân tách. Kết quả lưu vào vector.
2. Quá đơn giản cho việc tìm xâu có độ dài lớn nhất trong một vector chứa chúng.

Bài toán cần các hàm sau để giải.

1. Hàm **find_first_not_of()** - Hàm này dùng tìm kiếm sự vắng mặt (absence) tức là không có ký tự ở trong chuỗi. Tìm kiếm chuỗi cho ký tự đầu tiên không khớp với bất kỳ ký tự nào được chỉ định trong các đối số của nó. Khi pos được chỉ định, tìm kiếm chỉ bao gồm các ký tự tại hoặc sau vị trí pos, bỏ qua các ký tự trước vị trí pos đó.

   Kết quả trả về là vị trí đầu tiên không phù hợp, nếu không tìm thấy trả về hằng **string::npos**. Kiểu của trị trả về là **size_t** (số nguyên không âm).

   Xét ví dụ mã nguồn sau:

```
string str ("look for non-alphabetic characters...");
size_t found = str.find_first_not_of("abcdefghijklmnopqrstuvwxyz ");
if (found!=string::npos){
   cout << "The first non-alphabetic character is " << str[found];
   cout << " at position " << found << '\n';
}

Output:
The first non-alphabetic character is - at position 12.
```

   Như vậy với hàm trên ta lặp đi lặp lại việc tìm các ký tự phân cách, xác định được vị trí của chúng để sao chép các chuỗi con lưu vào vector nhưng cần thêm một địa chỉ nữa và được biết đến với hàm thứ hai.

2. Hàm thứ hai là **find_first_of()** - Hàm tìm ký tự trong chuỗi. Tìm kiếm chuỗi cho ký tự đầu tiên khớp với bất kỳ ký tự nào được chỉ định trong các đối số của nó.

   Kết quả trả về là vị trí đầu tiên tìm thấy, nếu không tìm thấy trả về hằng **string::npos**. Kiểu của trị trả về là **size_t** (số nguyên không âm).

   Xét ví dụ mã nguồn sau:

```
string str ("Please, replace the vowels in this sentence by asterisks.");
// Tìm các nguyên âm.
size_t found = str.find_first_of("aeiou");
while (found!=string::npos){
   // Tìm thấy và thay thế bằng dấu *.
   str[found]='*';
   //Tìm tiếp cho vị trí tiếp theo. lặp cho đến khi gặp ::npos.
   found=str.find_first_of("aeiou",found+1);
}
cout << str << endl;

Output:
   Pl**s*, r*pl*c* th* v*w*ls *n th*s s*nt*nc* by *st*r*sks.
```

3. Cuối cùng là hàm copy xâu con **string substr (size_t pos = 0, size_t len = npos) const;** - Gọi là hàm tạo xâu con. Trả về một đối tượng chuỗi được xây dựng mới với giá trị của nó được khởi tạo từ một bản sao (copy) của một chuỗi con của đối tượng chuỗi gọi hàm.

   Chuỗi con là một phần của đối tượng bắt đầu tại vị trí ký tự **pos** và kéo dài các ký tự với chiều dài **len** (hoặc cho đến khi kết thúc chuỗi, tùy theo điều kiện nào đến trước).

   Xét ví dụ mã nguồn sau:

```
string str="We think in generalities, but we live in details.";
string str2 = str.substr (3,5);     // Cắt được xâu con "think"
std::size_t pos = str.find("live"); // Trả về địa chỉ của xâu "live" trong str.
string str3 = str.substr (pos);     // cắt được xâu con "live" cho đến cuối.
cout << str2 << ' ' << str3 << '\n';
Output:
think live in details.
```

   Với sự kết hợp của ba hàm trên, thuật toán cho bài toán trình bày (accept) với mã nguồn:

a.cpp

```cpp
#include <bits/stdc++.h>
using namespace std;
using VS = vector<string>;

VS split(const string& text, const string& delim){
   VS tokens;
   size_t start = text.find_first_not_of(delim), end = 0;
   while((end = text.find_first_of(delim, start)) != string::npos){
      tokens.push_back(text.substr(start, end - start));
      start = text.find_first_not_of(delim, end);
   }
   if(start != string::npos) tokens.push_back(text.substr(start));
   return tokens;
}
int main()
{
   // Khai báo các ký tự phân tách.
   string delims="; :.!,";
   string s;
   //Vòng lặp đọc lần lượt các xâu.
   while (getline(cin, s)){
      // Phân tách và lưu vào vector.
      VS tokens = split(s, delims);
      //Tìm max.
      unsigned int max=0;
      for (auto &s: tokens) {
         if (s.size() > max) max =s.size();
      }
      cout<< max <<endl;
      //Xóa vector dọn chỗ cho lần lặp tiếp.
      tokens.clear();
   }
   return 0;
}
```

Nhận xét

+ Với bài toán này, nhiều bạn nghĩ đơn giản hơn nhiều và đã accept được, ví dụ duyệt qua các ký tự xâu và kiểm tra xem có phải ký tự alphabet (isalpha) hoặc ký tự mã #39, nếu thỏa thì tăng biến đếm. so sánh với max để in kết quả. Nhưng nếu xâu không phải alphabet thì bài toán không giải quyết được.

+ Ý thứ hai quan trọng là bài toán lưu được các từ để dùng xử lý sau này.

3.2 Xử lý xâu bằng các hàm regex

Bài toán thứ hai hoàn toàn tương tự, nội dung như sau:

Bài tập 2: Số từ và từ dài nhất

Một xâu ký tự có thể bao gồm các ký tự Alphabet, ký tự trắng, dấu phẩy (,), dấu chấm (.), dấu hai chấm(:). Một từ là một xâu con chỉ chứa các ký tự Alphabet, ngoài ra hai ký tự liền trước và liền sau của xâu con đó (nếu có) sẽ không phải là ký tự Alphabet. Hãy lập trình tìm từ có chiều dài lớn nhất trong xâu và số từ của xâu đó.

Input

+ Gồm một dòng duy nhất là một xâu ký tự có tối đa 200 ký tự.

Output

+ Dòng thứ nhất in chiều dài lớn nhất của các từ trong xâu.
+ Dòng thứ hai in số từ của xâu.

Samples

input	output
Programming Java in oj	11 6

Tham khảo:

Bài tập PT075, http://oj.hueuni.edu.vn/practice/problem/175/details

Nhận xét

+ Quá dễ cho bài toán này khi nội dung hoàn toàn tương tự, thậm chí các ký tự phân cách lại ít hơn.
+ Accept thôi, chỉnh lại tí nộp.

Mã nguồn:

bNotAc.cpp

```cpp
#include <bits/stdc++.h>
using namespace std;
using VS = vector<string>;
//Sử dụng hàm đã ac ở câu a.
VS split(const string& text, const string& delim){...
}
int main()
{
   string delims = " ,.:";
   string s;
   getline(cin, s);
   tokens = split(s, delims);
   unsigned int max=0;
   for (auto &s: tokens) {
      if (s.size() > max) max =s.size();
   }
   cout<< max <<endl;
   cout<<tokens.size();
   return 0;

}
```

Thách thức

+ Chiều nộp hơn 10 lần vẫn WA, mới phát hiện ra testcase bài 174 yếu hơn, thiếu những trường hợp là một loạt ký tự phân cách dính liền nhau và debug mãi không được. Thầy nhờ mấy em debug tiếp, ví dụ sau là sai: "Hoc, hoc nua, hoc mai... hehe", kết quả ra: 4, 6 mà thuật mình ra 6, 6. Sửa một hồi ra 5, 6 gần đúng, toát mồ hôi. Ở đây cũng gặp trường hợp chạy máy mình báo đúng, không dùng DevC mà dùng command Line, bực quá chạy trên IDEone lúc đúng, lúc sai nhưng nộp là sai.

+ Cuối cùng nghĩ phương án khác, hồi trước có làm về xử lý chuỗi nhớ có sử dụng cái là regex kiểu như compiler trên shell linux. Thế là nhờ Giáo sư Goggle, tìm và thử cuối cùng cũng accept được câu b. Có điều giải thích nó cũng hơi mệt. mà hôm nay ngồi nguyên một ngày. Nhờ các bạn đọc hộ và phản biện.

Mã nguồn:

bAccept.cpp

```
#include <bits/stdc++.h>
using namespace std;

int main()
{
   string s;
   getline(cin,s);
   regex words_regex("[^\\s.,:;!?]+");
   auto words_begin = sregex_iterator(s.begin(), s.end(), words_regex);
   auto words_end = sregex_iterator();

   unsigned int max=0,c=0;
   for (sregex_iterator i = words_begin; i != words_end; ++i){
      // Nhác quá cũng không lưu vector, chứ (*i).str() chính là các từ được tách ra.
      if ((*i).str().length() >max)
      max = (*i).str().length();
      c++;
   }
   cout<<max<<endl;
   cout<<c<<endl;
return 0;
}
```