MỤC LỤC

CHUYÊN ĐỀ DỮ LIỆU KIỂU XÂU TRONG C++

A. MỞ ĐẦU

1. Cơ sở khoa học về việc lựa chọn đề tài

Trong thực tiễn dữ liệu vào của các bài toán đều liên quan đến các kiểu dữ liệu khác nhau, để tiện cho việc lập trình và xử lý dữ liệu chúng ta thường đưa dữ liệu đó về các dạng kiểu dữ liệu chuẩn hoặc kiểu dữ liệu có cấu trúc, một trong những kiểu dữ liệu chuẩn đó là kiểu xâu.

Qua quá trình tham gia giảng dạy và bồi dưỡng học sinh giỏi chúng tôi nhận thấy dữ liệu kiểu xâu thường gặp rất nhiều trong các bài toán và vận dụng linh hoạt các thao tác xử lý trên kiểu dữ liệu này vào bài toán không phải là dễ. Với mong muốn phần nào giúp học sinh mới học trong việc tìm ra lời giải cho một số bài toán liên quan tới kiểu dữ liệu xâu và chuỗi được dễ dàng hơn, chúng tôi xin giới thiệu " Chuyên đề dữ liệu kiểu trong C++" mà chúng tôi đã áp dụng trong quá trình giảng dạy.

2. Mục đích nhiệm vụ của đề tài

Hiện nay có rất nhiều chuyên đề viết về dữ liệu kiểu xâu chuyên đề của tôi nhằm đưa ra những bài toán cơ bản nhất giúp cho học sinh mới học tiếp cận với kiểu dữ liệu kiểu xâu và mỗi bài toán được tham khảo từ các nguồn khác nhau.

Việc tìm lời giải cho các bài toán kiểu xâu, là rất cần thiết nhằm giúp cho giáo viên, cũng như học sinh hệ thống lại các kiến thức về các thao tác trên kiểu dữ liệu xâu và phân dạng bài tập, từ đó áp dụng cho các bài toán cụ thể. Chúng tôi đề ra mục đích, nhiệm vụ cụ thể của việc thực hiện đề tài:

- Giới thiệu về kiểu dữ liệu kiểu xâu trong C++
- Giới thiệu một số phép toán về dữ liệu kiểu xâu và string
- Hệ thống bài tập từ đơn giản đến phức tạp của kiểu dữ liệu xâu trong C++.

B. LÝ THUYẾT

1. DỮ LIỆU KIỂU XÂU

1.1. Khái niệm

- Xâu là một dãy kí tự trong bảng mã ASCII. Mỗi kí tự được gọi là một phần tử của xâu.
- Số lượng kí tự trong xâu được gọi là độ dài của xâu .
- Xâu có độ dài bằng 0 gọi là xâu rỗng

1.2. Khai báo mảng xâu

```
char <tên xâu>[độ dài] ;                 // không khởi tạo
char <tên xâu>[độ dài] = xâu kí tự ;     // có khởi tạo
char <tên xâu>[] = xâu kí tự ;           // có khởi tạo
```

- Độ dài mảng là số kí tự tối đa có thể có trong xâu. Độ dài thực sự của xâu chỉ tính từ đầu mảng đến dấu kết thúc xâu (không kể dấu kết thúc xâu '\0').

- Ví dụ:

char a[26] ; // mảng xâu a chứa tối đa 25 kí tự
char b[31] = “abcndd” ;

1.3. Phương thức nhập mảng xâu

Do toán tử nhập cin >> có hạn chế đối với xâu kí tự nên C++ đưa ra hàm riêng (còn gọi là phương thức) **cin.getline(s,n)** để nhập xâu kí tự.
Xét đoạn lệnh sau

```
char s[10] ;
cin.getline(s, 10) ;
cout << s << endl ;
```

1.4. Một số hàm xử lí xâu

- **strcpy(s, t)** // Gán nội dung của xâu t cho xâu s (thay cho phép gán = không được dùng).
- **strncpy(s, t, n)** // Sao chép n kí tự của t vào s.
- **strncat(s, t, n)** // Nối bản sao n kí tự đầu tiên của xâu t vào sau xâu s.
- **strcmp(s, t)** // Hàm so sánh 2 xâu s và t (thay cho các phép toán so sánh). **strncmp(s, t)** // Giống hàm strcmp(s, t) nhưng chỉ so sánh tối đa n kí tự đầu tiên của hai xâu.
- **strcmpi(s, t) ;** // Như strcmp(s, t) nhưng không phân biệt chữ hoa, thường.
- **strupr(s)** // Hàm đổi xâu s thành in hoa, và cũng trả lại xâu in hoa đó.
- **strlwr(s);** // Hàm đổi xâu s thành in thường, kết quả trả lại là xâu s.
- **strlen(s)** // Hàm trả giá trị là độ dài của xâu s.

1.5. Sau đây là một số ví dụ sử dụng tổng hợp các hàm trên.

Ví dụ 1 : Thống kê số chữ 'a' xuất hiện trong xâu s.

```
main()
{
   const int MAX = 100;
   char s[MAX+1];
      int sokitu = 0;
      cin.getline(s, MAX+1);
```

```
    for (int i=0; i < strlen(s); i++)   if (s[i] = 'a ') sokitu++;
    cout << "Số kí tự = " << sokitu << endl ;
}
```

Ví dụ 2 : Tính độ dài xâu bằng cách đếm từng kí tự (tương đương với hàm strlen(s))

```
main()
{
    char s[100];                          // độ dài tối đa là 99 kí tự
    cin.getline(s, 100);                  // nhập xâu s
    for (int i=0 ; s[i] != '\0' ; i++) ;  // chạy từ đầu đến cuối xâu
    cout << "Độ dài xâu = " << i ;
}
```

Ví dụ 3 : Sao chép xâu s sang xâu t (tương đương với hàm strcpy(t,s))

```
main()
{
    char s[100], t[100];
    cin.getline(s, 100);                  // nhập xâu s
      int i=0;
      while ((t[i] = s[i]) != '\0') i++;  // copy cả dấu kết thúc xâu '\0'
      cout << t << endl ;
}
```

2. DỮ LIỆU KIỂU STRING

2.1. Khái niệm:

Chuỗi ký tự là tập hợp các ký tự được đặt trong dấu ngoặc kép. Dùng để biểu diễn những thông báo, văn bản, … trong chương trình. Trong chương trình trên, "Hello, HowKteam.com!" chính là một chuỗi ký tự. Trong C++, kiểu chuỗi ký tự không được xây dựng sẵn (không phải là “built-in string”) mà được cài đặt trong một lớp của thư viện chuẩn STL (C++ Standard Template Library)

2.2. Khai báo chuỗi:

Trong C++, chuỗi được định nghĩa sẵn với từ khoá là string .

```
string s;
```

2.3. Các hàm xử lý chuỗi:

- Nhập xuất chuỗi:

+> Nhập chuỗi bằng lệnh cin

Lưu ý: Lệnh cin chỉ có thể dùng để nhập chuỗi không có khoảng trắng.

```
#include <bits/stdc++.h>
using namespace std ;
int main()
{
  string a;
  cin>>a;
  cout<<a;
}
```

+> Nhập chuỗi bằng câu lệnh getline

Lưu ý: Lệnh getline dùng để nhập chuỗi có tồn tại khoảng trắng.

```
#include <bits/stdc++.h>
using namespace std ;
int main()
{
  string a;
  getline(cin,a);
 cout<<a;
}
```

- Lấy chiều dài chuỗi:

Ta sử dụng 2 hàm là size() hoặc length() để lấy chiều dài chuỗi.

```
#include <bits/stdc++.h>
using namespace std ;
int main()
{
  string a;
  getline(cin,a);
 cout<<a.size();
}
```

- Truy cập một phần tử bất kỳ trong chuỗi:

Chuỗi thực chất là một mảng các phần tử kiểu char. Do đó, ta có thể truy xuất các phần tử trong chuỗi tương tự như truy xuất với mảng thông thường.

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

```
int main()
{
   string a;
   getline(cin,a);
  int t=a.size();
  for(int i=0;i<t;i++)
    cout<<a[i];
}
```

- Chèn chuỗi:

Để chèn chuỗi trong C++, ta dùng hàm insert theo cấu trúc sau:

*insert **(** vị trí cần chèn(kiểu nguyên) **,** chuỗi cần chèn **) ;***

```
#include <bits/stdc++.h>
using namespace std ;
int main()
{
   string a;
    getline(cin,a); // Viet
   int t=a.size();
   a.insert(t," Nam");
   cout<<a; // Viet Nam
}
```

- Xoá chuỗi:

Để xoá chuỗi trong C++, ta dùng hàm erase theo cấu trúc sau:

*erase **(** vị trí cần xóa **,** number **);***

```
#include <bits/stdc++.h>
using namespace std ;
int main()
{
   string a;
    getline(cin,a); // cong nghe thong tin
   int t=a.size();
   a.erase(5,5);
   cout<<a; // cong thong tin
}
```

C. BÀI TẬP VỀ DỮ LIỆU KIỂU XÂU, STRING TRONG C++

Bài 1: Vị trí xâu

Cho xâu S chỉ chứa các kí tự latin thường, có n yêu cầu, mỗi yêu cầu có dạng:

$1\ x\ y$: thêm xâu x vào xâu s sau vị trí thứ y của xâu s hiện tại, nếu y bằng 0 thì thêm x vào vị trí đầu xâu.

- $2\ x\ y$: xóa xâu của s bắt đầu từ vị trí thứ x, độ dài y.

Dữ liệu vào: bai1.inp

- Dòng đầu chứa xâu s ban đầu và n ($|s| \leq 100, n \leq 100$).
- N dòng tiếp theo chứa các yêu cầu.

Dữ liệu ra : bai2.out chứa n dòng, mỗi dòng là xâu s hiện tại.

Bai1.inp	Bai1.out
corona 2 1 beer 6 2 2 2	coronabeer conabeer

Thuật toán: Với yêu cầu bài 1 chúng ta sử dụng các hàm cơ bản của dữ liệu kiểu chuỗi để giải quyết bài toán ở đây ta sử dụng hàm insert và erase

Độ phức tạp O(N)

Code

```
#include <bits/stdc++.h>
using namespace std;
string s;
int n;
int main() {
  freopen("bai1.inp", "r", stdin);
  freopen("bai1.out", "w", stdout);
  cin >> s >> n;
  for (int i = 1; i <= n; i++) {
    int re;
    cin >> re;
    if (re == 1) {
      string x;
      int y;
      cin >> x >> y;
      s.insert(y, x);
    } else {
      int x, y;
```

```
    cin >> x >> y;
    s.erase(x - 1, y);
   }
   cout << s << endl;
  }
  return 0;
}
```

Test:
https://drive.google.com/file/d/1N5B_mSfpbVyzk7JfEPpv2LwSWBtqV7NX/view?usp=sharing

Bài 2: Chuẩn hóa xâu

Cho 1 xâu s gồm các ký tự $a, b, c, \dots z, A, B, C, \dots Z$ và dấu cách. Chuẩn hóa xâu S về dạng chuẩn biểu diễn tên người– chữ cái đầu tiên mỗi từ viết hoa, các chữ còn lại viết thường. Giữa các từ có thể có nhiều dấu dấu cách, ta chỉ được giữ lại 1 dấu cách.

Dữ liệu vào : bai 2.inp chứa 1 xâu s ($|s| \leq 100000$)

Dữ liệu ra : bai2.out chứa xâu s sau khi đã chuẩn hóa.

Bai2.inp	Bai2.out
TRAN QUOC TUAN	Tran Quoc Tuan

Thuật toán: Sử dụng các hàm chuyển từ chữ hoa thành chữ thường, Độ phức tạp O(N).

code

```
#include <bits/stdc++.h>
using namespace std;
const int N = 100010;
char s[N];
void upperTurn(char &u) {
  if (u >= 'A' && u <= 'Z') {
    return;
  }
  u += 'A' - 'a';
}
void lowerTurn(char &u) {
  if (u >= 'a' && u <= 'z') {
```

```
    return;
  }
  u += 'a' - 'A';
}
main()
{
  freopen("bai2.inp", "r" ,stdin);
  freopen("bai2.out", "w", stdout);
  while (cin >> (s + 1)) {
    int n = strlen(s + 1);
    upperTurn(s[1]);
    for (int i = 2; i <= n; i++) {
      lowerTurn(s[i]);
    }
    printf("%s ", s + 1);
  }
  return 0;
}
```

Test:
https://drive.google.com/file/d/1N5B_mSfpbVyzk7JfEPpv2LwSWBtqV7NX/view?usp=sharing

Bài 3: Tính tổng các số ghi trong xâu

Cho 1 xâu s gồm chữ cái in thường và số, tính tổng các số được ghi trong xâu s

Dữ liệu vào : bai3.inp chứa 1 xâu s ($|s| \leq 50$)

Dữ liệu ra : bai3.out chứa 1 số là tổng cần tìm.

Bai3.inp	Bai3.out
1as123as3x	127

Thuật toán: Độ phức tạp O(N)

Code:

```
#include <bits/stdc++.h>
using namespace std;
char s[100];
int n;
int main() {
```

```
  freopen("bai3.inp", "r", stdin);
  freopen("bai3.out", "w", stdout);
  scanf("%s", s + 1);
  n = strlen(s + 1);
  s[++n] = '$';
  long long sum = 0, cur = 0;
  for (int i = 1; i <= n; i++) {
    if (s[i] < '0' || s[i] > '9') {
      sum += cur;
      cur = 0;
    } else {
      cur = cur * 10 + s[i] - '0';
    }
  }
  cout << sum << endl;
  return 0;
}
```

Test:
https://drive.google.com/file/d/1N5B_mSfpbVyzk7JfEPpv2LwSWBtqV7NX/view?usp=sharing

Bài 4: Tìm vị trí xuất hiện trong xâu

Cho 2 xâu x và y, tìm vị trí đầu tiên xuất hiện của x trong y, nếu x không xuất hiện in ra -1.

Dữ liệu vào: bai4.inp chứa xâu x và y. ($|x|, |y| \leq 100000$)

Dữ liệu ra: bai4.out gồm 1 số là vị trí của xâu x trong y hoặc -1

Bai4.inp	Bai4.out
abc zabcd	2

Thuật toán: sử dụng hàm tìm kiếm find.

Code

```
#include <bits/stdc++.h>
using namespace std;
string x, y;
int main() {
```

```
  freopen("bai4.inp", "r", stdin);
  freopen("bai4.out", "w", stdout);
  cin >> x >> y;
  int u = y.find(x);
  cout << (u + 1 > 0 ? u + 1 : -1 )<< endl;
  return 0;
}
```

Test:

https://drive.google.com/file/d/1N5B_mSfpbVyzk7JfEPpv2LwSWBtqV7NX/view?usp=sharing

Bài 5: Xâu con

Cho 2 xâu x và y, hỏi xem xâu x có phải là xâu con không liên tiếp của y hay không? Xâu x là xâu con không liên tiếp của y nếu ta có thể nhận được xâu x sau khi xóa 1 số kí tự của xâu y (có thể không xóa kí tự nào).

Dữ liệu vào: INP.TXT chứa xâu x và y ($|x|, |y| \leq 1000$)

Dữ liệu ra: OUT.TXT in YES nếu x là xâu con của y, ngược lại in ra NO

Bai5.inp	Bai5.out
abc adbec	YES
abb abcd	NO

Code:

```
#include <bits/stdc++.h>
using namespace std;
const int N = 1010;
char x[N], y[N];
int nx, ny;
int main() {
  freopen("bai5.inp", "r", stdin);
  freopen("bai5.out", "w", stdout);
  scanf("%s %s", x + 1, y + 1);
  nx = strlen(x + 1);
  ny = strlen(y + 1);
  int cur = 1;
  int f = 0;
```

```
  for (int i = 1; i <= nx; i++) {
    while (cur <= ny && y[cur] != x[i]) {
      cur++;
    }
    if (y[cur] == x[i]) {
      cur++;
      continue;
    }
    puts("NO");
    return 0;
  }
  puts("YES");
  return 0;
}
```

Test:

https://drive.google.com/file/d/1N5B_mSfpbVyzk7JfEPpv2LwSWBtqV7NX/view?usp=sharing

Bài 6: Số lớn nhất

Cho số a viết ở hệ cơ số 2. Nhiệm vụ của bạn là phải xóa một chữ số để số còn lại xóa khi xóa là lớn nhất !!!

Input

Một dòng chứa số a viết ở hệ nhị phân.

Output

Số lớn nhất sau khi xóa 1 chữ số từ a.

H**ướng dẫn Thuật toán** : Xử lý xâu tìm ký tự '0' trái nhất để xóa trong trường hợp không có thì xóa ký tự bất kỳ chọn luôn ký tự đầu tiên

Bai6.inp	Bai6.out
1010101	110101

Code:

```
#include<bits/stdc++.h>
using namespace std;
int main()
{
   char x[100001],*s;
```

```
    scanf("%s",x);
    for(s=x;*s=='1';s++);
    if(*s==0) printf("%s",x+1);
    else
    {
        *s=0;
        printf("%s",x);
        printf("%s",s+1);
    }
}
```

Test: https://www.spoj.com/PTIT/submit/REMOVBIT/

Bài 7: Bóng đá

Petya rất thích bóng đá. Một ngày nọ, khi anh đang xem một trận bóng đá, anh đang viết các vị trí hiện tại của các cầu thủ trên một tờ giấy. Để đơn giản hóa tình huống, ông mô tả nó như một chuỗi bao gồm các số 0 và một số. Số 0 tương ứng với người chơi của một đội; một cái tương ứng với người chơi của một đội khác. Nếu có ít nhất 7 cầu thủ của một số đội đứng lần lượt, thì tình huống được coi là nguy hiểm. Ví dụ, tình hình 00100110111111101 là nguy hiểm và 11110111011101 không. Bạn được đưa ra tình hình hiện tại. Xác định xem nó có nguy hiểm hay không.

Input

Dòng đầu vào đầu tiên chứa một chuỗi không trống bao gồm các ký tự "0" và "1", đại diện cho người chơi.

Độ dài của chuỗi không vượt quá 100 ký tự. Có ít nhất một cầu thủ từ mỗi đội có mặt trên sân.

Output:

In "CÓ" nếu tình huống nguy hiểm. Nếu không, hãy in "KHÔNG".

Bai7.inp	Bai7.out
001001 1000000001	NO YES

Hướng dẫn thuật toán: chỉ cần duyệt và đếm số số 0 hoặc số số 1 liền nhau xem có ít nhất 7 số không?

Code:

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

```
{
      freopen("bai7.inp", "r", stdin);
      freopen("bai7.out", "w", stdout);
  char x[1005];
  scanf("%s",x);
  int d=1;
  for(char *p=x+1;*p;p++)
  {
     if(*p==*(p-1))
     {
       d++;
       if(d==7) {printf("YES"); return 0;}
     }
     else d=1;
  }
  printf("NO");
}
```

Test:https://drive.google.com/file/d/1N5B_mSfpbVyzk7JfEPpv2LwSWBtqV7NX/view?usp=sharing

Bài 8: Petya và Chuỗi

Little Petya thích quà. Mẹ anh đã mua cho anh hai chuỗi cùng kích cỡ cho ngày sinh nhật của anh. Các chuỗi bao gồm chữ hoa chữ thường và chữ thường. Bây giờ Petya muốn so sánh hai chuỗi đó theo từ vựng . Các trường hợp chữ cái không quan trọng, đó là một chữ cái viết hoa được coi là tương đương với chữ cái viết thường tương ứng. Giúp Petya thực hiện so sánh.

Đầu vào

Mỗi dòng trong hai dòng đầu tiên chứa một chuỗi đã mua. Độ dài của chuỗi bao gồm từ 1 đến 100 . Nó được đảm bảo rằng các chuỗi có cùng độ dài và cũng bao gồm các chữ cái Latinh viết hoa và viết thường.

Đầu ra

Nếu chuỗi thứ nhất nhỏ hơn chuỗi thứ hai, hãy in "-1".
Nếu chuỗi thứ hai nhỏ hơn chuỗi thứ nhất, hãy in "1". Nếu các chuỗi bằng nhau, in "0". Lưu ý rằng trường hợp của các chữ cái không được xem xét khi các chuỗi được so sánh.

Bai8.inp	Bai8.out
aaaa aaaA	0

abs Abz abcdefg AbCdEfF	-1 1

Ghi chú: Nếu bạn muốn biết thêm thông tin chính thức về thứ tự từ điển (còn được gọi là " thứ tự từ điển " hoặc " thứ tự chữ cái "), bạn có thể truy cập trang web sau:

- http://en.wikipedia.org/wiki/Lexicographic_order

Hướng dẫn thuật toán:

Viết hàm chuyển ký tự thường thành hoa, sau đó viết hàm chuyển xâu thường thành hoa
Dùng hàm strcmp trong string.h để so sánh xâu chú ý chuyển về -1, 0, 1 theo yêu cầu của đầu bài

Code

```
#include<bits/stdc++.h>
#include<string.h>
char Touper(char x)
{
  if('a'<=x && x<='z') return x+'A'-'a';
  return x;
}
void Up(char *x)
{
   while(*x)
   {
      *x=Touper(*x);
      x++;
   }
}
int main()
{
      freopen("bai8.inp", "r", stdin);
      freopen("bai8.out", "w", stdout);
   char x[1000],y[1000];
   scanf("%s%s",x,y);
   Up(x);
   Up(y);
   int k=strcmp(x,y);
```

```
    if(k<0) k=-1; else if(k>0) k=1;
    printf("%d",k);
}
```

Test:
https://drive.google.com/file/d/1N5B_mSfpbVyzk7JfEPpv2LwSWBtqV7NX/view?usp=sharing

Bài 9: String Task

Petya bắt đầu tham dự các bài học lập trình. Trong bài học đầu tiên, nhiệm vụ của anh là viết một chương trình đơn giản. Chương trình được yêu cầu thực hiện như sau: trong chuỗi đã cho, bao gồm nếu chữ hoa và chữ thường viết hoa, nó:

- xóa tất cả các nguyên âm,
- chèn một ký tự " . " trước mỗi phụ âm,
- thay thế tất cả các phụ âm chữ hoa bằng các chữ thường tương ứng.

Nguyên âm là các chữ cái "A", "O", "Y", "E", "U", "I" và phần còn lại là phụ âm. Đầu vào của chương trình chính xác là một chuỗi, nó sẽ trả về đầu ra dưới dạng một chuỗi, kết quả là sau khi chương trình xử lý chuỗi ban đầu.

Giúp Petya đối phó với nhiệm vụ dễ dàng này.

Đầu vào

Dòng đầu tiên đại diện cho chuỗi đầu vào của chương trình Petya. Chuỗi này chỉ bao gồm các chữ cái Latinh viết hoa và viết thường và độ dài của nó là từ 1 đến 100 , bao gồm.

Đầu ra

In chuỗi kết quả. Nó được đảm bảo rằng chuỗi này không trống.

Bai9.inp	Bai9.out
tour Codeforces	.t.r .c.d.f.r.c.s

Hướng dẫn thuật toán:

Viết hàm chuyển chữ hoa thành thường bằng cách nếu là x chữ hoa thì chữ thường là x-'A'+'a'

Duyệt từ đầu đến cuối xâu gặp nguyên âm thì bỏ qua gặp phụ âm thì xuất ra '.' và phụ âm đó

Code:

```
#include<bit/stdc++.h>
char Lower(char x)
{
  if('A'<=x &&x<='Z') return x-'A'+'a';
  return x;
}
using namespace std;
int main()
{
      freopen("bai9.inp", "r", stdin);
      freopen("bai9.out", "w", stdout);
    char x[1000];
    cin>>x;
    for(char *p=x;*p;p++)
    {
       *p=Lower(*p);
       if(*p=='a'||*p=='o'||*p=='y'||*p=='e'||*p=='u'||*p=='i') continue;
       cout<<"."<<*p;
    }
}
```

Test:

https://drive.google.com/file/d/1wAUBYBZrvWVsrUiz1PadSxQJnp1z-Zl9/view?usp=sharing

Bài 10: NKLETTER – Gửi thư

Vị Giám đốc công ty XYZ cần gửi một văn bản quan trọng tới một đối tác của mình. Văn bản là một xâu S các chữ cái la tinh in thường. Để bảo mật nội dung văn bản, ông Giám đốc gửi 2 bức thư. Bức thư thứ nhất là phần đầu Sb của xâu S, bức thư thứ 2 là phần cuối Se của S. Hai bức thư Sb và Se đảm bảo đầy đủ nội dung của S, tuy nhiên có thể một phần cuối của Sb có thể được viết lặp lại trong phần đầu của Se, song số kí tự được viết lặp lại không biết trước.

Ví dụ: với văn bản S=’truongnguyenduquannhat’ tạo ra hai bức thư:

Sb=truongngueNdu

ngueNduquanNhat=Se=

Sb=’truongnguyendu’ và Se=’nguyenduquannhat’

Yêu cầu:

Cho hai xâu Sb và Se, hãy xác định một xâu S có thể là nội dung của bức thư sao cho độ dài của xâu S là ngắn nhất.

Dữ liệu

Dòng đầu chứa xâu Sb, dòng thứ hai chứa xâu Se. Mỗi xâu có độ dài không quá 250.
Kết quả
Ghi ra độ dài của xâu S tìm được.
Hướng dẫn thuật toán: Vét cạn duyệt toàn bộ kiểm tra từng phần tiếp đuôi ngữ của Sb có là tiếp đầu của Se không. Dùng hàm strstr(s1,s2) kiểm tra xem xâu s2 có là xâu con của s1 không nếu có thì từ ở vị trí nào.
Ví dụ

input	output
truongnguyendu nguyenduquannhat	22

Code

```
#include <bits/stdc++.h>
#include <string.h>

int main() {
    char sb[300],se[300],*p;
    scanf("%s%s",sb,se);
    for(p=sb;*p;p++) if(strstr(se,p)==se) break;
    printf("%d",strlen(se)+strlen(sb)-strlen(p));
}
```

Test: https://www.spoj.com/PTIT/submit/REMOVBIT/

Bài 11: FINDCOW – Find the Cow!

Cô bò Bessie đã trốn thoát và đang trốn ở một đồi núi với những đồng cỏ cao. Nông dân John (FJ), người đang muốn tìm kiếm Bessie, đã quyết định bò trên đồng cỏ bằng tay và dầu gối để tìm ra dấu vết của Bessie. Không may thay, ông ta có vấn đề với việc tìm kiếm Bessie từ vị trí thuận lợi này. Dãy cỏ ở trước mặt FJ trông như một chuỗi ngoặc đơn có độ dài N (1 <= N <= 50,000); ví dụ:

)((()())())

FJ biết rằng Bessie chân sau của Bessie giống như một cặp dấu mở ngoặc đơn ((, và chân trước của cô ta giống như một cặp dấu đóng ngoặc đơn. Vị trí của Bessie có thể được diễn tả bởi mọt cặp x < y , trong đó ((được tìm ở vị trí x, và)) được tìm ở vị trí y. Hãy đếm có bao nhiêu vị trí mà Bessie có thể đang đứng.
Input
Dòng 1: Một chuỗi ngoặc đơn có độ dài là N (1 <= N <= 50,000).
Output
Dòng 1: Số vị trí mà Bessie có thể đứng – Có nghĩa là số cặp (x,y) khác nhau với x < y sao cho ((được tìm thấy ở x và)) được tìm thấy ở y.

input	output
)(()())())	4

Có 4 vị trí có thể của Bessie được thể hiện ở dưới

1.)(()())())
   ^^ ^^
2.)(()())())
    ^^ ^^
3.)(()())())
    ^^   ^^
4.)(()())())
   ^^   ^^

Hướng dẫn thuật toán:

1. Duyệt để kiểm tra xem tại mỗi vị trí có phải là dấu chân trước không ((thì true ngược lại là false lưu vào mảng L
2. Bước 2 đếm cuốn chiếu từ cuối lên nếu gặp dấu chân sau)) thì thì tăng lên một đơn vị ngược lại để nguyên lưu vào mảng R
3. Duyệt lại từ trái sang phải nếu gặp đúng dấu chân trước L[i]= true thì kiểm tra tại đó có bao nhiêu dấu chân sau ở phía sau tức là R[i+1] nhưng vì đây là vị trí ((nên R[i+1]=R[i] nên ta tổng vào R[i]

```
#include<bits/stdc++.h>
using namespace std;
int main()
{
char x[50005];
bool L[50005];
long R[50005],KQ=0;
scanf("%s",x+1);
long n=strlen(x+1);
for(int i=2;i<=n;i++)
if(x[i]=='(' && x[i-1]=='(')
L[i]=true; else L[i]=false;
R[n]=0;
for(int i=n-1;i>=1;i--)
```

```
if(x[i]==')' && x[i+1]==')')
R[i]=R[i+1]+1; else R[i]=R[i+1];
for(int i=2;i<=n;i++)
if(L[i]) KQ+=R[i];
printf("%ld",KQ);
}
```

Test: http://www.spoj.com/PTIT/problems/FINDCOW/

Bài 12: REPSTR – Replacing Digits

Cho số nguyên dương a có N chữ số và số dãy s có M chữ số. Chữ số ở vị trí j (1<=j<=M) của dãy s có thể chọn bất kì vị trí i (1<=i<=N) trong số a và thay thế bằng sj. Mỗi chữ số của dãy s chỉ được thay thế không quá một lần.

Nhiệm vụ của bạn là hãy tìm cách thay sao cho số a đạt giá trị lớn nhất. Bạn có thể không cần sử dụng tất cả các chữ số trong s.

Input

Dòng đầu chứa số nguyên dương a có độ dài N (không bắt đầu bằng chữ số 0).

Dòng 2 chứa dãy s có độ dài M $(1 \le N,M \le 10^5)$

Output

Số a lớn nhất có thể thay thế được.

input	output
1024 010	1124

Hướng dẫn:
Bước 1 : Duyệt số các chữ số từ 0 đến 9 của xâu s đếm số chữ '0', số chữ '1' … số chữ '9'
Bước 2 : Duyệt lần lượt xâu a từ đầu đên cuối, kiểm tra ký tự lớn nhất nếu thay vào làm tăng giá trị của a thì thay.

Code:

```
#include <bits/stdc++.h>
using namespace std;
int main()
{
char a[100005],s[100005],q='9';
int d[100];
```

```
scanf("%s",a);
scanf("%s",s);
for(char i='0';i<='9';i++) d[i]=0;
for(char *q=s;*q;q++) d[*q]++;
for(char *p=a;*p;p++)
{
while(q>'0' && d[q]==0) q--;
if(q=='0') break;
if(*p<q)
{*p=q;
  d[q]--;
} }
printf("%s",a);
}
```

Test: http://www.spoj.com/PTIT/problems/REPSTR/

Bài 13: Dãy đặc biệt

Long là một người rất yêu thích nghiên cứu các dãy số. Một hôm, Long nghĩ ra một dãy số (a_n) mà cậu gọi là **dãy đặc biệt** được xây dựng theo quy tắc sau:

- Cho trước số a_0 là số tự nhiên có tối đa 10 chữ số.

- Số a_i ($i \geq 1$) là một số tự nhiên nhận được từ a_{i-1} bằng cách viết thêm vào sau các chữ số của a_{i-1} chính a_{i-1} nhưng viết theo thứ tự ngược lại.

Long rất thích dãy số này và đem khoe nó với Hải, cậu bạn cũng thích tìm hiểu về dãy số như mình. Sau một lúc suy nghĩ, Hải liền đố Long một bài toán về dãy số của cậu như sau: "Với hai số nguyên dương N và M cho trước, hãy tìm chữ số thứ M của a_N trong dãy đặc biệt trên ". Bạn hãy giúp Long lập trình giải bài toán này nhé.

Dữ liệu: vào từ file văn bản "**CHUSO.inp**" có dạng:

- Dòng đầu ghi số tự nhiên a_0.

- Dòng thứ hai ghi hai số nguyên dương N, M cách nhau một dấu cách.

Kết quả: cho ra file văn bản "**CHUSO.out**" Trong trường hợp có lời giải, ghi số tìm được. Trong trường hợp không có lời giải, ghi số -1

Giới hạn: $1 \leq N \leq 30$; $1 \leq M \leq 10^{9.}$

Ví dụ:

chuso.inp	chuso.out
345 2 10	5

CODE

```
#include <bits/stdc++.h>
using namespace std;
char a[20],a0[10];
unsigned long long m;
int n;
void nhap()
{
   cin.getline(a,10);
   cin>>n>>m;
   int j=0;
   for( int i=strlen(a)-1;i>=0;i-- )
   {
         a0[j]=a[i];
         j++ ;
   }
   strcat(a,a0);
}
void dieukien()
{
   unsigned long long b=1;
   for( int i=1;i<=n;i++ )
      b*=2;
   b*=strlen(a0);
   if ( m < 1 || m > b )
   {
      cout<<"-1";
      exit(0);
   }
}
void xuli()
{
   int k,l,o;
```

```
    k=strlen(a);
    l=m%k;
    if ( l==0) o=k-1;
    else o=l-1;
   cout<<a[o];
}
 main()
{
   freopen("chuso.inp","r",stdin);
   freopen("chuso.out","w",stdout);
   nhap();
   dieukien();
   xuli();
}
```

Test:
https://drive.google.com/file/d/1N5B_mSfpbVyzk7JfEPpv2LwSWBtqV7NX/view?usp=sharing

Bài 14: Xâu nhị phân

Một xâu gọi là xâu nhị phân nếu chỉ chứa hai ký tự “0” hoặc 1

Xâu *v* gọi là xâu con của *w* nếu xâu v có độ dài khác 0 và gồm các ký tự liên tiếp trong xâu w. Ví dụ: xâu “010” có các xâu con là “0”, “1”, “0”, “01”, “10”, “010”.

Cho trước một giá trị k, hãy đếm xem có bao nhiêu xâu con chứa đúng k ký tự “1”

INPUT: SUBSTR.INP

- Dòng 1 chứa một số nguyên k ($0 <= k <= 10^6$)
- Dòng 2 chứa một xâu nhị phân có độ dài $<= 10^6$

OUTPUT: SUBSTR.OUT

- Một số nguyên duy nhất là kết quả tìm được.

Ví dụ:

SUBSTR.INP	SUBSTR.OUT
2 01010	4

Code

```
#include <bits/stdc++.h>
using namespace std;
const int32_t N=1e6+2;
int32_t k,d;
int64_t dem;
int32_t dd[N];
string s;
int main()
{
   ios_base::sync_with_stdio(0);
   freopen("substr.inp","r",stdin);
   freopen("substr.out","w",stdout);
   dd[0]=1;
   cin>>k;
   cin>>s;
   int32_t n=s.size();
   while(n--)
   {
      d+=s[n]-48;
      if ( d >= k ) dem+=dd[d-k];
      dd[d]++;
   }
   cout<<dem;
}
```

Test:
https://drive.google.com/file/d/1N5B_mSfpbVyzk7JfEPpv2LwSWBtqV7NX/view?usp=sharing

Bài 15: Tổng chẵn

Do mới được chọn vào đội tuyển Học sinh giỏi Tin học lớp 10, đội tuyển lại chưa có nhiều bài tập nên Bi và Bo rảnh quá, trong lúc không biết làm gì, các bạn đã tổ chức một trò chơi như sau: Bi viết một dãy số nguyên dương, rồi đố Bo tìm một dãy số liên tiếp trong dãy số của Bi viết ra mà có tổng là số chẵn. Dĩ nhiên câu đố này quá dễ với Bo, nên Bo đã đố lại Bi là hãy tính xem có bao nhiêu dãy như thế mà có tổng là số chẵn. Do tin tưởng vào khả năng lập trình của mình nên Bi yêu cầu cho mượn một chiếc máy tính để lập trình tính toán và trả lời kết quả của Bo trong không quá 1 giây. *Chú ý dãy có tổng chẵn thỏa mãn yêu cầu có thể chỉ có 1 phần tử.*

Em hãy thử xem Bi đã làm như thế nào mà có thể viết ra một chương trình để tính toán tuyệt diệu đến vậy nhé!

Dữ liệu vào:

- Dòng đầu tiên là n số lượng của dãy số mà Bi cho ban đầu.
- Dòng tiếp theo là các số nguyên dương $a_i, (i = 1..n)$.

Kết quả ra:

- Một số duy nhất là số dãy có tổng chẵn.

Ví dụ:

SUMEVEN.INP	SUMEVEN.OUT
4 1 1 1 1	4
6 1 2 3 4 5 6	9

Ràng buộc:

20% test có $n \leq 100, a_i \leq 10^9$

30% test có $100 < n \leq 10000, a_i \leq 10^9$

30% test có $10000 < n \leq 10^5, a_i \leq 10^9$

20% test có $10000 < n \leq 10^5, a_i \geq 10^{19}$

Code

```
#include <bits/stdc++.h>
#define maxn 100005
#define f first
#define s second
#define pp pair<int, int>
#define task "sumeven"
#define reset(a, b) memset(a, b, sizeof(a))
using namespace std;
long long mod[5], n, ans;
string sum = "0", a[maxn];
string add(string a, string b)
{
   string c = "";
   int hold=0, sum=0;
   if (a.size() < b.size())
      a.insert(0, b.size() - a.size(), '0');
   if (a.size() > b.size())
      b.insert(0, a.size() - b.size(), '0');
   for (int i = a.size() - 1; i >= 0; i--)
   {
      sum = (a[i] - 48) + (b[i] - 48) + hold;
```

```
        hold = sum / 10;
        sum %= 10;
        c = (char) (sum + 48) + c;
    }
    if (hold > 0)
        c = (char) (hold + 48) + c;
    return c;
}

int main()
{
    ios_base::sync_with_stdio(NULL);
    cin.tie(0); cout.tie(0);
    freopen(task".inp", "r", stdin);
    freopen(task".out", "w", stdout);
    mod[0]=1;
    cin >> n;
    for (int i = 1; i <= n; i++)
        cin >> a[i];
    for (int i = 1; i <= n; i++)
    {
        sum = add(sum, a[i]);
        int j = (sum[sum.size()-1] - 48) % 2;
        ans += mod[j];
        mod[j]++;
    }
    cout << ans;
    return 0;
}
```

Test:
https://drive.google.com/file/d/1N5B_mSfpbVyzk7JfEPpv2LwSWBtqV7NX/view?usp=sharing

Bài 16: P165PROI Đột biến gen

Các nhà sinh học vừa phát hiện ra một loại cấu trúc DNA alpha đặc biệt được cấu trúc bởi chỉ các Nucleotit A và B. Đặc biệt hơn, họ còn phát hiện ra một loại gen omega chỉ gồm nucleotit A nhưng vô cùng hiếm, để nghiên cứu sâu hơn về omega họ quyết định tổng hợp omega từ alpha bằng các phương pháp đột biến nhân tạo. May mắn thay

họ tìm ra hai phương pháp đột biến có thể gây ảnh hưởng đến alpha: họ có thể làm thay đổi 1 nuleotit ở một vị trí bất kỳ trên đoạn gen alpha (A thành B hoặc B thành A), phương pháp thứ hai là thay đổi cả đoàn đầu của gen (những nuc A chuyển thành B và ngược lại). Để tiết kiệm chi phí họ quyết định tìm ra số lần gây đột biến ít nhất có thể để biến alpha thành omega.

Hãy giúp họ.

Input
Dòng đầu tiên chứa số tự nhiên n là số nucleotit của gen alpha (1 <= n <= 1000000)
Dòng thứ hai là biểu diễn của gen alpha.

Output
Kết quả bài toán.

input	output
12 AAABBBAAABBB	4

Hướng dẫn: Ý tưởng Quy hoạch động
Đặt A[n] là số phép biến đổi ít nhất để đưa xâu x1…xn về toàn ‘A’
Đặt B[n] là số phép biến đổi ít nhất để đưa xâu x1…xn về toàn ‘B’

Ta có công thức truy hồi A[0] = B[0] = 0 vì xâu rỗng
+ Nếu x[n]==’A’ thì A[n]=A[n-1] vì ta chỉ biến đổi n-1 ký tự đầu về ‘A’ chữ còn lại hiển nhiên đã là ‘A’;
+ Ngược lại nếu x[n] == ‘B’ thì có hai cách biến xâu này về toàn ‘A’: Cách 1 là biến đổi hết n-1 chữ đầu về ‘A’ và thêm phép biến đổi cuối cùng đưa x[n] đang là ‘B’ về ‘A’ ta có A[n-1]+1 phép ít nhất: Cách 2 là biến đổi hết n-1 chữ đầu về ‘B’ phép cuối cùng đổi n chữ ‘B’ này sang n chữ ‘A’ có B[n-1]+1 cách lấy min của 2 cách này ra ra số cách ít nhất.
Một cách tương tự đối với công thức truy hồi cho B[n] ta có code như sau:

Code để giải thích ý tưởng thuật toán

```
#include <bits/stdc++.h>
using namespace std;
long min(long a,long b) {return a<b?a:b;}
int main()
{
char x[1000006];
long n;
scanf("%ld%s",&n,x+1);
long *A=new long[n+5];
```

```
long *B=new long[n+5];
A[0]=B[0]=0;
for(long i=1;i<=n;i++)
{
if(x[i]=='A')
{
A[i]=A[i-1];
B[i]=1+min(A[i-1],B[i-1]);
}
else
{
A[i]=1+min(A[i-1],B[i-1]);
B[i]=B[i-1];
}
}
printf("%ld",A[n]);
}
```

Test: https://www.spoj.com/PTIT/problems/P165PROI/

D. Kết luận

Chuyên đề được tác giả viết không đi về thuật toán cụ thể như HASH, Z, KMP…mà chú trọng đến kĩ năng ban đầu về các thao tác xử lí xâu ở mức độ đơn giản nhằm giúp học sinh mới bắt đầu học có cái nhìn về dữ liệu kiểu xâu để tiến tới giải quyết các bài toán phức tạp hơn, chuyên đề viết còn nhiều hạn chế mong được sự giúp đỡ và đóng góp ý kiến của quý thầy cô.

Xin chân thành cảm ơn!

E. Tài liệu tham khảo

- Học sinh có thể tham khảo tài liệu mở trên các website…

http://www.spoj.com

https://codeforces.com;

https://www.geeksforgeeks.org