Xâu (**string**) xuất hiện rất nhiều trong các bài toán. Bài viết này giới thiệu sơ qua một số thuật ngữ cũng như thuật toán về xâu.

Thuật ngữ

- Một xâu *X*
  là **xâu con** (**substring**) của một xâu *Y*
  nếu *X*
  là một chuỗi các ký tự liên tiếp của *Y*
  . Ví dụ: `ab` và `bc` là 2 xâu con của `abcd`. Nhưng `ac` thì không phải là xâu con của `abcd`.
- Một xâu *X*
  là **tiền tố** (**prefix**) của một xâu *Y*
  nếu *X*
  là xâu con của *Y*
  và *X*
  xuất hiện ở đầu của xâu *Y*
  . Ví dụ: `ab` là tiền tố của `abcd`, nhưng `bc` **không** phải là tiền tố của `abcd`. Một xâu *X*
  là một **tiền tố không tầm thường** (**proper prefix**) của xâu *Y*
  nếu nó là tiền tố của xâu *Y*
  và khác xâu *Y*
  .
- Một xâu *X*
  là **hậu tố** (**hậu tố**) của một xâu *Y*
  nếu *X*
  là xâu con của *Y*
  và *X*
  xuất hiện ở cuối của xâu *Y*
  . Ví dụ: `cd` là hậu tố của `abcd`, nhưng `bc` **không** phải là hậu tố của `abcd`. Một xâu *X*
  là một **hậu tố không tầm thường** (**proper suffix**) của xâu *Y*
  nếu nó là hậu tố của xâu *Y*
  và khác xâu *Y*
  .

Các dạng bài

So khớp chuỗi (string matching)

Cho một xâu *T*
và xâu *S*
. Tìm tất cả các lần xuất hiện của xâu *S*
trong xâu *T*
.

Ví dụ:

```
S = abc
T = abcabcabc

Các lần xuất hiện: 1, 4, 7.
```

Bài toán này còn được gọi là tìm kiếm **cây kim** (**needle**) trong **đống rơm** (**haystack**), vì nó xuất hiện trong thực tế khi ta cần tìm một xâu rất nhỏ trong một lượng dữ liệu rất lớn (ví dụ Google cần tìm từ khóa trong hàng tỉ tỉ trang web).

Có 3 thuật toán chính để giải quyết bài này, đó là:

- Thuật toán KMP
- Hash
- Z Algorithm

Xâu đối xứng (Palindrome)

Palindrome hay còn gọi là xâu đối xứng, xâu đối gương là tên gọi của những xâu kí tự mà khi viết từ phải qua trái hay từ trái qua phải thì xâu đó không thay đổi. VD: MADAM, IOI,...

Có rất nhiều bài tập liên quan đến xâu đối xứng. Các bạn có thể tìm đọc ở trong các bài viết:

- Một vài bài tập QHD về Palindrome
- Hash
- Palindrome Tree

Cấu trúc dữ liệu

- Trie là CTDL cơ bản nhất trong xử lý xâu. Nó giúp giải quyết các bài toán về tìm kiếm xâu.
- Lớp CTDL được gọi chung là Suffix Structures gồm:
  - Suffix Array
  - Suffix Automaton
  - Suffix Tree
  - Aho Corasick

  Gọi chung như vậy vì các CTDL này có thể dùng thay thế nhau để giải quyết cùng một lớp bài toán liên quan đến các suffix của cây.

Các bài Ad-hoc

Trong xử lý xâu còn một vài thuật toán chỉ áp dụng được cho 1 bài toán (ad-hoc).

Thuật toán Manacher

Bài toán

Cho xâu S
.

- Với mỗi vị trí i
  của xâu S
  , tìm xâu đối xứng dài nhất nhận i
  là tâm.
- Với mỗi cặp i
  , $i + 1$
  của xâu S
  , tìm xâu đối xứng dài nhất nhận i
  và $i + 1$
  là tâm.

Mô tả thuật toán

Tham khảo thêm ở link

Code

```
const char DUMMY = '.';

int manacher(string s) {
 // Để tránh phải xét riêng trường hợp độ dài xâu đối xứng chẵn / lẻ,
 // ta thêm 1 ký tự DUMMY vào giữa các ký tự của s.
 // CHÚ Ý: Phải đảm bảo DUMMY không có trong xâu s

 int n = s.size() * 2 - 1;
 vector <int> f = vector <int>(n, 0);

 // Tạo xâu a bằng cách chèn ký tự DUMMY vào giữa các ký tự của s.
 // Ví dụ:
 // s = aabcb
 // a = a.a.b.c.b
 string a = string(n, DUMMY);
 for (int i = 0; i < n; i += 2) a[i] = s[i / 2];

 int l = 0, r = -1, center, res = 0;
 for (int i = 0, j = 0; i < n; i++) {
  j = (i > r ? 0 : min(f[l + r - i], r - i)) + 1;
  while (i - j >= 0 && i + j < n && a[i - j] == a[i + j]) j++;
  f[i] = --j;
  if (i + j > r) {
   r = i + j;
   l = i - j;
  }

  int len = (f[i] + i % 2) / 2 * 2 + 1 - i % 2;
  if (len > res) {
   res = len;
   center = i;
  }
 }
 // Với mỗi vị trí i, xâu đối xứng dài nhất nhận i là tâm là [i - f[i], i + f[i]].
 // Ví dụ:
 // s = aabcb
 // a = a.a.b.c.b
 // f = 011010200
 return res;
}
```

Minimal string rotation

Bài toán

Cho một xâu S
. Xét các xâu thu được từ xâu S
bằng phép xoay. Ví dụ: `S = abcd`, thì các xâu thu được từ S
bằng phép xoay là:

- `abcd`
- `bcda`
- `cdab`
- `dabc`

Tìm xâu có thứ tự từ điển nhỏ nhất.

Mô tả thuật toán

Bạn có thể xem ở đây

Code

```
// Tính vị trí của xâu xoay vòng có thứ tự từ điển nhỏ nhất của xâu s[]
int minmove(string s) {
 int n = s.length();
 int x, y, i, j, u, v; // x is the smallest string before string y
 for (x = 0, y = 1; y < n; ++ y) {
  i = u = x;
  j = v = y;
  while (s[i] == s[j]) {
   ++ u; ++ v;
   if (++ i == n) i = 0;
   if (++ j == n) j = 0;
   if (i == x) break; // All strings are equal
  }
  if (s[i] <= s[j]) y = v;
  else {
   x = y;
   if (u > y) y = u;
  }
 }
 return x;
}
```

Lyndon Decomposition

Bài toán

Lyndon word là các xâu khác rỗng, mà có thứ tự từ điển nhỏ hơn tất cả các xâu thu được bằng phép xoay của nó.

Cho một xâu S
. Tìm cách tách S
thành ít nhất các xâu, sao cho mỗi xâu đều là Lyndon word.

Code

```
void lyndon(string s) {
 int n = (int) s.length();
 int i = 0;
 while (i < n) {
  int j = i + 1, k = i;
  while (j < n && s[k] <= s[j]) {
   if (s[k] < s[j]) k = i;
   else ++k;
   ++j;
  }
  while (i <= k) {
   cout << s.substr(i, j - k) << ' ';
   i += j - k;
  }
 }
 cout << endl;
}
```

Last edited by **Thanh Trung Nguyen**, 2016-08-23 09:06:08

Một vài bài tập về Palindrome

Nguyễn Hoành Tiến

Palindrome hay còn gọi là xâu đối xứng, xâu đối gương là tên gọi của những xâu kí tự mà khi viết từ phải qua trái hay từ trái qua phải thì xâu đó không thay đổi. VD: MADAM, IOI,... Nhờ tính chất đặc biệt đó mà có khá nhiều bài tập có liên quan đến Palindrome, phần lớn trong chúng thường đi kèm với QHĐ. Tôi xin giới thiệu với các bạn một vài bài tập như vậy.

Bài 1: Xem một xâu có phải là Palindrome hay không?

Đây là một bài cơ bản, nhưng quan trọng vì nó được đề cập đến trong nhiều bài tập khác. Cách làm tốt nhất là duyệt đơn thuần mất O(N).

```
function is_palindrome(s: string): boolean;
var i, n : integer;
begin
n := length(s);
for i := 1 to (n div 2) do
if s[i] <> s[n+1-i] then
begin is_palindrome := false; exit; end;
is_palindrome := true;
end;
```

Một đoạn chương trình khác :

```
function is_palindrome(s : string) : boolean;
var i, j : integer;
begin
i := 1;
j := length(n);
while i
begin
if s[i] <> s[j] then
begin is_palindrome := false; exit; end;
inc(i);
dec(j);
end;
is_palindrome := true;
end;
```

Bài 2: Cho một xâu S <= 1000 kí tự; tìm palindrome dài nhất là xâu con của S (Xâu con là một dãy các kí tự liên tiếp).

Đây cũng là một bài cơ bản với nhiều cách làm.

Cách 1: QHĐ

Dùng mảng F[i, j] có ý nghĩa: F[i, j] = true/false nếu đoạn gồm các kí tự từ i đến j của S có/không là palindrome.

Ta có công thức là:

* F[i, i] = True
* F[i, j] = F[i+1, j-1]; (nếu s[i] = s[j])
* F[i, j] = False; (nếu s[i] <> s[j])

Đoạn chương trình như sau:

```
FillChar( F, sizeof(F), false );
for i := 1 to n do F[i, i] := True;
for k := 1 to (n-1) do
for i := 1 to (n-k) do
begin
j := i + k;
F[i, j] := ( F[i+1, j-1] ) and (s[i] = s[j] );
end;
```

i∀Kết quả là : Max(j-i+1) <=j thỏa F[i,j] = True.

Độ phức tạp thuật toán là 0(N2).

Chú ý : Với N lớn, ta phải thay mảng 2 chiều F bằng 3 mảng 1 chiều và dùng thêm biến max lưu giá trị tối ưu.

Cách 2: Duyệt có cận.

Ta xét từng vị trí i:

+ xem a[i] có phải là tâm của Palindrome có lẻ kí tự không?

(ví dụ Palindrome MADAM có tâm là kí tự D)

+ xem a[i] và a[i+1] có phải là tâm của Palindrome có chẵn kí tự không?

(ví dụ Palindrome ABBA có tâm là 2 kí tự BB)

với mỗi kí tự ta tìm palindrome dài nhất nhận nó là tâm, cập nhập lại kết quả khi duyệt.

Ta duyệt từ giữa ra để dùng kết quả hiện tại làm cận.

Đoạn chương trình như sau:

```
procedure Lam;
var i, j : Longint ;
{ }
procedure try( first, last : Longint );
var đ : Longint;
begin
if first = last then
begin đ := 1; dec(first); inc(last); end
else đ := 0;
repeat
if (first < 1) or (last > N) then break;
if s[i] = s[j] then
begin
đ := đ + 2;
first := first - 1;
last := last + 1;
end
else break;
until false;
if max < dd then max := dd;
end;
{ }
begin
i := n div 2;
```

```
j := n div 2 + 1;
max := 1;
while (i > max div 2) and (j <= N-max div 2) do
begin
if i > max div 2 then
begin
try( i, i );
try( i, i+1 );
end;
if j <= N - max div 2 then
begin
try( j, j );
try( j, j+1 );
end;
i := i - 1;
j := j + 1;
end;
end;
```

Cách làm này có độ phức tạp: max*(N-max). Vì vậy nó chạy nhanh hơn cách QHĐ trên, thời gian chậm nhất khi max = N/2 cũng chỉ mất N2/4 nhanh gấp 4 lần cách dùng QHĐ. Nhờ vậy, chúng ta biết là: không phải lúc nào QHĐ cũng chấp nhận được về mặt thời gian và không phải lúc nào duyệt lúc nào cũng chậm.

Bài trên còn có một cách NlogN nữa là dùng Suffix Aray, thậm chí có cách O(N) là sử dụng Suffix Tree và thuật toán tìm LCA. Đương nhiên cách cài đặt không hề dễ dàng, tôi sẽ thảo luận với các bạn vào một dịp khác.

Bài 3: Chia một xâu thành ít nhất các Palindrome (độ dài <=1000).Bài này phức tạp hơn bài trên, cách làm thì vẫn là QHĐ.

Gọi F[i] là số palindrome ít nhất mà đoạn 1..j chia thành được.

Ta có công thức:

F[i] = max(F[j] + 1; ∀j < i thỏa mãn:đoạn j+1..i là palindrome)

Đoạn chương trình như sau:

```
F[0] := 0;
for i := 1 to n do
begin
for j := i-1 downto 0 do
if (đoạn j+1..i là palindrome) then F[i] := max( F[i], F[j]+1 );
end;
```

Hai vòng for lồng nhau mất O(N2), phần kiểm tra đoạn j+1..i là palindrome hay không mất O(N), vậy độ phức tạp thuật toán là O(N3). Sẽ không được khả thi nếu N = 1000.Để giảm độ phức tạp thuật toán, ta sử dụng mảng L[i, j] có ý nghĩa tương tự như mảng F[i, j] ở bài 1. QHĐ lập mảng L[i, j] mất N2. Tổng cộng là O(N2) vì mỗi lần kiểm tra chỉ mất O(1).

Nhưng đến đây lại nảy sinh vấn đề: mảng L[i, j] không thể lưu được khi N=1000 vì bộ nhớ của chúng ta chỉ có 640KB. Một cách khắc phục là dùng xử lý bít. Nhưng có cách đơn giản hơn là dùng hai mảng một chiều L[i] và C[i] có ý nghĩa:

* L[i] là độ dài lớn nhất của palindrome độ dài lẻ nhận s[i] làm tâm;

* C[i] là độ dài lớn nhất của palindrome độ dài chẵn nhận s[i] và s[i+1] làm tâm;
L[i] và C[i] có thể tính được bằng cách 2 bài 2 trong O(N2). Phần kiểm tra ta viết lại như sau:

```
function is_palindrome(i, j : integer) : boolean;
var đ : integer;
begin
đ := j-i+1;
if ođ (đ) then is_palindrome := (L[(i+j) div 2] >= n)
else is_palindrome := (C[(i+j) div 2] >= n)
end;
```

Vậy thuật toán của chúng ta có độ phức tạp tính toán là O(N2), chi phí bộ nhớ là O(N).

Bài 4 *: Pal - Ioicamp - Marathon 2005-2006- tuần 17*

Cho một xâu, hỏi nó có bao nhiêu xâu con là palindrome; xâu con ở đây gồm các kí tự không cần liên tiếp (độ dài <= 120).

Ví Dụ:

Pal.inp
IOICAMP
Pal.out
9

Đây là một bài tập rất thú vị. Phương pháp là dùng QHĐ.
Gọi F[i, j] là số palindrome là xâu con của đoạn i..j.
Ta có công thức :
* F[i, i] = 1;
* F[i, j] = F[i+1, j] + F[i, j-1] - F[i+1, j-1] + T;
Nếu s[i] = s[j] thì T = F[i+1, j-1] + 1;
Nếu s[i] <> s[j] thì T = 0;
Đoạn chương trình như sau :

```
procedure lam;
var k, i, j : integer;
begin
n := length(s);
for i := 1 to n do F[i, i] := 1;
for k := 1 to n-1 do
for i := 1 to n-k do
begin
j := i+k;
F[i, j] := F[i, j-1] + F[i+1, j] - F[i+1, j-1];
if s[i] = s[j] then F[i, j] := F[i, j] + F[i+1, j-1] + 1;
end;
end;
```

Để chương trình chạy nhanh hơn, chúng ta sửa lại đoạn mã một chút như sau :

```
procedure lam2;
var k, i, j : integer;
begin
n := length(s);
for i := 1 to n do F[i, i] := 1;
```

```
for k := 1 to n do
for i := 1 to n-k do
begin
j := i+k;
F[i, j] := F[i, j-1] + F[i+1, j];
if s[i] = s[j] then F[i, j] := F[i, j] + 1
else F[i, j] := F[i, j] - F[i+1, j-1];
end;
end;
```

Đoạn chương trình trên chỉ có tính mô phỏng, muốn hoàn thiện bạn phải cài đặt các phép tính cộng trừ số lớn vì kết quả có thể lên tới 2n-1.Độ phức tạp của thuật toán là O(N2). Vì vậy, chúng ta hoàn toàn có thể làm với N = 1000, khí đó cần rút gọn mảng F thành ba mảng một chiều.

Bài 5: *Palindrome - IOI 2000*

Cho một xâu, hỏi phải thêm vào nó ít nhất bao nhiêu xâu kí tự để nó trở thành một palindrome (độ dài <= 500).

Bài này cũng sử dụng QHĐ:

Gọi F[i, j] là số phép biến đổi ít nhất cần thêm vào đoạn i..j để đoạn i..j trở thành palindrome.

Ta có công thức :

* F[i, i] = 0;
* Nếu s[i] = s[j] thì F[i, j] = F[i+1, j-1]
* Nếu s[i] <> s[j] thì F[i, j] = Min(F[i, j-1], F[i+1, j]) + 1;

Muốn chương trình chạy với n = 500 thì cần rút gọn F thành ba mảng một chiều. Muốn truy vết, bạn phải dùng mảng bít hoặc dùng dữ liệu động.

Để thực hành, bạn hãy làm bài tập sau :

Bài 6: The next palindrome - SPOJ

Cho nhiều số <= 106, với mỗi số, tìm số bé nhất có dạng palindrome lớn hơn số đã cho.

Mở rộng với câu hỏi: Tìm số bé thứ k?

Ví Dụ :

Input:

2

808

2133

Output:

818

2222

Gợi ý: dùng phương pháp đếm kết hợp QHĐ.