11 tháng, 2 tuần trước

Mở đầu

* 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 exit(false);
* exit(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<j)
* begin
* if s[i] <> s[j] then exit(false);
* inc(i);
* dec(j);
* end;
* exit(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 Si = Sj )
    - F[i, j] = False; ( nếu Si <> Sj )
  + 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;
    - Kết quả là : Max(j-i+1) <=j thỏa F[i,j] = True.
    - Độ phức tạp thuật toán là O(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 Si có phải là tâm của Palindrome có lẻ kí tự không? ( ví dụ Palindrome MADAM có tâm là kí tự D )
    - Xem Si và Si+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:
    - rocedure Lam;
    - var i, j : Longint ;
    - procedure try( first, last : Longint );
    - var dd : Longint;
    - begin
    - if first = last then
    - begin
    - dd := 1;
    - dec(first);
    - inc(last);
    - end
    - else dd := 0;
    - repeat
    - if (first < 1) or (last > N) then break;
    - if s[i] = s[j] then
    - begin
    - dd := dd + 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[i], 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
  + for j := i-1 downto 0 do
  + if (đoạn j+1..i là palindrome) then F[i] := max( F[i], F[j]+1 );
* Hai vòng for lồng nhau mất O(N​2), phần kiểm tra đoạn j+1..i là palindrome hay không mất Ổ(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 Li và Ci có ý nghĩa:
  + Li là độ dài lớn nhất của palindrome độ dài lẻ nhận Si làm tâm;
  + Ci là độ dài lớn nhất của palindrome độ dài chẵn nhận Si và Si+1 làm tâm;  Li và Ci 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 dd : integer;
    - begin
    - dd := j-i+1;
    - if odd (dd) 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 Si = Sj thì T = F[i+1, j-1] + 1;
    - Nếu Si <> Sj 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 Si = Sj thì F[i, j] = F[i+1, j-1]
  + Nếu Si <> Sj 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 <= 10^6, 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.