### Palindrome

*Olympic Tin học Quốc tế, 2000, Bắc Kinh.*

Dãy ký tự s được gọi là đối xứng (palindrome) nếu các phần tử cách đều đầu và cuối giống nhau. Cho dãy s tạo bởi n ký tự gồm các chữ cái hoa và thường phân biệt và các chữ số. Hãy cho biết cần xoá đi từ s ít nhất bao nhiêu ký tự để thu được một dãy đối xứng. Giả thiết rằng sau khi xoá một số ký tự khỏi s thì các ký tự còn lại sẽ tự động xích lại sát nhau.

Ví dụ, với dãy s gồm 9 ký tự, s = "baeadbadb" thì cần xoá ít nhất 4 ký tự, chẳng hạn, các ký tự thứ 4, 6, 7 và 8, để thu được dãy đối xứng chiều dài 5 là "baeab": "baeadbadb" → "baeab".

Dĩ nhiên là có nhiều cách xoá. Tuy nhiên đáp số là số ít nhất các ký tự cần loại bỏ khỏi s là duy nhất và bằng 4.

#### Thuật toán

Bài toán này đã được nhiều bạn đọc công bố lời giải với một mảng hai chiều kích thước *n2* hoặc vài ba mảng một chiều kích thước *n*, trong đó *n* là chiều dài của dữ liệu vào.

Với một nhận xét nhỏ ta có thể phát hiện ra rằng chỉ cần dùng một mảng một chiều kích thước *n* và một vài biến đơn là đủ.

#### Lập hệ thức quy hoạch động

Gọi *p*(*i*, *j*) là chiều dài của dãy con đối xứng dài nhất thu được khi giải bài toán với dữ liệu vào là đoạn *s*[*i*..*j*]. Khi đó *p*(0, *n-*1) là chiều dài của dãy con đối xứng dài nhất trong dãy *n* ký tự *s*[0..*n-*1] và do đó, số ký tự cần loại bỏ khỏi dãy *s* sẽ là *n-p(0,n-*1*).* Đây chính là đáp số của bài toán.

Ví dụ, với dãy s = "baeadbadb", n = 9 thì p(0,8) = 5 và số ký tự cần xóa sẽ là n – p(0,8) = 9 – 5 = 4.

Ta liệt kê một số tính chất quan trọng của hàm hai biến *p*(*i, j*). Ta có:

* Nếu *i* > *j,* tức là chỉ số đầu trái lớn hơn chỉ số đầu phải, ta quy ước *p*(*i, j*) = 0.
* Nếu *i* = *j* thì *p*(*i, i*) = 1 vì dãy khảo sát chỉ chứa duy nhất một ký tự s[i] nên nó là đối xứng.
* Nếu *i* < *j* và *s*[*i*] = *s*[*j*] thì *p*(*i, j*) = *p*(i + *1*, *j* – 1) + 2. Vì hai ký tự đầu và cuối dãy *s*[*i..j*] giống nhau nên chỉ cần xác định chiều dài của khúc giữa rồi cộng thêm 2 đơn vị ứng với hai ký tự đầu và cuối dãy.
* Nếu *i* < *j* và *s*[*i*] ≠ *s*[*j*], tức là hai ký tự đầu và cuối của dãy con *s*[*i*..*j*] là khác nhau thì ta khảo sát hai dãy con là *s*[*i*..*j*–1] và *s*[*i*+1..*j*] để lấy chiều dài của dãy con đối xứng dài nhất trong hai dãy này làm kết quả:

*p(i,j) = max(p(i,j-1), p(i+1,j))*

Vấn đề đặt ra là cần tính *p*(0, *n-1*). Mà muốn tính được *p*(0, *n-*1) ta phải tính được các *p*(*i, j*) với mọi *i, j* = 0..*n-*1*.*

#### Phương án đệ quy

string s; // string s la bien tong the

// so ki tu nhieu nhat con lai

int P(int i, int j) {

if (i > j) return 0;

if (i == j) return 1;

return (s[i] == s[j]) ? P(i+1, j-1) + 2

: max(P(i, j-1), P(i+1,j));

}

// so ki tu it nhat can xoa

int Pal(string inps) {

s = inps;

return s.length()-P(0, s.length()-1);

}

#### Cải tiến 1: dùng mảng 2 chiều

Phương án đệ quy sẽ phát sinh các lời gọi hàm trùng lặp như đã phân tích trong bài toán trước.

Ta khắc phục điều này bằng cách sử dụng một mảng hai chiều v để tính trước các giá trị của hàm *p*(*i, j*), mỗi giá trị được tính tối đa một lần: v[i][j] = p(i,j).

Ta có, theo hệ thức tính p(i,j),

v[i][j] = 0, nếu i > j: nửa tam giác dưới đường chéo chính toàn 0

v[i][i] = 1: đường chéo chính toàn 1

Nếu i < j

v[i][j] = v[i+1][j-1] + 2, nếu s[i] = s[j]

v[i][j] = max(v[i][j-1], v[i+1][j]), nếu s[i] ≠ s[j]

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 1 |  |  |  |  |  |  | ? ô v[i][j] cần tính  + ô v[i+1][j-1] (chéo trái)  \* ô v[i][j-1] (kề trái)  # ô v[i+1][j] (kề dưới)  *Điền trị cho ma trận v* |
| 0 | 1 |  |  | \* | ? |  |
| 0 | 0 | 1 |  | + | # |  |
| 0 | 0 | 0 | 1 |  |  |  |
| 0 | 0 | 0 | 0 | 1 |  |  |
| 0 | 0 | 0 | 0 | 0 | 1 |  |
| 0 | 0 | 0 | 0 | 0 | 0 | 1 |

Như vậy, ô v[i][j] sẽ được điền trị khi ta biết trị của ba ô bao quanh nó là

+ ô chéo trái v[i+1][j-1]

\* ô kề trái v[i][j-1]

# ô kề dưới v[i+1][j]

Từ đây ta suy ra quy trình điền mảng hai chiều v như sau:

Điền 1 tại đường chéo chính,

Điền 0 vào tam giác dưới đường chéo chính,

Điền các dòng i từ dưới lên trên: i = n-2..0.

Trên mỗi dòng, ta điền trị từ trái qua phải: j = i+1..n-1.

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  | b | a | e | a | d | b | a | d | b |
|  |  | *0* | *1* | *2* | *3* | *4* | *5* | *6* | *7* | *8* |
| b | *0* | 1 | 1 | 1 | 3 | 3 | 5 | 5 | 5 | 5 |
| a | *1* | 0 | 1 | 1 | 3 | 3 | 3 | 3 | 3 | 3 |
| e | *2* | 0 | 0 | 1 | 1 | 1 | 1 | 3 | 3 | 3 |
| a | *3* | 0 | 0 | 0 | 1 | 1 | 1 | 3 | 3 | 3 |
| d | *4* | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 3 | 3 |
| b | *5* | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 3 |
| a | *6* | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 |
| d | *7* | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 |
| b | *8* | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| *Gía trị của hàm p(i,j) đối với dãy baeadbadb*  *i,j=0..8* | | | | | | | | | | |

#### Cải tiến 2: dùng hai mảng 1 chiều

Ta sẽ không theo đuổi phương án dùng mảng hai chiều mà hãy căn cứ vào quy luật điền mảng hai chiều để vận dụng cho hai mảng một chiều là *a và b*, trong đó *a* là mảng ứng với dòng dưới v[i+1][\*], *b* là mảng ứng với dòng trên cần điền trị v[i][\*]*,* i = n-2..0. Cụ thể là

a[j] = v[i+1][j] // dòng i+1

b[j] = v[i][j]) = // dòng i

Sau mỗi dòng ta hoán vị a và b như hai con trỏ mảng.

#### Chương trình

// Palindrome. Phuong an 3: Dung 2 mang 1 chieu

#include <bits/stdc++.h>

using namespace std;

// so ki tu int nhat can xoa

int Pal(const string &s) {

int n = s.length();

if(n == 1) return 0;

int \*a = new int[n]; // dong duoi

int \*b = new int[n]; // dong tren

int \*c;

memset(a, 0, n\*sizeof(int));

memset(b, 0, n\*sizeof(int));

a[n-1] = 1;

for(int i = n-2; i >= 0; --i) {

// dien b qua a

b[i] = 1;

for(int j = i+1; j < n; ++j)

b[j] = (s[i]==s[j]) ? a[j-1] + 2

: max(b[j-1], a[j]);

c = a; a = b; b = c; // hoan vi hai dia chi a, b

}

return n - a[n-1];

}

main() {

cout << Pal("abcvcbaabcvcbaabcvcbaabcvcba") << endl; // xoa 0

cout << Pal("ab1cvcbaa2bcvc3baab1cvcbaa2bcvc3ba") << endl; // xoa 6

cout << Pal("ab1cvcba2abcvc3baab1cvcba2abcvc3ba"); // xoa 4

cout << "\n T h e E n d";

return 0;

}

Nếu muốn, bạn có thể dùng một mảng một chiều với hai biến đơn c và d ghi nhận lại giá trị của dòng chéo dưới (c) và dòng dưới (d).

// so ki tu int nhat can xoa

int Pal(const string &s) {

int n = s.length();

if(n == 1) return 0;

int a[n]; // dong duoi

memset(a, 0, sizeof(a));

int c, d; // c: o cheo duoi, d: o duoi

a[n-1] = 1; // dong cuoi cung n-1

for(int i = n-2; i >= 0; --i) {

// dien dong i

c = 0; a[i] = 1;

// dien o (i,j)

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

d = a[j]; // cat tam dong duoi

a[j] = (s[i]==s[j]) ? c + 2

: max(a[j-1], a[j]);

c = d; // dong cheo duoi

} // j

} // i

return n - a[n-1];

}