The longest palindromes string of Manacher's Algorithm to solve the string

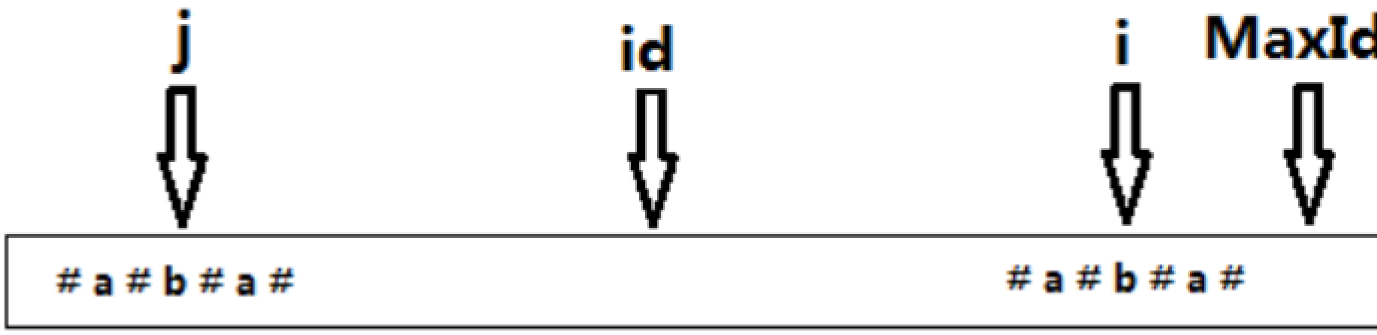
Category: ACM _ data collection2012-11-05 23:11 55 people read the comments (0) Favorites Report

Manacher algorithm: O (n) and the string longest palindromic string 1: The algorithm can be calculated in O (n) time per one character as the center of the longest palindromic string 2: algorithm odd palindrome string and even-numbered palindrome string unify consider : algorithms approximate the process. First insert a separator in the middle of each of the two adjacent characters, of course, this separator does not appear in the original string over. General can use the '#' separated. This very clever odd-length the palindrome string with an even number length palindrome string unify considered (see an example of the following the palindrome string length is odd), and then use an auxiliary array rad record for each character the center of the longest palindromic string information. rad [i] records based on character str [i] is the center of the longest palindrome string str [i] for the first character, the longest palindrome string extends to the right rad [i] characters . The original string: w aa bwsw the fd a new string: $ # w # a # a # b # w # s # w # f # d # \ n; auxiliary array P: 1 2 1 2 3 2 1 2 1 2 1 4 1 212,121 4 There is a good nature, rad [i] -1 is the palindrome substring in the original string length ('#'). 5 This algorithm is linear sweep from front to back. Then when we are prepared to seek rad [i], i rad [j] before we have been. In the the palindrome string before i mx Vocabulary, extends to the rightmost position. This optimal mx id value at the same time made note of this variable id.

6 in order to prevent the character comparisons out of bounds, the first position of a string plus another special character '$' last or '\ 0' do not change, so the new string subscript 1.
7 by the following sentence, the algorithm avoids unnecessary repeat match.

```
if (mx> i)
   rad [i] = min (rad [2 * id-i], mx-i);
```

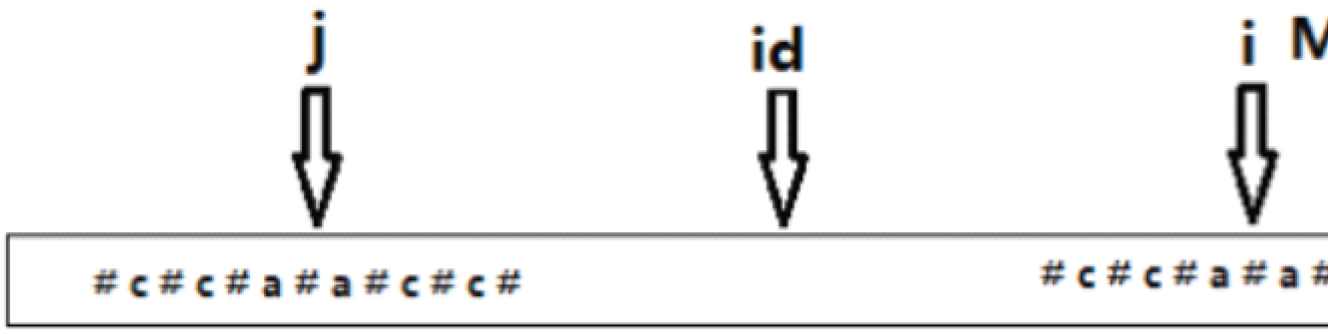
then this sentence is how come it, in fact, is the use of the symmetry of the palindrome string, as follows Figure,



j=2*id-1 即为 i 关于 id 的对称点，根据对称性，P[j]的回文串也是可

但是如果 P[j]的回文串对称过来以后超过 MaxId 的话，超出部分就不能

图，所以这里 P[i]为的下限为两者中的较小者，p[i]=Min(p[2*id-i],Max



算法的有效比较次数为 MaxId 次，所以说这个算法的时间复杂度为 O(n)。

Template:

[Cpp] view plaincopy

```
# Define MAXN 240010

int  ans;
the char  tmpStr [MAXN];
char  String [MAXN];
int  rad MAXN];

/ * Seeking the rad an array * /
void  manacher () {

    ans = 0;
    int  mx = 0;
    int  id;
    int  len = strlen (String);

    for ( int  i = 1; i <len; i + +) {
        if (mx> i)
          rad [i] = min (rad [2 * id-i], mx-i);
        else
          rad [i] = 1;
        for (; String [i + RAD [i]] == String [i-Rad [i]]; the RAD [i] + +);
        if (rad [i] + i> MX) {
          mx = rad [i] + i;
          id = i;
        }
        ans = max (ans, rad [i]);
    }
}

int  main () {
    while (scanf ( "% s"  , tmpStr)! = EOF) {
       / * Seek String * /
       int  pos = 0;
       int  len = strlen (tmpStr);
       String [pos + +] =  '$' ;
       for ( int  i = 0; i <= len; i + +) { / * Here is an enumeration to len * /
          String [pos + +] =  '#' ;
          String [pos + +] = tmpStr [i];
       }
       manacher ();
       printf ( "% d \ n"  , ans-1); / * output the longest length * /
    }
    return  0;
}
```