题目描述：

给出一些字符串，分别求其最长回文子串的长度。

解题思路：

模板题。O(n)的算法有后缀数组和 Manacher 两种。本题用的是 Manacher（后缀数组可能会 T）

代码:

```
#include<cstdio>

#include<cstring>

#include<cstdlib>

#include<iostream>

#include<algorithm> //一堆头文件


using namespace std;

const int maxn=5000010;//没搞清楚到底该开多大。。。反正不会 RE 了

int rad[maxn];
char str[maxn],s[2*maxn+2];
/*str 是读入的字符串所在的数组，而 s 则是根据 str 处理得到的数组
处理方式为：s[0]为'$',除了占个位子没什么别的作用，也就是让下标从1开始更直观。
        从 s[1]开始则是 str 中每两个自字符间插入一个'#'所得的新字符串（含开头末尾空格）。
        例:str 为"abcabc"，则 s 为"$#a#b#c#a#b#c#"。
        这么做的目的是避免对回文子串奇偶性的讨论。str 中每个回文子串都对应 s 中一个长度为奇数的回文子串。
        rad[i]为以 s[i]为中心的最长回文子串的回文半径。
        具体实现见下面的代码。*/
void Manacher(int rad[],char str[],int n)//对于每个 s[i]求 rad[i]
{
   int i,mx=0,id;//我们用 mx 记在 i 之前的回文串中，延伸至最右端的位置。同时用 id 这个变量记下取得这个最优 mx 时的 id 值
    for(i=1;i<n;i++)
   {
     if(mx>i) rad[i]=min(rad[2*id-i],mx-i);//这步为算法核心，由于回文子串的对称性，在以 id 为中心，id 到 mx 的长度为回文半径的范围内凡是关于 s[i]对称的也关于 s[2*id-i]对称，超出此范围则应具体判断.
     else rad[i]=1;
```

```
        for(;str[i+rad[i]]==str[i-rad[i]];rad[i]++)
            ;//每次循环都导致该轮结束后 mx 后移量加1,而 mx 后移量最多为 o(n)，可保证算法是 o(n)的
        if(mx<rad[i]+i)//更新 mx 与 id
        {
            mx=rad[i]+i;
            id=i;
        }
    }
}

int main()
{
    int cnt = 1;

    while(scanf("%s",str)!=EOF && !(str[0] == 'E' && str[1] == 'N' && str[2] == 'D'))
    {
        memset(rad,0,sizeof(rad));
        int len=strlen(str);

        s[0]='$',s[1]='#';

        for(int i=0;i<=len;i++)
        {
            s[2*i+2]=str[i];

            s[2*i+3]='#';

        }
        Manacher(rad,s,2*len+2);

        int ans=1;
        for(int i=0;i<2*len+2;i++)
            ans=max(ans,rad[i]);

        printf("Case %d: %d\n",cnt++, ans-1);//根据 s 与 str 的对应规则，s 中回文半径为 r 的子串对应 str 中长度为 r-1。
    }

    return 0;
}
```

参考资料：http://blog.csdn.net/ggggiqnypgjg/article/details/6645824