# MANACHER

算法简介:算法的目的是在O(n)的时间复杂度内找到一个字符串中各个字母所在的最大长度的回文串。

此算法用到了一个Rad[]数组的定义，Rad[i]表示回文的半径，即最大的j满足str[i-j+1...i] = str[i+1...i+j]。

我们的工作就是把全部的Rad[]求出来。

有两个结论: (1)： 对于一个整数k,如果 (1<=k<=len[i] && len[i-k] < len[i]-k ) 那么, len[i+k] = min( len[i-k], len[i]-k ).

(2) : 对于一个整数k,如果 (1<=k<=len[i] && len[i-k] > len[i]-k ) 那么, len[i+k] = min( len[i-k], len[i]-k ).

const int maxn=1000010;

char str[maxn];//原字符串

char tmp[maxn<<1];//转换后的字符串

int Len[maxn<<1];

//转换原始串

int INIT(char \*st)

{

int i,len=strlen(st);

tmp[0]='@';//字符串开头增加一个特殊字符，防止越界

for(i=1;i<=2\*len;i+=2)

{

tmp[i]='#';

tmp[i+1]=st[i/2];

}

tmp[2\*len+1]='#';

tmp[2\*len+2]='$';//字符串结尾加一个字符，防止越界

tmp[2\*len+3]=0;

return 2\*len+1;//返回转换字符串的长度

}

//Manacher算法计算过程

int MANACHER(char \*st,int len)

{

int mx=0,ans=0,po=0;//mx即为当前计算回文串最右边字符的最大值

for(int i=1;i<=len;i++)

{

if(mx>i)

Len[i]=min(mx-i,Len[2\*po-i]);//在Len[j]和mx-i中取个小

else

Len[i]=1;//如果i>=mx，要从头开始匹配

while(st[i-Len[i]]==st[i+Len[i]])

Len[i]++;

if(Len[i]+i>mx)//若新计算的回文串右端点位置大于mx，要更新po和mx的值

{

mx=Len[i]+i;

po=i;

}

ans=max(ans,Len[i]);

}

return ans-1;//返回Len[i]中的最大值-1即为原串的最长回文子串额长度

}

//对于len[i]的解释：如果i为偶数，则tmp[i]是原串中的字母，（len[i]-1）表示在原串中以i/2为中心， 最长奇回文串长度.如果i为奇数，则tmp[i]为#，(len[i-1]) 表示在原串中以i/2和i/2+1为中心， 最长偶回文串长度.

Hdu 5340

题意，是给出一个字符串，能否分成三个非空回文串。

题解：我们可以发现 第一个和第三个串，一定是最大回文串的某个串,Manacher 求出最大回文串的长度，枚举第一个和最后一个，中间直接判断，中点的最大回文串是否包括了就可以了。复杂度为o(n \* n)无耻卡常数。.