马拉车$manacher$算法习题。

1.模板题

```
#include<bits/stdc++.h>
using namespace std;
typedef long long ll;
const int N=3e7+5,M=1e6+5,inf=0x3f3f3f3f,mod=1e9+7;
#define mst(a) memset(a,0,sizeof a)
#define lx x<<1
#define rx x<<1|1
#define reg register
#define PII pair<int,int>
#define fi first
#define se second
int p[N],n,mx,id;
char a[N],b[N];
int Manacher(){
    int len=0,n=strlen(a);
    b[0]='@',b[1]='#';
    for(int i=0;i<n;i++)
        b[(i<<1)+2]=a[i],b[(i<<1)+3]='#';
    n=(n<<1)+2;
    //cout<<b<<endl;
    for(int i=1;i<n-1;i++){
        p[i]=(mx>i)?min(p[2*id-i],mx-i):1;
        while(b[i+p[i]]==b[i-p[i]]) p[i]++;
        if(i+p[i]>mx) mx=i+p[i],id=i;
        if(p[i]-1>len) len=p[i]-1;
    }
    return len;
}
int main(){
    scanf("%s",a);
    printf("%d\n",Manacher());
    return 0;
}
```

加快时间的方法：1.乘法用位运算。
2.取min用比较来取
3.使用$scanf,printf$读入。

2.P4555 [国家集训队]最长双回文串

传送门

思路：考虑枚举每个$'\#'$位置向左的最大回文串长度和向右的最大回文串长度，取最大值即可。

因为每个$'\#'$都在奇数位置，对于中心i，

他的边界$[i-(p[i]-1),i+(p[i]-1)]$位置肯定是$'\#'$，

所以$i-(p[i]-1)$为左端点的最长回文长度可以被更新$max(p[i]-1,l[i-(p[i]-1)])$

右端同理也可以被更新。

所以我们可以在$manacher$时维护这两个数组。

但是这不是最后的答案，因为对于$l[i]$，可能有$l[i-2]-2>l[i]$，因为回文串去头和尾仍然是回文串，所以可以更新$l[i]=max(l[i],l[i-2]-2)$。

右端同理。

最终取$max(l[i]+r[i])$，即可。

另外此题有个WA点时，$l[i],r[i]$不能为0，根据题目要求不能为空串。

实际上，$l[1],r[1],r[n-1],l[n-1]$外其他都不可能为空串。

```
#include<bits/stdc++.h>
using namespace std;
typedef long long ll;
const int N=2e5+10,M=1e6+5,inf=0x3f3f3f3f,mod=1e9+7;
#define mst(a) memset(a,0,sizeof a)
#define lx x<<1
#define rx x<<1|1
#define reg register
#define PII pair<int,int>
#define fi first
#define se second
char a[N],b[N];
int p[N],l[N],r[N],ans,n;
void Manacher(){
    int mx=0,id=0;
    n=strlen(a);
    b[0]='@',b[1]='#';
    for(int i=0;i<n;i++) b[(i<<1)+2]=a[i],b[(i<<1)+3]='#';
    n=(n<<1)+2;
    //cout<<b<<endl;
    for(int i=1;i<n;i++){
        p[i]=(mx>i)?min(p[(id<<1)-i],mx-i):1;
        while(b[i+p[i]]==b[i-p[i]]) p[i]++;
        if(p[i]+i>mx) mx=p[i]+i,id=i;
        r[i+p[i]-1]=max(p[i]-1,r[i+p[i]-1]);
        l[i-(p[i]-1)]=max(p[i]-1,l[i-(p[i]-1)]);
    }
}
int main(){
    scanf("%s",a);
    Manacher();
    for(int i=n-1;i>0;i-=2) r[i]=max(r[i+2]-2,r[i]);
    for(int i=3;i<n;i+=2) l[i]=max(l[i-2]-2,l[i]);
    for(int i=3;i<n-1;i+=2) ans=max(ans,l[i]+r[i]);
    printf("%d\n",ans);
    return 0;
}
```

3.SP7586 NUMOFPAL - Number of Palindromes

传送门

题意：求字符串中回文串的个数，长度为1的字母也算以一个回文串。

思路1：马拉车算法，显然我们可以求得每个以每个位置i中心的最长回文半径$p[i]$，显然回文串的长度是$p[i]-1$，当回文串的长度是奇数时，其回文子串个数为$\frac{p[i]-1+1}{2}=\frac{p[i]}{2}$，当长度为偶数时，长度为$\frac{p[i]-1}{2}=\frac{p[i]}{2}$

$Update(8.19)$：这里解释下为什么$max_len=p[i]-1$,因为这里的回文串都是奇回文串，所以对于i为中心的回文串原始长度为：$2p[i]-1$，又因为#的总个数为：$p[i]$个，所以该回文串长度为：$2p[i]-1-p[i]=p[i]-1$

所以综上，我们将所有位置的$\frac{p[i]}{2}$求即可。

时间复杂度：$O(n)$

```
#include<bits/stdc++.h>
using namespace std;
typedef long long ll;
const int N=2e3+5,M=1e6+5,inf=0x3f3f3f3f,mod=1e9+7;
#define mst(a) memset(a,0,sizeof a)
#define lx x<<1
#define rx x<<1|1
#define reg register
#define PII pair<int,int>
#define fi first
#define se second
char a[N],b[N];
int p[N];
int Manacher(){
    int mx=0,id=0,n=strlen(a);
    b[0]='@',b[1]='#';
    for(int i=0;i<n;i++)
        b[(i<<1)+2]=a[i],b[(i<<1)+3]='#';
    n=(n<<1)+2;
    for(int i=1;i<n-1;i++){
        p[i]=mx>i?min(p[(id<<1)-i],mx-i):1;
        while(b[i+p[i]]==b[i-p[i]]) p[i]++;
        if(p[i]+i>mx) mx=p[i]+i,id=i;
    }
    int ans=0;
    for(int i=1;i<n-1;i++) ans+=p[i]/2;
    return ans;
}
int main(){
    scanf("%s",a);
    printf("%d\n",Manacher());

    return 0;
}
```

思路2：PAM,显然用PAM求出以每个位置结尾回文后缀个数$num[i]$，也即回文串的个数，然后求和$num[i]$即可。

时间复杂度:$O(n)$

```cpp
#include<bits/stdc++.h>
using namespace std;
typedef long long ll;
const int N=5e5+5,M=1e6+5,inf=0x3f3f3f3f,mod=1e9+7;
#define mst(a) memset(a,0,sizeof a)
#define lx x<<1
#define rx x<<1|1
#define reg register
#define PII pair<int,int>
#define fi first
#define se second
char a[N];
int ch[N][26],len[N],fail[N],cnt=1,last,ans,num[N];
//num[i] 结点i的回文后缀个数.
int main(){
    scanf("%s",a);
    int n=strlen(a);
    fail[0]=1,len[1]=-1;
    for(int i=0;i<n;i++){
        if(a[i]>0) a[i]=(a[i]+ans-97)%26+97;
        while(i-len[last]-1<0||a[i-len[last]-1]!=a[i]) last=fail[last];
        if(!ch[last][a[i]-'a']){
             len[++cnt]=len[last]+2;
             int j=fail[last];  // 此步是为了求 当前子串的fail指针.
             while(a[i-len[j]-1]!=a[i]) j=fail[j];
             fail[cnt]=ch[j][a[i]-'a'];//求出fail指针.
             ch[last][a[i]-'a']=cnt;//插入该结点.
             num[cnt]=num[fail[cnt]]+1; //更新num.
        }
        last=ch[last][a[i]-'a'];//last变为当前结点.
        ans=num[last];
        printf("%d ",ans);
    }
    return 0;
}
```

待补待补.....