Problem A
Palindrome Again

- 对于回文串 首先我们要快速预处理出来每个中心向左右最长能延伸的回文串长度
- 所有长度为奇数的回文串 , 对称中心必然是某个字符 , 长度为偶数的回文串 , 对称中心必然是两个字符间的空隙
- 这个可以用后缀数组 ,hash, 或者 Manacher 算法求解
- 然后我们要处理一个新问题
- 给一个字符串 s, 处理一些询问 (i,j,k),i 开头 i<=j j<=k 问 s.substr(i,j) 到 s.substr(i,k) 之间那么多字符串有多少是回文

- 举个例子
- s="ababa",i=1,j=3,k=5
- 那么 s.substr(i,j) 到 s.substr(i,k)
- 为 "aba","abab","ababa"
- "aba","ababa" 两个串是回文
- 问某个子串是不是回文 , 可以等价为问这个子串的对称中心延伸的长度是不是 >= 子串长度的一半
- 这样 分长度奇偶讨论 就对奇数来说吧
- 子串 (i,j) 到 (i,k) 之间的回文中心的下标对应了一串连续的区间

- 这样 就相当于问 len[x]>=x-i (x>=(i+j)/2,x<=(i+k)/2) 中 x 的个数
- len[x]-x>=i (x>=(i+j)/2,x<=(i+k)/2) 中 x 的个数
- 相当于一个数组 每一次一个询问 (i,j,k)
- 问区间 (i,j) 中 >=k 的数的个数
- 这个问题在线可以用树状数组套平衡树解决 , 离线可以按 k 排序解决

- 下面 对于输入的两个串 s1,s2
- 构建新串 s=s1+"#"+s2+"@".
- 先考虑一个暴力一点的做法
- 把 s 的每个后缀插入字典树
- 当 s1="aba",s2="aba" 的时候 s=aba#aba@
- 对于字典树的每个节点 , 可以统计出来他有多少个儿子是 @, 记为 two
- 可以统计出来他有多少个儿子是 #, 记为 one
- 那么 sigma 每个回文节点 (two-one)*one 就是答案

- 但是字典树是 n^2 的复杂度 怎么办
- 考虑后缀树解决
- s 的后缀树就是把 s 的每个后缀插入字典树, 但通过压缩公共结点使得时间空间复杂度都是 O(n) 的
- 但是此时每个节点表示的不是一个字符 而是一段字符区间
- 于是第一个问题就可以用上了

核心代码

```
void walkTree(int pl,sufNode *p)
{
    int c = 0,cd;
    ll &one=p->one,&two=p->two;
    one=two=0;
    if (p->s > -1) pl += p->t-p->s+1;
    for (int i = 0;i <setSize;i++)
        if (p->son[i])
        {
            walkTree(pl,p->son[i]);
            one+=p->son[i]->one;
            two+=p->son[i]->two;
            ++c;
        }
    if (c == 0)
    {
        cd = p->t-pl+1;
        two++;
        if (cd<=l1)one++;
    }
    cd = p->t-pl+1;
    if (p->s>-1&&p->t<=l1)ans+=(ll)query(cd,max(p->s,cd+lim-1),p->t)*(two-one)*one;
}
```