引自洛谷

题目描述：~~不描述子直接放个网页自己去看~~

阿狸喜欢收藏各种稀奇古怪的东西，最近他淘到一台老式的打字机。打字机上只有28个按键，分别印有26小写英文字母和B□P两个字母。经阿狸研究发现，这个打字机是这样工作的：

1 输入小写字母，打字机的一个凹槽中会加入这个字母(这个字母加在凹槽的最后)。

2 按一下印有 B 的按键，打字机凹槽中最后一个字母会消失。

3 按一下印有 P 的按键，打字机会在纸上打印出凹槽中现有的所有字母并换行，但凹槽中的字母不会消失。

例如，阿狸输入 aPaPBbP，纸上被打印的字符如下： a aa ab

我们把纸上打印出来的字符串从 1 开始顺序编号，一直到 n□打字机有一个非常有趣的功能，在打字机中暗藏一个带数字的小键盘，在小键盘上输入两个数 (x,y)(其中 1≤x,y≤n,1≤x,y≤n)□打字机会显示第 x 个打印的字符串在第 y 个打印的字符串中出现了多少次。

分析

简单来说，我们的目标来说是求，一个串中模式串的个数，很容易想到kmp和ac自动机，多个模式串批配，想到ac自动机。于是便想到模拟建立出ac自动机，然后再做考虑□ac自动机和kmp的区别无非是kmp链上建立fail边，而ac自动机是在trie树上建立fail边。其中ac自动机字符串匹配的方法就是跳fail边从一个起点不断往回跳记录个数。往回跳是啥？不就是树上动嘛，**我们考虑，将fail[j]=i□转化为一条有向边，由i指向j**，这样问题就~~很自然~~转化成了，询问fail树上，以x串结尾的节点为根的子树中有多少节点为y中的节点。于是这便是一道dfs序的题啦啦啦，具体来说，先dfs fail树，得到dfs序，保证任何一个根节点的子树在一段连续的序列上。连续序列单点修改维护区间，那必须是树状数组啊。建立一个BIT□维护dfs序的序列，然后对trie树进行dfs□注意如果是写拆trie的那种ac自动机的话要先复制一份trie□□遇到一个节点，就将其对应dfs序+1，一个节点dfs完后再-1，保证始终只维护一个字符串的贡献。最后，离线处理查询，用类似链式前向星的方式查询一个字符串对另一个字符串的贡献。(记录每个查询对象为y的x字符串终止点，在dfs完一个字符串后统一查询和)

几个坑点

1 不接受任何暴力，一不小心就t qaq 2 trie树如果多加了一个0号节点□BIT维护的序列就要+1(被坑死了555)
3 这tm太难了，~~不愧是NOI~~，怎么想得到这么多技巧和数据结构……

几个好处

做完这题绝堆加深了对ac自动机以及dfs序，树上差分（勉强算吧）的理解，重新意识到nlogn数据结构的强大……

最后贴个代码

```
#include <bits/stdc++.h>
using namespace std;
```

```
const int maxn=1e5+13;
char s[maxn];
int low[maxn];
int ch[maxn][26];
int CH[maxn][26];
int dfn[maxn];//dfs序
queue<int> que;
int fail[maxn];
int a[maxn];//用于树状数组维护
int rt=0,now;
int he1[maxn],he2[maxn];
struct node{
    int nx,v;
}tt[maxn];//fail存边
struct node1{
    int nx,v;
    int num;//询问存边
}q[maxn];
int cnt,cnt1,cnt2,cnt3,p;
int mp[maxn],rmp[maxn];
int fa[maxn],ans[maxn];
void add1(int u,int v)
{
    tt[++cnt1]=(node){he1[u],v},he1[u]=cnt1;
}
void add2(int u,int v,int ci)
{
    q[++cnt2]=(node1){he2[u],v,ci},he2[u]=cnt2;
}
int insert(char s)
{
    if(s=='B') return fa[now];
    if(s=='P')
    {
        mp[now]=++cnt;
        rmp[cnt]=now;
        return now;
    }
    int lo=s-'a';
    if(!ch[now][lo])
    {
        CH[now][lo]=ch[now][lo]=++p;
        fa[p]=now;
        now=p;
    }
    else now=ch[now][lo];
    return now;
 }
void pre_ac()
{
```

```
    int u=rt;
    for(int i=0;i<=25;i++)
    {
        if(ch[u][i]) que.push(ch[u][i]);
    }
    while(!que.empty())
    {
        u=que.front();
        que.pop();
        for(int i=0;i<=25;i++)
        {
            if(ch[u][i])
            {
                fail[ch[u][i]]=ch[fail[u]][i],que.push(ch[u][i]);
            }
            else
            {
                ch[u][i]=ch[fail[u]][i];
            }
        }
    }
}
struct BIT{
    int l(int x)
    {
        return x&(-x);
    }
    void add(int x,int ad) {
    for(;x<=p+1;x+=l(x)) a[x]+=ad;}//实际多算了一个节点……所以p+1
    int query(int x) {
    int ans=0;for(;x;x-=l(x)) ans+=a[x];return ans;}
}bt;
void dfs_1(int u)//对fail树的dfs
{
    dfn[u]=++cnt3;
    for(int i=he1[u];i;i=tt[i].nx)
    {
        int v=tt[i].v;
        dfs_1(v);
    }
    low[u]=cnt3;
}
void dfs(int u)//对trie树的dfs
{
    bt.add(dfn[u],1);
    if(mp[u])
    {
        for(int i=he2[mp[u]];i;i=q[i].nx)
        {
            int v=q[i].v;
            int nw=rmp[v];
```

```
            ans[q[i].num]=bt.query(low[nw])-bt.query(dfn[nw]-1);
        }
    }
    for(int i=0;i<=25;i++)
    {
        int y=CH[u][i];
        if(y) dfs(y);
    }
    bt.add(dfn[u],-1);
}
int main()
{
    int n,u,v;
    scanf("%s",s);
    for(int i=0;s[i];i++) now=insert(s[i]);
    pre_ac();
    for(int i=1;i<=p;i++)
    {
        add1(fail[i],i);
    }
    scanf("%d",&n);
    for(int i=1;i<=n;i++)
    {
        scanf("%d%d",&u,&v);
        add2(v,u,i);
    }
    dfs_1(0);
    dfs(0);
    for(int i=1;i<=n;i++)
    {
        printf("%d\n",ans[i]);
    }
 }
```

这是洛谷上这题的链接阿狸的打字机