```
void Build_AC_automation(Node *root)
{
    int head=0,tail=0;//队列头、尾指针
    queue[head++]=root;//先将root入队
    while(head!=tail)
    {
        Node *p=NULL;
        Node *temp=queue[tail++];//弹出队头结点
        for(int i=0;i<26;i++)
        {
            if(temp->next[i]!=NULL)//找到实际存在的字符结点
            { //temp->next[i] 为该结点，temp为其父结点
                if(temp==root)//若是第一层中的字符结点，则把该结点的失败指针指向root
                    temp->next[i]->fail=root;
                else
                {
                    //依次回溯该节点的父节点的失败指针直到某节点的next[i]与该节点相同，
                    //则把该节点的失败指针指向该next[i]节点；
                    //若回溯到 root 都没有找到，则该节点的失败指针指向 root
                    p=temp->fail;//将该结点的父结点的失败指针给p
                    while(p!=NULL)
                    {
                        if(p->next[i]!=NULL)
                        {
                            temp->next[i]->fail=p->next[i];
                            break;
                        }
                        p=p->fail;
                    }
                    //让该结点的失败指针也指向root
                    if(p==NULL)
                        temp->next[i]->fail=root;
                }
                queue[head++]=temp->next[i];//每处理一个结点，都让该结点的所有孩子依次入队
            }
        }
    }
}
int query(Node *root)
{ //i为主串指针，p为模式串指针
    int i,v,count=0;
    Node *p=root;
    int len=strlen(s);
    for(i=0;i<len;i++)
    {
        v=s[i]-'a';
        //由失败指针回溯查找，判断s[i]是否存在于Trie树中
        while(p->next[v]==NULL && p!=root)
            p=p->fail;
        p=p->next[v];//找到后p指针指向该结点
        if(p==NULL)//若指针返回为空，则没有找到与之匹配的字符
            p=root;
        Node *temp=p;//匹配该结点后，沿其失败指针回溯，判断其它结点是否匹配
        while(temp!=root)//匹配结束控制
        {
            if(temp->cnt>=0)//判断该结点是否被访问
            {
                count+=temp->cnt;//由于cnt初始化为 0，所以只有cnt>0时才统计了单词的个数
                temp->cnt=-1;//标记已访问过
            }
            else//结点已访问，退出循环
                break;
            temp=temp->fail;//回溯 失败指针 继续寻找下一个满足条件的结点
        }
    }
    return count;
}
```