Problem F　　ZOJ 3228

一.题目大意

给出一个长串和 n 个短串，让你找出这 n 个短串在长串中出现了多少次。同时，在这 n 个串中，一种是可以重叠匹配（例"aba"在 ababa 中出现了 2 次），另一种不可重叠匹配("aba"在"ababa"中只计为出现 1 次）。

二.题目算法

题目是基于 AC 自动机的模型的，都是找出长串中匹配的多个子串。但是我们在每一个单词的末尾节点加上两个标记，一个是"重叠匹配"，一个是上一次出现的位置。所以，我们在进行 AC 自动机查找匹配的时候，就可以通过处理得出答案。

三.Trick

题目的 Sample Input 出现了某个字串同时为重叠串和非重叠串，同时推测还有某个字串多次出现的情况。所以应用并查集，才能得出最终的正确解。

四.参考代码

```
//Result:2012-01-28 12:49:29   Accepted 3228   C++ 2130MS 25416KB   Wizmann
#include <cstdio>
#include <cstdlib>
#include <cstring>
#include <algorithm>
#include <string>
#include <iostream>
#include <queue>

using namespace std;

#define print(x) cout<<x<<endl
#define input(x) cin>>x
#define STRLEN 100001
#define N 100010
#define SHORTSTR 8
#define ALPHA 26
#define SIZE 200000
#define ROOT 0

struct node
```

```
{
    int next[ALPHA];
    int fail,last,len;
    int olap,nolap;
};

node trie[SIZE];
int num[N];
char str[STRLEN];
int n,ind;
int father[N];

int findFather(int x)
{
    if(father[x]==x) return x;
    else return father[x]=findFather(father[x]);
}

void trieInsert(char *instr,int endMark,int type)
{
    int ptr=ROOT;
    for(int i=0;instr[i];i++)
    {
        int now=instr[i]-'a';
        if(!trie[ptr].next[now])
        {
            trie[ptr].next[now]=ind++;
        }
        ptr=trie[ptr].next[now];
    }
    //type = 0 denotes substring a is allowed to overlap
    //and type = 1 denotes not.
    if(type)
    {
        father[trie[ptr].nolap]=endMark;
        trie[ptr].nolap=endMark;
    }
    else
    {
        father[trie[ptr].olap]=endMark;
```

```
        trie[ptr].olap=endMark;
    }
    trie[ptr].last=-1;
    trie[ptr].len=strlen(instr);
}

void makeAC()
{
    queue<int> q;
    for(int i=0;i<ALPHA;i++)
    {
        if(trie[ROOT].next[i])
        {
            trie[trie[ROOT].next[i]].fail=ROOT;
            q.push(trie[ROOT].next[i]);
        }
    }

    while(!q.empty())
    {
        int now=q.front();
        q.pop();
        for(int i=0;i<ALPHA;i++)
        {
            if(trie[now].next[i])
            {
                int v=trie[now].next[i];
                int fail=trie[now].fail;
                q.push(v);
                trie[v].fail=trie[fail].next[i];
            }
            else
            {
                int fail=trie[now].fail;
                trie[now].next[i]=trie[fail].next[i];
            }
        }
    }
}
```

```
void search()
{
    int ptr=ROOT;
    for(int i=0;str[i];i++)
    {
        int now=str[i]-'a';
        while(trie[ptr].fail && !trie[ptr].next[now]) ptr=trie[ptr].fail;
        ptr=trie[ptr].next[now];

        int t=ptr;
        while(t)
        {
            if(trie[t].olap||trie[t].nolap)
            {
                if(trie[t].olap) num[findFather(trie[t].olap)]++;
                if(trie[t].nolap && i-trie[t].last>=trie[t].len)
                {
                    num[findFather(trie[t].nolap)]++;
                    trie[t].last=i;
                }
            }
            t=trie[t].fail;
        }
    }
}

int main()
{
    freopen("input.txt","r",stdin);
    char sstr[SHORTSTR];
    int type;
    int cas=1;
    while(scanf("%s",str)!=EOF)
    {
        print("Case "<<cas++);
        ind=1;
        input(n);

        for(int i=0;i<=n;i++) father[i]=i;
        memset(trie,0,sizeof(trie));
```

```
        memset(num,0,sizeof(num));

        for(int i=1;i<=n;i++)
        {
            scanf("%d%s",&type,sstr);
            trieInsert(sstr,i,type);
        }
        makeAC();
        search();
        for(int i=1;i<=n;i++)
        {
            printf("%d\n",num[findFather(i)]);
        }
        print("");
    }
    return 0;
}
```