Z Algorithm

```
#include<bits/stdc++.h>
using namespace std;
string txt,pat,ztxt;

int z[10000000];
void zzz()
{
 int l=0,r=0,k,x=ztxt.size();

    for(int i=1;i<x;i++){

     if(i>r){
         l=r=i;
     while(r<x and ztxt[r]==ztxt[r-l])
                r++;
          z[i]=r-l;r--;
     }

     else{
        k=i-l;
        if(z[k]<r-i+1)z[i]=z[k];

        else{
           l=i;
      while(r<x and ztxt[r]==ztxt[r-l])
                    r++;
             z[i]=r-l;r--;
             }
        }
    }
}

int main()
{
    int t,cs=0;
    cin>>t;

    while(t--){
        cin>>txt>>pat;
        ztxt+=pat;ztxt+='$';ztxt+=txt;
        zzz();
        int a=pat.size(),b=ztxt.size();
       int count=0;

        for(int i=a+1;i<b;i++){
            if(z[i]==a) count++;
        }

    printf("Case %d: %d\n",++cs,count);
        ztxt.clear();
    }
}
```

KMP Algorithm

```
#include<bits/stdc++.h>
using namespace std;
int lps[1000010],n,x;
string txt,pat;

int find_match()
{
    int count=0,i=0,j=0;
    while(i<n){
        if(txt[i]==pat[j]){
            i++;j++;
        }
        else{
            if(j>0)j=lps[j-1];
            else i++;
        }
        if(j==x){
            count++;
            j=lps[j-1];
        }
    }
    return count;
}

void computeLPS()
{
    for(int i=1,len=0;i<x;)
    {
        if(pat[i]==pat[len])
           lps[i++]=++len;
        else{
            if(len>0)len=lps[len-1];
            else lps[i++]=0;
        }
    }
}

int main()
{
    int t,cs=0;
    cin>>t;
    while(t--){
        cin>>txt>>pat;
        n=txt.size(); x=pat.size();
        computeLPS();
        printf("Case %d:
      %d\n",++cs,find_match());
    }
}
```