$exKMP$

算法思想

$exKMP$ 可以求得模式串和文本串的后缀的最长公共前缀

z 数组是自己的后缀和自己的最长公共前缀

$extend$ 数组是文本串的后缀和模式串的最长公共前缀

这里 s 是文本串， t 是模式串

代码实现

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

const int maxn=20000100;
char s[maxn],t[maxn];
int lens,lent,z[maxn],extend[maxn];
void solve_z(){
    z[1]=lent;
    for(int i=2,l=0,r=0;i<=lent;i++){
        if(i<=r) z[i]=min(z[i-l+1],r-i+1);
        while(i+z[i]<=lent&&t[i+z[i]]==t[z[i]+1]) ++z[i];
        if(i+z[i]-1>r) l=i,r=i+z[i]-1;
    }
    long long ans1=0,ans2=0;
    for(int i=1;i<=lent;i++) ans1^=1ll*i*(z[i]+1);
}
void exkmp(){
    solve_z();
    for(int i=1,l=0,r=0;i<=lens;i++){
        if(i<=r) extend[i]=min(z[i-l+1],r-i+1);
        while(i+extend[i]<=lens&&s[i+extend[i]]==t[extend[i]+1])
++extend[i];
        if(i+extend[i]-1>r) l=i,r=i+extend[i]-1;
    }
}
int main(){
    scanf("%s%s",s+1,t+1);
    lens=strlen(s+1);lent=strlen(t+1);
    exkmp();
    long long ans1=0,ans2=0;
    //for(int i=1;i<=lent;i++) printf("%d\n",z[i]);
    //for(int i=1;i<=lens;i++) printf("%d\n",extend[i]);
```

```
    for(int i=1;i<=lent;i++) ans1^=1ll*i*(z[i]+1);
    for(int i=1;i<=lens;i++) ans2^=1ll*i*(extend[i]+1);
    printf("%lld\n%lld",ans1,ans2);
    return 0;
}
```

备用：

```
#include<bits/stdc++.h>

#define N 1000010

using namespace std;

int q,nxt[N],extend[N];
string s,t;

void getnxt()
{
    nxt[0]=t.size();//nxt[0]一定是T的长度
    int now=0;
    while(t[now]==t[1+now]&&now+1<(int)t.size())now++;//这就是从1开始暴力
    nxt[1]=now;
    int p0=1;
    for(int i=2;i<(int)t.size();i++)
    {
        if(i+nxt[i-p0]<nxt[p0]+p0)nxt[i]=nxt[i-p0];//第一种情况
        else
        {//第二种情况
            int now=nxt[p0]+p0-i;
            now=max(now,0);//这里是为了防止i>p的情况
            while(t[now]==t[i+now]&&i+now<(int)t.size())now++;//暴力
            nxt[i]=now;
            p0=i;//更新p0
        }
    }
}

void exkmp()
{
    getnxt();
    int now=0;
    while(s[now]==t[now]&&now<min((int)s.size(),(int)t.size()))now++;//暴力
    extend[0]=now;
    int p0=0;
    for(int i=1;i<(int)s.size();i++)
    {
        if(i+nxt[i-p0]<extend[p0]+p0)extend[i]=nxt[i-p0];//第一种情况
```

```
        else
        {//第二种情况
            int now=extend[p0]+p0-i;
            now=max(now,0);//这里是为了防止i>p的情况
while(t[now]==s[i+now]&&now<(int)t.size()&&now+i<(int)s.size())now++;//暴力
            extend[i]=now;
            p0=i;//更新p0
        }
    }
}

int main()
{
    cin>>s>>t;
    exkmp();
    int len=t.size();
    for(int i=0;i<len;i++)printf("%d ",nxt[i]);//输出nxt
    puts("");
    len=s.size();
    for(int i=0;i<len;i++)printf("%d ",extend[i]);//输出extend
    return 0;
}
```