// EXKMP extend[i] 表示 T 与 S[i,n-1]的最长公共前缀

#include<iostream>#include<stdio.h>#include<string.h>#include<math.h>#include<algorithm>#include<stdlib.h>#include<queue>#include<stack>#include<map>#include<vector>const double PI = acos(-1.0);

template<class T> T gcd(T a, T b) { return b ? gcd(b, a % b) : a; }

template<class T> T lcm(T a, T b) { return a / gcd(a, b) \* b; }

using namespace std;

const int maxn=100010; //字符串长度最大值

int next[maxn],ex[maxn]; //ex数组即为extend数组

//预处理计算next数组

void GETNEXT(char \*str)

{

int i=0,j,po,len=strlen(str);

next[0]=len;//初始化next[0]

while(str[i]==str[i+1]&&i+1<len)//计算next[1]

i++;

next[1]=i;

po=1;//初始化po的位置

for(i=2;i<len;i++)

{

if(next[i-po]+i<next[po]+po)//第一种情况，可以直接得到next[i]的值

next[i]=next[i-po];

else//第二种情况，要继续匹配才能得到next[i]的值

{

j=next[po]+po-i;

if(j<0)j=0;//如果i>po+next[po],则要从头开始匹配

while(i+j<len&&str[j]==str[j+i])//计算next[i]

j++;

next[i]=j;

po=i;//更新po的位置

}

}

}

//计算extend数组

void EXKMP(char \*s1,char \*s2)

{

int i=0,j,po,len=strlen(s1),l2=strlen(s2);

GETNEXT(s2);//计算子串的next数组

while(s1[i]==s2[i]&&i<l2&&i<len)//计算ex[0]

i++;

ex[0]=i;

po=0;//初始化po的位置

for(i=1;i<len;i++)

{

if(next[i-po]+i<ex[po]+po)//第一种情况，直接可以得到ex[i]的值

ex[i]=next[i-po];

else//第二种情况，要继续匹配才能得到ex[i]的值

{

j=ex[po]+po-i;

if(j<0)j=0;//如果i>ex[po]+po则要从头开始匹配

while(i+j<len&&j<l2&&s1[j+i]==s2[j])//计算ex[i]

j++;

ex[i]=j;

po=i;//更新po的位置

}

}

}

//int main()

{

char s1[105], s2[105];

scanf("%s", s1);

scanf("%s", s2);

EXKMP(s1, s2);

for(int i = 0; i <= 10; i++)

cout << ex[i] << ' ' ;

extend[i] 表示 T 与 S[i,n-1]的最长公共前缀

要求求出所有extend[i] (0 <= i < n)

S : aaaabaa

T : aaaaa

4 3 2 1 0 2 1 0 0 0 0

}