int f[2][5005]={},r[2][5005]={};//必须开滚动数组，不开绝对爆空间,由于动态规划中，一个阶段的决策只受上一个阶段的影响，

//因此可以将以前的状态覆盖掉。只要存两个状态，故只开第一个下标为0~1的二维数组

//r[i][j]存第一个字串前i个（一直被轮换），第二个字串前j个最长子序列的个数

char s1[5005]={},s2[5005]={};

int n,m;

int main()

{

scanf("%s",s1+1);//s1+1代表从s1[1]开始读入数据，s[0]不读入

n=strlen(s1+1)-1;

scanf("%s",s2+1);

m=strlen(s2+1)-1;

re int now=1,pre=0;//这两个变量用来判断两个状态中哪个是以前状态（pre），哪个是现在状态（now）（滚动数组的产物）

for(re int k=0;k<=m;k++)

r[0][k]=1;//初始化，长度为0最长子序列方案数为1（第一个序列长度为0）

r[1][0]=1;//长度为0最长子序列方案数为1（第二个序列长度为0）

for(re int i=1;i<=n;i++)

{

for(re int j=1;j<=m;j++)

{

f[now][j]=max(f[pre][j],f[now][j-1]);

r[now][j]=0;//后面的方案数都是由前面加过来的

if(s1[i]==s2[j])f[now][j]=max(f[now][j],f[pre][j-1]+1);

if(s1[i]==s2[j]&&f[now][j]==f[pre][j-1]+1) r[now][j]+=r[pre][j-1];

if(f[pre][j]==f[now][j]) r[now][j]+=r[pre][j];

if(f[now][j-1]==f[now][j]) r[now][j]+=r[now][j-1];//加上f[i-1][j]和f[i][j-1]中=k的方案数

if(f[pre][j-1]==f[now][j]) r[now][j]-=r[pre][j-1];//如果a[i]!=b[j]且f[i-1][j-1]=k，就要减去它的方案数

r[now][j]=(r[now][j]+mod)%mod;//+mod可以省略

}

now=pre;pre=1-pre;//滚动

}

printf("%d\n%d",f[pre][m],r[pre][m]);

return 0;

}