线性类DP

主讲老师：党东

【线型动态规划】

一、线型动态规划

一个问题的状态可以描绘在一条直线上，即状态可以用一维数组来表示，第i个元素的状态与前i-1个元素的状态有关，前i-1个状态组成一个决策序列，它是其它类型动态规划的基础。

典型的线型动态规划有LIS（最长不下降子序列）,LCS（最长公共子序列），部分和问题，区间选择问题等等。

LCS模型

最长公共子序列(Longest Common Subsequence)

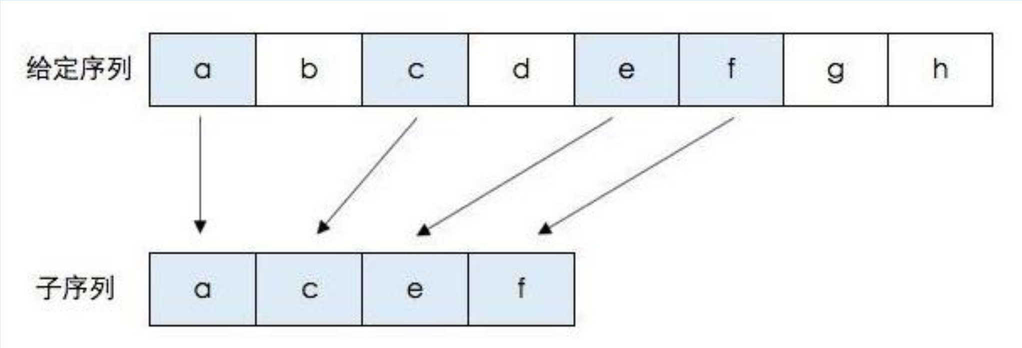
巴蜀中學 BASHU SECONDARY SCHOOL

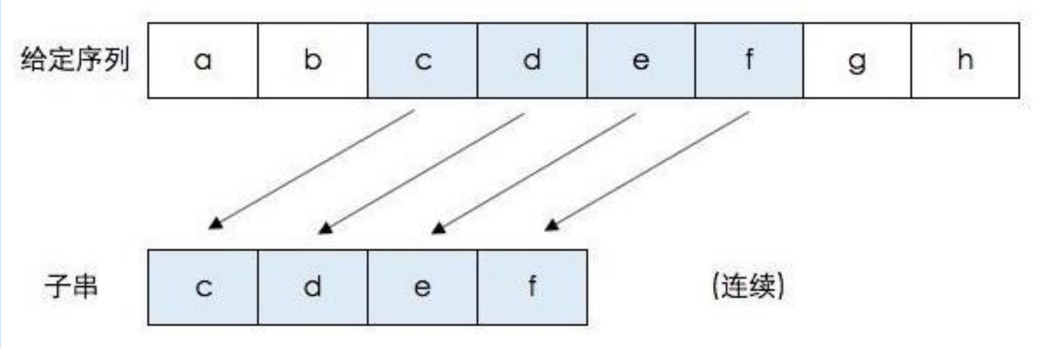
最长公共子序列（longest common sequence）

最长公共子串（longest common substring）

1）什么是子序列呢？即一个给定的序列的子序列，就是将给定序列中**零个或多个元素去掉**之后得到的结果。

2）什么是子串呢？给定串中任意个**连续**的字符组成的子序列称为该串的子串。





【例1】最长公共子序列 --1353

【问题简述】

给定的字符序列 X ="x0，x1，…，xm-1" ，序列Y="y0，y1，…，yk-1"是X的子序列，存在X的一个严格递增下标序列<i0，i1，…，ik-1>，使得对所有的j=0，1，…，k-1，有xij = yj。

例如，X="ABCBDAB"，Y="BCDB"是X的一个子序列。

给出两个字符串S1和S2，长度不超过5000

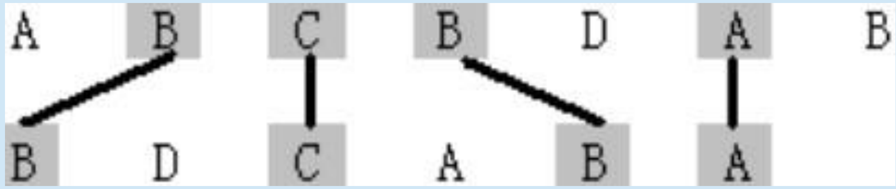
求这两个字符串的最长公共子串长度。

【例1】最长公共子序列 --1353

【问题分析】

S1=“ABCBDAB”

S2=“BDCABA”



可以看出他们的最长公共子序列有BCBA,BDAB，长度为4

从样例分析，我们思考的方式为要找出S1串与S2串的公共子序列，假设将S1固定，从第1个位置开始直到最后一个位置为止，与S2的各个部分不断找最长公共子序列

当然S1也可以变化，这样我们即得出了思路:

枚举S1的位置 i

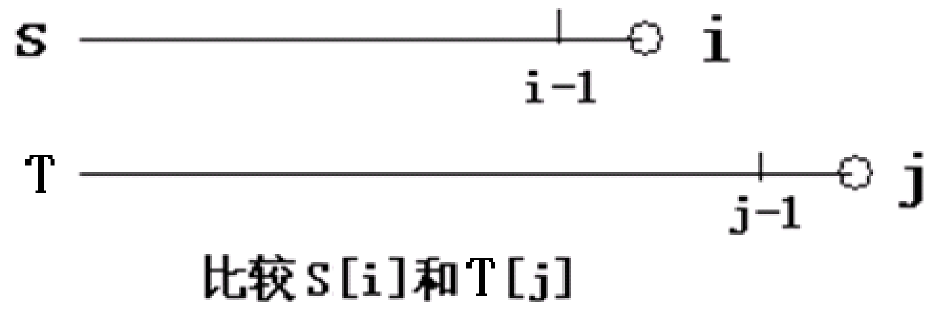
枚举S2的位置 j

找出S1的前i位与S2的前j位的最长公共子序列，直到两个串的最后一个位置为止。

【例1】最长公共子序列 --1353

【问题分析】

S1=设f[i,j]表示S的前i位与T的前j位的最长公共子串长度。则有，



$$f[i,j]=\begin{cases}\max\{f[i-1,j],f[i,j-1]\}, & S[i]\neq T[j]\\ f[i-1,j-1]+1 & S[i]=T[j]\end{cases}$$

时间复杂度O(n*m)

【例1】最长公共子序列 --1353

【参考代码】

```
int f[2005][2005]={0};
int main()
{    int i,j,l1,l2;string s1,s2;
     cin>>s1>>s2;l1=s1.length();l2=s2.length();
     s1=' '+s1;s2=' '+s2;
     for(i=1;i<=l1;i++)
         for(j=1;j<=l2;j++)
             if(s1[i]==s2[j])f[i][j]=f[i-1][j-1]+1;
                     else f[i][j]=max(f[i][j-1],f[i-1][j]);
     cout<<f[l1][l2]<<endl;
}
```

【例2】编辑距离 --1380

Description

设A和B是两个字符串。我们要用最少的字符操作次数，将字符串A转换为字符串B。这里所说的字符操作共有三种：

1、删除一个字符；

2、插入一个字符；

3、将一个字符改为另一个字符。

对任意的两个字符串A和B，计算出将字符串A变换为字符串B所用的最少字符操作次数。

Input

第一行为字符串A；第二行为字符串B；字符串A和B的长度均小于2000。

Output

只有一个正整数，为最少字符操作次数。

Sample Input

sfdqxbw

gfdgw

Sample Output

4

【例2】编辑距离 --1380

【问题分析】

1、阶段和状态：

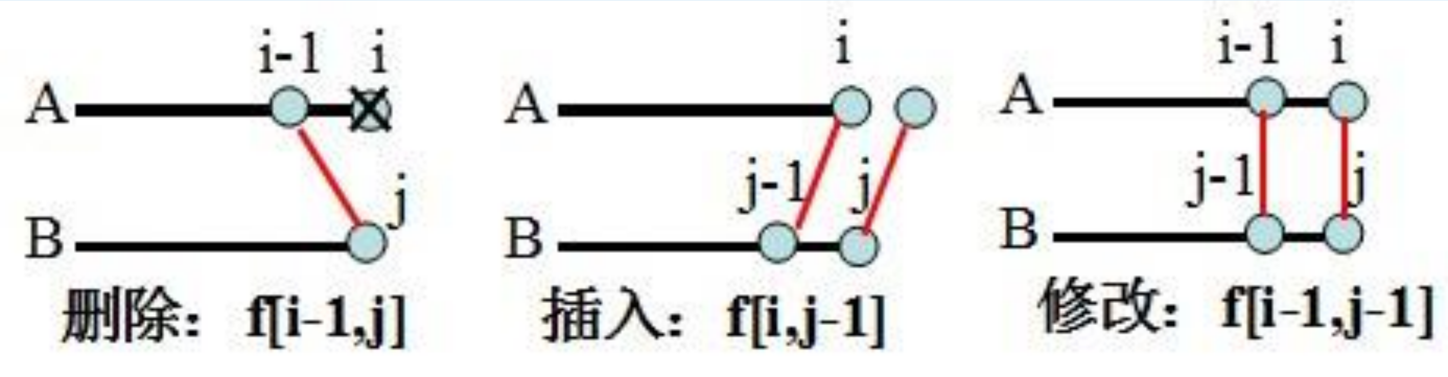
f[i][j]：表示将A的前i个字符变成B的前j个字符的最少操作次数

求f[i][j]要考虑3种操作：

1）在A中删除一个字符：f[i-1][j]+1

2）在A中插入一个字符：f[i][j-1]+1

3）在A中将一个字符改为另一个字符，如果a[i]=b[i]为f[i-1][j-1]，如果a[i]!=b[i]为f[i-1][j-1]+1



【例2】编辑距离 --1380

【问题分析】

2、状态转移方程：

$$f[i][j]=\begin{cases} f[i-1][j-1] & \text{当a[i]= b[i]时} \\ \min\{f[i-1][j], f[i][j-1], f[i-1][j-1]\}+1 & \text{当a[i] <> b[i]时} \end{cases}$$

初始化：f[0][i]=f[i][0]=i;

Answer=f[L1][L2]

【例2】编辑距离 --1380

【参考代码】

```
cin>>s1>>s2;
l1=s1.length();s1=' '+s1;
l2=s2.length();s2=' '+s2;
memset(f,0,sizeof(f));
for(i=1;i<=l1;i++)f[i][0]=i;//初始化
for(i=1;i<=l2;i++)f[0][i]=i;
for(i=1;i<=l1;i++)
    for(j=1;j<=l2;j++)
        if(s1[i]==s2[j])f[i][j]=f[i-1][j-1];
            else f[i][j]=min(f[i-1][j-1]+1,min(f[i-1][j]+1,f[i][j-1]+1));
cout<<f[l1][l2]<<endl;
```

【例2】回文词 --1507

Description

回文词是一种对称的字符串——也就是说，一个回文词，从左到右读和从右到左读得到的结果是一样的。任意给定一个字符串，通过插入若干字符，都可以变成一个回文词。你的任务是写一个程序，求出将给定字符串变成回文词所需插入的最少字符数。比如字符串“Ab3bd”，在插入两个字符后可以变成一个回文词（“dAb3bAd”“Adb3bdA”）。然而，插入两个以下的字符无法使它变成一个回文词。

Input

第一行包含一个整数N，表示给定字符串的长度（3≤N≤50 00）。

第二行是一个长度为N的字符串。字符串仅由大写字母“A”到“Z”，小写字母“a”到“z”和数字“0”到“9”构成。大写字母和小写字母将被认为是不同的。

Output

只有一行，包含一个整数，表示需要插入的最少字符数。

Sample Input

5
Ab3bd

Sample Output

2

【例3】回文词 --1507

【问题分析】

ab3bd

只需变为a**d**b3bd**a**即可，在前面插入d，在后面插入a;

我们分几种情况讨论:

1）若A形如 ?A?，(问号代表任意一个相同字符，下同)则只需将A变为回文串。

2）若A形如?A 再在A的后面插入一个“?”

3）若A形如A? 再在A的前面插入一个“?”

【例3】回文词 --1507

【问题分析】

阶段和状态：设f[i,j]表示使得S的[1..i]和[j..n]部分构成回文需要添加的最少字母个数。

$$f[i,j]=\begin{cases} f[i-1,j+1],\text{当}a[i]=a[j]\text{时} \\ \min\{f[i-1,j],f[i,j+1]\}+1,\text{当}a[i]\neq a[j]\text{时} \end{cases}$$

初始条件：f[0,i]=n-i+1;f[i,n+1]=i;(1<=i<=n)

Ans=min{f[i,i],f[i,i+1]}(1<=i<=n)

时间复杂度为：$O(n^2)$

【例3】回文词 --1507

【问题分析】

```
cin>>n>>s;s=' '+s; //让字符串下标从1开始
for(i=1;i<=n;i++){f[0][i]=n-i+1;f[i][n+1]=i;}//初始化
for(i=1;i<=n;i++)
      for(j=n;j>=i;j--)
          if(s[i]==s[j])f[i][j]=f[i-1][j+1];
                  else f[i][j]=min(f[i-1][j],f[i][j+1])+1;
Ans=0x7fffffff/2;
for(i=1;i<=n;i++)
{   if(f[i][i+1]<Ans)  Ans=f[i][i+1];
    if(f[i][i]<Ans)  Ans=f[i][i];
}
cout<<Ans<<endl;
```

【例4】交错匹配 --1508

Description

有两行非负整数，用 A[i]表示第一行的第 i 个元素，B[j]表示第二行的第 j 个元素，如果A[i]=B[j]=K,那么可以在A[i]和B[j]之间连一条线，称为一个 K匹配，每个数最多只能连一条线。另外，每个K匹配都必须跟一个 L匹配相交且K≠L。小 Z同学想知道，对于任意的两行数，最大匹配是多少?

例如：以下两行数的最大匹配数为8。

Input

输入由三行组成。第一行包含两个用空格隔开的整数 N和M。
第二行包含N个自然数，表示 A[i]。
第三行包含M个自然数，表示 B[i]。

Output

输出一个整数，表示最大匹配数。

【例4】交错匹配 --1508

【样例输入1】

12 11

1 2 3 3 2 4 1 5 1 3 5 10

3 1 2 3 2 4 12 1 5 5 3

【样例输出1】

8

【样例输入2】

4 4

1 1 3 3

1 1 3 3

【样例输出2】

0

【例4】交错匹配 --1508

【LCS类动规】

F[i,j]表示A[]的前i个数与B[]的前j个交错匹配的最多匹配线段的个数。

F[i,j]=max{ F[i-1,j], F[i,j-1], F[la-1,lb-1]+2}

la：表示当必须要求A[]第i个与B[]第j个形成交错时，A[i]在B[j]之前找到的第一个与之相等的数的位置。

lb：表示当必须要求A[]第i个与B[]第j个形成交错时，B[j]在A[i]之前找到的第一个与之相等的数的位置。

解释：F[i,j]是放弃i为F[i-1,j]，放弃j为F[i,j-1]，和必须让A[i]，B[j]形成交错三种情况下最大的一种情况。第三种情况等同求“最长公共子序列”中A串的第i个与B串的第j个相等的情况。