# 序列比对

# 序列比对

- 为确定两个序列之间的相似性或同源性，将它们按照一定的规律排列，进行比对.

- 应用：

  生物信息学中用于研究同源性，如蛋白质序列或 DNA 序列. 在比对中，错配与突变相对应，空位与插入或缺失相对应.

  计算语言学中用于语言进化或文本相似性的研究.

# 序列之间的编辑距离

**编辑距离：**

给定两个序列 $S_1$ 和 $S_2$，
通过一系列字符编辑（插入、删除、替换）等操作，将$S_1$ 转变成 $S_2$.

完成这种转换所需要的最少的编辑操作个数称为 $S_1$ 和$S_2$ 的编辑距离.

# 实例

vintner 转变成 writers，

编辑距离 ≤ 6

vintner

删除 v: –intner

插入 w: wintner

插入 r: wrintner

删除 n: wri–tner

删除 n: writ–er

插入 s: writers

# 子问题界定和归约

$S_1[1..n]$ 和 $S_2[1..m]$ 表示两个序列

子问题：$S_1[1..i]$ 和 $S_2[1..j]$，边界$(i, j)$

| 操作 | 归约子问题 | 编辑距离 |
|---|---|---|
| 删除$S_1[i]$ | $(i\text{-}1, j)$ | +1 |
| $S_1[i]$后插入$S_2[j]$ | $(i, j\text{-}1)$ | +1 |
| $S_1[i]$替换为$S_2[j]$ | $(i\text{-}1, j\text{-}1)$ | +1 |
| $S_1[i]=S_2[j]$ | $(i\text{-}1, j\text{-}1)$ | +0 |

# 优化函数的递推方程

$C[i,j]$：　$S_1[1..i]$ 和 $S_2[1..j]$ 的编辑距离

$$C[i,j]=\min\{C[i-1,j]+1, C[i,j-1]+1,$$
$$C[i-1,j-1]+t[i,j]\}$$

$$t[i,j]=\begin{cases}0 & S_1[i]=S_2[j]\\ 1 & S_1[i]\neq S_2[j]\end{cases}$$

$C[0,j]=j$,

$C[i,0]=i$

# 计算复杂度分析

- 子问题 由 $i, j$界定，
  有$O(m\ n)$ 个子问题
- 每个子问题的计算
  为常数时间
- 算法的时间复杂度是$O(n\ m)$

# 动态规划算法设计要点

(1) 引入参数来界定子问题的边界. 注意子问题的重叠程度.

(2) 给出带边界参数的优化函数定义与优化函数的递推 关系，找到递推关系的初值.

(3) 判断该优化问题是否满足优化原则.

(4) 考虑是否需要标记函数.

# 动态规划算法设计要点(续)

(5) 采用自底向上的实现技术，从最小的子问题开始迭代计算，计算中用备忘录保留优化函数和标记函数的值.

(6) 动态规划算法的时间复杂度是对所有子问题(备忘录)的计算工作量求和(可能需要追踪解的工作量)

(7) 动态规划算法一般使用较多的存储空间，这往往成为限制动态规划算法使用的瓶颈因素.