

## SECTION 6.6

# 6. DYNAMIC PROGRAMMING II

---

- ▶ *sequence alignment*

# String similarity

---

Q. How similar are two strings?

Ex. occurrence and occurance.

o	c	u	r	r	a	n	c	e	-
o	c	c	u	r	r	e	n	c	e

6 mismatches, 1 gap

o	c	-	u	r	r	a	n	c	e
o	c	c	u	r	r	e	n	c	e

1 mismatch, 1 gap

o	c	-	u	r	r	-	a	n	c	e
o	c	c	u	r	r	e	-	n	c	e

0 mismatches, 3 gaps

# Edit distance

---

**Edit distance.** [Levenshtein 1966, Needleman-Wunsch 1970]

- Gap penalty  $\delta$ ; mismatch penalty  $\alpha_{pq}$ .
- Cost = sum of gap and mismatch penalties.

C	T	-	G	A	C	C	T	A	C	G
C	T	G	G	A	C	G	A	A	C	G

$$\text{cost} = \delta + \alpha_{CG} + \alpha_{TA}$$

**Applications.** Unix diff, speech recognition, computational biology, ...

# Sequence alignment

---

**Goal.** Given two strings  $x_1 x_2 \dots x_m$  and  $y_1 y_2 \dots y_n$  find min cost alignment.

**Def.** An **alignment**  $M$  is a set of ordered pairs  $x_i - y_j$  such that each item occurs in at most one pair and no crossings.

$x_i - y_j$  and  $x_{i'} - y_{j'}$  cross if  $i < i'$ , but  $j > j'$

**Def.** The **cost** of an alignment  $M$  is:

$$\text{cost}(M) = \underbrace{\sum_{(x_i, y_j) \in M} \alpha_{x_i y_j}}_{\text{mismatch}} + \underbrace{\sum_{i: x_i \text{ unmatched}} \delta + \sum_{j: y_j \text{ unmatched}} \delta}_{\text{gap}}$$

$x_1$	$x_2$	$x_3$	$x_4$	$x_5$		$x_6$
C	T	A	C	C	-	G
	T	A	C	A	T	G
	$y_1$	$y_2$	$y_3$	$y_4$	$y_5$	$y_6$

**an alignment of CTACCG and TACATG:**

$$M = \{ x_2 - y_1, x_3 - y_2, x_4 - y_3, x_5 - y_4, x_6 - y_6 \}$$

# Sequence alignment: problem structure

---

**Def.**  $OPT(i, j) = \text{min cost of aligning prefix strings } x_1 x_2 \dots x_i \text{ and } y_1 y_2 \dots y_j.$

**Case 1.**  $OPT$  matches  $x_i - y_j$ .

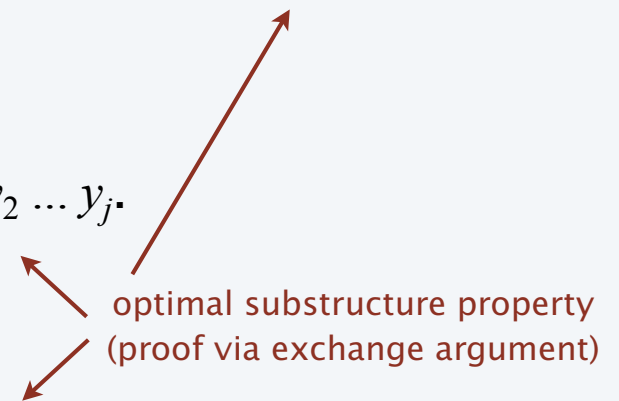
Pay mismatch for  $x_i - y_j$  + min cost of aligning  $x_1 x_2 \dots x_{i-1}$  and  $y_1 y_2 \dots y_{j-1}$ .

**Case 2a.**  $OPT$  leaves  $x_i$  unmatched.

Pay gap for  $x_i$  + min cost of aligning  $x_1 x_2 \dots x_{i-1}$  and  $y_1 y_2 \dots y_j$ .

**Case 2b.**  $OPT$  leaves  $y_j$  unmatched.

Pay gap for  $y_j$  + min cost of aligning  $x_1 x_2 \dots x_i$  and  $y_1 y_2 \dots y_{j-1}$ .



$$OPT(i, j) = \begin{cases} j\delta & \text{if } i = 0 \\ \min \begin{cases} \alpha_{x_i y_j} + OPT(i-1, j-1) \\ \delta + OPT(i-1, j) \\ \delta + OPT(i, j-1) \end{cases} & \text{otherwise} \\ i\delta & \text{if } j = 0 \end{cases}$$

# Sequence alignment: algorithm

---

SEQUENCE-ALIGNMENT ( $m, n, x_1, \dots, x_m, y_1, \dots, y_n, \delta, \alpha$ )

---

FOR  $i = 0$  TO  $m$

$M[i, 0] \leftarrow i\delta.$

FOR  $j = 0$  TO  $n$

$M[0, j] \leftarrow j\delta.$

FOR  $i = 1$  TO  $m$

FOR  $j = 1$  TO  $n$

$M[i, j] \leftarrow \min \{ \alpha[x_i, y_j] + M[i-1, j-1],$   
 $\delta + M[i-1, j],$   
 $\delta + M[i, j-1] \}.$

RETURN  $M[m, n].$

---

## Sequence alignment: analysis

---

**Theorem.** The dynamic programming algorithm computes the edit distance (and optimal alignment) of two strings of length  $m$  and  $n$  in  $\Theta(mn)$  time and  $\Theta(mn)$  space.

**Pf.**

- Algorithm computes edit distance.
- Can trace back to extract optimal alignment itself. ■