

6.6 Sequence Alignment



String Similarity

How **similar** are two strings?

- **ocurrance**
- **occurrence**

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

Applications.

- . Basis for Unix diff.
- . Speech recognition.
- . Computational biology.

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

. Input Parameters :

- . **Gap penalty** δ ;
 - . **Mismatch penalty** α_{pq} .
- . **Cost** = **sum** of **gap** and **mismatch** penalties.

C T G A C C T A C C T

C C T G A C T A C A T

$$\alpha_{TC} + \alpha_{GT} + \alpha_{AG} + 2\alpha_{CA}$$

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

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

$$2\delta + \alpha_{CA}$$



Feasible Solution: Sequence Alignment

INPUT: parameter δ ; matrix $\alpha_{i,j}$;
two strings $X = x_1 x_2 \dots x_m$ and $Y = y_1 y_2 \dots y_n$,
GOAL: find an **alignment** of **minimum cost**.

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**.

Def. The pair x_i - y_j and $x_{i'}$ - $y_{j'}$ **cross** if $i < i'$, but $j > j'$.

$$\text{Cost}(M) = \underbrace{\sum_{x_i-y_j \in M} \alpha_{i,j}}_{\text{Mismatches}} + \underbrace{\sum_{x_i \notin M} \delta}_{\text{Gaps}} + \sum_{y_i \notin M} \delta$$

Ex: CTACCG vs. TACATG.

Sol: $M = x_2-y_1, x_3-y_2, x_4-y_3, x_5-y_4, x_6-y_6$.

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



Designing the Dynamic Programming

FACT. Let M be any Alignment of X and Y .

IF the pair (x_m, y_n) is not in M **THEN**

either x_m is not matched in M or y_n is not matched in M .

Proof.

Otherwise, a **cross** would occur!!!!



Sequence Alignment: Problem Structure

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

- **Case 1:** OPTIMAL SOL matches $x_i - y_j$. THEN:
 - pay for $x_i - y_j$ + min cost of aligning the two strings
 - mismatch** $x_1 x_2 \dots x_{i-1}$ and $y_1 y_2 \dots y_{j-1}$
- **Case 2a:** OPTIMAL SOL leaves x_i unmatched. THEN
 - 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:** OPTIMAL SOL leaves y_j unmatched.
 - pay gap for y_j + min cost of aligning $x_1 x_2 \dots x_i$ and $y_1 y_2 \dots y_{j-1}$

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



Sequence Alignment: Algorithm

```
Sequence-Alignment(m, n,  $x_1x_2\dots x_m$ ,  $y_1y_2\dots y_n$ ,  $\delta$ ,  $\alpha$ ) {  
  for i = 0 to m  
    M[i, 0] =  $i\delta$   
  for j = 0 to n  
    M[0, j] =  $j\delta$   
  
  for i = 1 to m  
    for j = 1 to n  
      M[i, j] = 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]  
}
```

Analysis. $\Theta(mn)$ time and space.

English words or sentences: $m, n \leq 10$.

Computational biology: $m = n = 100,000$. 10 billions ops OK, but 10GB array?



HOMEWORKS

- Give the formal definition of the **SEQUENCE ALIGNMENT PROBLEM**
- Improve the **space** complexity of the Dynamic Programming Algorithm (**Hint**: Do you need to store **all** matrix elements, for **all** the steps?)

