

COL 351 : ANALYSIS & DESIGN OF ALGORITHMS

LECTURE 23

DYNAMIC PROGRAMMING IV: SEQUENCE ALIGNMENT

OCT 06, 2026

|

ROHIT VAISH

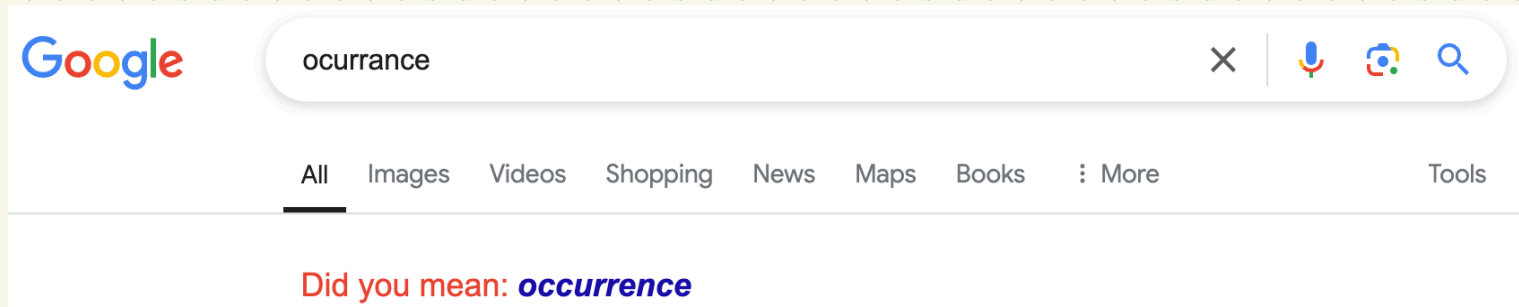
SEQUENCE ALIGNMENT

SEQUENCE ALIGNMENT

How similar are two given strings?

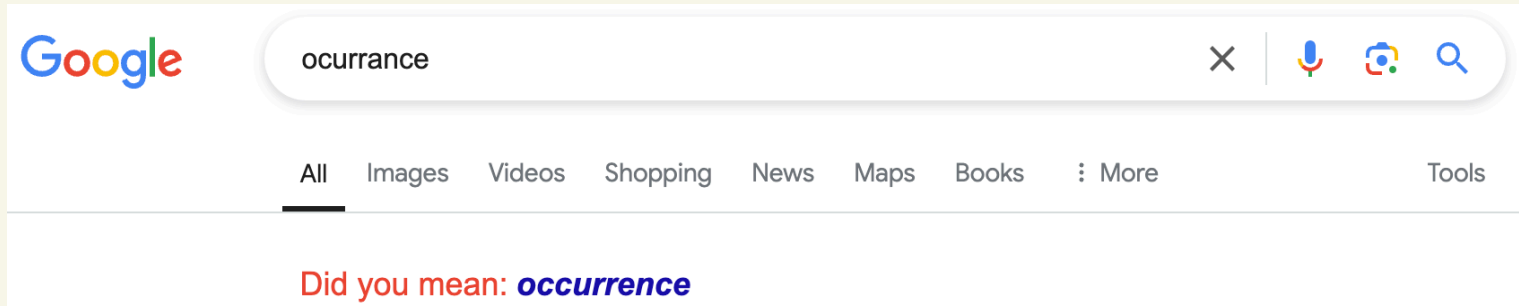
SEQUENCE ALIGNMENT

How similar are two given strings?



SEQUENCE ALIGNMENT

How similar are two given strings?



O _ c u r r a n c e

O C c u r r e n c e

SEQUENCE ALIGNMENT

How similar are two given strings?

ATTGCCCTTACAA TGCTAGGCACCTAGC

ATTGTCCCTTACACTGCTAGGCAGTAGC

SEQUENCE ALIGNMENT

How similar are two given strings?

ATTG CCTTACAA TGCTAGGCAC TAGC

ATTG TCCTTACAC TGCTAGGCAG TAGC

Understanding the functions of specific regions of genomes

Understanding phylogenetics

SEQUENCE ALIGNMENT

input: strings

$$X = x_1 x_2 \dots x_m \quad Y = y_1 y_2 \dots y_n$$

can be of different lengths

SEQUENCE ALIGNMENT

input: strings $X = x_1 x_2 \dots x_m$ $Y = y_1 y_2 \dots y_n$ over alphabet Σ

SEQUENCE ALIGNMENT

input: strings $X = x_1 x_2 \dots x_m$ $Y = y_1 y_2 \dots y_n$ over alphabet Σ

output: an alignment of X and Y with minimum total penalty.

SEQUENCE ALIGNMENT

input: strings $X = x_1 x_2 \dots x_m$ $Y = y_1 y_2 \dots y_n$ over alphabet Σ

output: an alignment of X and Y with minimum total penalty.

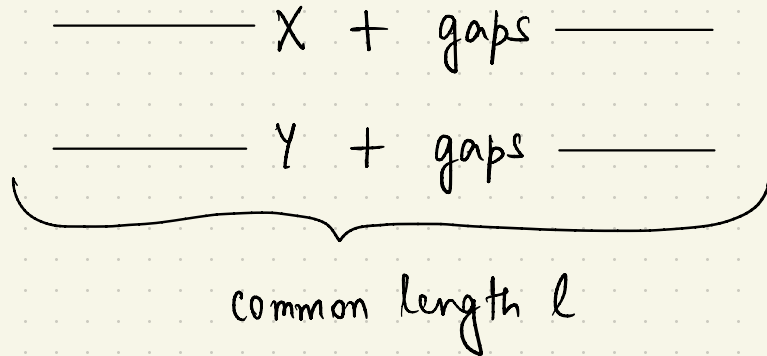
a way of inserting gaps into one or both strings so that they have equal length

SEQUENCE ALIGNMENT

input: strings $X = x_1 x_2 \dots x_m$ $Y = y_1 y_2 \dots y_n$ over alphabet Σ

output: an alignment of X and Y with minimum total penalty.

a way of inserting gaps into one or both strings so that they have equal length



SEQUENCE ALIGNMENT

input: strings $X = x_1 x_2 \dots x_m$ $Y = y_1 y_2 \dots y_n$ over alphabet Σ

output: an alignment of X and Y with minimum total penalty.

a way of inserting gaps into one or both strings so that they have equal length

_____ X + gaps _____
_____ Y + gaps _____
⏟
common length l

e.g.,

O _ c u r r a n c e

O c c u r r e n c e

SEQUENCE ALIGNMENT

input: strings $X = x_1 x_2 \dots x_m$ $Y = y_1 y_2 \dots y_n$ over alphabet Σ

output: an alignment of X and Y with minimum total penalty.

$X = A G G C T$

$Y = A C G G C C$

SEQUENCE ALIGNMENT

input: strings $X = x_1 x_2 \dots x_m$ $Y = y_1 y_2 \dots y_n$ over alphabet Σ

output: an alignment of X and Y with minimum total penalty.

$X = A G G C T$

$Y = A C G G C C$

Which is better:

A G G C T _

A C G G C C

or

A _ G G C _ T

A C G G C C _

?

SEQUENCE ALIGNMENT

input: strings $X = x_1 x_2 \dots x_m$ $Y = y_1 y_2 \dots y_n$ over alphabet Σ

output: an alignment of X and Y with minimum total penalty.

Gap penalty α_{gap}

Mismatch penalty α_{xy}

where $x, y \in \Sigma$

SEQUENCE ALIGNMENT

input: strings $X = x_1 x_2 \dots x_m$ $Y = y_1 y_2 \dots y_n$ over alphabet Σ

output: an alignment of X and Y with minimum total penalty.

Gap penalty $\alpha_{\text{gap}} \geq 0$

Why?

Mismatch penalty α_{xy}

where $x, y \in \Sigma$

SEQUENCE ALIGNMENT

input: strings $X = x_1 x_2 \dots x_m$ $Y = y_1 y_2 \dots y_n$ over alphabet Σ

output: an alignment of X and Y with minimum total penalty.

Gap penalty $\alpha_{\text{gap}} \geq 0$

c _ a _ t
_ d _ o _ g

Mismatch penalty α_{xy}

where $x, y \in \Sigma$

SEQUENCE ALIGNMENT

input: strings $X = x_1 x_2 \dots x_m$ $Y = y_1 y_2 \dots y_n$ over alphabet Σ
a gap penalty $\alpha_{\text{gap}} \geq 0$, a mismatch penalty $\alpha_{xy} \forall x, y \in \Sigma$

output: an alignment of X and Y with minimum total penalty.

SEQUENCE ALIGNMENT

input: strings $X = x_1 x_2 \dots x_m$ $Y = y_1 y_2 \dots y_n$ over alphabet Σ
a gap penalty $\alpha_{\text{gap}} \geq 0$, a mismatch penalty $\alpha_{xy} \forall x, y \in \Sigma$
output: an alignment of X and Y with minimum total penalty.

Sum of all gap and mismatch penalties

Needleman - Wunsch (NW) score

SEQUENCE ALIGNMENT

input: strings $X = x_1 x_2 \dots x_m$ $Y = y_1 y_2 \dots y_n$ over alphabet Σ
a gap penalty $\alpha_{\text{gap}} \geq 0$, a mismatch penalty $\alpha_{xy} \forall x, y \in \Sigma$

output: an alignment of X and Y with minimum total penalty.

SEQUENCE ALIGNMENT

input: strings $X = x_1 x_2 \dots x_m$ $Y = y_1 y_2 \dots y_n$ over alphabet Σ
a gap penalty $\alpha_{\text{gap}} \geq 0$, a mismatch penalty $\alpha_{xy} \forall x, y \in \Sigma$

output: an alignment of X and Y with minimum total penalty.

Example: $\alpha_{\text{gap}} = 1$ $\alpha_{xy} = 2 \forall x, y \in \Sigma$

$X =$ A G T A C G

$Y =$ A C A T A G

SEQUENCE ALIGNMENT

input: strings $X = x_1 x_2 \dots x_m$ $Y = y_1 y_2 \dots y_n$ over alphabet Σ
a gap penalty $\alpha_{\text{gap}} \geq 0$, a mismatch penalty $\alpha_{xy} \forall x, y \in \Sigma$

output: an alignment of X and Y with minimum total penalty.

Example: $\alpha_{\text{gap}} = 1$ $\alpha_{xy} = 2 \forall x, y \in \Sigma$

$X =$ A G T A C G

$Y =$ A C A T A G

NW score = ?

SEQUENCE ALIGNMENT

input: strings $X = x_1 x_2 \dots x_m$ $Y = y_1 y_2 \dots y_n$ over alphabet Σ
a gap penalty $\alpha_{\text{gap}} \geq 0$, a mismatch penalty $\alpha_{xy} \forall x, y \in \Sigma$

output: an alignment of X and Y with minimum total penalty.

Example: $\alpha_{\text{gap}} = 1$ $\alpha_{xy} = 2 \forall x, y \in \Sigma$

$X =$ A G T A C G

$Y =$ A C A T A G

SEQUENCE ALIGNMENT

input: strings $X = x_1 x_2 \dots x_m$ $Y = y_1 y_2 \dots y_n$ over alphabet Σ
a gap penalty $\alpha_{\text{gap}} \geq 0$, a mismatch penalty $\alpha_{xy} \forall x, y \in \Sigma$

output: an alignment of X and Y with minimum total penalty.

Example: $\alpha_{\text{gap}} = 1$ $\alpha_{xy} = 2 \forall x, y \in \Sigma$

$X =$ A G T A C G

$Y =$ A C A T A G

SEQUENCE ALIGNMENT

input: strings $X = x_1 x_2 \dots x_m$ $Y = y_1 y_2 \dots y_n$ over alphabet Σ
a gap penalty $\alpha_{\text{gap}} \geq 0$, a mismatch penalty $\alpha_{xy} \forall x, y \in \Sigma$

output: an alignment of X and Y with minimum total penalty.

Example: $\alpha_{\text{gap}} = 1$ $\alpha_{xy} = 2 \forall x, y \in \Sigma$

$X =$ A G T A C G

$Y =$ A C A T A G

No gaps $\Rightarrow$ penalty = 8

SEQUENCE ALIGNMENT

input: strings $X = x_1 x_2 \dots x_m$ $Y = y_1 y_2 \dots y_n$ over alphabet Σ
a gap penalty $\alpha_{\text{gap}} \geq 0$, a mismatch penalty $\alpha_{xy} \forall x, y \in \Sigma$

output: an alignment of X and Y with minimum total penalty.

Example: $\alpha_{\text{gap}} = 1$ $\alpha_{xy} = 2 \forall x, y \in \Sigma$

$X =$ A G T A C G

A _ _ G T A C G

$Y =$ A C A T A G

A C A _ T A _ G

Four gaps $\Rightarrow$ penalty = 4

SEQUENCE ALIGNMENT

input: strings $X = x_1 x_2 \dots x_m$ $Y = y_1 y_2 \dots y_n$ over alphabet Σ
a gap penalty $\alpha_{\text{gap}} \geq 0$, a mismatch penalty $\alpha_{xy} \forall x, y \in \Sigma$

output: an alignment of X and Y with minimum total penalty.

Example: $\alpha_{\text{gap}} = 1$ $\alpha_{xy} = 2 \forall x, y \in \Sigma$

$X =$ A G T A C G

$Y =$ A C A T A G

Is penalty ≤ 3 possible?

SEQUENCE ALIGNMENT

input: strings $X = x_1 x_2 \dots x_m$ $Y = y_1 y_2 \dots y_n$ over alphabet Σ
a gap penalty $\alpha_{\text{gap}} \geq 0$, a mismatch penalty $\alpha_{xy} \forall x, y \in \Sigma$

output: an alignment of X and Y with minimum total penalty.

Example: $\alpha_{\text{gap}} = 1$ $\alpha_{xy} = 2 \forall x, y \in \Sigma$

$X =$ A G T A C G

$Y =$ A C A T A G

Is penalty ≤ 3 possible? No!

SEQUENCE ALIGNMENT

input: strings $X = x_1 x_2 \dots x_m$ $Y = y_1 y_2 \dots y_n$ over alphabet Σ
a gap penalty $\alpha_{\text{gap}} \geq 0$, a mismatch penalty $\alpha_{xy} \forall x, y \in \Sigma$

output: an alignment of X and Y with minimum total penalty.

Example: $\alpha_{\text{gap}} = 1$ $\alpha_{xy} = 2 \forall x, y \in \Sigma$

$X =$ A G T A C G

$Y =$ A C A T A G

Is penalty ≤ 3 possible? No!

Strings of equal length $\Rightarrow$ #gaps is even

SEQUENCE ALIGNMENT

input: strings $X = x_1 x_2 \dots x_m$ $Y = y_1 y_2 \dots y_n$ over alphabet Σ
a gap penalty $\alpha_{\text{gap}} \geq 0$, a mismatch penalty $\alpha_{xy} \forall x, y \in \Sigma$

output: an alignment of X and Y with minimum total penalty.

Example: $\alpha_{\text{gap}} = 1$ $\alpha_{xy} = 2 \forall x, y \in \Sigma$

$X =$ A G T A C G

$Y =$ A C A T A G

Is penalty ≤ 3 possible? No!

Strings of equal length $\Rightarrow$ #gaps is even

If #gaps = 0, then #mismatches = 4

SEQUENCE ALIGNMENT

input: strings $X = x_1 x_2 \dots x_m$ $Y = y_1 y_2 \dots y_n$ over alphabet Σ
a gap penalty $\alpha_{\text{gap}} \geq 0$, a mismatch penalty $\alpha_{xy} \forall x, y \in \Sigma$

output: an alignment of X and Y with minimum total penalty.

Example: $\alpha_{\text{gap}} = 1$ $\alpha_{xy} = 2 \forall x, y \in \Sigma$

$X =$ A G T A C G

$Y =$ A C A T A G

Is penalty ≤ 3 possible? No!

Strings of equal length $\Rightarrow$ #gaps is even

If #gaps = 0, then #mismatches = 4

If #gaps = 4, then penalty ≥ 4

SEQUENCE ALIGNMENT

input: strings $X = x_1 x_2 \dots x_m$ $Y = y_1 y_2 \dots y_n$ over alphabet Σ
a gap penalty $\alpha_{\text{gap}} \geq 0$, a mismatch penalty $\alpha_{xy} \neq 0, x, y \in \Sigma$

output: an alignment of X and Y with minimum total penalty.

Example: $\alpha_{\text{gap}} = 1$ $\alpha_{xy} = 2 \neq 0, x, y \in \Sigma$

$X =$ A G T A C G

$Y =$ A C A T A G

Is penalty ≤ 3 possible? No!

Strings of equal length $\Rightarrow$ #gaps is even

If #gaps = 0, then #mismatches = 4

If #gaps = 4, then penalty ≥ 4

If #gaps = 2, then #mismatches ≥ 1 .

SEQUENCE ALIGNMENT

input: strings $X = x_1 x_2 \dots x_m$ $Y = y_1 y_2 \dots y_n$ over alphabet Σ
a gap penalty $\alpha_{\text{gap}} \geq 0$, a mismatch penalty $\alpha_{xy} \forall x, y \in \Sigma$

output: an alignment of X and Y with minimum total penalty.

Example: $\alpha_{\text{gap}} = 1$ $\alpha_{xy} = 2 \forall x, y \in \Sigma$

$X =$ A G T A C G

$Y =$ A C A T A G

NW score = 4.

SEQUENCE ALIGNMENT

input: strings $X = x_1 x_2 \dots x_m$ $Y = y_1 y_2 \dots y_n$ over alphabet Σ
a gap penalty $\alpha_{\text{gap}} \geq 0$, a mismatch penalty $\alpha_{xy} \forall x, y \in \Sigma$
output: an alignment of X and Y with minimum total penalty.

(Exercise)

alignments between X and Y : **exponential** in $(n + m)$.

Brute force is prohibitive 😞

OPTIMAL SUBSTRUCTURE

OPTIMAL SUBSTRUCTURE

_____ X + gaps _____

_____ Y + gaps _____

OPTIMAL SUBSTRUCTURE

equal length $\left\{ \begin{array}{l} \text{---} X + \text{gaps ---} \\ \text{---} Y + \text{gaps ---} \end{array} \right.$

OPTIMAL SUBSTRUCTURE

equal length { $\text{---} X + \text{gaps}$ --- 

$\text{---} Y + \text{gaps}$ --- 

OPTIMAL SUBSTRUCTURE

How many relevant cases?

equal length { $\text{---} X + \text{gaps}$ 
 $\text{---} Y + \text{gaps}$ 

OPTIMAL SUBSTRUCTURE

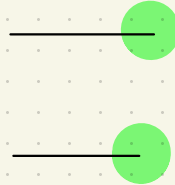
How many relevant cases? Three

equal length { $\text{---} X + \text{gaps}$ 
 $\text{---} Y + \text{gaps}$ 

OPTIMAL SUBSTRUCTURE

How many relevant cases? Three

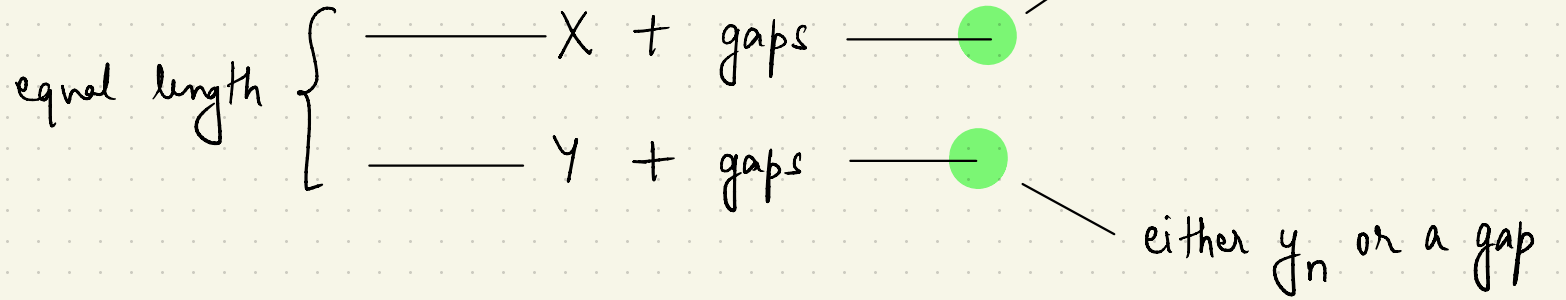
equal length {
——— X + gaps
——— Y + gaps



either n_m or a gap

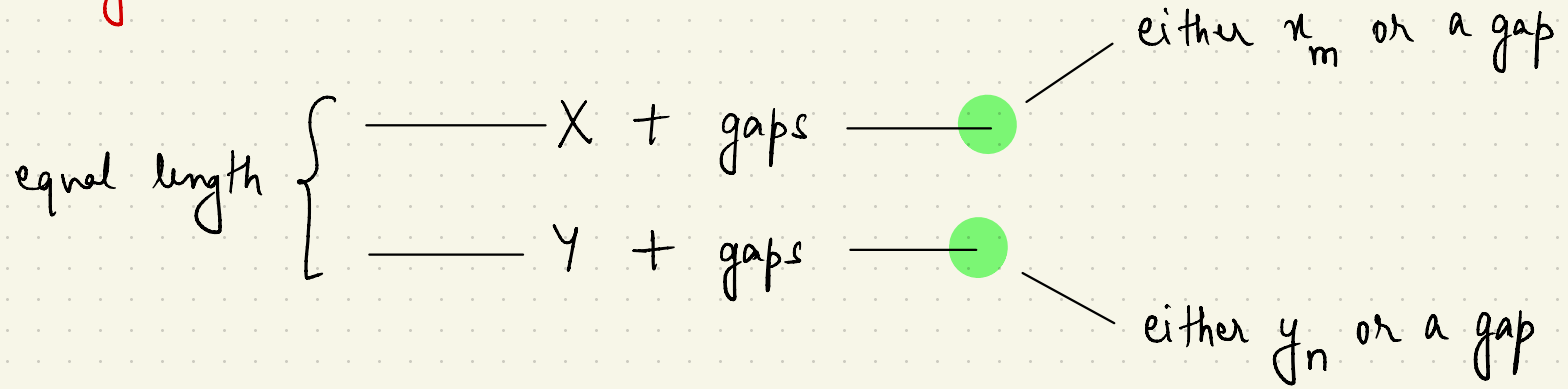
OPTIMAL SUBSTRUCTURE

How many relevant cases? Three



OPTIMAL SUBSTRUCTURE

How many relevant cases? Three



$\alpha_{\text{gap}} \geq 0 \Rightarrow$ both can't be gaps!

OPTIMAL SUBSTRUCTURE

— X + gaps — ●

— Y + gaps — ●

OPTIMAL SUBSTRUCTURE

Let $X' := X - x_m$ $Y' := Y - y_n$

— $X + \text{gaps}$ — 
— $Y + \text{gaps}$ — 

OPTIMAL SUBSTRUCTURE

Let $X' := X - x_m$ $Y' := Y - y_n$

— $X + \text{gap}$ — 
— $Y + \text{gap}$ — 

Case I: Final characters are x_m and y_n

Case II: Final characters are x_m and gap

Case III: Final characters are gap and y_n

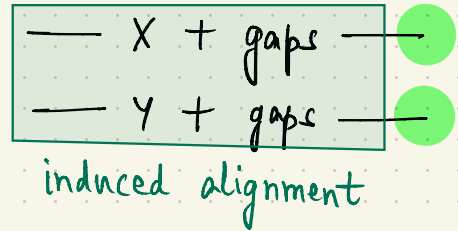
OPTIMAL SUBSTRUCTURE

Let $X' := X - x_m$ $Y' := Y - y_n$

Case I: Final characters are x_m and y_n

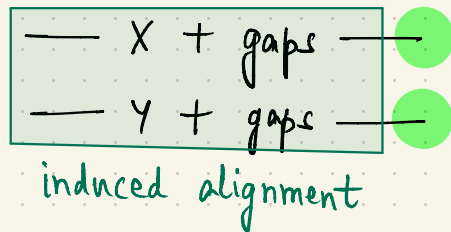
Case II: Final characters are x_m and gap

Case III: Final characters are gap and y_n



OPTIMAL SUBSTRUCTURE

Let $X' := X - x_m$ $Y' := Y - y_n$



Case I: Final characters are x_m and y_n

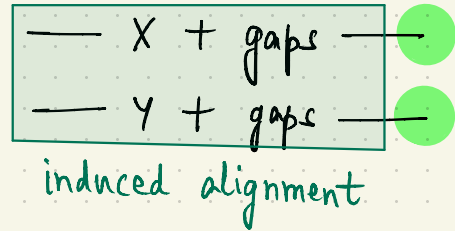
$\Rightarrow$ induced alignment of X' and Y' is optimal.

Case II: Final characters are x_m and gap

Case III: Final characters are gap and y_n

OPTIMAL SUBSTRUCTURE

Let $X' := X - x_m$ $Y' := Y - y_n$



Case I: Final characters are x_m and y_n

$\Rightarrow$ induced alignment of X' and Y' is optimal.

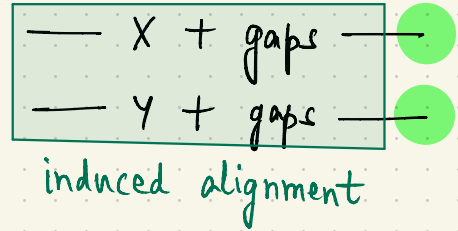
Case II: Final characters are x_m and gap

$\Rightarrow$ induced alignment of X' and Y is optimal.

Case III: Final characters are gap and y_n

OPTIMAL SUBSTRUCTURE

Let $X' := X - x_m$ $Y' := Y - y_n$



Case I: Final characters are x_m and y_n

$\Rightarrow$ induced alignment of X' and Y' is optimal.

Case II: Final characters are x_m and gap

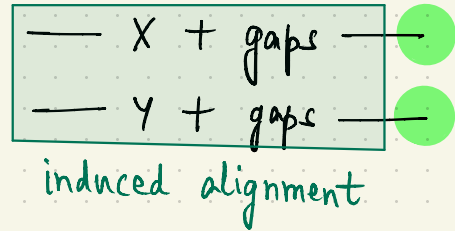
$\Rightarrow$ induced alignment of X' and Y is optimal.

Case III: Final characters are gap and y_n

$\Rightarrow$ induced alignment of X and Y' is optimal.

OPTIMAL SUBSTRUCTURE

Let $X' := X - x_m$ $Y' := Y - y_n$



Case I: Final characters are x_m and y_n

$\Rightarrow$ induced alignment of X' and Y' is optimal.

Case II: Final characters are x_m and gap

$\Rightarrow$ induced alignment of X' and Y is optimal.

Case III: Final characters are gap and y_n

$\Rightarrow$ induced alignment of X and Y' is optimal.

Exercise: Prove formally.

RELEVANT SUBPROBLEMS

Subproblems are **two-dimensional** (like knapsack)

$X_i :=$ first i letters of X

$Y_j :=$ first j letters of Y

THE RECURRENCE

THE RECURRENCE

Define $P_{i,j} :=$ penalty of optimal alignment of x_i and y_j

THE RECURRENCE

Define $P_{i,j} :=$ penalty of optimal alignment of X_i and Y_j

For all $i = 1, 2, \dots, m$ and $j = 1, 2, \dots, n$

THE RECURRENCE

Define $P_{i,j} :=$ penalty of optimal alignment of X_i and Y_j

For all $i = 1, 2, \dots, m$ and $j = 1, 2, \dots, n$

$$P_{i,j} =$$

// case I

// case II

// case III

THE RECURRENCE

Define $P_{i,j} :=$ penalty of optimal alignment of x_i and y_j

For all $i = 1, 2, \dots, m$ and $j = 1, 2, \dots, n$

$$\alpha_{x_i y_j} + P_{(i-1), (j-1)} \quad // \text{ case I}$$

$$P_{i,j} = \quad // \text{ case II}$$

$$// \text{ case III}$$

THE RECURRENCE

Define $P_{i,j} :=$ penalty of optimal alignment of x_i and y_j

For all $i = 1, 2, \dots, m$ and $j = 1, 2, \dots, n$

$$P_{i,j} = \begin{aligned} & \alpha_{x_i y_j} + P_{(i-1), (j-1)} \quad // \text{ case I} \\ & \alpha_{\text{gap}} + P_{(i-1), j} \quad // \text{ case II} \\ & \quad // \text{ case III} \end{aligned}$$

THE RECURRENCE

Define $P_{i,j} :=$ penalty of optimal alignment of x_i and y_j

For all $i = 1, 2, \dots, m$ and $j = 1, 2, \dots, n$

$$P_{i,j} = \begin{aligned} & \alpha_{x_i y_j} + P_{(i-1), (j-1)} \quad // \text{ case I} \\ & \alpha_{\text{gap}} + P_{(i-1), j} \quad // \text{ case II} \\ & \alpha_{\text{gap}} + P_{i, (j-1)} \quad // \text{ case III} \end{aligned}$$

THE RECURRENCE

Define $P_{i,j} :=$ penalty of optimal alignment of x_i and y_j

For all $i = 1, 2, \dots, m$ and $j = 1, 2, \dots, n$

$$P_{i,j} = \min \begin{cases} \alpha_{x_i y_j} + P_{(i-1), (j-1)} & // \text{ case I} \\ \alpha_{\text{gap}} + P_{(i-1), j} & // \text{ case II} \\ \alpha_{\text{gap}} + P_{i, (j-1)} & // \text{ case III} \end{cases}$$

THE RECURRENCE

Define $P_{i,j} :=$ penalty of optimal alignment of x_i and y_j

For all $i = 1, 2, \dots, m$ and $j = 1, 2, \dots, n$

$$P_{i,j} = \min \begin{cases} \alpha_{x_i y_j} + P_{(i-1), (j-1)} & // \text{ case I} \\ \alpha_{\text{gap}} + P_{(i-1), j} & // \text{ case II} \\ \alpha_{\text{gap}} + P_{i, (j-1)} & // \text{ case III} \end{cases}$$

Base cases: $P_{i,0} := ?$

THE RECURRENCE

Define $P_{i,j} :=$ penalty of optimal alignment of x_i and y_j

For all $i = 1, 2, \dots, m$ and $j = 1, 2, \dots, n$

$$P_{i,j} = \min \begin{cases} \alpha_{x_i y_j} + P_{(i-1), (j-1)} & // \text{ case I} \\ \alpha_{\text{gap}} + P_{(i-1), j} & // \text{ case II} \\ \alpha_{\text{gap}} + P_{i, (j-1)} & // \text{ case III} \end{cases}$$

Base cases: $P_{i,0} := i \cdot \alpha_{\text{gap}}$

THE RECURRENCE

Define $P_{i,j} :=$ penalty of optimal alignment of x_i and y_j

For all $i = 1, 2, \dots, m$ and $j = 1, 2, \dots, n$

$$P_{i,j} = \min \begin{cases} \alpha_{x_i y_j} + P_{(i-1), (j-1)} & // \text{ case I} \\ \alpha_{\text{gap}} + P_{(i-1), j} & // \text{ case II} \\ \alpha_{\text{gap}} + P_{i, (j-1)} & // \text{ case III} \end{cases}$$

Base cases:

$$P_{i,0} := i \cdot \alpha_{\text{gap}}$$

$$P_{0,j} := j \cdot \alpha_{\text{gap}}$$

THE ALGORITHM

THE ALGORITHM

A = two-dimensional array of size $(m+1) \times (n+1)$

THE ALGORITHM

A = two-dimensional array of size $(m+1) \times (n+1)$

// base cases

$$A[i, 0] = i \cdot \alpha_{\text{gap}} \quad \forall i$$

$$A[0, j] = j \cdot \alpha_{\text{gap}} \quad \forall j$$

THE ALGORITHM

A = two-dimensional array of size $(m+1) \times (n+1)$

// base cases

$$A[i, 0] = i \cdot \alpha_{\text{gap}} \quad \forall i$$

$$A[0, j] = j \cdot \alpha_{\text{gap}} \quad \forall j$$

// main loop

for $i = 1, 2, \dots, m$

for $j = 1, 2, \dots, n$

THE ALGORITHM

A = two-dimensional array of size $(m+1) \times (n+1)$

// base cases

$$A[i, 0] = i \cdot \alpha_{\text{gap}} \quad \forall i$$

$$A[0, j] = j \cdot \alpha_{\text{gap}} \quad \forall j$$

// main loop

for $i = 1, 2, \dots, m$

for $j = 1, 2, \dots, n$

$$A[i, j] = \min \begin{cases} A[i-1, j-1] + \alpha_{x_i, y_j} \\ A[i-1, j] + \alpha_{\text{gap}} \\ A[i, j-1] + \alpha_{\text{gap}} \end{cases}$$

THE ALGORITHM

Correctness : By induction (Exercise)

THE ALGORITHM

Correctness : By induction (Exercise)

Running time : $O(m \cdot n)$ $O(1)$ work each of $O(mn)$ subproblems

THE ALGORITHM

Correctness : By induction (Exercise)

Running time : $O(m \cdot n)$ $O(1)$ work each of $O(mn)$ subproblems

Reconstruction : Easy Case I, II, III explicitly tells us where a gap is inserted.