Lecture 12: Dynamic Programming 2

CS 341: Algorithms

Tuesday, Feb 14th 2019

Outline For Today

1. Sequence Alignment

Sequence Alignment

- Fundamental Problem is Computational Genomics.
- Given 2 DNA strands, i.e., strings of A, G, T, C
- Want to determine how “similar” they are.

Q: Are these 2 strands similar?

X	A	G	G	A	A	T	T

Y	A	G	G	C	T	T

Underdefined. Need to define what similar means.

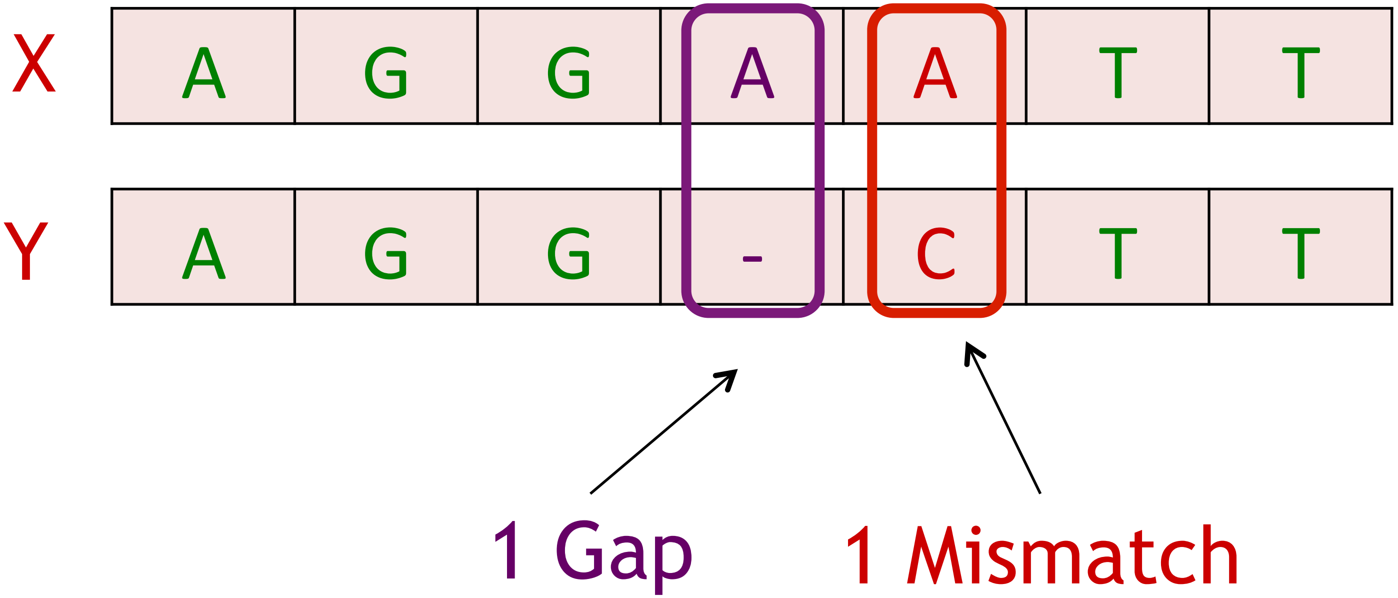
High-level idea: If we can align them well, they’re similar, o.w., they’re not similar.

Applications of Sequence Alignment

- Extrapolation
  - E.g: You know the function of a DNA fragment in a mouse
  - You want to find the most similar part in a human genome
- Similarity reflects proximity of species

Similarity Intuition

- X and Y can be aligned “quite well”.
- Alignment 1:



X	A	G	G	A	A	T	T
Y	A	G	G	-	C	T	T

1 Gap

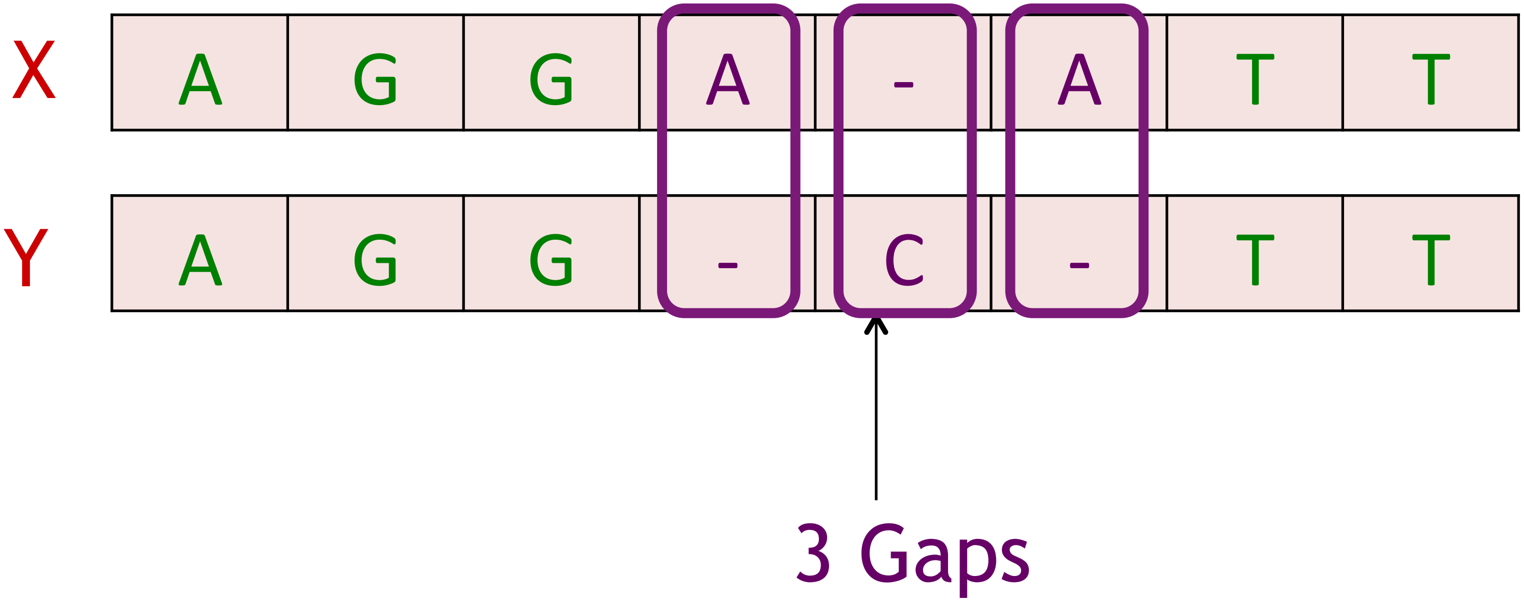
1 Mismatch

Still underdefined.

There can be multiple ways to align.

Similarity Intuition

- Alignment 2:



X	A	G	G	A	-	A	T	T
Y	A	G	G	-	C	-	T	T

Need to define which is better:

(1 gap, 1 mismatch) OR (3 gaps, 0 mismatches)

Formal Computational Problem

- Assume we have experimentally determined penalties:
  1. Gaps: $\delta \geq 0$
  2. Mismatches for (i, j): $\alpha(i, j) \geq 0$ (assume: $\alpha(i, i) = 0$)
- Penalty of an alignment A: sum over penalties
- Input: (1) $X=x_1 \ldots x_m$; (2) $Y=y_1 \ldots y_n$; (3) $\delta \geq 0$; and (4) $\alpha(i,j) \geq 0$
- Output: NW-Score(X, Y), min penalty of any alignment

  (with final equal string lengths)

Formal Name in Genomics: Needleman-Wunsch score (1970s)

Small NW Score

between X and Y

=> X and Y are similar

Interesting Note

- NW Score definition is intrinsically algorithmic
- I.e. without fast algorithms, NW score is useless

Why?

You cannot even answer the question of what's the NW score of two strings X and Y without computing the best alignment between them!

Recall Recipe of a DP Algorithm

1: Identify small # of subproblems

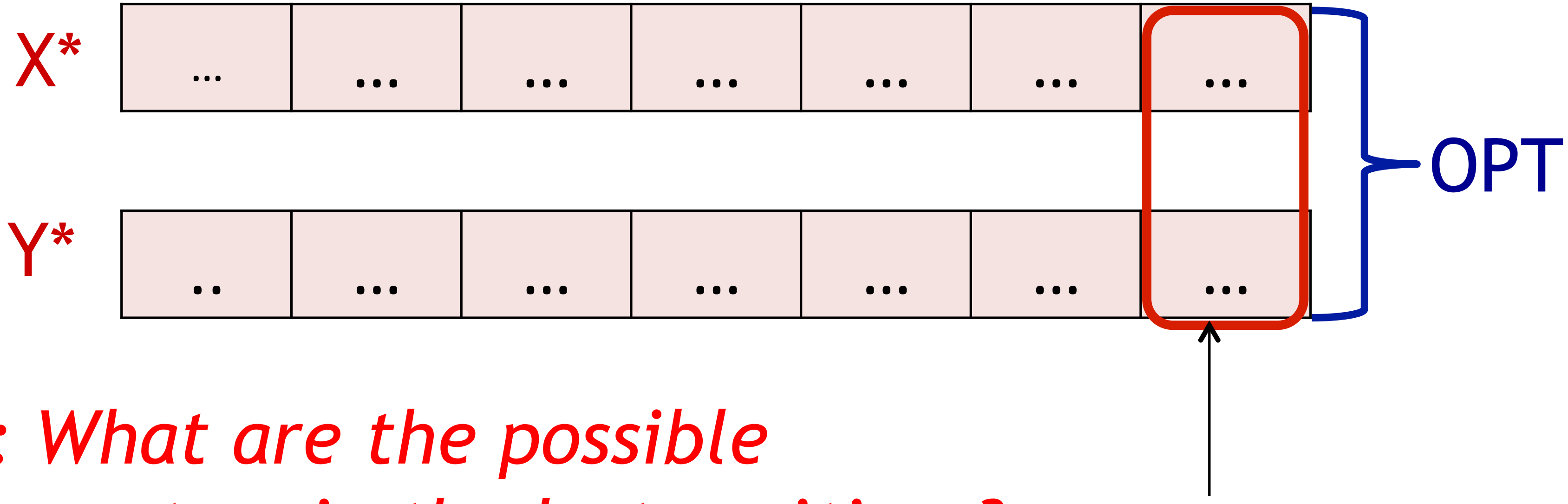
2: quickly + correctly solve “larger” subproblems given solutions to smaller ones

3: After solving all subproblems, can quickly compute final solution

Question: What subproblems should we be thinking of?

Subproblems

Consider the last position in OPT (opt alignment of X,Y).

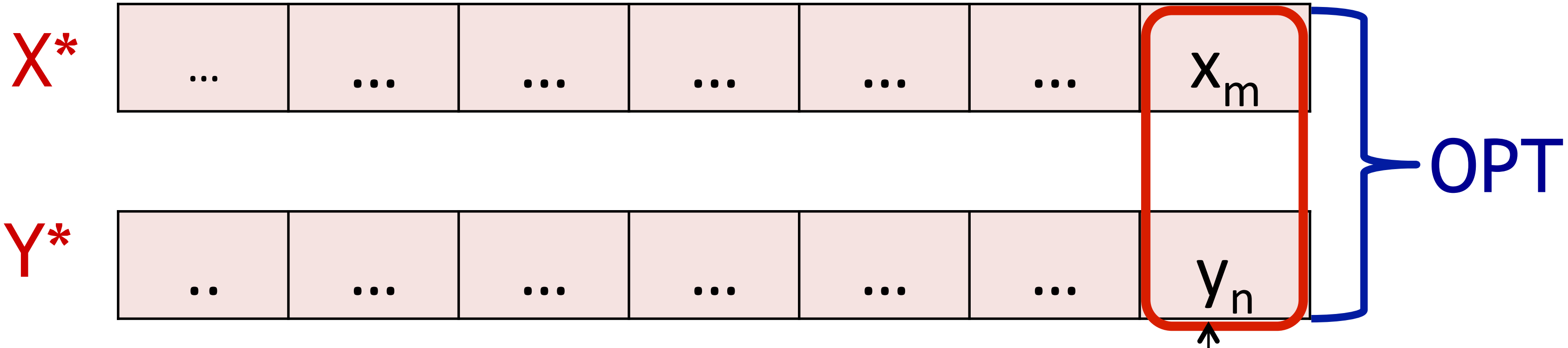


Q: What are the possible characters in the last positions?

last position

Subproblems

Consider the last position in OPT (opt alignment of X,Y).



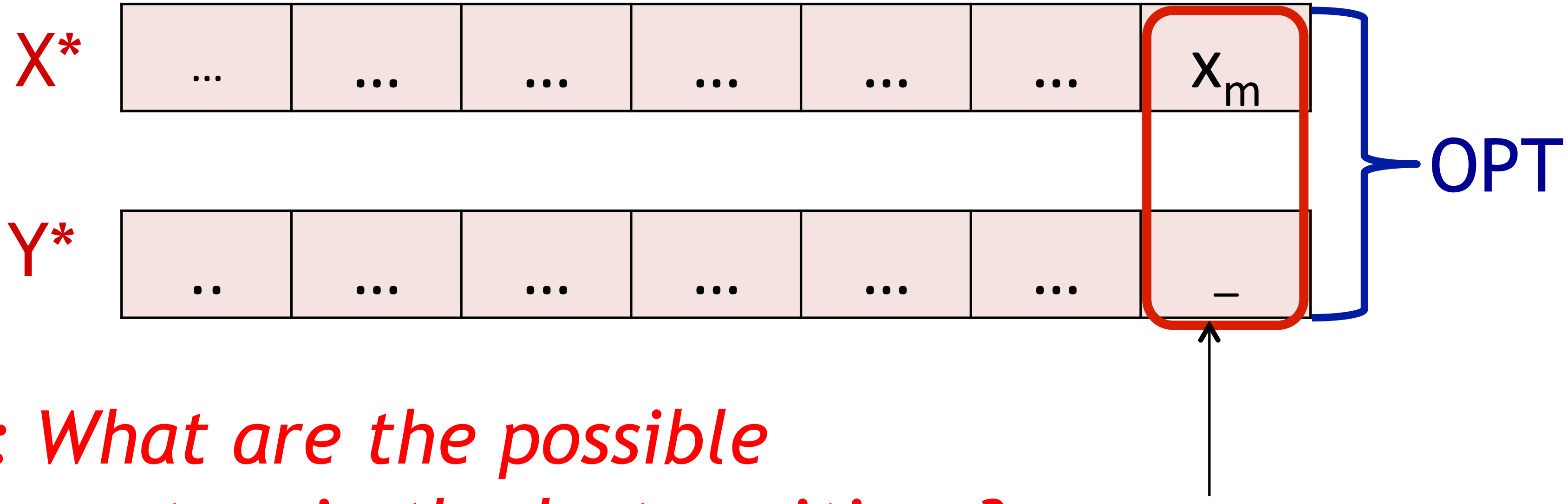
Q: What are the possible characters in the last positions?

last position

Case 1: x_m y_n

Subproblems

Consider the last position in OPT (opt alignment of X,Y).



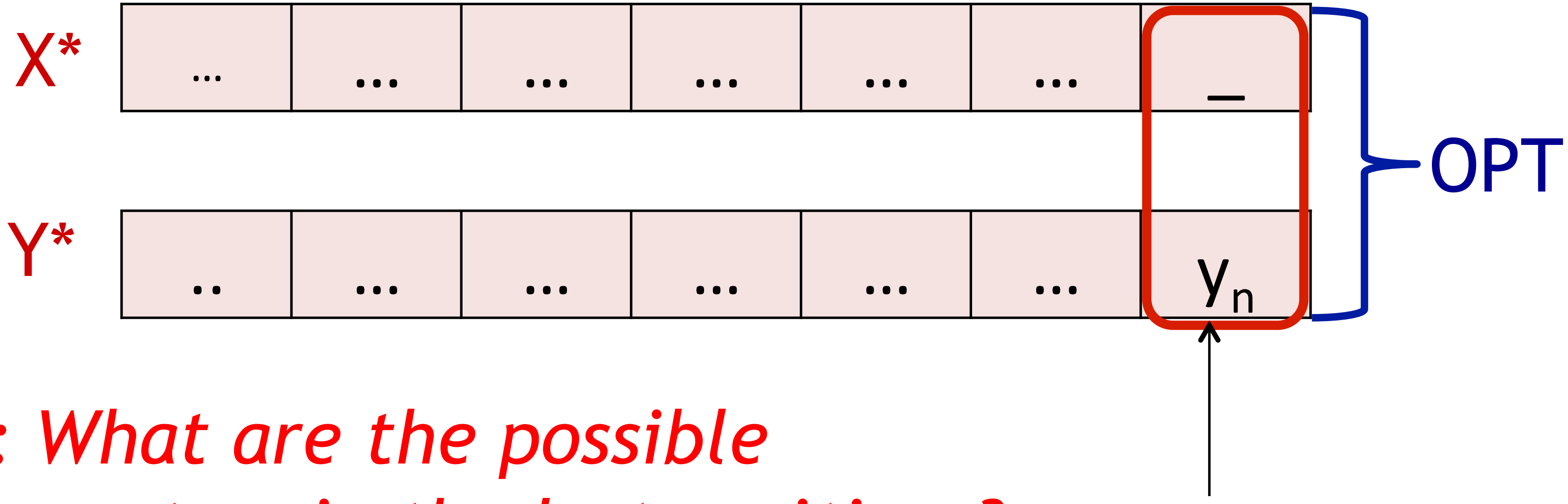
Q: What are the possible characters in the last positions?

Case 1: x_m y_n

Case 2: x_m --

Subproblems

Consider the last position in OPT (opt alignment of X,Y).



X*

… … … … … … _

Y*

.. … … … … … y_n

OPT

Q: What are the possible characters in the last positions?

last position

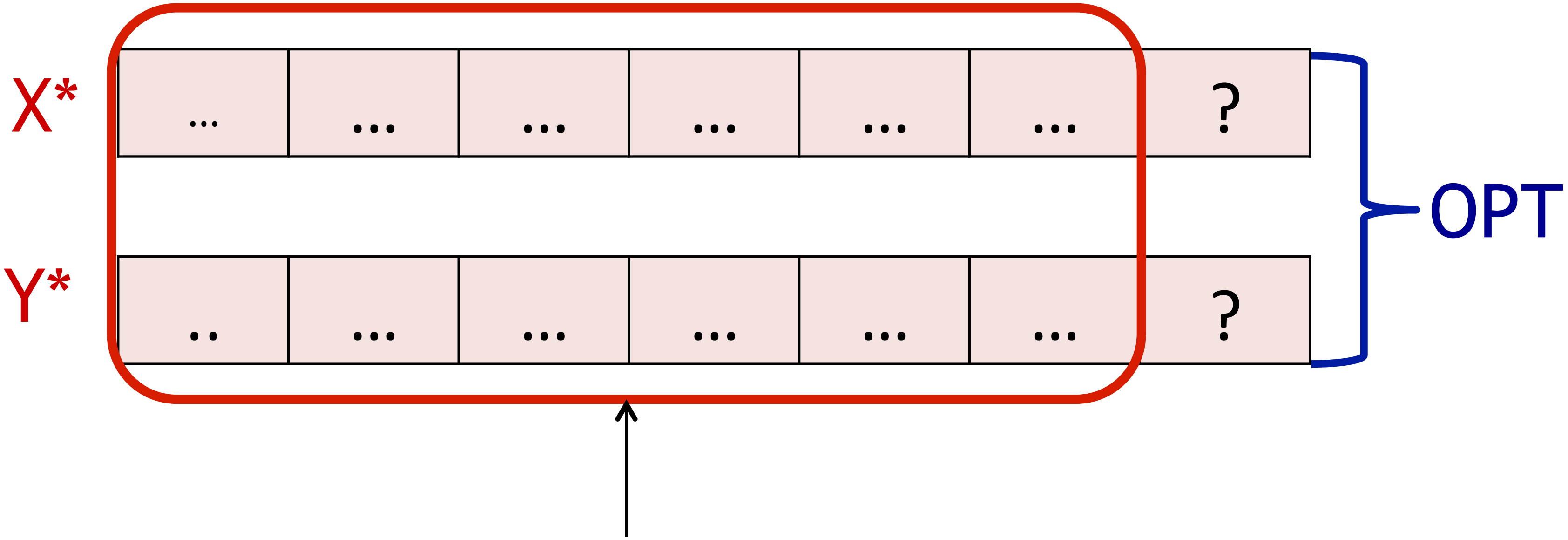
Case 1: x_m y_n

Case 2: x_m --

Case 3: -- y_n

Subproblems

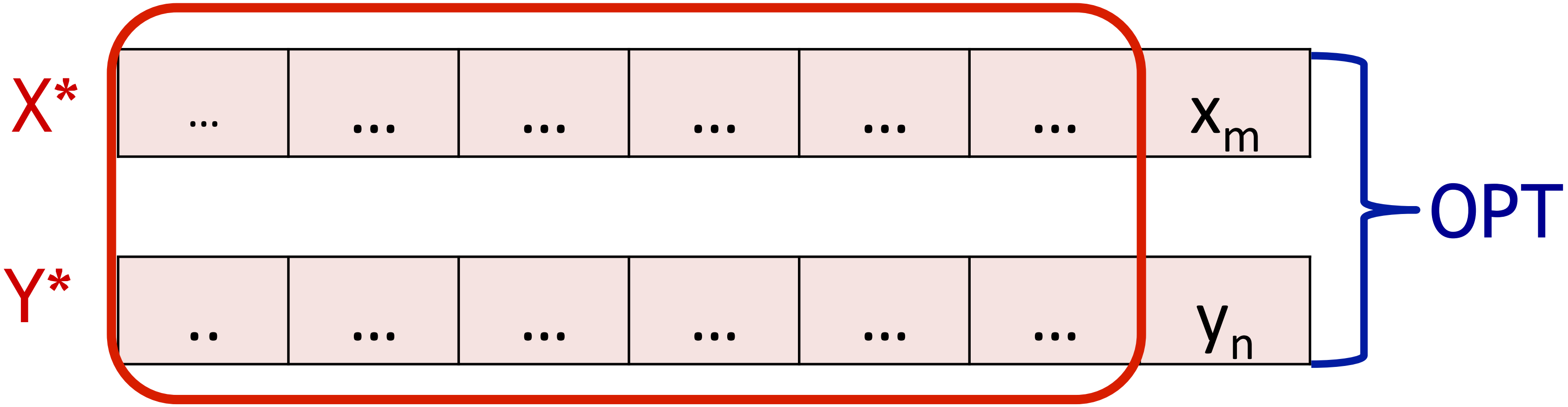
Let X`=X-{$x_m$}, and Y`=Y-{y_n}.



Induced alignment (IA) of two strings. Depending on which case we're at, the strings IA aligns are different.

Subproblems

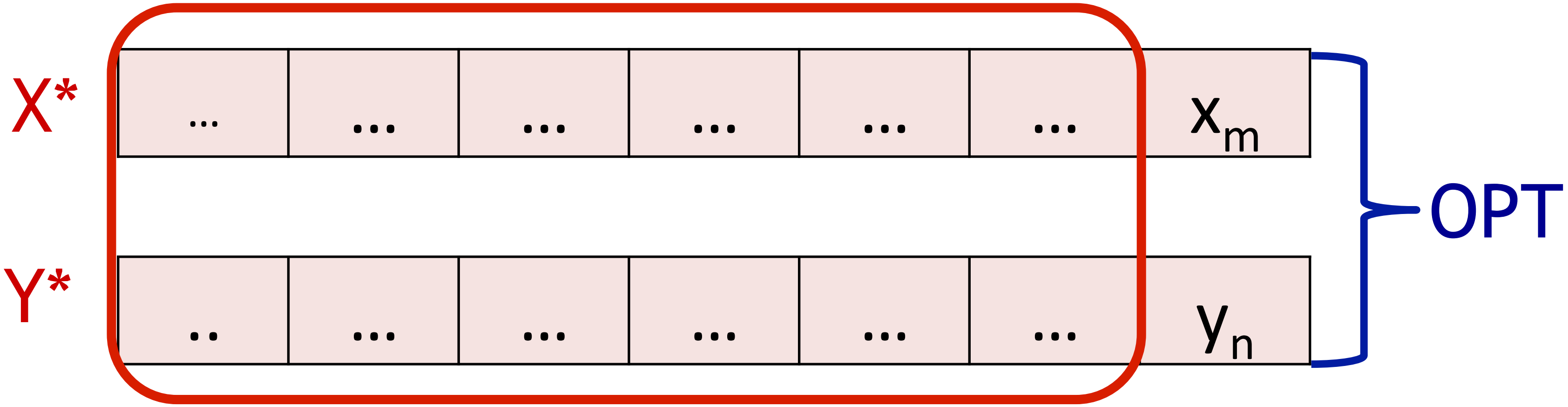
Let X`=X-{$x_m$}, and Y`=Y-{y_n}.



If Case 1: x_m y_n =>

Subproblems

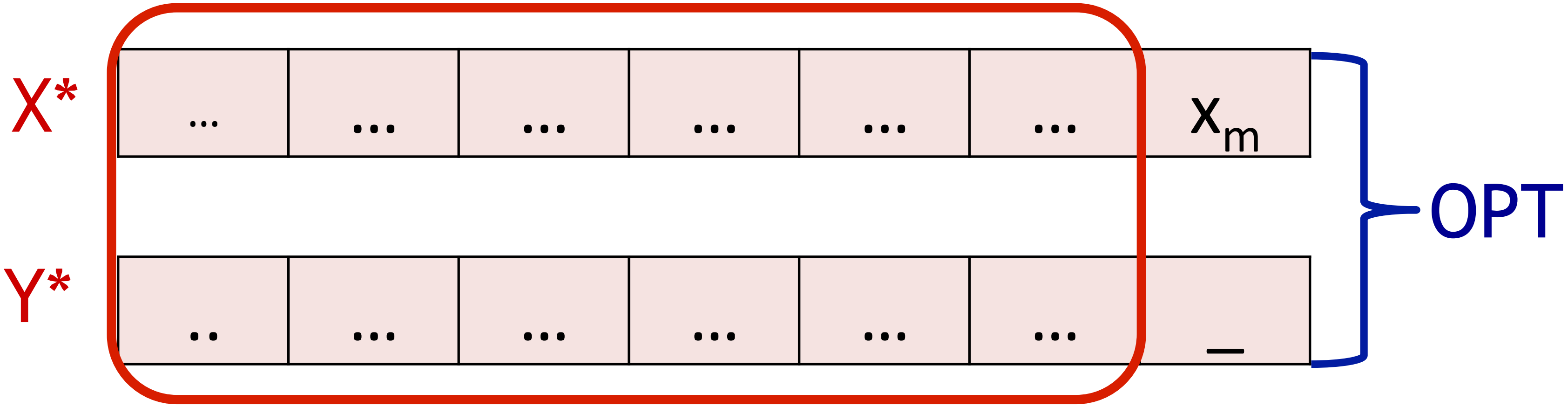
Let X`=X-{$x_m$}, and Y`=Y-{y_n}.



If Case 1: x_m y_n => IA should be optimal X`,Y` alignment

Subproblems

Let X` =X-{$x_m$}, and Y` =Y-{y_n}.

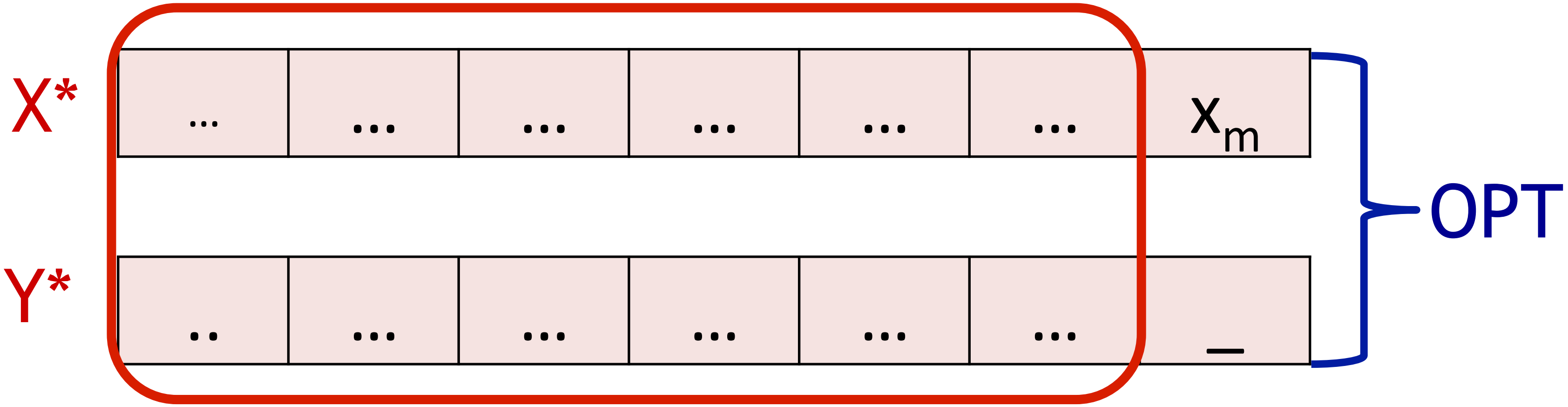


If Case 1: x_m y_n => IA should be optimal X`,Y` alignment

If Case 2: x_m -- =>

Subproblems

Let X`=X-{$x_m$}, and Y`=Y-{y_n}.

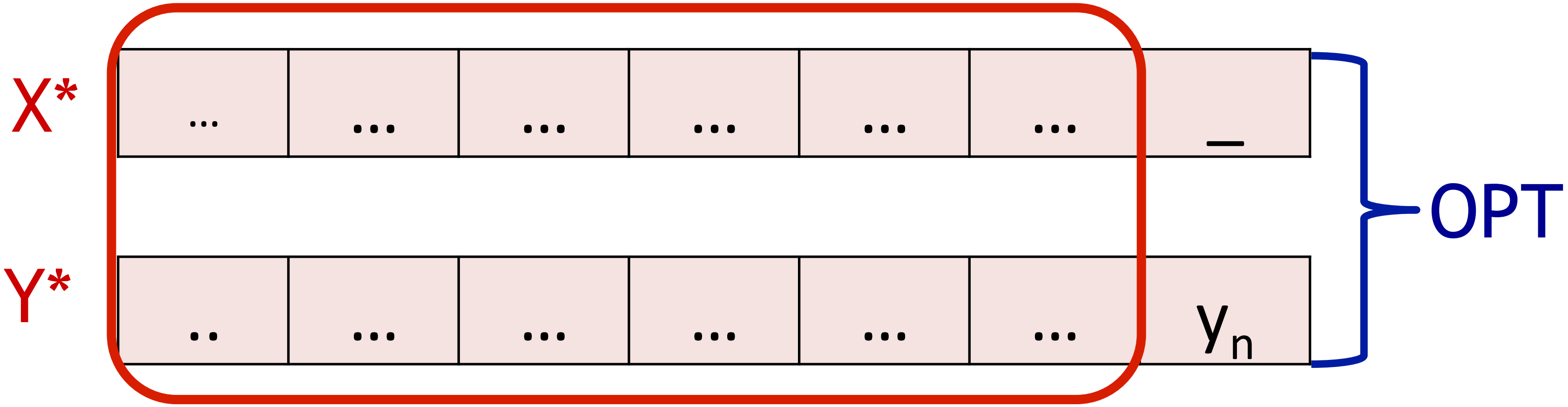


If Case 1: x_m y_n => IA should be optimal X`,Y` alignment

If Case 2: x_m -- => IA should be optimal X`, Y alignment

Subproblems

Let X`=X-{$x_m$}, and Y`=Y-{y_n}.



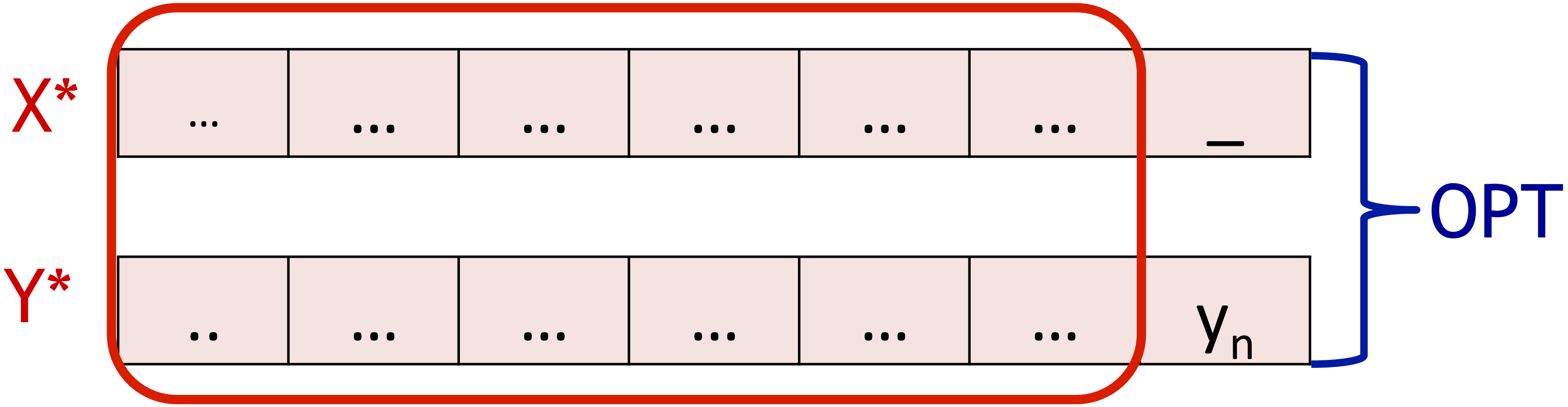
If Case 1: x_m y_n => IA should be optimal X`, Y` alignment

If Case 2: x_m -- => IA should be optimal X`, Y alignment

If Case 3: -- y_n =>

Subproblems

Let X`=X-{$x_m$}, and Y`=Y-{y_n}.



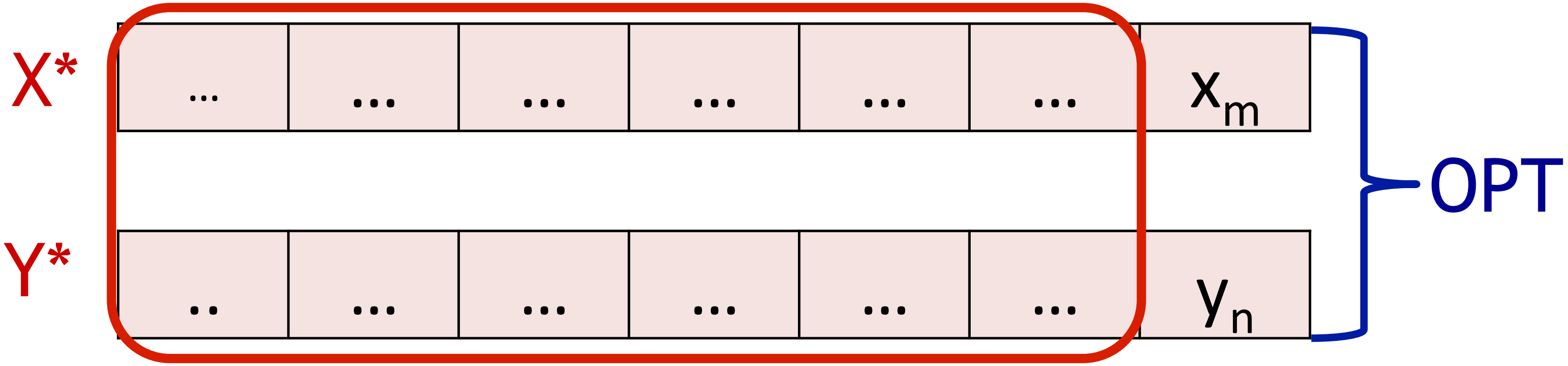
If Case 1: x_m y_n => IA should be optimal X`, Y` alignment

If Case 2: x_m -- => IA should be optimal X`, Y alignment

If Case 3: -- y_n => IA should be optimal X, Y` alignment

Proof of Case 1

Let X`=X-{$x_m$}, and Y`=Y-{y_n}.



X*	…	…	…	…	…	…	x_m	OPT
Y*	..	…	…	…	…	…	y_n	

Claim: If x_m y_n => IA should be optimal X`, Y` alignment

Proof: Assume the contrary that IA is not optimal.

Let penalty(IA) = p. Assume opt X`, Y` alignment is A* with penalty p* < p.

Append x_m and y_n to A*, get an alignment for X, Y with penalty:

$$p^* + \alpha_{\{xm, yn\}} < p + \alpha_{\{xm, yn\}} = \text{penalty(OPT)}$$ Q.E.D

A Possible Recursive Algorithm

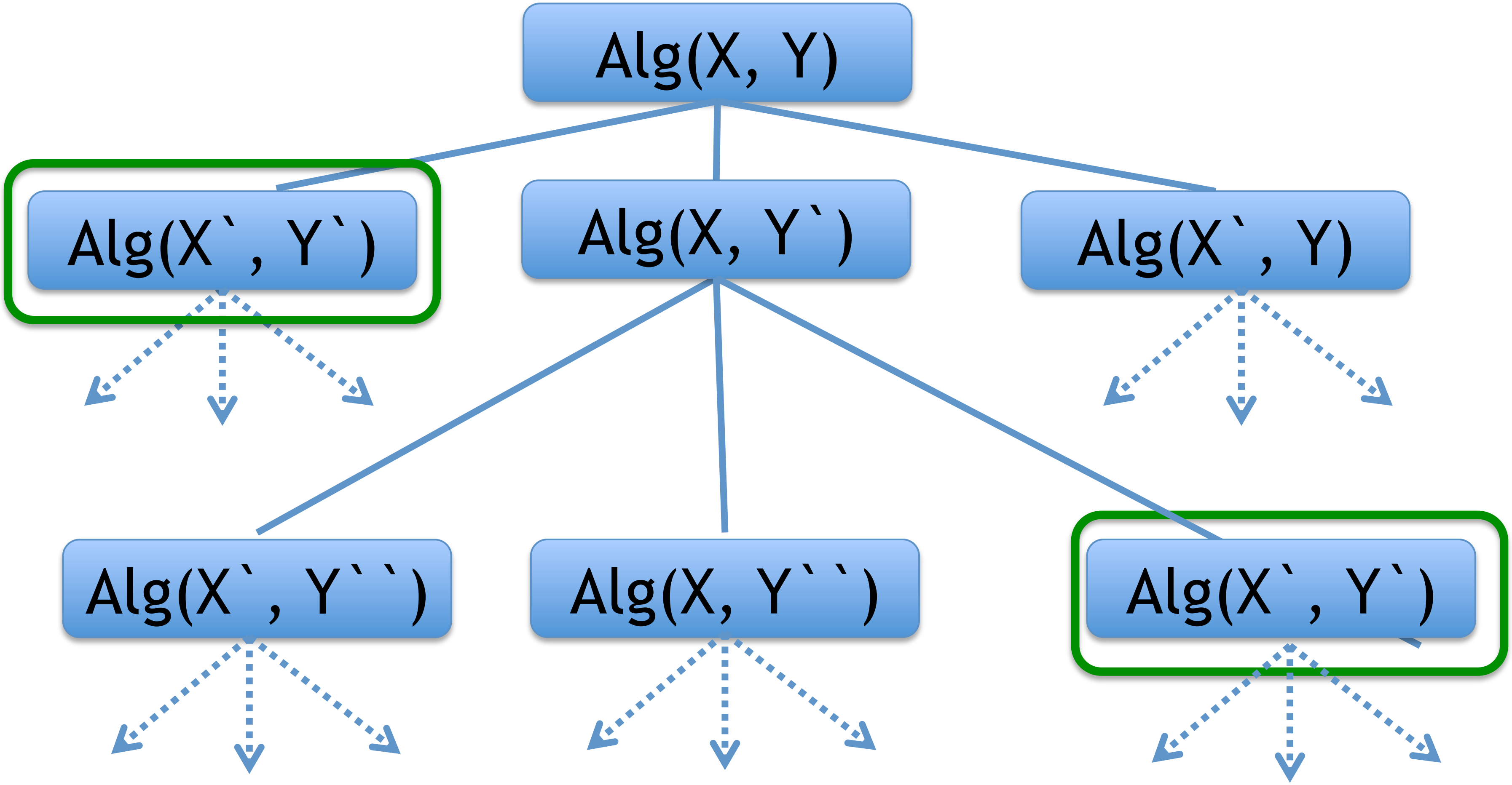
```
Recursive-Alignment-1: (X, Y, $\delta$, $\alpha_{ij}$)
    1. Base Case 1: If |X| = 0 return |Y| * $\delta$;
    2. Base Case 2: If |Y| = 0 return |X| * $\delta$;
    3. Let $A_1$ = Recursive-Alignment-1(X`, Y`)
    4. Let $A_2$ = Recursive-Alignment-1(X`, Y)
    5. Let $A_3$ = Recursive-Alignment-1(X, Y`)
    6. return the min cost of: (1) $A_1 \cup (x_m, y_n)$
                               (2) $A_2 \cup (x_m, \_)$
                               (3) $A_3 \cup (\_, y_n\}$
```

Good news: The algorithm is correct!

$$\min(\mathrm{cost}(A_1) + \alpha(x_m, y_n), \mathrm{cost}(A_2) + \delta, \mathrm{cost}(A_3) + \delta)$$

Problem: brute-force search!

Q: How many distinct recursive calls?



Answer: only mn
b/c each input is a prefix of X, and Y.

Put All Subproblems in An Array And Solve!

Let X_i be the substring x_1, …, x_i (let X_0 = empty string)

Let Y_j be the substring y_1, …, y_j (let Y_0 = empty string)

Let P_{ij} be the penalty of the optimal algnmt for X_i, $Y_{j.}$

Then we showed

$$\forall i,j\ P_{ij} = \min \begin{cases} P_{i-1,\, j-1} + \boxed{\alpha(x_i, y_j)} \\ P_{i-1,\, j} + \delta \\ P_{i,\, j-1} + \delta \end{cases}$$

Note, might be 0 if $x_i = y_j$

Bottom-up Iterative Reformulation

Let A[][] be an mxn matrix

A[i][j] is penalty of optimal X_i and Y_j alignment

```
procedure DP-Alignment(X, Y, δ, α(i,j)):
  Base Cases: A[i][0] =
```

Bottom-up Iterative Reformulation

Let A[][] be an nxm matrix,

A[i][j] is penalty of optimal X_i and Y_j alignment

```
procedure DP-Alignment(X, Y, δ, α(i, j)):
 Base Cases: A[i][0] = iδ
             A[0][j] =
```

Bottom-up Iterative Reformulation

Let A[][] be an mxn matrix,

A[i][j] is penalty of optimal X_i and Y_j alignment

```
procedure DP-Alignment(X, Y, δ, $α_{ij}$):
 Base Cases: A[i][0] = iδ
             A[0][j] = jδ
 for i = 1 … m
    for j = 1 … n
       A[i][j] =
```

Bottom-up Iterative Reformulation

Let A[][] be an m+1xn+1 matrix,

A[i][j] is penalty of optimal X_i and Y_j alignment

```
procedure DP-Alignment(X, Y, δ, $α_{ij}$):
 Base Cases: A[i][0] = iδ
             A[0][j] = jδ
 for i = 1 … m
    for j = 1 … n
        A[i][j] = min { A[i-1][j-1] + α($x_i$, $y_j$)
                        A[i-1][j] + δ,
                        A[i][j-1] + δ}
 return A[m][n]
```

DP Alignment Simulation

X	A	G	G	A	A	T	T

Y	A	G	G	C	T	T

$\delta=1$, $\alpha(i, j)=5$ for $i \neq j$, 0 o.w

A

	0	1	2	3	4	5	6	7
0								
1								
2								
3								
4								
5								
6								

DP Alignment Simulation

X	A	G	G	A	A	T	T

Y	A	G	G	C	T	T

δ=1, α_{ij}=5 for i ≠j, 0 o.w

A

	0	1	2	3	4	5	6	7
0	0	1	2	3	4	5	6	7
1	1							
2	2							
3	3							
4	4							
5	5							
6	6							

DP Alignment Simulation

X	A	G	G	A	A	T	T

Y	A	G	G	C	T	T

$\delta=1$, $\alpha_{ij}=5$ for $i \neq j$, 0 o.w

A

	0	1	2	3	4	5	6	7
0	0	1	2	3	4	5	6	7
1	1	🟥						
2	2							
3	3							
4	4							
5	5							
6	6							

DP Alignment Simulation

X	A	G	G	A	A	T	T

Y	A	G	G	C	T	T

δ=1, α_{ij}=5 for i ≠j, 0 o.w

A

	0	1	2	3	4	5	6	7
0	0	1	2	3	4	5	6	7
1	1	0						
2	2							
3	3							
4	4							
5	5							
6	6							

DP Alignment Simulation

X	A	G	G	A	A	T	T

Y	A	G	G	C	T	T

$\delta=1$, $\alpha_{ij}=5$ for $i \neq j$, 0 o.w

A

	0	1	2	3	4	5	6	7
0	0	1	2	3	4	5	6	7
1	1	0						
2	2	🟥						
3	3							
4	4							
5	5							
6	6							

DP Alignment Simulation

X							
	A	G	G	A	A	T	T

Y						
	A	G	G	C	T	T

δ=1, α_{ij}=5 for i ≠j, 0 o.w

A

	0	1	2	3	4	5	6	7
0	0	1	2	3	4	5	6	7
1	1	0						
2	2	1						
3	3							
4	4							
5	5							
6	6							

DP Alignment Simulation

X	A	G	G	A	A	T	T

Y	A	G	G	C	T	T

δ=1, α_{ij}=5 for i ≠j, 0 o.w

A

	0	1	2	3	4	5	6	7
0	0	1	2	3	4	5	6	7
1	1	0						
2	2	1						
3	3	🟥						
4	4							
5	5							
6	6							

DP Alignment Simulation

X

A	G	G	A	A	T	T

Y

A	G	G	C	T	T

δ=1, α_{ij}=5 for i ≠j, 0 o.w

A

	0	1	2	3	4	5	6	7
0	0	1	2	3	4	5	6	7
1	1	0						
2	2	1						
3	3	2						
4	4							
5	5							
6	6							

DP Alignment Simulation

X	A	G	G	A	A	T	T

Y	A	G	G	C	T	T

$\delta=1$, $\alpha_{ij}=5$ for $i \neq j$, 0 o.w

A

	0	1	2	3	4	5	6	7
0	0	1	2	3	4	5	6	7
1	1	0						
2	2	1						
3	3	2						
4	4	🟥						
5	5							
6	6							

DP Alignment Simulation

X							
	A	G	G	A	A	T	T

Y						
	A	G	G	C	T	T

$\delta=1$, $\alpha_{ij}=5$ for $i \neq j$, 0 o.w

A

	0	1	2	3	4	5	6	7
0	0	1	2	3	4	5	6	7
1	1	0						
2	2	1						
3	3	2						
4	4	3						
5	5							
6	6							

DP Alignment Simulation

X	A	G	G	A	A	T	T

Y	A	G	G	C	T	T

$\delta=1$, $\alpha_{ij}=5$ for $i \neq j$, 0 o.w

A

	0	1	2	3	4	5	6	7
0	0	1	2	3	4	5	6	7
1	1	0	🟥					
2	2	1						
3	3	2						
4	4	3						
5	5	4						
6	6	5						

DP Alignment Simulation

X	A	G	G	A	A	T	T

Y	A	G	G	C	T	T

$\delta=1$, $\alpha_{ij}=5$ for $i \neq j$, 0 o.w

A

	0	1	2	3	4	5	6	7
0	0	1	2	3	4	5	6	7
1	1	0	1					
2	2	1						
3	3	2						
4	4	3						
5	5	4						
6	6	5						

DP Alignment Simulation

X

A	G	G	A	A	T	T

Y

A	G	G	C	T	T

$\delta=1$, $\alpha_{ij}=5$ for $i \neq j$, 0 o.w

A

	0	1	2	3	4	5	6	7
0	0	1	2	3	4	5	6	7
1	1	0	1					
2	2	1	🟥					
3	3	2						
4	4	3						
5	5	4						
6	6	5						

DP Alignment Simulation

X	A	G	G	A	A	T	T

Y	A	G	G	C	T	T

$\delta=1$, $\alpha_{ij}=5$ for i $\neq$ j, 0 o.w

A

	0	1	2	3	4	5	6	7
0	0	1	2	3	4	5	6	7
1	1	0	1					
2	2	1	0					
3	3	2						
4	4	3						
5	5	4						
6	6	5						

DP Alignment Simulation

X							
	A	G	G	A	A	T	T

Y						
	A	G	G	C	T	T

$\delta=1$, $\alpha_{ij}=5$ for $i \neq j$, 0 o.w

A

	0	1	2	3	4	5	6	7
0	0	1	2	3	4	5	6	7
1	1	0	1					
2	2	1	0					
3	3	2	🟥					
4	4	3						
5	5	4						
6	6	5						

DP Alignment Simulation

X	A	G	G	A	A	T	T

Y	A	G	G	C	T	T

δ=1, α_{ij}=5 for i ≠j, 0 o.w

A

	0	1	2	3	4	5	6	7
0	0	1	2	3	4	5	6	7
1	1	0	1					
2	2	1	0					
3	3	2	1					
4	4	3						
5	5	4						
6	6	5						

DP Alignment Simulation

X							
	A	G	G	A	A	T	T

Y						
	A	G	G	C	T	T

$\delta=1$, $\alpha_{ij}=5$ for $i \neq j$, 0 o.w

A

	0	1	2	3	4	5	6	7
0	0	1	2	3	4	5	6	7
1	1	0	1	🟥				
2	2	1	0					
3	3	2	1					
4	4	3	2					
5	5	4	3					
6	6	5	4					

DP Alignment Simulation

X	A	G	G	A	A	T	T

Y	A	G	G	C	T	T

δ=1, α_{ij}=5 for i ≠j, 0 o.w

A	0	1	2	3	4	5	6	7
0	0	1	2	3	4	5	6	7
1	1	0	1	2				
2	2	1	0	1				
3	3	2	1	0				
4	4	3	2					
5	5	4	3					
6	6	5	4					

DP Alignment Simulation

X	A	G	G	A	A	T	T

Y	A	G	G	C	T	T

$\delta=1$, $\alpha_{ij}=5$ for $i \neq j$, 0 o.w

A

	0	1	2	3	4	5	6	7
0	0	1	2	3	4	5	6	7
1	1	0	1	2	3			
2	2	1	0	1	2			
3	3	2	1	0	1			
4	4	3	2	1	🟥			
5	5	4	3	2				
6	6	5	4	3				

DP Alignment Simulation

X	A	G	G	A	A	T	T

Y	A	G	G	C	T	T

$\delta=1$, $\alpha_{ij}=5$ for i ≠j, 0 o.w

A

	0	1	2	3	4	5	6	7
0	0	1	2	3	4	5	6	7
1	1	0	1	2	3			
2	2	1	0	1	2			
3	3	2	1	0	1			
4	4	3	2	1	2			
5	5	4	3	2				
6	6	5	4	3				

DP Alignment Simulation

X	A	G	G	A	A	T	T

Y	A	G	G	C	T	T

$\delta=1$, $\alpha_{ij}=5$ for $i \neq j$, 0 o.w

A

	0	1	2	3	4	5	6	7
0	0	1	2	3	4	5	6	7
1	1	0	1	2	3	4	5	6
2	2	1	0	1	2	3	4	5
3	3	2	1	0	1	2	3	4
4	4	3	2	1	2	3	4	5
5	5	4	3	2	3	4	3	4
6	6	5	4	3	4	5	4	3

Correctness

Claim: At termination A[m][n] = P_{mn} = penalty of opt X, Y alignment or the NW score of X and Y

Proof: By induction on i, and j. (exercise)

```
procedure DP-Alignment(X, Y, $\delta$, $\alpha_{ij}$):
 Base Cases: A[i][0] = i$\delta$
             A[0][j] = j$\delta$
 for i = 1 … m
    for j = 1 … n
       A[i][j] = min { A[i-1][j-1] + $\alpha_{ij}$,
                       A[i-1][j] + $\delta$
                       A[i][j-1] + $\delta$}
```

Runtime

O(mn)

b/c O(1) cost for each mn loop iterations!

```
procedure DP-Alignment(X, Y, $\delta$, $\alpha_{ij}$):
 Base Cases: A[i][0] = i$\delta$
             A[0][j] = j$\delta$
 for i = 1 … m
    for j = 1 … n
       A[i][j] = min { A[i-1][j-1] + $\alpha_{ij}$,
                       A[i-1][j] + $\delta$
                       A[i][j-1] + $\delta$}
```

How to Reconstruct Actual Alignment?

Same trick of tracing back in the array A.

```
procedure DP-Alignmt-Reconst(X, Y, $\delta$, $\alpha_{ij}$):
  let i = m, j = n
  let B be an empty linked list
  while (i > 0 && j > 0):
      if (A[i][j] was filled by case 1):
        B.append(<$x_i$, $y_j$>) i--, j--;
      else if(A[i][j] was filled by case 2):
        B.append(<$x_i$, _>) i--;
      else if(A[i][j] was filled by case 3):
        B.append(<_, $y_j$>) j--;
```

DP Alignment Reconstruction

$\delta=1$, $\alpha_{ij}=5$ for $i \neq j$, 0 o.w

A

	0	1	2	3	4	5	6	7
0	0	1	2	3	4	5	6	7
1	1	0	1	2	3	4	5	6
2	2	1	0	1	2	3	4	5
3	3	2	1	0	1	2	3	4
4	4	3	2	1	2	3	4	5
5	5	4	3	2	3	4	3	4
6	6	5	4	3	4	5	4	3

DP Alignment Reconstruction

δ=1, α_{ij}=5 for i ≠j, 0 o.w

A

	0	1	2	3	4	5	6	7
0	0	1	2	3	4	5	6	7
1	1	0	1	2	3	4	5	6
2	2	1	0	1	2	3	4	5
3	3	2	1	0	1	2	3	4
4	4	3	2	1	2	3	4	5
5	5	4	3	2	3	4	3	4
6	6	5	4	3	4	5	4	3

Q: Which case was A[7][6]?

A: Case 1

Output:

T

T

DP Alignment Reconstruction

$\delta=1$, $\alpha_{ij}=5$ for $i \neq j$, 0 o.w

A

	0	1	2	3	4	5	6	7
0	0	1	2	3	4	5	6	7
1	1	0	1	2	3	4	5	6
2	2	1	0	1	2	3	4	5
3	3	2	1	0	1	2	3	4
4	4	3	2	1	2	3	4	5
5	5	4	3	2	3	4	3	4
6	6	5	4	3	4	5	4	3

Q: Which case was A[6][5]?

A: Case 1

Output:

T

T

DP Alignment Reconstruction

$\delta=1$, $\alpha_{ij}=5$ for $i \neq j$, 0 o.w

A

	0	1	2	3	4	5	6	7
0	0	1	2	3	4	5	6	7
1	1	0	1	2	3	4	5	6
2	2	1	0	1	2	3	4	5
3	3	2	1	0	1	2	3	4
4	4	3	2	1	2	3	4	5
5	5	4	3	2	3	4	3	4
6	6	5	4	3	4	5	4	3

Q: Which case was A[6][5]?

A: Case 1

Output:

T T

T T

DP Alignment Reconstruction

$\delta=1$, $\alpha_{ij}=5$ for i ≠j, 0 o.w

A

	0	1	2	3	4	5	6	7
0	0	1	2	3	4	5	6	7
1	1	0	1	2	3	4	5	6
2	2	1	0	1	2	3	4	5
3	3	2	1	0	1	2	3	4
4	4	3	2	1	2	3	4	5
5	5	4	3	2	3	4	3	4
6	6	5	4	3	4	5	4	3

Q: Which case was A[5][4]?

A: Case 2 or 3 (say 2)

Output:

T T

T T

DP Alignment Reconstruction

$\delta=1$, $\alpha_{ij}=5$ for i ≠j, 0 o.w

A

	0	1	2	3	4	5	6	7
0	0	1	2	3	4	5	6	7
1	1	0	1	2	3	4	5	6
2	2	1	0	1	2	3	4	5
3	3	2	1	0	1	2	3	4
4	4	3	2	1	2	3	4	5
5	5	4	3	2	3	4	3	4
6	6	5	4	3	4	5	4	3

Q: Which case was A[5][4]?

A: Case 2 or 3 (say 2)

Output:

A T T

- T T

DP Alignment Reconstruction

$\delta=1$, $\alpha_{ij}=5$ for $i \neq j$, 0 o.w

A

	0	1	2	3	4	5	6	7
0	0	1	2	3	4	5	6	7
1	1	0	1	2	3	4	5	6
2	2	1	0	1	2	3	4	5
3	3	2	1	0	1	2	3	4
4	4	3	2	1	2	3	4	5
5	5	4	3	2	3	4	3	4
6	6	5	4	3	4	5	4	3

Q: Which case was A[4][4]?

A: Case 2 or 3 (say 2)

Output:

A T T

- T T

DP Alignment Reconstruction

δ=1, α_{ij}=5 for i ≠j, 0 o.w

A

	0	1	2	3	4	5	6	7
0	0	1	2	3	4	5	6	7
1	1	0	1	2	3	4	5	6
2	2	1	0	1	2	3	4	5
3	3	2	1	0	1	2	3	4
4	4	3	2	1	2	3	4	5
5	5	4	3	2	3	4	3	4
6	6	5	4	3	4	5	4	3

Q: Which case was A[4][4]?

A: Case 2 or 3 (say 2)

Output:

A A T T

- - T T

DP Alignment Reconstruction

$\delta=1$, $\alpha_{ij}=5$ for i ≠j, 0 o.w

A

	0	1	2	3	4	5	6	7
0	0	1	2	3	4	5	6	7
1	1	0	1	2	3	4	5	6
2	2	1	0	1	2	3	4	5
3	3	2	1	0	1	2	3	4
4	4	3	2	1	2	3	4	5
5	5	4	3	2	3	4	3	4
6	6	5	4	3	4	5	4	3

Final Output

AGG-AATT

AGGC--TT