Sequence Alignment

https://cs.pomona.edu/classes/cs140/

Change Return Possibilities

How many ways can you return amount A using n kinds of coins?

All the ways returning amount A using all but the first kinds of coins (using the other (n – 1) kinds of coins)

+

All the ways returning amount (A – d) using n kinds of coins, where d is the denomination for the first kind of coin

Does this seem like a “hard” problem?

Outline

Topics and Learning Objectives

- Discuss the dynamic programming paradigm
- Investigate the sequence alignment problem

Assessments

- None

Sequence Alignment

- Compute the similarity between two strings.
- For example, using the Needleman-Wunsch Similarity Score

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

- Total penalty (dissimilarity) = $p_{gap} + p_{AT}$
- Assume these penalties are based on biological principles

Sequence Alignment

Input:

- Two strings X = x_1, …, x_m; and Y = y_1, …, y_n; over the alphabet Σ
  - For example, Σ = {A, C, G, T} for genomes
- Also given a penalty value for each possible error
  - For example, p_{gap} , p_{AC} , p_{AG} , p_{AT} , p_{CG} , p_{CT} , p_{GT}

Output:

- Out of all possible alignments, output the one that minimizes penalties

Sequence Alignment

Input:

- Two strings X = x_1, …, x_m; and Y = y_1, …, y_n; over the alphabet Σ
  - For example, Σ = {A, C, G, T} for genomes
- Also given a penalty value for each possible error
  - For example, p_{gap} , p_{AC} , p_{AG} , p_{AT} , p_{CG} , p_{CT} , p_{GT}

All pairwise combinations

$O(2^n)$

Output:

- Out of all possible alignments, output the one that minimizes penalties

How many possible alignments exist?

Example

Assume a penalty of

- 1 for each gap and
- 2 for a mismatch between symbols

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

What is the minimum penalty for these two strings?

Example

Assume a penalty of

- 1 for each gap and
- 2 for a mismatch between symbols

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

We'll say that these sequences have a common length of L

What is the minimum penalty for these two strings?

- Minimum penalty: 4

Optimal Substructure

- Let's zoom in on the last column of the alignment

X has m values

Y has n values

X=	A	G	G	G	C	…	X_m?
Y=	A	G	G	--	C	…	Y_n?

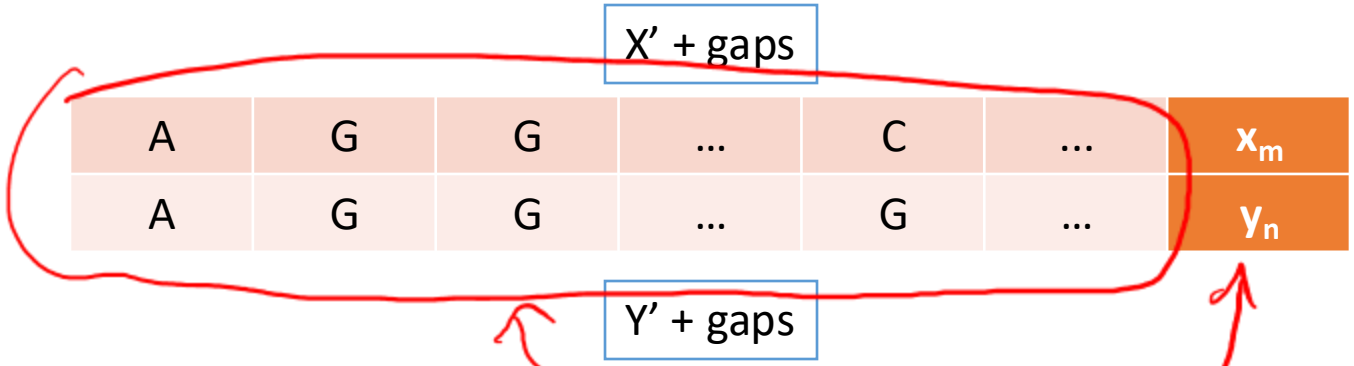
- How many possibilities are there for the contents of the final column of an *optimal* alignment?
  - Case 1: x_m and y_n
  - Case 2: x_m and gap (handles case where y_n is matched with something else)
  - Case 3: gap and y_n (handles case where x_m is matched with something else)

$X_m = G$
$Y_n = A$
$P_{AG} < P_{gap}$

What would you say about optimal substructure assuming we can find the lowest penalty for these three cases?

Case 1: x_m and y_n (no gap at the end)

- Let P denote the final alignment penalty after matching x_m and y_n
- Let X' and Y' denote the sequences without x_m and y_n



- Then the total penalty is: $P = P_{first} + P_{end}$
- We then want P_{first} to be optimal with respect to X' and Y'

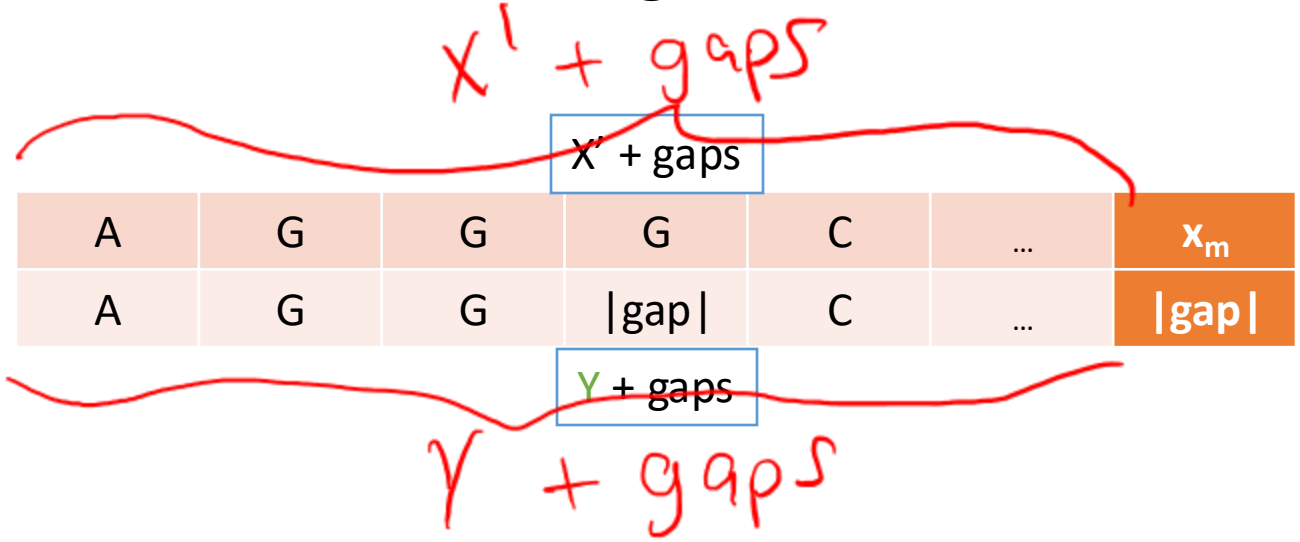
Optimal Substructure

Case 2: x_m and gap

m + n symbols

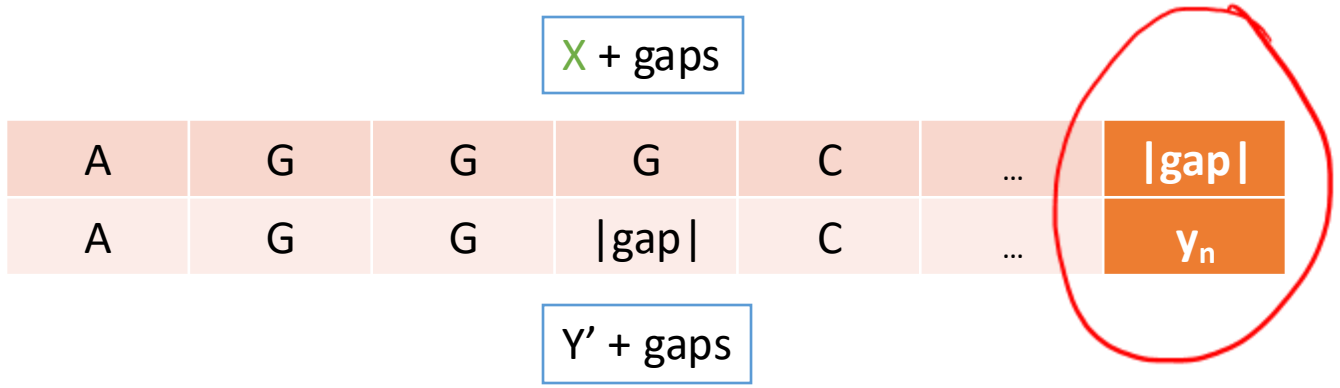
(m-1) + n symbols

- In this case we match x_m with a gap
- We've removed one symbol from X (call it X')
- But we still have the entire Y string



Case 3: gap and y_n

- In this case we match y_n with a gap
- We've removed one symbol from Y (call it Y')
- But we still have the entire X string



Optimal Substructure

An optimal alignment of two strings X and Y is one of

1. An optimal alignment of X' and Y' with x_m and y_n at the end
2. An optimal alignment of X' and Y with x_m and a gap at the end
3. An optimal alignment of X and Y' with a gap and y_n at the end

What if one of X' or Y' is empty at this stage?

What is the Recurrence?

Let us introduce some more formal notation

$P_{m,n} =$

Capital 'P' denotes the total penalty up to the ith and jth items in X and Y.

$$P_{i,j} = P_{i-1,j-1} + P_{x_i, y_j}$$

Lowercase p denotes a specific type of penalty for a mismatch

Recurrence

Top-Down : recursive

Bottom-Up :

$x_i + y_j$

$$P_{i,j} = \min \begin{cases} P_{i-1,j-1} + p_{x_i,y_j} \\ P_{i-1,j} + p_{gap} \\ P_{i,j-1} + p_{gap} \end{cases}$$

This happens for many (all) possible alignments

Case 3

Code and Running Time

$P_{m,n}$

A good practice problem

$$P_{i,j} = \min \begin{cases} P_{i-1,j-1} + p_{x_i,y_j} \\ P_{i-1,j} + p_{gap} \\ P_{i,j-1} + p_{gap} \end{cases}$$

Things to consider

- What size is the dynamic programming table? $(m, n) \rightarrow (m+1, n+1)$
- What are the base cases?
- How do we initialize the table?
- How many loops do we need?
- What is the running time? $O(n \cdot m)$

Proof

A good practice problem

$$P_{i,j} = \min \begin{cases} P_{i-1,j-1} + p_{x_i,y_j} \\ P_{i-1,j} + p_{gap} \\ P_{i,j-1} + p_{gap} \end{cases}$$

Things to consider

- What kind of proof seems natural?
- What are the base cases?
- What is our inductive hypothesis?
- What reasoning do we need for the inductive step?

```
FUNCTION ReconstructSequence(penalties, X, Y)
    i = penalties.x_length - 1
    j = penalties.y_length - 1
    alignedX = ""
    alignedY = ""
    WHILE i > 0 && j > 0
        MATCH penalties[i][j]
            IF case 1
                alignedX += X[i]; i -= 1
                alignedY += Y[j]; j -= 1
            IF case 2
                alignedX += X[i]; i -= 1
                alignedY += "gap"
            IF case 3
                alignedX += "gap"
                alignedY += Y[j]; j -= 1
    fillAsNeeded(X, alignedX, Y, alignedY)
```

{1, 5, 6}

# of coins to return that much in change