

String Alignment

Definition: An Alignment of two strings $x = x_1 \dots x_n$ and $y = y_1 \dots y_m$ is obtained by first inserting spaces, and then placing the two resulting strings one above the other so that every character or space has a counterpart in the other string.

Example: string ACBCDDDB and CADBDAD. One possibility:

```
A C - - B C D D D B
  |         |   |   |
- C A D B - D A D -
```

another one

```
- A C B C D D D B
  |   |   |
C A D B D A D - -
```

Optimal Alignment

Given: two strings x and y over alphabet \mathcal{A} .

In order to formalize **optimality of an alignment**, one has to define

- the costs for substituting each letter by one other letter (possibly the same letter) \Rightarrow **substitution matrix**;
- the costs for insertion/deletion \Rightarrow **gap penalties**.

Example: Levenshtein's distance:

Match count : 0

Mismatch count : +1

Insertion/Deletion count : +1

Optimal Alignment (formal)

Theorem: Let $X = (x_1, x_2, \dots, x_m)$ and $Y = (y_1, y_2, \dots, y_n)$ be sequences. Let $M_{i,j}$ denote the minimum penalty of aligning X and Y . The optimal alignment is found as follows:

Base conditions: $M(i, 0) = i$, $M(0, j) = j$

Recurrence: for $1 \leq i \leq n$, $1 \leq j \leq m$:

$$M(i, j) = \min \begin{cases} M(i-1, j-1) + \delta_{x_i, y_j} \\ M(i-1, j) + 1 \\ M(i, j-1) + 1 \end{cases}$$

Optimal Alignment: proof

Base condition: Only way to align first i elements of sequence x with 0 elements of y is to align each element with a space (gap) in $y \Rightarrow$ penalty for one element is $+1 \Rightarrow$ total score is $M(i, 0) = \sum_{k=1}^i 1 = i$.

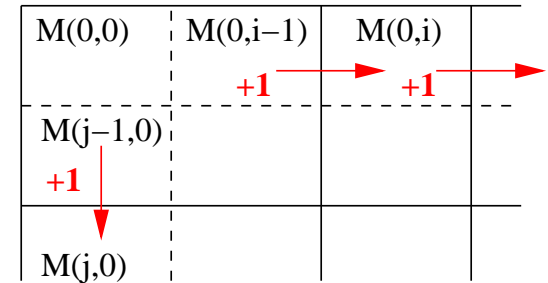
Recurrence: three cases:

- **Aligning x_i with y_j** \Rightarrow total costs are $c(x_i, y_j) = \delta_{x_i, y_j}$ plus costs of aligning $i - 1$ elements of x with $j - 1$ elements of y : $M(i - 1, j - 1) + 1$.
- **Aligning x_i with space in y :** $c(x_i, -) = 1$ plus costs of aligning $i - 1$ elements of x with j elements of y : $M(i - 1, j) + 1$.
- **Aligning y_j with space in x :** $M(i, j - 1) + 1$ by analogy.

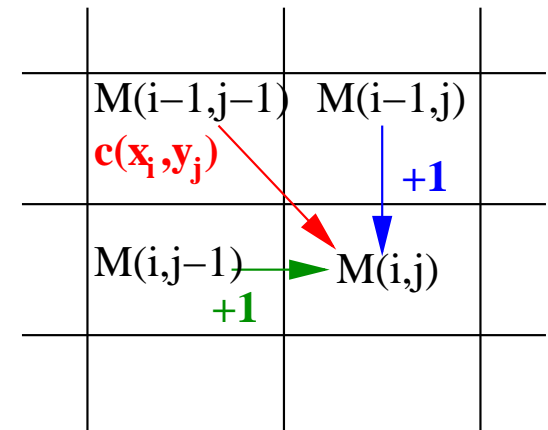
Tabular Computation of Optimal Alignment

Set $M(0,0) = 0$. Systematically fill matrix:

for $i = 0$ or $j = 0$, calculate new value from left-hand (upper) value.



for $i, j \geq 1$, calculate the bottom right-hand corner of each square of 4 cells from one of the 3 other cells:



Keep **traceback pointer** to cell from which it was derived.

Example

| | | B | I | O | L | O | G | I | S | C | H | E | M | E | D | I | Z | I | N |
|---|----|----------|----------|----------|----------|----------|----------|----------|----------|----------|----------|----------|----------|----------|----------|----------|----------|----------|----------|
| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 |
| B | 1 | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 |
| I | 2 | 1 | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 |
| O | 3 | 2 | 1 | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
| L | 4 | 3 | 2 | 1 | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 |
| O | 5 | 4 | 3 | 2 | 1 | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 |
| G | 6 | 5 | 4 | 3 | 2 | 1 | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 |
| I | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 |
| C | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 1 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
| A | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 2 | 2 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
| L | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 3 | 3 | 3 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
| | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 4 | 4 | 4 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 |
| M | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 5 | 5 | 5 | 5 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
| E | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 6 | 6 | 6 | 6 | 5 | 4 | 5 | 6 | 7 | 8 | 9 |
| D | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 7 | 7 | 7 | 7 | 6 | 5 | 4 | 5 | 6 | 7 | 8 |
| I | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 8 | 8 | 8 | 8 | 7 | 6 | 5 | 4 | 5 | 6 | 7 |
| C | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 9 | 9 | 9 | 9 | 8 | 7 | 6 | 5 | 5 | 6 | 7 |
| I | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 10 | 10 | 10 | 10 | 9 | 8 | 7 | 6 | 6 | 5 | 6 |
| N | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 11 | 11 | 11 | 11 | 10 | 9 | 8 | 7 | 7 | 6 | 5 |
| E | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 12 | 12 | 12 | 12 | 11 | 10 | 9 | 8 | 8 | 7 | 6 |

The traceback procedure

- The value in the lower right corner $M(n, m)$ is by definition the best score for an alignment of $x_{1\dots n}$ to $y_{1\dots m}$
- To find the alignment itself: find path back to $M(0, 0)$ by following the traceback pointers.
- In each traceback step, add a pair of symbols onto the current front of the alignment: $\swarrow: (x_i, y_j)$, $\uparrow: (-, y_j)$, $\leftarrow: (x_i, -)$
- Traceback finds just one optimal alignment. If at any point two of the derivations are equal, an arbitrary choice is made between equal options.

Time and Space Complexity

Satz 1 *Time complexity of optimal alignment is $O(nm)$. Space complexity is $O(n)$, if only $M(x, y)$ is required, and $O(nm)$ for the reconstruction of the alignment.*

Proof Time complexity: when computing value of the cell (i, j) , only cells $(i - 1, j - 1)$, $(i, j - 1)$, $(i - 1, j)$ are examined \Rightarrow constant time. There are $(n + 1)(m + 1)$ cells $\Rightarrow O(nm)$ time complexity.

Space complexity: row-wise computation of the matrix: for computing row k , only row $k - 1$ must be stored $\Rightarrow O(n)$ space. For reconstructing the alignment, all traceback pointers must be stored $\Rightarrow O(nm)$ space.

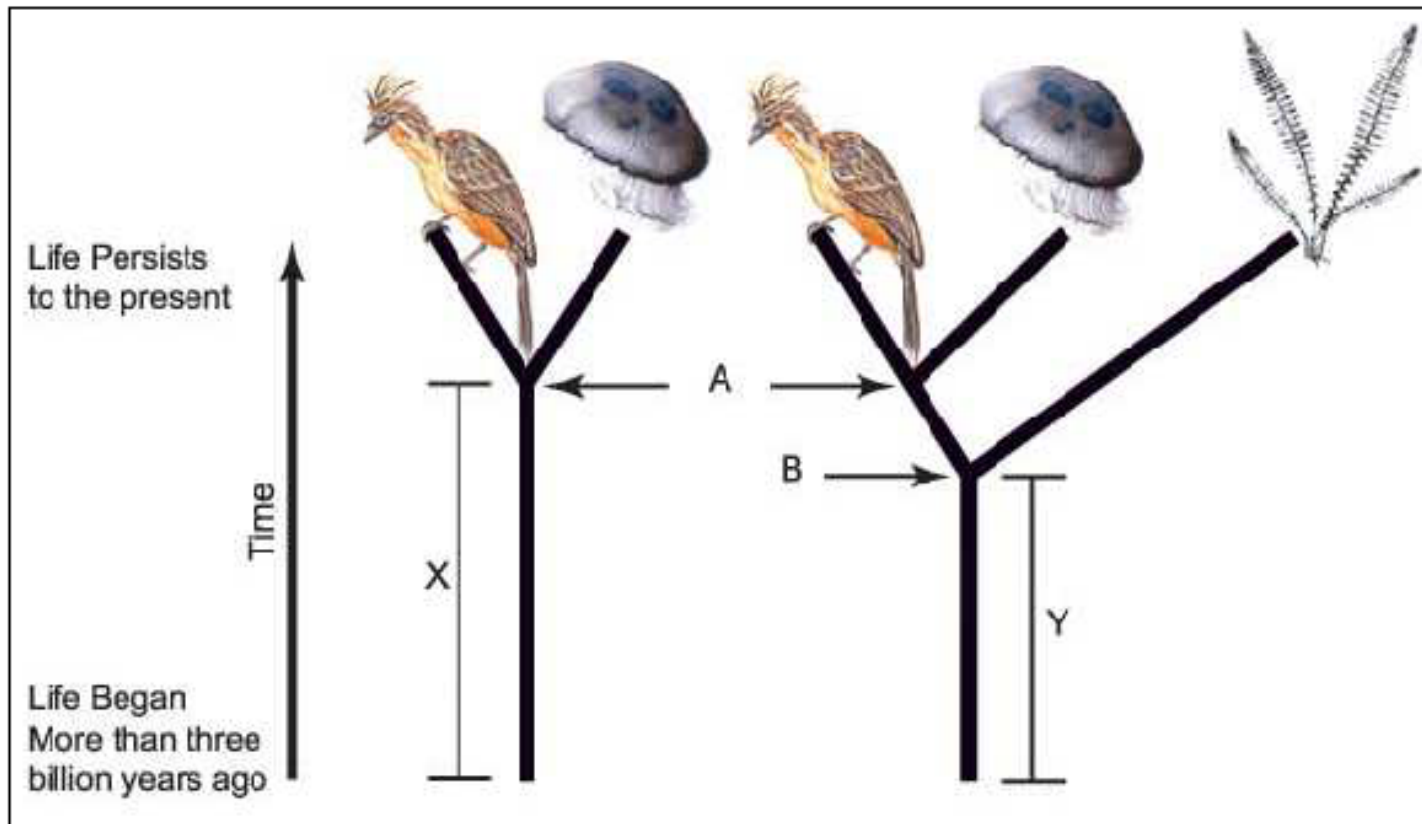
Global Alignment in Linear Space

- Problem: genomic scale sequence analysis: comparing two large genomic sequences: $m, n \approx 10^6 \Rightarrow$ space complexity 10^{12} is clearly unacceptable!
- **Solution:** there exist algorithms with space complexity $O(m + n)$.
- Basic idea: **divide and conquer**. Let $u = \lfloor \frac{n}{2} \rfloor$
 - Assume $\exists v$: the cell (u, v) is on the optimal alignment.
 - Split dynamic programming problem into two parts: (i) $(0, 0) \rightarrow (u, v)$ and $(u, v) \rightarrow (n, m)$. Optimal alignment will be concatenation of individual sub-alignments.
 - Repeat splitting until until $u = 0 \leftarrow$ trivial, region already completely specified.

What is Phylogenetics?

- Phylogenetics: Find the *genetic relationships* between species.
- **Basic idea:** compare specific features of the species. Assumption: similar species are genetically close.
- The term phylogeny refers to these relationships, usually presented as a phylogenetic tree.
- Classic phylogenetics: physical or morphological features - size, color, ...
- Modern phylogeny uses information extracted from genetic material - mainly DNA and protein sequences.
- Features (characters): DNA or protein sites within conserved blocks of multiple alignments.

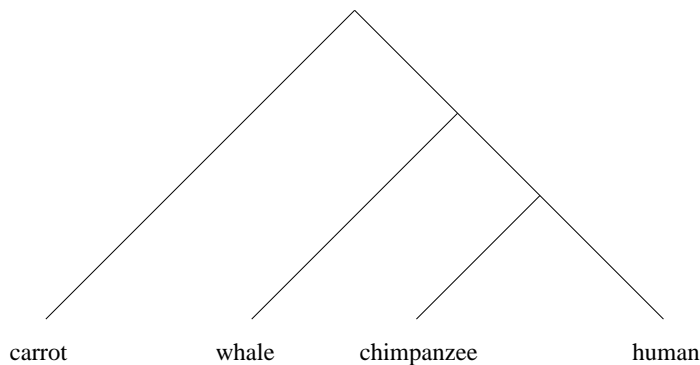
What is Phylogenetics?



- A: most recent common ancestor of bird and jellyfish
- X: portion of history shared by bird and jellyfish

The Problem of Phylogenetic Tree Construction

Problem: Find tree which best describes the relationship between a set of objects.



Cladistics: systematic classification of groups of organisms on the basis of shared characteristics being derived from a common ancestor.

Assumptions:

- Any group of organisms are related by descent from a common ancestor.
- There is a bifurcation (binary) pattern of cladogenesis.
- Changes in characteristics occur in lineages over time.

Approaches To Phylogenetic Tree Construction

Distance based methods:

require definition of a *distance function between objects*. Construct tree so that the pairwise distances can be mapped over the tree *as accurately as possible*.

Character based methods:

character or *trait* = a discrete property of an object. E.g.

- “mammal” (all are either mammals or not)
- “unicellular” (all are either unicellular or multicellular)

Species are grouped according to the maximum number of matching characters.

Probabilistic methods (maximum likelihood methods):

Classification is based on the *likelihood* (probability) of a certain tree explaining the set of objects. \rightsquigarrow look for ML tree

Application Areas

Research in biology, linguistics, archaeology,

- **The Tree of Life:** (Systematics)

1st generation: Linnaeus (1758) Independent of evolutionary history

2nd generation: Lamarck, Darwin, Haeckel (1800s)

Based on phylogenetic relationships (no objective criteria).

3rd generation: Zimmerman, Henning et al.

(50s and 60s) Phylogenies based on shared attributes (↔ character compatibility models).

4th generation: Many people (since the 1970s) Molecular sequence data available in huge quantities

- **The Indo-European tree of languages** by Ringe, Warnow et al. (1995)

Phylogenies with Protein Sequences

portions of peptide sequences of Triosephosphate Isomerase:

| | |
|----------|---|
| Spinach | CNGT KESITKL VSDLNSATLEAD--VDVVVAPP FVYIDQVKSS LTGRVEISA |
| Rice | CNGT TDQVDKIVKIL NEGQIASTDV VEVVV SPPY VFLPVVKS QLRPEIQVAA |
| Mosquito | MNGDKASIALDLCKVLTTGPLNAD--TEVVVGC PAPYL TLARSQLPDSVCVAA |
| Monkey | MNGRKQ N LGELIGTLNAAKVPAD--TEVVCAPPTAYIDFARQKLDPKIAVAA |
| Human | MNGRKQ S LGELIGTLNAAKVPAD--TEVVCAPPTAYIDFARQKLDPKIAVAA |

(Differences between Spinach and Rice = **orange**, between monkey and human = **green**, and "-" = gap).

Basis of Phylogenetic Inference: the more differences the less related are species. \rightsquigarrow Find tree *best explaining* differences.

A Simple Solution?

Trivial solution: enumerate over all possible trees and calculate the target function for each one.

Problem: number of non-isomorphic, labeled, binary, rooted trees containing n leaves, can be shown to be super-exponential:

$$(2n - 3)!! = \prod_{i=2}^n (2i - 3)$$

(or $(2n - 5)!!$ for unrooted trees). For $n = 20$: about 10^{21} trees
→ infeasible even for a relatively small number of species.

Theorem: Phylogenetic Tree Construction (for almost all reasonable models) is NP-Complete.

Note: *Uncountably infinite* many trees with real edge lengths.