CSI5126. Algorithms in bioinformatics

Pairwise Sequence **Alignment**

Marcel Turcotte

uOttawa

School of Electrical Engineering and Computer Science (EECS)
University of Ottawa

Version September 27, 2018

Summary

In this lecture, we learn that molecular sequences suffer mutations. We distinguish between two kinds of similar sequences: orthologues and paralogues. We derive an algorithm to compare molecular sequences taking into account their mode of evolution.

General objective

- Describe in your own words the pairwaise sequence alignment problem and explains its asumptions.

Reading

- Bernhard Haubold and Thomas Wiehe (2006). *Introduction to computational biology: an evolutionary approach.* Birkhäuser Basel. Pages 11-15, 30-33.

Comparative sequence analysis

Comparative sequence analysis

Why?

Comparative sequence analysis

Why?

*"Determining **function** for a sequence is a matter of tremendous complexity, requiring biological experiments of the highest order of creativity. Nevertheless, with only DNA sequence it is possible to execute a computer-based algorithm **comparing the sequence to a database of previously characterized genes**. In about 50% of the cases, such a mechanical comparison will indicate a sufficient degree of **similarity** to suggest a putative enzymatic or structural function that might be possessed by the unknown gene."*
*Caskey et al. (1995) Genome Digest **2**:6-9.*

Comparative sequence analysis

A **molecular sequence alignment** aims

- to identify **similar regions** between two sequences
- to determine if two sequences have a **common origin**

Comparative sequence analysis

- Molecular sequences are the result of **evolutionary processes**.

Comparative sequence analysis

- Molecular sequences are the result of **evolutionary processes**.
- **Speciation** (the formation of new and distinct species in the course of evolution) is the main process for creating new, yet **related**, sequences.

Comparative sequence analysis

- Molecular sequences are the result of **evolutionary processes**.
- **Speciation** (the formation of new and distinct species in the course of evolution) is the main process for creating new, yet **related**, sequences.
- Evolution transforms the sequences: **point mutations** (insertions, deletions, substitutions), **duplications**, **inversions**, **transpositions**, etc. Consequently, making it more difficult (interesting) to find the common origins.

Comparative sequence analysis

- Molecular sequences are the result of **evolutionary processes**.
- **Speciation** (the formation of new and distinct species in the course of evolution) is the main process for creating new, yet **related**, sequences.
- Evolution transforms the sequences: **point mutations** (insertions, deletions, substitutions), **duplications**, **inversions**, **transpositions**, etc. Consequently, making it more difficult (interesting) to find the common origins.
- Information (function, structure, etc.) that is known about a sequence can generally be **transferred to** "similar" sequences.

Comparative sequence analysis

- Molecular sequences are the result of **evolutionary processes**.
- **Speciation** (the formation of new and distinct species in the course of evolution) is the main process for creating new, yet **related**, sequences.
- Evolution transforms the sequences: **point mutations** (insertions, deletions, substitutions), **duplications**, **inversions**, **transpositions**, etc. Consequently, making it more difficult (interesting) to find the common origins.
- Information (function, structure, etc.) that is known about a sequence can generally be **transferred to** "similar" sequences.
- Comparative sequence analysis is therefore an **essential** and **powerful tool**.

Caveat

- **All forms of life** are believed to have evolved from a **common origin**.

Caveat

- **All forms of life** are believed to have evolved from a **common origin**.
- The genomic content of the proto-cell (proto-organism) certainly arose by a series of events, including duplication content, from smaller sequence fragments.

Caveat

- **All forms of life** are believed to have evolved from a **common origin**.
- The genomic content of the proto-cell (proto-organism) certainly arose by a series of events, including duplication content, from smaller sequence fragments.
- **Conclusion**: all the sequences are related one to another.

Caveat

- **All forms of life** are believed to have evolved from a **common origin**.
- The genomic content of the proto-cell (proto-organism) certainly arose by a series of events, including duplication content, from smaller sequence fragments.
- **Conclusion**: all the sequences are related one to another.
- **This is not a very productive statement.** The evolutionary relationships that are considered interesting are those that can be explained by the techniques presented here, for which there are convincing statistical evidences.

Requirements

What are the **requirements** (necessities, difficulties)?

Requirements

What are the **requirements** (necessities, difficulties)?

- Which **metric** is adequate to compare molecular sequences?

Requirements

What are the **requirements** (necessities, difficulties)?

- Which **metric** is adequate to compare molecular sequences?
- How to compute the alignment **efficiently**?

Requirements

What are the **requirements** (necessities, difficulties)?

- Which **metric** is adequate to compare molecular sequences?
- How to compute the alignment **efficiently**?
- How to align sequences **when** $|S_1| << |S_2|$?

Requirements

What are the **requirements** (necessities, difficulties)?

- Which **metric** is adequate to compare molecular sequences?
- How to compute the alignment **efficiently**?
- How to align sequences **when** $|S_1| << |S_2|$?
- How to score **substitutions**?

Requirements

What are the **requirements** (necessities, difficulties)?

- Which **metric** is adequate to compare molecular sequences?
- How to compute the alignment **efficiently**?
- How to align sequences **when** $|S_1| << |S_2|$?
- How to score **substitutions**?
- How to score **insertions** and **deletions**?

Requirements

What are the **requirements** (necessities, difficulties)?

- Which **metric** is adequate to compare molecular sequences?
- How to compute the alignment **efficiently**?
- How to align sequences **when** $|S_1| << |S_2|$?
- How to score **substitutions**?
- How to score **insertions** and **deletions**?
- Any two sequences can be aligned, how to evaluate (the **likelihood** of) an alignment?

Requirements (continued)

Are these two sequences similar?

```
A       VLSAADKGNVKAAWGKVGGHAAEYGAEALERMFLSFPTTKTYFPHFDLSHGSAQVKG
B       SLSAAQKDNVKSSWAKASAAWGTAGPEFFMALFDAHDDVFAKFSGLFSGAAKGTVKN
```

Requirements (continued)

Are these two sequences similar?

```
A       VLSAADKGNVKAAWGKVGGHAAEYGAEALERMFLSFPTTKTYFPHFDLSHGSAQVKG
B       SLSAAQKDNVKSSWAKASAAWGTAGPEFFMALFDAHDDVFAKFSGLFSGAAKGTVKN
         !!!! ! !!!  ! ! !       !       !         !            !
```

14 out of 57 (25 % of) amino acids are **identical**.

Requirements (continued)

Are these two sequences similar?

```
A     VLSAADKGNVKAAWGKVGGHAAEYGAEALERMFLSFPTTKTYFPHFD-LSHGSAQ--VKG
B     SLSAAQKDNVKSSWAKA---SAAWGTAGPEFFMALFDAHDDVFAKFSGLFSGAAKGTVKN
         !!!! ! !!!  ! !    ! ! !   !    !     ! ! ! ! !  !!
```

Insertion/deletions, two evolutionary events, must be taken into account

21 out of 60 (35 % of) positions are identical.

What's an **indel**?

- **Indel** stands for **insertion or deletion**.

What's an **indel**?

- **Indel** stands for **insertion or deletion**.
- **Given exactly two sequences, I am claiming that insertions cannot be distinguished from deletions, hence the use of the word indel.**

What's an **indel**?

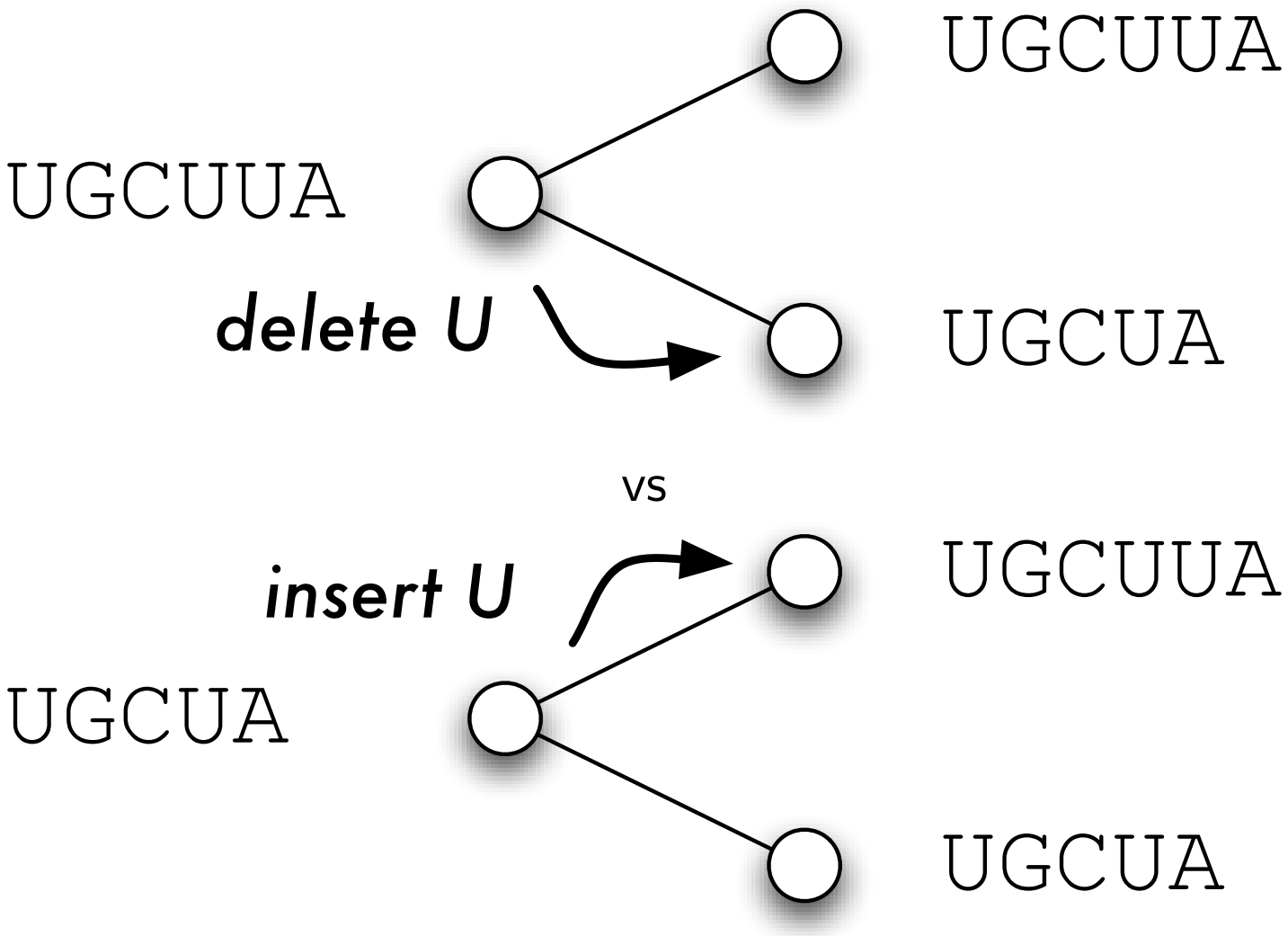
- **Indel** stands for **insertion or deletion**.
- **Given exactly two sequences, I am claiming that insertions cannot be distinguished from deletions, hence the use of the word indel.**

What's an **indel**?

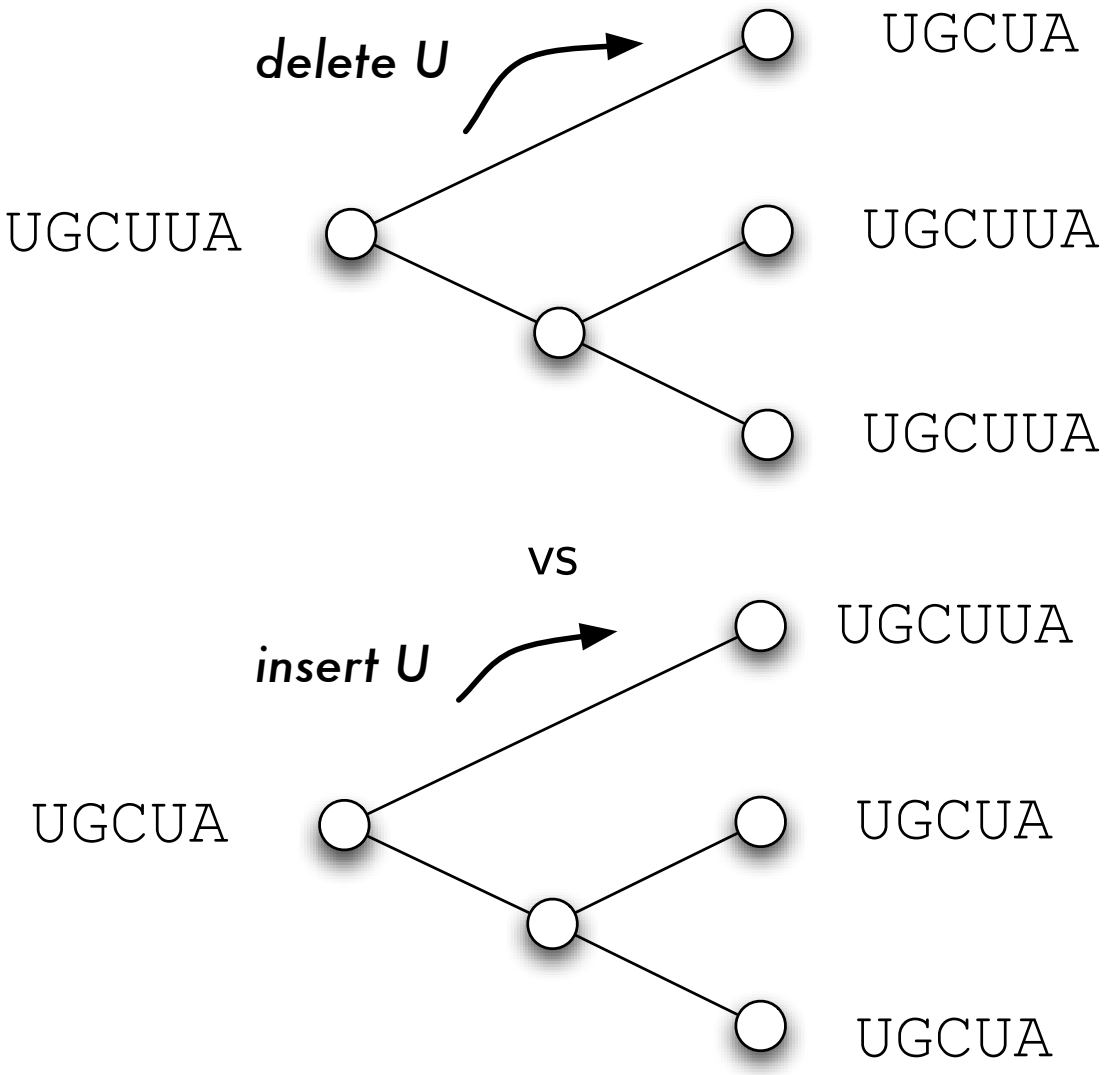
- **Indel** stands for **insertion or deletion**.
- **Given exactly two sequences, I am claiming that insertions cannot be distinguished from deletions, hence the use of the word indel.** What do I mean?
- Consider the following pairwise alignment, was the **U**, present in **S1**, deleted, to produce **S2**? Or, was is a **U** inserted into **S2** to produce **S1**?

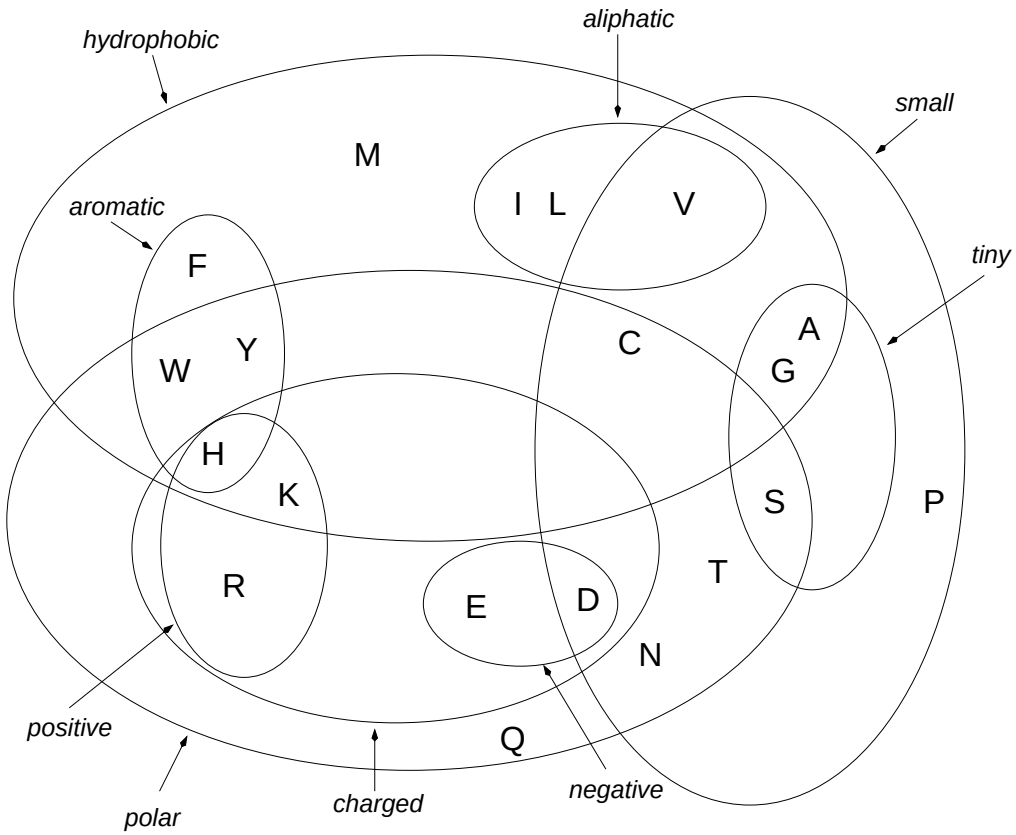
```
S1 = UGCUUA
S2 = UGC-UA
```

What's an **indel**?



What's an **indel**?





VLSAADKGNVKAAWGKVGGHAAEYGAEALERMFLSFPTTKTYFPHFD–LSHGSAQ
SLSAAQKDNVKSSWAKA——SAAWGTAGPEFFMALFDAHDDVFAKFSGLFSGAAK
! !

Requirements

Are these two sequences **similar**?

```
VLSAADKGNVKAAWGKVGGHAAEYGAEALERMFLSFPTTKTYFPHFD–LSHGSAQ
SLSAAQKDNVKSSWAKA——SAAWGTAGPEFFMALFDAHDDVFAKFSGLFSGAAK
  ! ! ! ! . !   ! ! ! . . ! . ! .     .!  .!.  .  !  ..   !  .      !  .!.  !   !.!.
```

38 out of 60 (63 % of) positions have the **same or similar properties**.

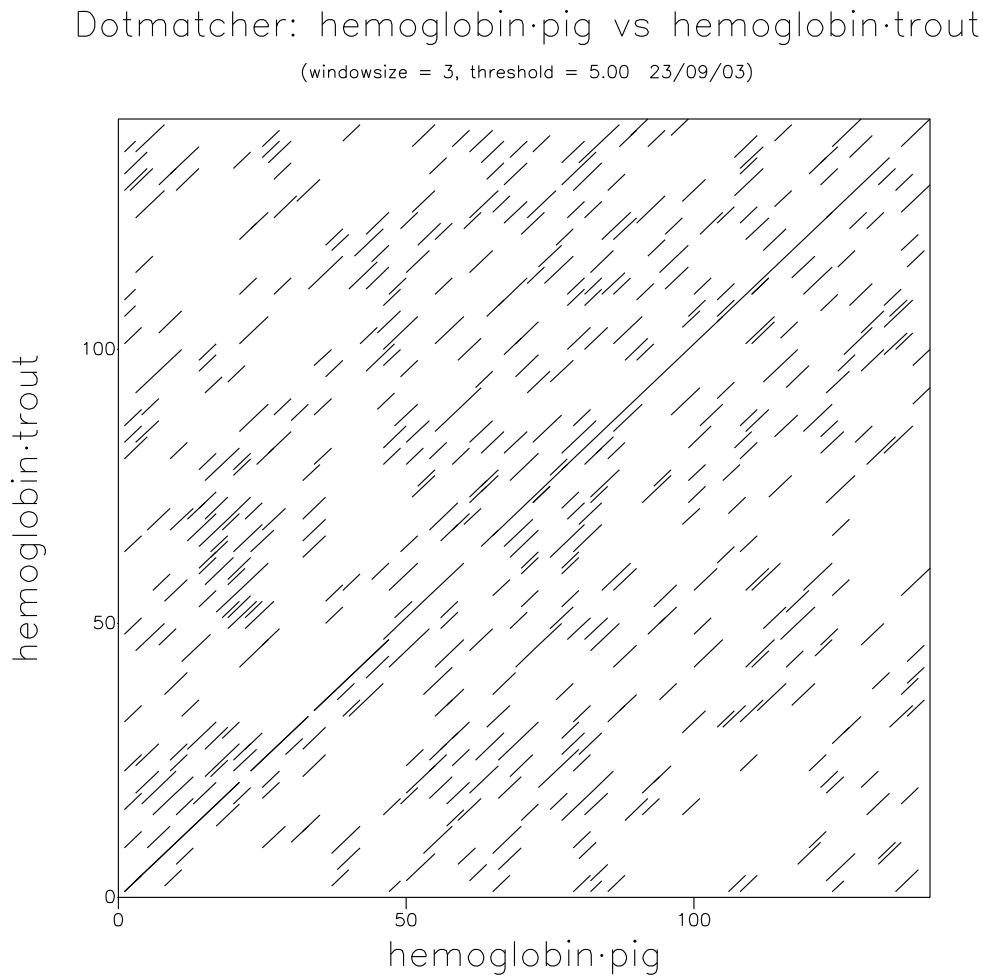
Requirements

Are these two sequences **similar**? (...have similar regions?)

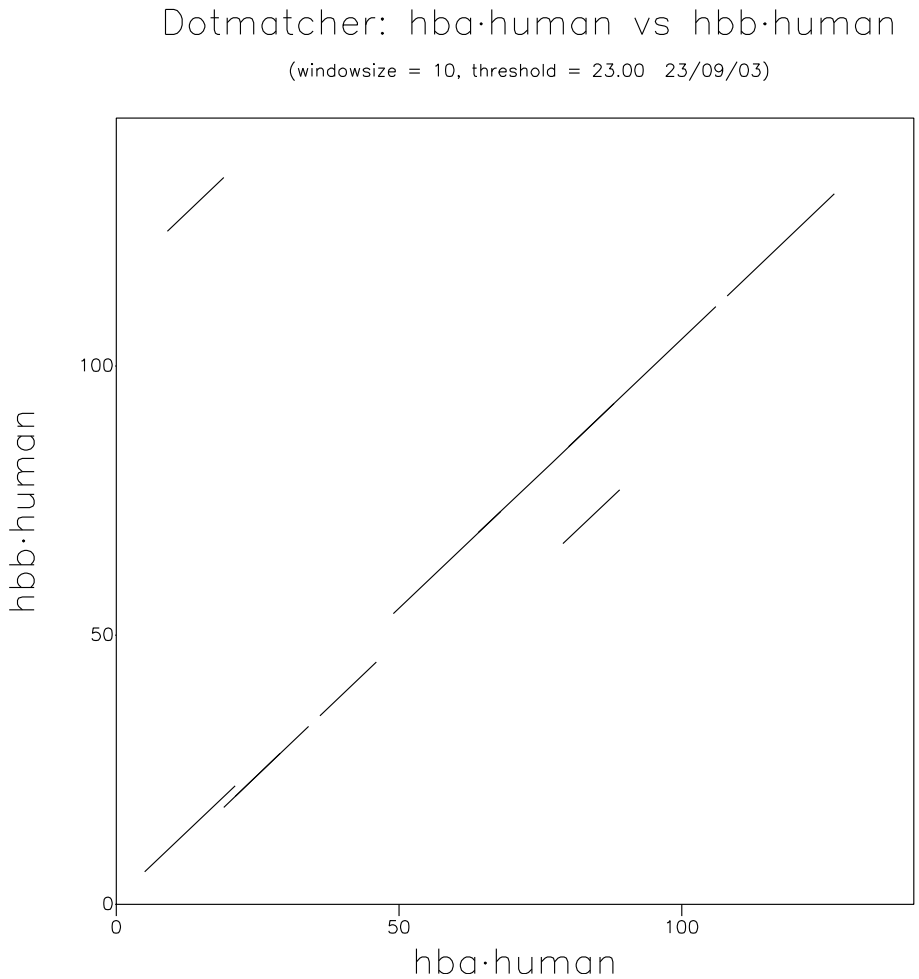
```
> Escherichia coli (K-12), complete genome
AGCTTTTCATTCTGACTGCAACGGGCAATATGTCTCTGTGTGGATTAAAAAAAGAGTGTC
TGATAGCAGCTTCTGAACTGGTTACCTGCCGTGAGTAAATTAAAATTTTATTGACTTAGG
TCACTAAATACTTTAACCAATATAGGCATAGCGCACAGACAGATAAAAATTACAGAGTAC
ACAACATCCATGAAACGCATTAGCACCACCATTACCACCACCATCACCATTACCACAGGT
AACGGTGCGGGCTGACGCGTACAGGAAACACAGAAAAAAGCCCGCACCTGACAGTGCGGG
CTTTTTTTTTCGACCAAAGGTAACGAGGTAACAACCATGCGAGTGTTGAAGTTCGGCGGT
ACATCAGTGGCAAATGCAGAACGTTTTCTGCGTGTTGCCGATATTCTGGAAAGCAATGCC
...(4,639,675)...
GCATGATATTGAAAAAAATATCACCAAATAAAAAACGCCTTAGTAAGTATTTTTC

> Methanococcus vannielii SB, DNA-directed RNA polymerase
ATGGATAGATTTGATGTTCCAAAGGAAATCGGAGATATTACATTTGGATTGCTCTCTCCA
GAACAGATAAGGACAATGTCTGTTGCAAAAATCGTTACAGCAGATACTTATGATGACGAT
... (2,670) ...
ACAAAAGTCATTTCAAAATATGAAAATTAA
```

Dot plot

- A **dot plot** is a useful tool to compare two sequences.
- It consists of a **two dimensional diagram**, such that one sequence is written along one of its axes, and the other sequence along the other axis.
- In its simplest form, **a dot is plotted at position i and j if the characters i and j of the two respective strings are identical**.



The resulting diagram often contain too much noise (is too busy).



A **window-based** approach is generally used to circumvent the problem, i.e. a **dot** is plotted only if x **characters** (amino acids or nucleotides) **out** w **characters** are **identical**, where w is the window size.

Dot plots

- **Insertions/deletions** show up as slightly shifted diagonal

Dot plots

- **Insertions/deletions** show up as slightly shifted diagonal
- Shows **duplications**

Dot plots

- **Insertions/deletions** show up as slightly shifted diagonal
- Shows **duplications**
- Identifies **local** similarity

Dot plots

- **Insertions/deletions** show up as slightly shifted diagonal
- Shows **duplications**
- Identifies **local** similarity
- Shows **inverted repeats** (anti-diagonals)

Dot plots

- **Insertions/deletions** show up as slightly shifted diagonal
- Shows **duplications**
- Identifies **local** similarity
- Shows **inverted repeats** (anti-diagonals)
- **Not suitable for automated analyses**

Finding an appropriate **metric**

When insertions and deletions are allowed, there are many possible alignments of the two input sequences.

Finding an appropriate **metric**

When insertions and deletions are allowed, there are many possible alignments of the two input sequences.

```
S1 A T T C G      S1 A T T C G -      S1 A T T C G - -
S2 T T C C A      S2 - T T C C A      S2 - T T C - C A
     x   x             x x x              x x x
```

- How many alignments for two input sequences of length 5?

Finding an appropriate **metric**

When insertions and deletions are allowed, there are many possible alignments of the two input sequences.

```
S1 A T T C G      S1 A T T C G -      S1 A T T C G - -
S2 T T C C A      S2 - T T C C A      S2 - T T C - C A
     x   x             x x x              x x x
```

- How many alignments for two input sequences of length 5? 1,683

Finding an appropriate **metric**

When insertions and deletions are allowed, there are many possible alignments of the two input sequences.

```
S1 A T T C G      S1 A T T C G -      S1 A T T C G - -
S2 T T C C A      S2 - T T C C A      S2 - T T C - C A
     x   x             x x x              x x x
```

- How many alignments for two input sequences of length 5? 1,683
- Which one to choose?

Finding an appropriate **metric**

When insertions and deletions are allowed, there are many possible alignments of the two input sequences.

```
S1 A T T C G      S1 A T T C G -      S1 A T T C G - -
S2 T T C C A      S2 - T T C C A      S2 - T T C - C A
     x   x             x x x              x x x
```

- How many alignments for two input sequences of length 5? 1,683
- Which one to choose?
- The **edit distance** is the minimum number of edit operations that are needed to transform one string into the other.

Finding an appropriate **metric**

When insertions and deletions are allowed, there are many possible alignments of the two input sequences.

```
S1 A T T C G      S1 A T T C G -      S1 A T T C G - -
S2 T T C C A      S2 - T T C C A      S2 - T T C - C A
     x   x             x x x              x x x
```

- How many alignments for two input sequences of length 5? 1,683
- Which one to choose?
- The **edit distance** is the minimum number of edit operations that are needed to transform one string into the other.
- The edit distance is sometimes referred to as **Levenshtein distance**.

Edit distance

The edit operations that are useful to model evolutionary processes are **insertions** (I), **deletions** (D) and the **substitutions** (S).

```
S1        A T T C G
S2          T T C C A
```

- The set of operations can be augmented with the **match** (M) operation, which simply rewrites a letter from the input onto the output.
- However, the match operation will not be counted when calculating the edit distance; in other words, it can be seens as having a weight of 0.

Edit distance

- What are the **assumptions**?

Edit distance

- What are the **assumptions**?
- **Independent**. These operations are independent one from another. The likelihood of a substitution at position i is not affected by the identity of the residue found at position j. Is this realistic?

Edit distance

- What are the **assumptions**?
- **Independent**. These operations are independent one from another. The likelihood of a substitution at position i is not affected by the identity of the residue found at position j. Is this realistic?
- **Identically distributed**. The likelihood does not depend on the specific value of i, the position.

Edit distance

- What are the **assumptions**?
- **Independent**. These operations are independent one from another. The likelihood of a substitution at position i is not affected by the identity of the residue found at position j. Is this realistic?
- **Identically distributed**. The likelihood does not depend on the specific value of i, the position.
- **Is this realistic?**

Edit **transcript**

An **edit transcript** is a string over $\{I, D, S, M\}$ that summarizes the edit operations that are applied to the first string in order to produce the second one.

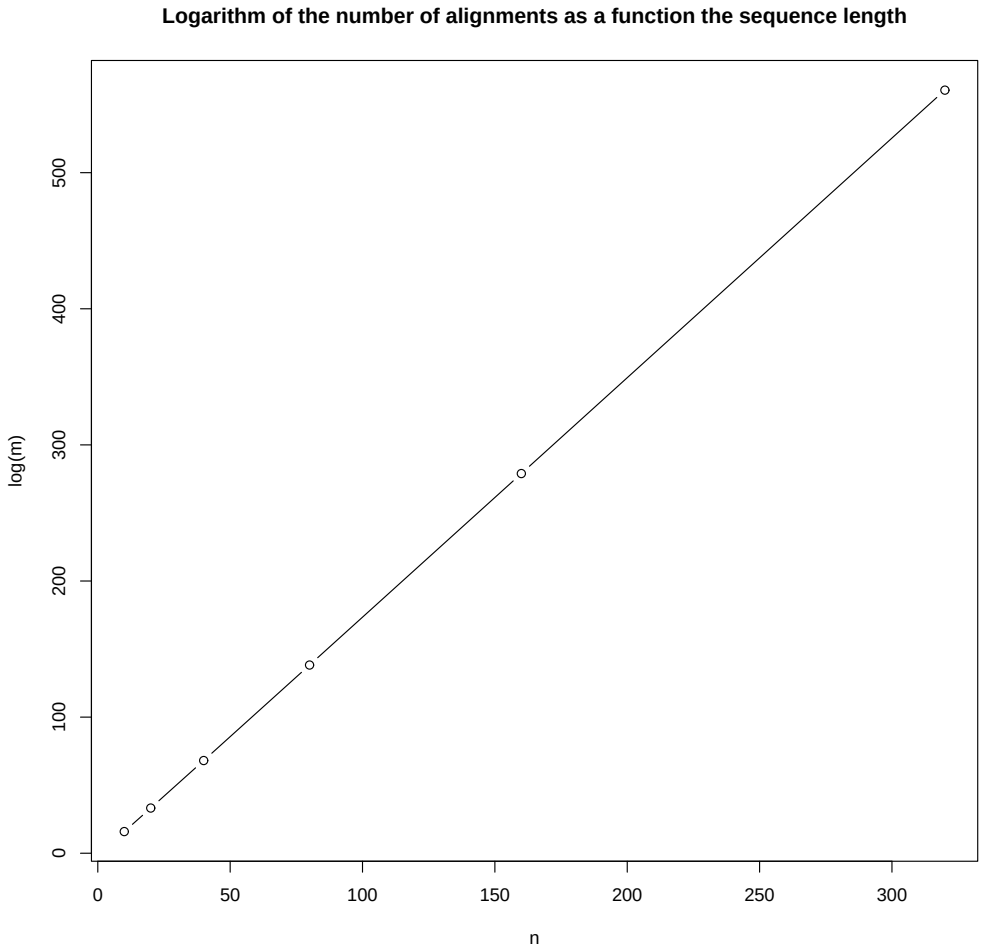
```
Transcript : D M M M S I
S1         : A T T C G -
S2         : - T T C C A
```

Pairwise alignment problem

- A **string alignment** consists of two input strings, written one on the top of the other, such that space (or dash) symbols have been added to the first, or second, string when insertions, or deletions, are seen in the edit transcript.
- The **edit distance problem** consists in finding the alignment (or equivalently the edit transcript) that minimizes the edit distance.

Size of the search space

Length	# alignments
10	8.097453e+06
20	2.605438e+14
40	3.781502e+29
80	1.121607e+60
160	1.392368e+121
320	3.031221e+243



Edit distance (continued)

- Uses of the edit distance occur **outside of the context of biological sequence comparisons**, examples are: spelling correction methods or textual database retrieval.
- The **Unix program diff** is an example of a program that is based on the notion of edit distance. It is a program that compares the content of two files.
- When ran with the argument **e** the program program produces a series of commands for the editor **ed** to transform the first file into the other.

Computing the optimal alignment

- To find an answer, it will help to **formalize this problem**, to find a mathematical formulation.

Computing the optimal alignment

- To find an answer, it will help to **formalize this problem**, to find a mathematical formulation.
- We're given **two strings** S_1 **and** S_2 of length n and m respectively.

Computing the optimal alignment

- To find an answer, it will help to **formalize this problem**, to find a mathematical formulation.
- We're given **two strings** S_1 **and** S_2 of length n and m respectively.
- Let $D(S_1, S_2)$ denote the **edit distance** of S_1 and S_2 (the **minimum** number of edit operations needed to transform S_1 into S_2).

Computing the optimal alignment

- To find an answer, it will help to **formalize this problem**, to find a mathematical formulation.
- We're given **two strings** S_1 **and** S_2 of length n and m respectively.
- Let $D(S_1, S_2)$ denote the **edit distance** of S_1 and S_2 (the **minimum** number of edit operations needed to transform S_1 into S_2).
- The notation $S_1(i)$ stands for the i-th character of S_1, e.g. $S_1 = TATAAT$, $S_1(3) = T$.

Computing the optimal alignment

- To find an answer, it will help to **formalize this problem**, to find a mathematical formulation.
- We're given **two strings** S_1 **and** S_2 of length n and m respectively.
- Let $D(S_1, S_2)$ denote the **edit distance** of S_1 and S_2 (the **minimum** number of edit operations needed to transform S_1 into S_2).
- The notation $S_1(i)$ stands for the i-th character of S_1, e.g. $S_1 = TATAAT$, $S_1(3) = T$.
- The notation $S_1[i, j]$ stands for the substring of S_1 starting at position i and ending at position j, $S_1[i, j] = S_1(i)S_1(i+1)\dots S_1(j)$, e.g. $S_1 = TATAAT$, $S_1[3, 5] = TAA$.

Computing the optimal alignment

- To find an answer, it will help to **formalize this problem**, to find a mathematical formulation.
- We're given **two strings** S_1 **and** S_2 of length n and m respectively.
- Let $D(S_1, S_2)$ denote the **edit distance** of S_1 and S_2 (the **minimum** number of edit operations needed to transform S_1 into S_2).
- The notation $S_1(i)$ stands for the i-th character of S_1, e.g. $S_1 = TATAAT$, $S_1(3) = T$.
- The notation $S_1[i, j]$ stands for the substring of S_1 starting at position i and ending at position j, $S_1[i, j] = S_1(i)S_1(i+1)\ldots S_1(j)$, e.g. $S_1 = TATAAT$, $S_1[3, 5] = TAA$.
- I like considering the problem from the point of view of the edit transcript.

- The edit transcript of the optimal alignment will **end** with one of the four edit operations, **I, D, S** or **M**.

- The edit transcript of the optimal alignment will **end** with one of the four edit operations, **I, D, S** or **M**.
- We don't know which one! Therefore, let's **consider all 4 possibilities**.

- The edit transcript of the optimal alignment will **end** with one of the four edit operations, **I, D, S** or **M**.
- We don't know which one! Therefore, let's **consider all 4 possibilities**.
- First, consider a transcript ending with the operation **I**.

$$\begin{array}{cc} & I \\ S_1 & - \\ S_2[1, m-1] & S_2(m) \end{array}$$

where $S_2(m)$ is the last symbol of S_2. Make sure to understand the details of above illustration. S_2 has been decomposed into a prefix and the last symbol, $S_2[1, m-1]S_2(m) = S_2$, a dash symbol has been added to the end of S_1.

- The edit transcript of the optimal alignment will **end** with one of the four edit operations, **I, D, S** or **M**.
- We don't know which one! Therefore, let's **consider all 4 possibilities**.
- First, consider a transcript ending with the operation **I**.

$$\begin{array}{cc} & I \\ S_1 & - \\ S_2[1, m-1] & S_2(m) \end{array}$$

where $S_2(m)$ is the last symbol of S_2. Make sure to understand the details of above illustration. S_2 has been decomposed into a prefix and the last symbol, $S_2[1, m-1]S_2(m) = S_2$, a dash symbol has been added to the end of S_1.

- Assuming this transcript leads to an optimal alignment, **how many edit operations are needed to transform S_1 into S_2?**

- Similarly for D (deletion),

$$\begin{array}{cc} & D \\ S_1[1, n-1] & S_1(n) \\ S_2 & - \end{array}$$

- and S (substitution),

$$\begin{array}{cc} & S \\ S_1[1, n-1] & S_1(n) \\ S_2[1, m-1] & S_2(m) \end{array}$$

- and M (match),

$$\begin{array}{cc} & M \\ S_1[1, n-1] & S_1(n) \\ S_2[1, m-1] & S_2(m) \end{array}$$

Obviously, only if $S_1(n) = S_2(m)$!

- and M (match),

$$\begin{array}{cc} & M \\ S_1[1, n-1] & S_1(n) \\ S_2[1, m-1] & S_2(m) \end{array}$$

  Obviously, only if $S_1(n) = S_2(m)$!
- For each case, how many edit operations are needed to transform S_1 into S_2?

$$\begin{array}{cc} & I \\ S_1 & - \\ S_2[1, m-1] & S_2(m) \end{array}$$

The number of edit operations required is?

$$\begin{array}{cc} & I \\ S_1 & - \\ S_2[1, m-1] & S_2(m) \end{array}$$

The number of edit operations required is?

$$D(S_1, S_2[1, m-1]) + 1$$

$$\begin{array}{cc} & D \\ S_1[1, n-1] & S_1(n) \\ S_2 & - \end{array}$$

The number of edit operations required is?

$$\begin{array}{cc} & D \\ S_1[1, n-1] & S_1(n) \\ S_2 & - \end{array}$$

The number of edit operations required is?

$$D(S_1[1, n-1], S_2) + 1$$

$$\begin{array}{cc} & S \\ S_1[1, n-1] & S_1(n) \\ S_2[1, m-1] & S_2(m) \end{array}$$

The number of edit operations required is?

$$D(S_1[1, n-1], S_2[1, m-1]) + 1$$

$$\begin{array}{cc} & M \\ S_1[1, n-1] & S_1(n) \\ S_2[1, m-1] & S_2(m) \end{array}$$

The number of edit operations required is?

$$D(S_1[1, n-1], S_2[1, m-1]) + 0$$

- Let's change the representation slightly so that $D(i, j)$ denotes the edit distance of $S_1[1, i]$ and $S_2[1, j]$

- Let's change the representation slightly so that $D(i, j)$ denotes the edit distance of $S_1[1, i]$ and $S_2[1, j]$
- In other words, $D(i, j)$ represents the minimum number of edit operations that are necessary to transform the first i characters of S_1 into the first j characters of S_2

- Let's change the representation slightly so that $D(i, j)$ denotes the edit distance of $S_1[1, i]$ and $S_2[1, j]$
- In other words, $D(i, j)$ represents the minimum number of edit operations that are necessary to transform the first i characters of S_1 into the first j characters of S_2
- Does it mean that $S_1(i)$ and $S_2(j)$ are aligned?

- Let's change the representation slightly so that $D(i, j)$ denotes the edit distance of $S_1[1, i]$ and $S_2[1, j]$
- In other words, $D(i, j)$ represents the minimum number of edit operations that are necessary to transform the first i characters of S_1 into the first j characters of S_2
- Does it mean that $S_1(i)$ and $S_2(j)$ are aligned?
- Consider $S_1 = \underline{ATT}GC$, $S_2 = \underline{A}GC$, and $D(3, 1)$, it does not mean that $S_1(3) = T$ is aligned against $S_2(1) = A$

- Let's change the representation slightly so that $D(i, j)$ denotes the edit distance of $S_1[1, i]$ and $S_2[1, j]$
- In other words, $D(i, j)$ represents the minimum number of edit operations that are necessary to transform the first i characters of S_1 into the first j characters of S_2
- Does it mean that $S_1(i)$ and $S_2(j)$ are aligned?
- Consider $S_1 = \underline{ATT}GC$, $S_2 = \underline{A}GC$, and $D(3, 1)$, it does not mean that $S_1(3) = T$ is aligned against $S_2(1) = A$
- Any other alignment than the one below would involve 3 or more edit operations (2 deletions and one substitution)

```
S1    ATT
      |
S2    A--
```

- Let's change the representation slightly so that $D(i, j)$ denotes the edit distance of $S_1[1, i]$ and $S_2[1, j]$
- In other words, $D(i, j)$ represents the minimum number of edit operations that are necessary to transform the first i characters of S_1 into the first j characters of S_2
- Does it mean that $S_1(i)$ and $S_2(j)$ are aligned?
- Consider $S_1 = \underline{ATT}GC$, $S_2 = \underline{A}GC$, and $D(3, 1)$, it does not mean that $S_1(3) = T$ is aligned against $S_2(1) = A$
- Any other alignment than the one below would involve 3 or more edit operations (2 deletions and one substitution)

```
S1    ATT
      |
S2    A--
```

- Here, the edit transcript of the optimal alignment is ending with a deletion (D)

- Let's see if we can find some base conditions.

- Let's see if we can find some base conditions. $D(0,0) = ?$

- Let's see if we can find some base conditions. $D(0,0) = ?$
- Surely, $D(0,0) = 0$,

- Let's see if we can find some base conditions. $D(0, 0) = ?$
- Surely, $D(0, 0) = 0$, no operations are needed to transform the first **zero** characters of S_1 into the first **zero** characters of S_2.

- Let's see if we can find some base conditions. $D(0, 0) = ?$
- Surely, $D(0, 0) = 0$, no operations are needed to transform the first **zero** characters of S_1 into the first **zero** characters of S_2.
- $D(i, 0)$ means transforming the first i characters of S_1 into the first **zero** characters of S_2. How many operations?

- Let's see if we can find some base conditions. $D(0,0) = ?$
- Surely, $D(0,0) = 0$, no operations are needed to transform the first **zero** characters of S_1 into the first **zero** characters of S_2.
- $D(i,0)$ means transforming the first i characters of S_1 into the first **zero** characters of S_2. How many operations? One needs to delete i characters, we therefore have,

$$D(i,0) = i$$

- Let's see if we can find some base conditions. $D(0,0) = ?$
- Surely, $D(0,0) = 0$, no operations are needed to transform the first **zero** characters of S_1 into the first **zero** characters of S_2.
- $D(i,0)$ means transforming the first i characters of S_1 into the first **zero** characters of S_2. How many operations? One needs to delete i characters, we therefore have,

$$D(i,0) = i$$

- Similarly, to transform the first j characters of S_2 into the first **zero** characters of S_1, i.e. $D(0,j)$, we have delete the first j characters of S_2,

$$D(0,j) = j$$

- For the general case, how was $D(i, j)$ obtained?

- For the general case, how was $D(i, j)$ obtained? Clearly, it was obtained by applying one of the three (four) possible edit operations: insertion, deletion, substitution (match), to a smaller alignment

- For the general case, how was $D(i, j)$ obtained? Clearly, it was obtained by applying one of the three (four) possible edit operations: insertion, deletion, substitution (match), to a smaller alignment
- Given, two sequences S_1 and S_2 of length m and n respectively, which particular value of D solves the problem?

- For the general case, how was $D(i, j)$ obtained? Clearly, it was obtained by applying one of the three (four) possible edit operations: insertion, deletion, substitution (match), to a smaller alignment
- Given, two sequences S_1 and S_2 of length m and n respectively, which particular value of D solves the problem?
- $D(m, n)$ **is the value that we are looking for. It is the minimum number of edit operations that are needed to transform the first** m **characters of** S_1 **into the first** n **characters of** S_2

Recurrence equation for the edit distance problem

Base conditions,

$$D(0,0) = 0$$

$$D(i,0) = i,\ \ i \in 1 \ldots n$$

$$D(0,j) = j,\ \ j \in 1 \ldots m$$

General case,

$$D(i,j) = \min \begin{cases} D(i-1,j) + 1, \\ D(i,j-1) + 1, \\ D(i-1,j-1) + 1, \text{if } S_1(i) \neq S_2(j), \\ D(i-1,j-1) + 0, \text{if } S_1(i) = S_2(j), \end{cases}$$

Solution,

$$D(m,n)$$

Algorithm for solving the edit distance recurrence equation

Two strategies:

- Top-down
- Bottom-up

Top-down computation

- In the top-down computation, a first call is made to compute $D(m, n)$, which will force the computation of $D(m-1, n)$, $D(m, n-1)$ and $D(m-1, n-1)$

Top-down computation

- In the top-down computation, a first call is made to compute $D(m, n)$, which will force the computation of $D(m-1, n)$, $D(m, n-1)$ and $D(m-1, n-1)$
- The computation of $D(m-1, n)$ forces the computation of $D(m-2, n)$, $D(m-1, n-2)$ and $D(m-2, n-1)$

Top-down computation

- In the top-down computation, a first call is made to compute $D(m, n)$, which will force the computation of $D(m-1, n)$, $D(m, n-1)$ and $D(m-1, n-1)$
- The computation of $D(m-1, n)$ forces the computation of $D(m-2, n)$, $D(m-1, n-2)$ and $D(m-2, n-1)$
- The computation of $D(m, n-1)$ forces the computation of $D(m-1, n-1)$, $D(m, n-2)$ and $D(m-1, n-2)$

Top-down computation

- In the top-down computation, a first call is made to compute $D(m, n)$, which will force the computation of $D(m-1, n)$, $D(m, n-1)$ and $D(m-1, n-1)$
- The computation of $D(m-1, n)$ forces the computation of $D(m-2, n)$, $D(m-1, n-2)$ and $D(m-2, n-1)$
- The computation of $D(m, n-1)$ forces the computation of $D(m-1, n-1)$, $D(m, n-2)$ and $D(m-1, n-2)$
- The computation of $D(m-1, n-1)$ forces the computation of $D(m-2, n-1)$, $D(m-1, n-2)$ and $D(m-2, n-2)$

Top-down computation

- In the top-down computation, a first call is made to compute $D(m, n)$, which will force the computation of $D(m-1, n)$, $D(m, n-1)$ and $D(m-1, n-1)$
- The computation of $D(m-1, n)$ forces the computation of $D(m-2, n)$, $D(m-1, n-2)$ and $D(m-2, n-1)$
- The computation of $D(m, n-1)$ forces the computation of $D(m-1, n-1)$, $D(m, n-2)$ and $D(m-1, n-2)$
- The computation of $D(m-1, n-1)$ forces the computation of $D(m-2, n-1)$, $D(m-1, n-2)$ and $D(m-2, n-2)$
- Many values will be re-computed several times in the top-down computation

Top-down computation

- In the top-down computation, a first call is made to compute $D(m, n)$, which will force the computation of $D(m-1, n)$, $D(m, n-1)$ and $D(m-1, n-1)$
- The computation of $D(m-1, n)$ forces the computation of $D(m-2, n)$, $D(m-1, n-2)$ and $D(m-2, n-1)$
- The computation of $D(m, n-1)$ forces the computation of $D(m-1, n-1)$, $D(m, n-2)$ and $D(m-1, n-2)$
- The computation of $D(m-1, n-1)$ forces the computation of $D(m-2, n-1)$, $D(m-1, n-2)$ and $D(m-2, n-2)$
- Many values will be re-computed several times in the top-down computation
- It is easy to see that an exponential number of operations will be performed!

Top-down computation

- In the top-down computation, a first call is made to compute $D(m, n)$, which will force the computation of $D(m-1, n)$, $D(m, n-1)$ and $D(m-1, n-1)$
- The computation of $D(m-1, n)$ forces the computation of $D(m-2, n)$, $D(m-1, n-2)$ and $D(m-2, n-1)$
- The computation of $D(m, n-1)$ forces the computation of $D(m-1, n-1)$, $D(m, n-2)$ and $D(m-1, n-2)$
- The computation of $D(m-1, n-1)$ forces the computation of $D(m-2, n-1)$, $D(m-1, n-2)$ and $D(m-2, n-2)$
- Many values will be re-computed several times in the top-down computation
- It is easy to see that an exponential number of operations will be performed!
- A complete 3-way tree of depth m has $\Theta(3^m)$ nodes.

Bottom-up (tabular) computation

- Hum, but there are only $(n+1) \times (m+1)$ distinct $D(i,j)$ values!

Bottom-up (tabular) computation

- Hum, but there are only $(n+1) \times (m+1)$ distinct $D(i,j)$ values!
- The bottom-up computation proceeds with the small values of i and j first.

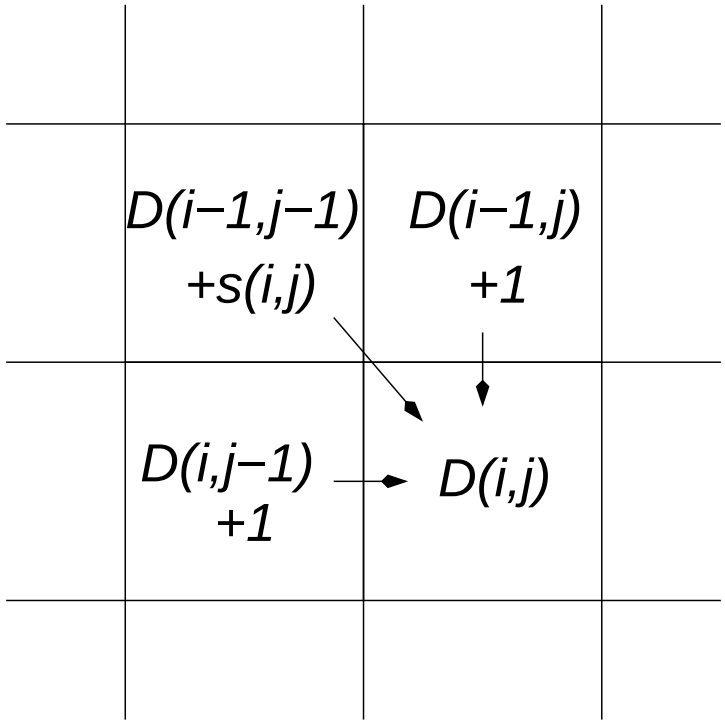
Bottom-up (tabular) computation

- Hum, but there are only $(n+1) \times (m+1)$ distinct $D(i,j)$ values!
- The bottom-up computation proceeds with the small values of i and j first.
- Furthermore, the algorithm memorizes (caches) the values of $D(i,j)$ so that a given $D(i,j)$ is computed only once.

Bottom-up (tabular) computation (continued)

1. This technique is known as **dynamic programming**;
2. Dynamic programming can only be applied to problems with a structure known as the **Bellman principle**.

Bottom-up computation



where $s(i,j) = 1$ if $S_1(i) \neq S_2(j)$ and 0 otherwise.

Bottom-up computation

		0	1	2	3	4	5
		–	A	T	C	G	C
0	–	0	1	2	3	4	5
1	A	1					
2	G	2					
3	G	3					
4	C	4					

$\Rightarrow$ **Base conditions**.

Bottom-up computation

		0	1	2	3	4	5
		–	A	T	C	G	C
0	–	0	1	2	3	4	5
1	A	1	0	1	2	3	4
2	G	2	1	1	2		
3	G	3					
4	C	4					

$\Rightarrow$ Notice the two **alternatives**: $D(1,2)+1 = D(2,2)+1 = 2$

Bottom-up computation

		0	1	2	3	4	5
		–	A	T	C	G	C
0	–	0	1	2	3	4	5
1	A	1	0	1	2	3	4
2	G	2	1	1	2	2	3
3	G	3	2	2	2	2	3
4	C	4	3	3	2	3	2

$\Rightarrow$ The **final result** is $D(4,5) = 2$. What does it tell us?

Bottom-up computation

		0	1	2	3	4	5
		–	A	T	C	G	C
0	–	0	1	2	3	4	5
1	A	1	0	1	2	3	4
2	G	2	1	1	2	2	3
3	G	3	2	2	2	2	3
4	C	4	3	3	2	3	2

$\Rightarrow$ We now know that **one sequence can be transformed into the other** with as little as **2 edit operations**!

Remarks

- **How** do you fill up the matrix: by **row**? by **column**? by **diagonal**? it is not important?

Remarks

- **How** do you fill up the matrix: by **row**? by **column**? by **diagonal**? it is not important?
- **How** many cells can be filled simultaneously? Leading to parallel computation.

Remarks

- **How** do you fill up the matrix: by **row**? by **column**? by **diagonal**? it is not important?
- **How** many cells can be filled simultaneously? Leading to parallel computation.
- $D(4,5) = 2$, it's possible to transform S_1 into S_2 with two edit operations, **which ones**?

Remarks

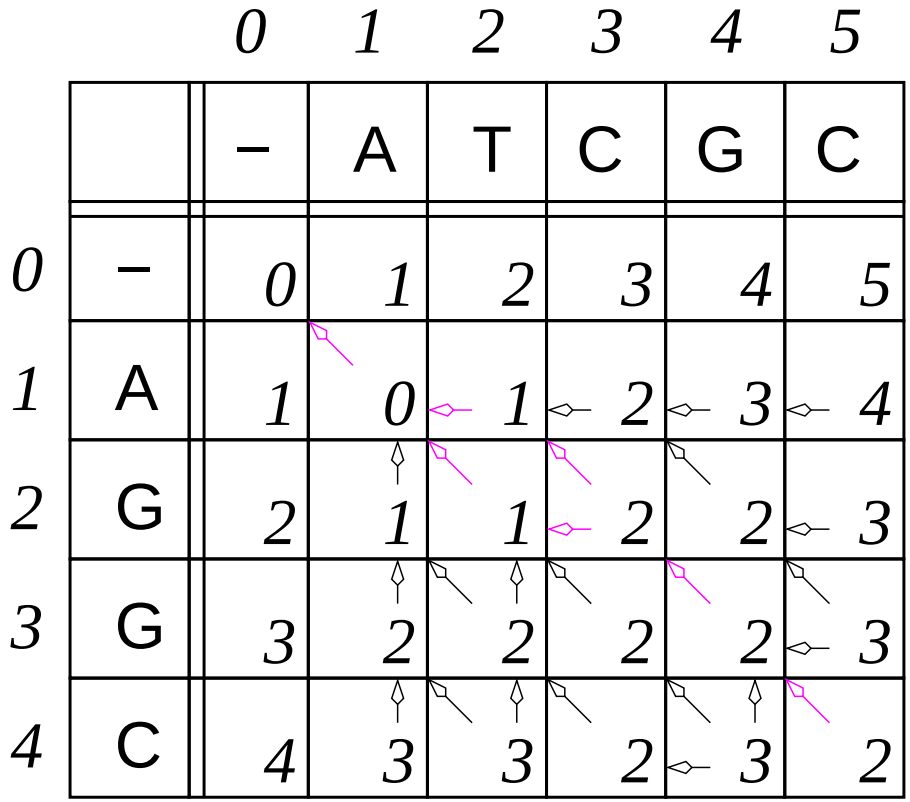
- **How** do you fill up the matrix: by **row**? by **column**? by **diagonal**? it is not important?
- **How** many cells can be filled simultaneously? Leading to parallel computation.
- $D(4,5) = 2$, it's possible to transform S_1 into S_2 with two edit operations, **which ones**?
- **How** to compute the actual alignment?

Dynamic Programming

		0	1	2	3	4	5
		–	A	T	C	G	C
0	–	0	1	2	3	4	5
1	A	1	0	1	2	3	4
2	G	2	1	1	2	2	3
3	G	3	2	2	2	2	3
4	C	4	3	3	2	3	2

$\Rightarrow$ **How to recover the underlying alignment?**

Traceback


0 1 2 3 4 5
– A T C G C
0 – 0 1 2 3 4 5
1 A 1 0 1 2 3 4
2 G 2 1 1 2 2 3
3 G 3 2 2 2 2 3
4 C 4 3 3 2 3 2

Traceback

In cell $D(i,j)$:

- set $\nwarrow$ if $D(i-1,j-1)+s(S_1(i),S_2(j))=D(i,j)$,
- set $\leftarrow$ if $D(i,j-1)+1=D(i,j)$,
- set $\uparrow$ if $D(i-1,j)+1=D(i,j)$.

Traceback (continued)

To recover the edit transcript, the alignment, follow a path from $D(n, m)$ to $D(0, 0)$. Interpreting each pointer as follows:

- $\leftarrow$: deletion of $S_1(j)$,
- $\uparrow$: insertion of $S_2(i)$,
- $\nwarrow$: match of $S_1(i)$ and $S_2(j)$ if $S_1(i) = S_2(j)$ and substitution otherwise.

The two optimal alignments:

```
ATCGC        ATCGC
A-GGC   or   AG-GC
```

$\Rightarrow$ It takes $\mathcal{O}(n + m)$ time to compute the traceback for one path.

Remarks

- There was **more than one optimal alignment**.

Remarks

- There was **more than one optimal alignment**.
- Only one solution was recovered, but we could have recorded all of them.

Remarks

- There was **more than one optimal alignment**.
- Only one solution was recovered, but we could have recorded all of them.
- **How many optimal alignments are there**?

Remarks

- There was **more than one optimal alignment**.
- Only one solution was recovered, but we could have recorded all of them.
- **How many optimal alignments are there**?
- **Can you enumerate them**?

References

- Gusfield, D. (1997) *Algorithms on strings, trees, and sequences: computer science and computational biology*. Cambridge Press, pp. 215–224. (MRT General QA 76.9 .A43 G87 1997)
- Jones N.C. and Pevzner P.A. (2004) *An Introduction to Bioinformatics Algorithms*, MIT Press, pp. 147–178. (QH324.2 b.J66 2004)
- Durbin, R. *et al* (1998,2000) *Biological sequence analysis: probabilistic models of proteins and nucleic acids*. Cambridge University Press. §2 (MRT General QP 620 .B576 1998)

References

Pensez-y!

L'impression de ces notes n'est probablement pas nécessaire!