

# Sequence Alignment

Ulf Leser

# This Lecture

---

- **Approximate String Matching**
- Edit distance and alignment
- Computing global alignments
- Local alignment

# Gene Function

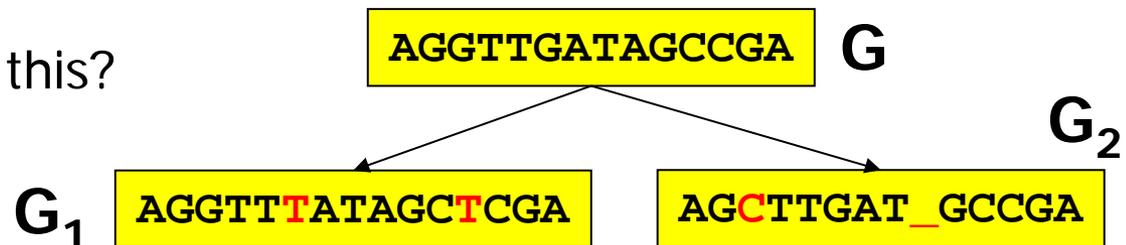
---

- A fundamental principle of bioinformatics
  - The function of a protein depends on its **physical structure**
  - The physical structure depends on the **protein sequence**
  - The protein sequence depends on the **gene sequence**
  - If the sequence of two genes is only slightly different, so will be the protein sequence
  - If the sequence of two proteins is only slightly different, so will be their structure
    - If the structure of two proteins is only moderately different, they **likely have the same (or at least share some) function**
- Studying the sequence of genes allows the generation of **hypotheses about their function**

# How Genes Evolve

---

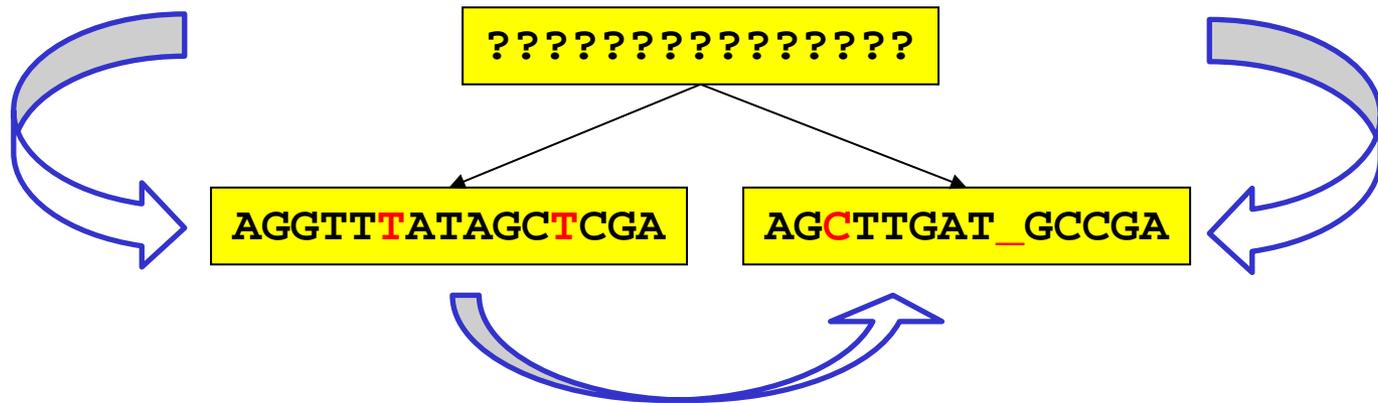
- Evolution, sequences, and function
  - Any two species  $X_1$ ,  $X_2$  have a **common ancestor A**
  - Any gene  $G$  from  $A$  will undergo **independent evolution** in  $X_1$  and  $X_2$ , leading to genes  $G_1$  and  $G_2$
  - The more similar  $G_1$  and  $G_2$  are, the more likely do they still have the **same function** (that of  $G$ )
  - For any two genes of non-trivial length, the chance that they have a very similar sequence **by chance** is extremely small
  - **Corollary:** If two genes  $G_1$  and  $G_2$  today are very similar, they most likely derive from the **same ancestor** and most likely have the **same function**
  - How can we quantify this?



# Basic Evolutionary Events

---

- The simplest model: Single bases can be **replaced (R)**, **inserted (I)**, or **deleted (D)** (or kept (M))
- Any changes must be explained by sequences of I, D, R
  - I.e., by singular evolutionary events accumulating over time
  - We call this an **edit script**
- Very simple yet quite powerful model
- One more simplification

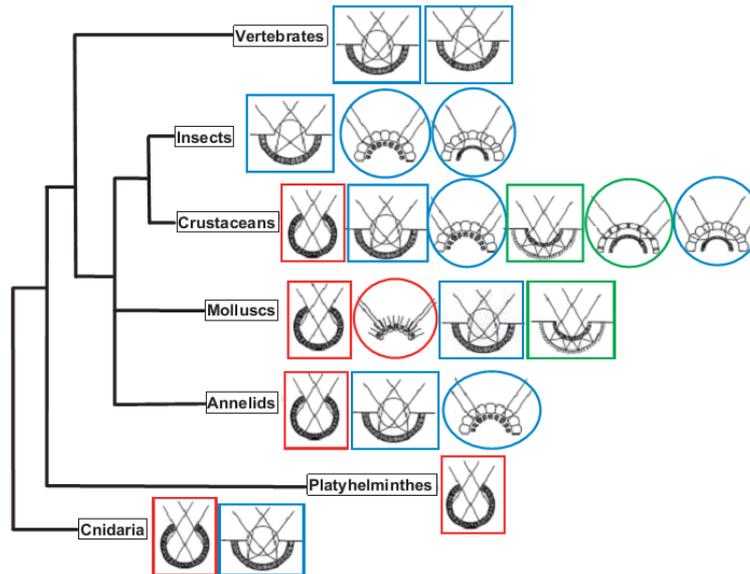
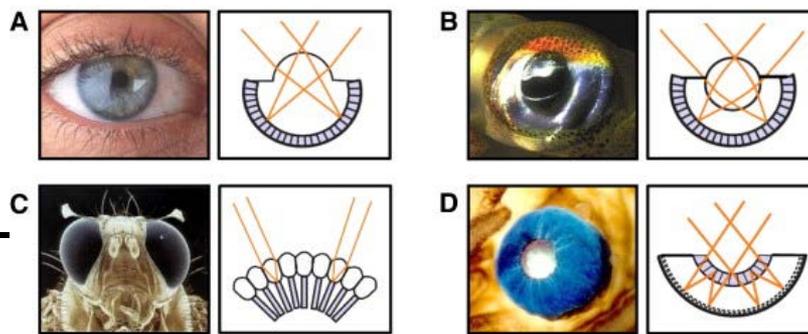


# Example: Eyeless (ey)

---

- **Family of genes** identified first in *Drosophila*
- When activated in arbitrary cells, non functional eyes start to grow at various places of the body
- *ey* is a “master gene” – controls a **cascade of activations** of other genes eventually leading to eye development
- Also inflicted with several other neural developments

# Eyes



Red: Only shadow

Blue: Lenses etc.

Green: Mirrors

Oval: Compound eyes

Rectangle: Single chamber

Source: Treisman (2004).

- Eyes probably are an example of **convergent evolution**
- However, genes controlling eye development are highly conserved across a **wide range of species**

# Homologues of "eyeless isoform D" (DM)



MFTLQPTPTAIGTVVPPWSAGTLIERLPSLEDMAHKDNVIAMRNLPCLTAGGSGLG  
GIAGKPSPTMEAVEASTASHPHSTSSYFATYYHLTDDECHSGVNQLGGVFGGRPL  
PDSTROKIVELAHSGARPCDISRILOQVSNQCVSKILGRYYETGSIRPRAIGGSKPRVAT  
AEVVKISIQYKRECPISIFAWEIRDRLLENVCTNDNIPSVSSINRVLRLNLAQKEQQST  
GSGSSSTSAGNSISAKVSVSIGGNVSNVAGSRGTLSSSTDLMTATPLNSESSEGGAS  
NSGEGSEQEAIYEKLRLLNTQHAAGPGPLEPARAAPLVGQSPNHLGTRSSHQPVLVHG  
NHOALQQHQQQSWPPRHYSWYPTSLSEIPISSAPNIASVTAYASGPLAHSLSP  
NDIESLASIGHQRNCPVATEDIHLKKELDGHQSDETGSGEGENSNGGASNIGNTEDD  
QARLILKRKLQRNRTSFTNDQIDSLEKEFERHTHYDPVFARERLAGKIGLPEARIQVWF  
NRRAKWRREEKLRNQRRTPNSTGASATSSSTSATASLTDSPNLSACSSLLSGSAGG  
PSVSTINGLSSPSTLSTNVNAPTLGAGIDSESPTPIPIRPSCTSDNDNGROSEDCRR  
VCSPCPLGVGGHONTHHIQSNQHAQGHALVPAISPRLNFSFGAMYSNMHHTAL  
SMSDSYGAVTPIPSFNHSAVGLAPPSPIQOQDLTPSSLYPCHMTLRPPMAPAHHH  
IVPGDGGRPAGVGLGSGQSANLGASCSGSGYEVLSAYALPPPMASSAADSSFSAA  
SASANVTPHHTIAQESCPSPCSSASHFGVAHSSGFSSDPISPAVS...

- 250 most similar protein sequences in UniProt

- Sequence identities all >50%,
- All p-Values < 1E-50

# This Lecture

---

- Approximate String Matching
- Edit distance and alignment
- Computing global alignments
- Local alignment



# Alignment

---

- Edit scripts are intuitive from an evolutionary point-of-view, but somewhat clumsy from a computational point-of-view
- Definition
  - A *(global) alignment* of strings  $A$ ,  $B$  is an arrangement of  $A$  and  $B$ , enriched with „\_“ at arbitrary positions, under each other such that no column contains two „\_“
  - The *score of an alignment* is the number of „\_“ plus the number of mismatching columns it contains
  - The *alignment distance* between  $A$  and  $B$  is the minimal score of any alignment of  $A$  and  $B$
- Edit distance and alignment distance are essentially identical
- Examples

– A\_TGT\_A  
AGTGTC\_

A\_T\_GTA  
\_AGTGTC

\_AGAGAG  
GAGAGA\_

AGAGAG\_  
\_GAGAGA

**Score: 3**

**5**

**2**

**2**

# A Visual Approach: Dotplots

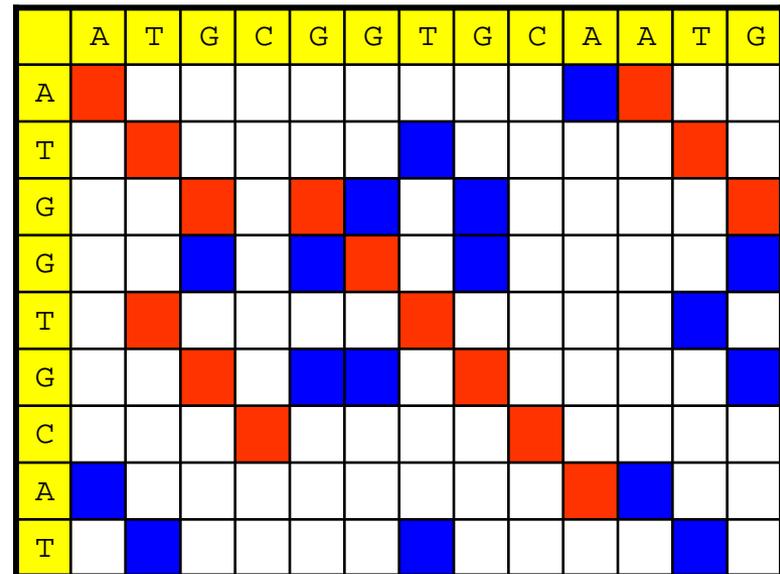
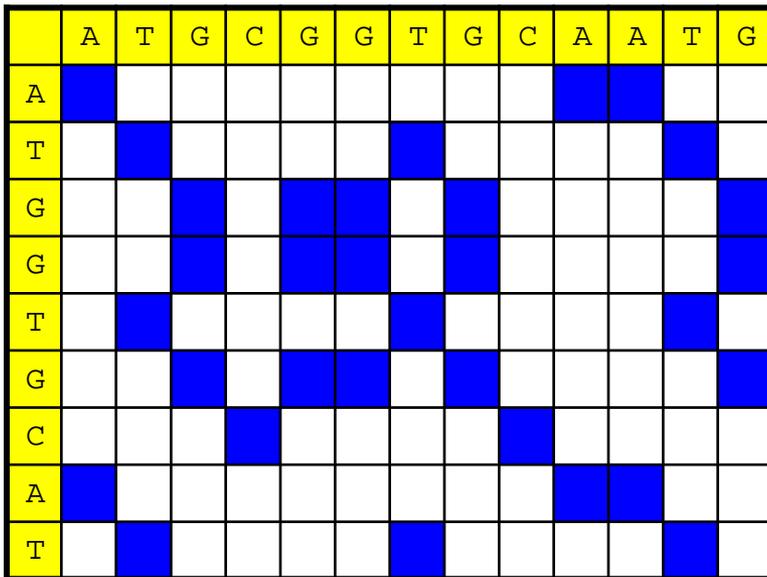
---

- A *dotplot* of two strings  $A$ ,  $B$  is a matrix  $M$  with
  - The  $i$ 'th character in  $A$  is represented by the  $i$ 'th column
  - The  $j$ 'th character in  $B$  is represented by the  $j$ 'th row
  - $M[i,j]=1$  (blue) iff  $A[i] = B[j]$

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

# Dotplot and Identical Substrings

- How do identical substrings look like in a dotplot?

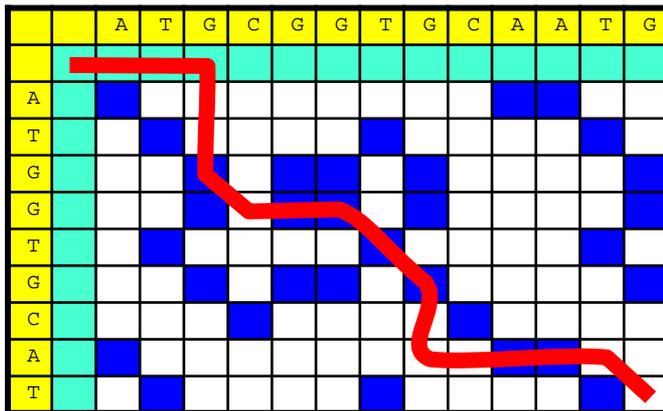


- Diagonals from up-left to down-right
  - Longest diagonal is the longest common substring

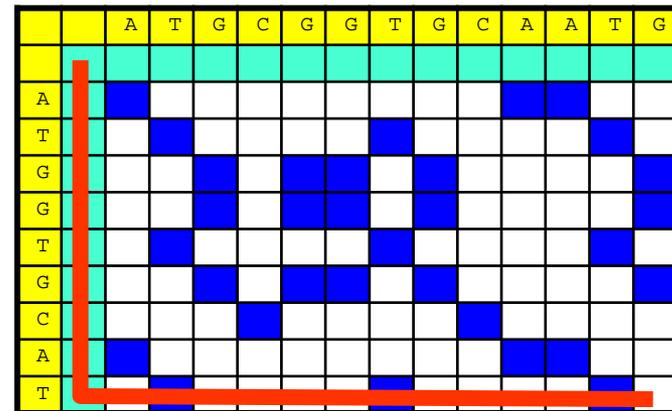
# Alignments and Dotplots

- Every alignment of A, B can be **uniquely mapped into a path** through M
  - The path starts in the upper-left corner (coord: 0,0)
  - Go through the alignment column by column
  - Next column is "X,\_" – move to the right
  - Next column is "\_ , X" – move down
  - Next column is "X, Y" – move right-down

ATG\_\_CGGTG\_\_CAATG  
 \_\_ATGG\_\_TGCA\_\_T

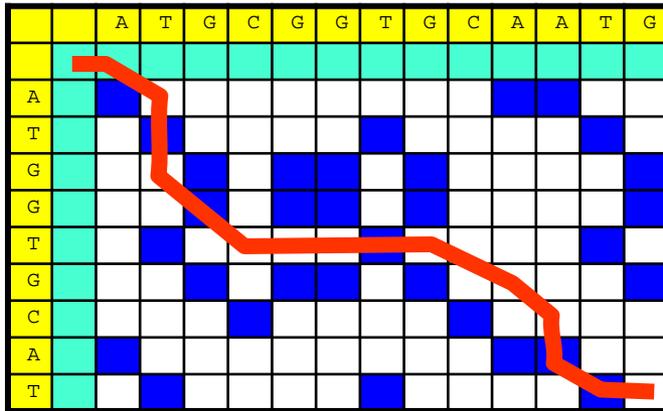


\_\_\_\_\_ATGCGGTGCAATG  
 ATGGTGCCAT\_\_\_\_\_

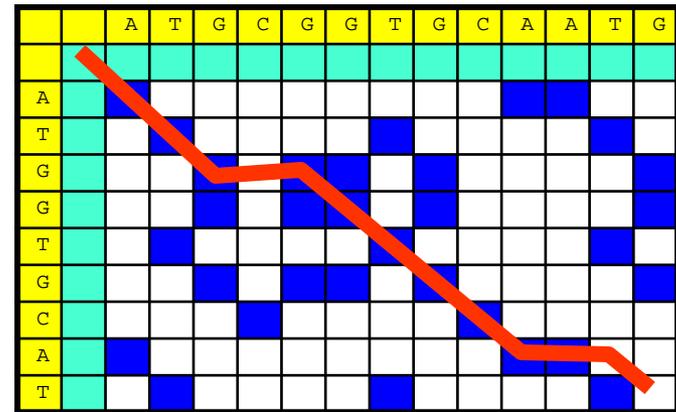


# Examples

AT\_\_GCGGTGCAA\_TG  
\_ATGGT\_\_\_\_GCAT\_\_



ATGCGGTGCAATG  
ATG\_\_GTGCA\_\_T



- Clearly, the number  $c(P)$  of 1's crossed in a diagonal step by a path  $P$  is the same as  $|P| - e(A, B)$
- Finding the path that **minimizes**  $|P| - c(P)$  also solves the problem of computing the edit distance

# This Lecture

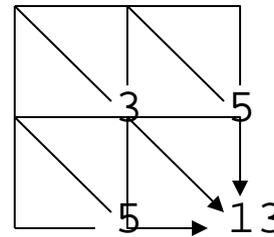
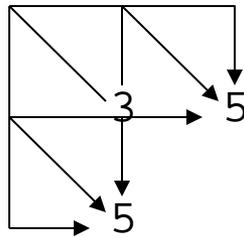
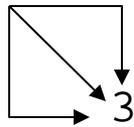
---

- Approximate String Matching
- Edit distance and alignment
- Computing global alignments
- Local alignment

# Algorithm

---

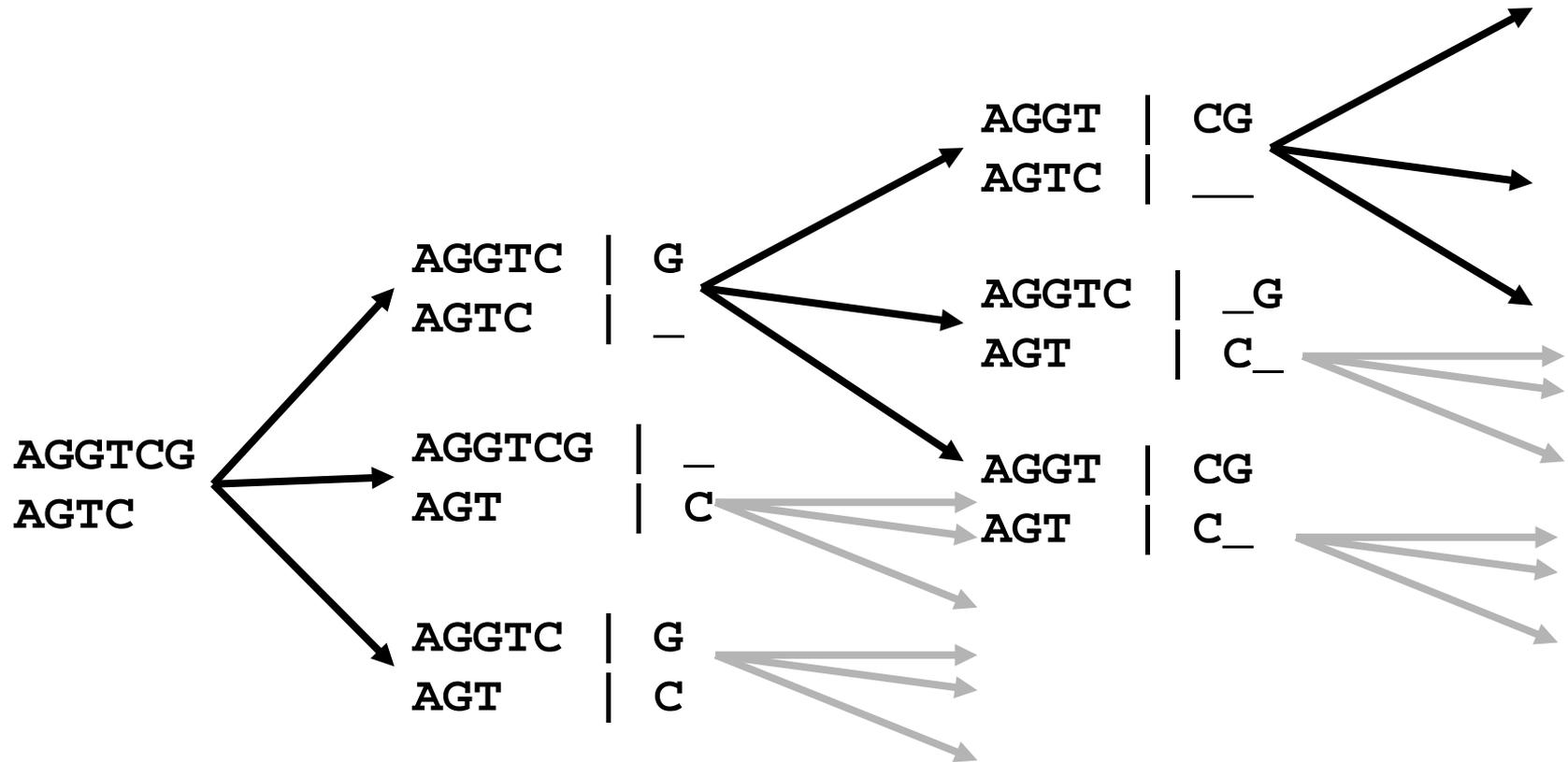
- How do we **compute** the edit distance of two strings?
- Naive: Enumerate all paths, compute  $c(P)$  for each



- Bad news: There exist  $> 3^{\min(m,n)}$  paths
- Good news: We can compute  $e(A,B)$  with  $\sim 3 * m * n$  operations

# Enumerating all Paths Recursively

---



# The naïve (recursive) Way

---

- Observation

- Let  $|A|=n$ ,  $|B|=m$
- Let  $d(i,j)=e(A[...i], B[...j])$  for  $0 \leq i \leq n$  and  $0 \leq j \leq m$  with  $d(i, 0)=i$  and  $d(0,j)=j$
- We can compute  $e(A,B) = d(n,m)$  recursively as follows

$$d(i, j) = \min \begin{cases} d(i, j-1) + 1 \\ d(i-1, j) + 1 \\ d(i-1, j-1) + t(i, j) \end{cases}$$

$$t(i, j) = \begin{cases} 1 : \text{if } A[i] \neq B[j] \\ 0 : \text{else} \end{cases}$$

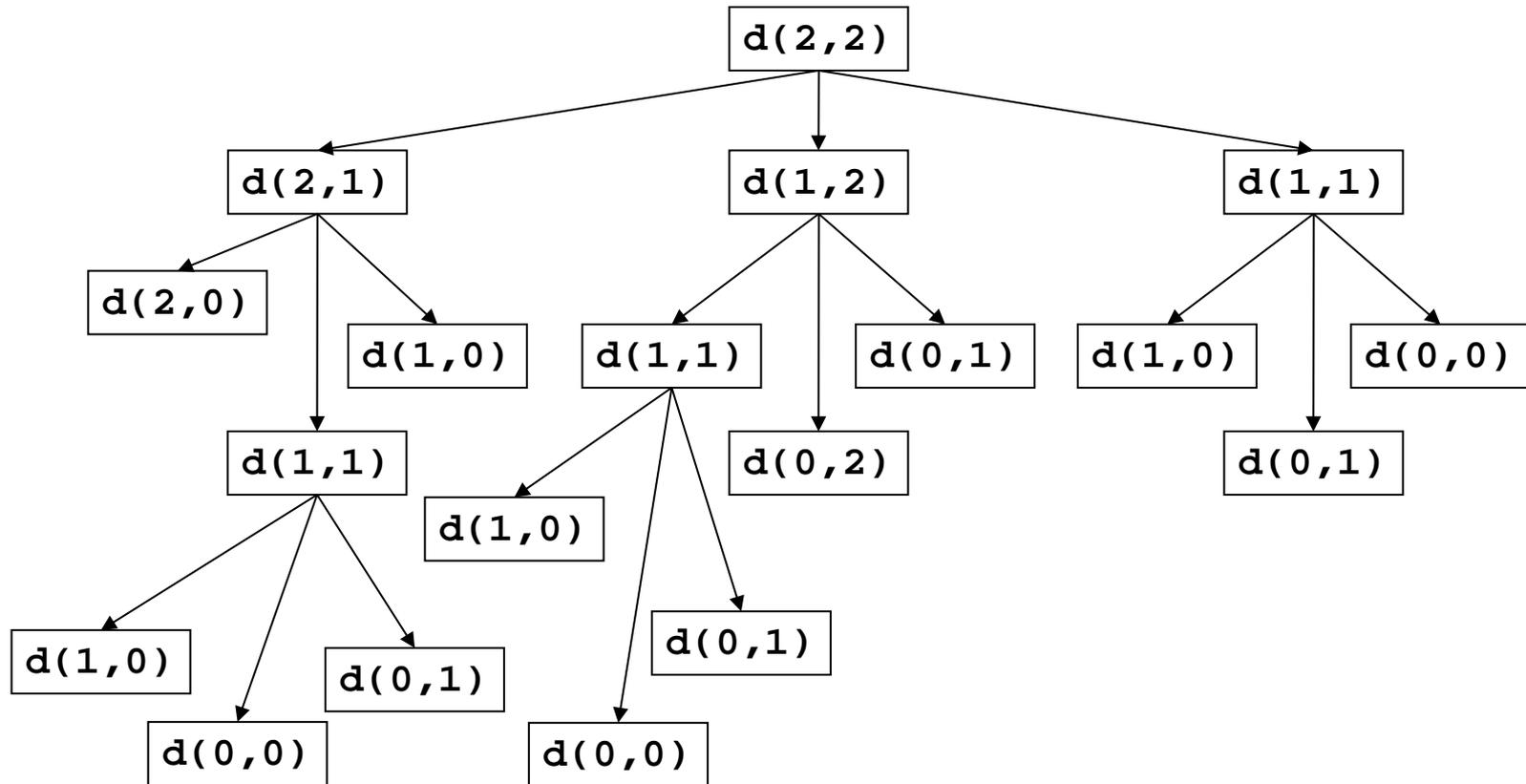
# Algorithm

---

```
function d(i,j) {
    if (i = 0)          return j;
    else if (j = 0)    return i;
    else
        return min ( d(i,j-1) + 1,
                     d(i-1,j) + 1,
                     d(i-1,j-1) + t(A[i],B[j]));
}
function t(c1, c2) {
    if (c1 = c2)      return 0;
    else                return 1;
}
```

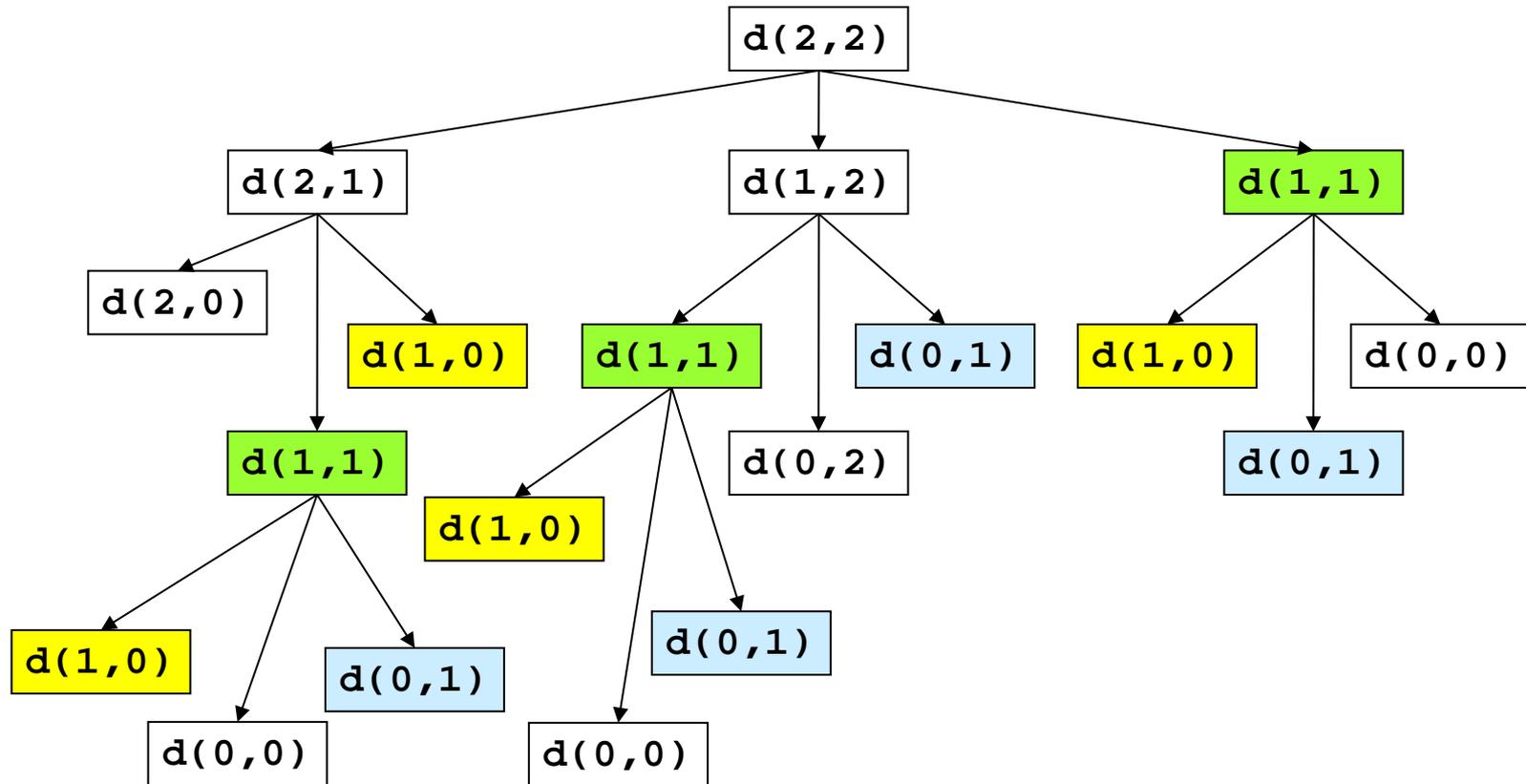
# What is Happening?

---



# Much Redundant Computation

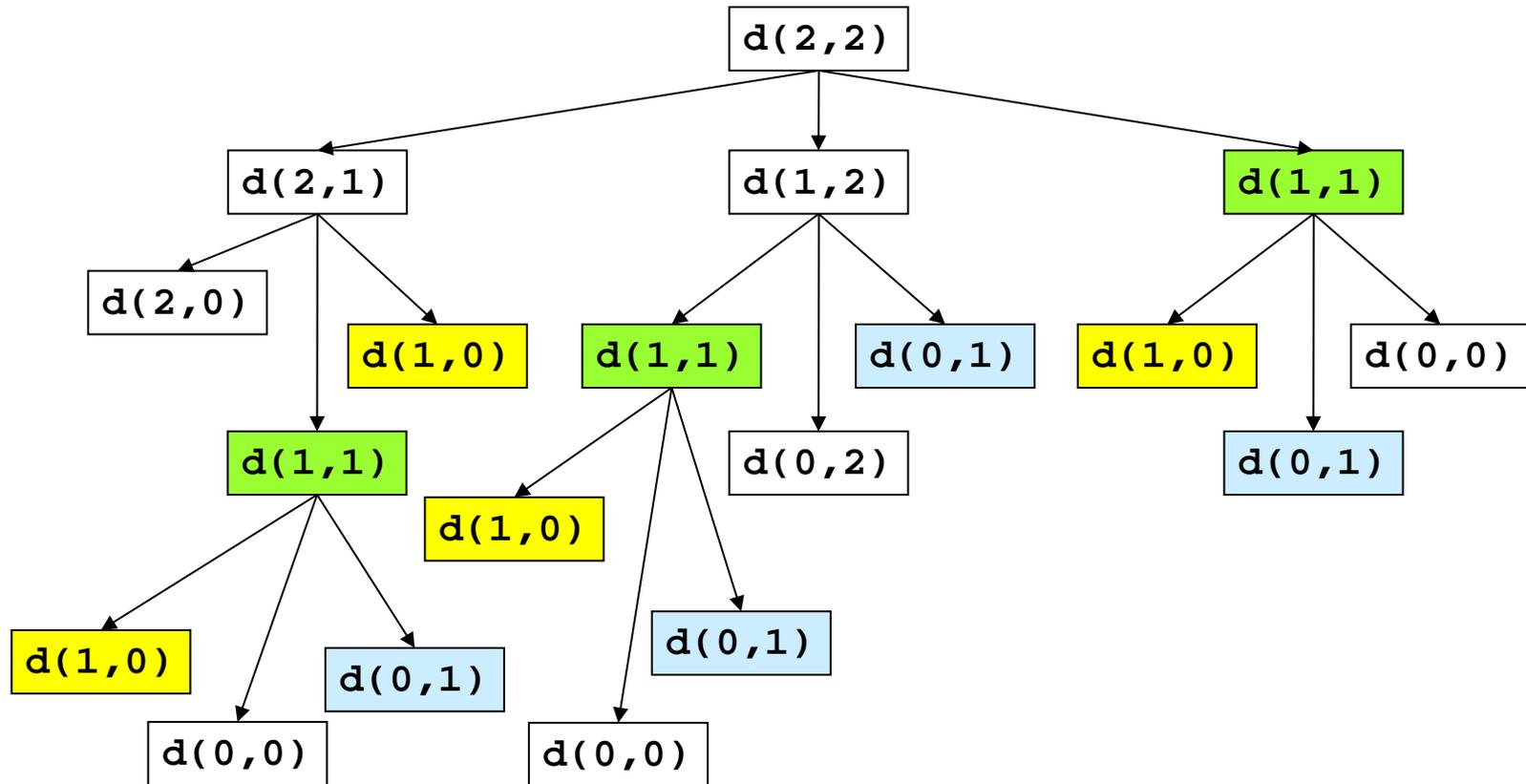
---



There are only  $\sim n*m$  different parameter combinations

# Much Redundant Computation

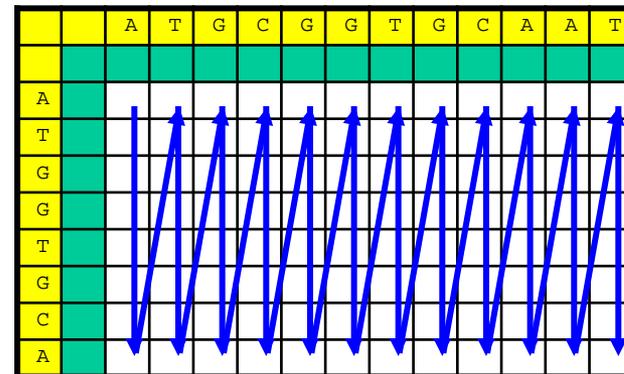
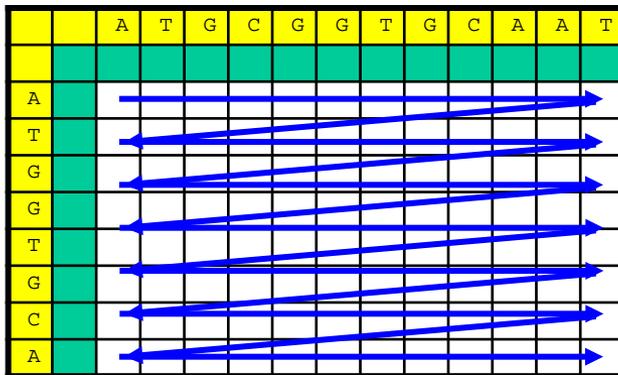
---



There are only  $\sim n*m$  different parameter combinations

# Dynamic Programming – Using a Table

- Instead of computing top-down (from  $n,m$ ), we compute all different values for  $d(i,j)$  **bottom-up**
  - We store all values in a table
- We can immediately “compute”  $d(i,0)$  and  $d(0,j)$
- Which values can we compute next?



# Example

$$d(i, j) = \min \left\{ \begin{array}{l} d(i, j-1) + 1 \\ d(i-1, j) + 1 \\ d(i-1, j-1) + t(i, j) \end{array} \right\}$$

		A	T	G	C	G	G	T
	0	1	2	3	4	5	6	7
A	1							
T	2							
G	3							
G	4							

		A	T	G	C	G	G	T
	0	1	2	3	4	5	6	7
A	1	0						
T	2							
G	3							
G	4							

		A	T	G	C	G	G	T
	0	1	2	3	4	5	6	7
A	1	0	1	2	3	4	5	6
T	2							
G	3							
G	4							

		A	T	G	C	G	G	T
	0	1	2	3	4	5	6	7
A	1	0	1	2	3	4	5	6
T	2	1	0	1	2	3	4	5
G	3							
G	4							

		A	T	G	C	G	G	T
	0	1	2	3	4	5	6	7
A	1	0	1	2	3	4	5	6
T	2	1	0	1	2	3	4	5
G	3	2	1	0	1	2	3	4
G	4							

		A	T	G	C	G	G	T
	0	1	2	3	4	5	6	7
A	1	0	1	2	3	4	5	6
T	2	1	0	1	2	3	4	5
G	3	2	1	0	1	2	3	4
G	4	3	2	1	1	1	2	3

# Finding the (an) optimal Alignment(s)

- Traceback
  - We find the path from back to front
  - Start at cell (n,m)
  - See which cells were used to compute  $d(n,m)$
  - Walk any of these – finds one **optimal path**
  - Walking all means finding all optimal paths
- Alternative: Store **pointers** while filling the table

		A	T	G	C	G	G	T
	0	1	2	3	4	5	6	7
A	1	0	1	2	3	4	5	6
T	2	1	0	1	2	3	4	5
G	3	2	1	0	1	2	3	4
G	4	3	2	1	1	1	2	3

		A	T	G	C	G	G	T
	0	1	2	3	4	5	6	7
A	1	0	1	2	3	4	5	6
T	2	1	0	1	2	3	4	5
G	3	2	1	0	1	2	3	4
G	4	3	2	1	1	1	2	3

		A	T	G	C	G	G	T
	0	1	2	3	4	5	6	7
A	1	0	1	2	3	4	5	6
T	2	1	0	1	2	3	4	5
G	3	2	1	0	1	2	3	4
G	4	3	2	1	1	1	2	3

# Complexity

---

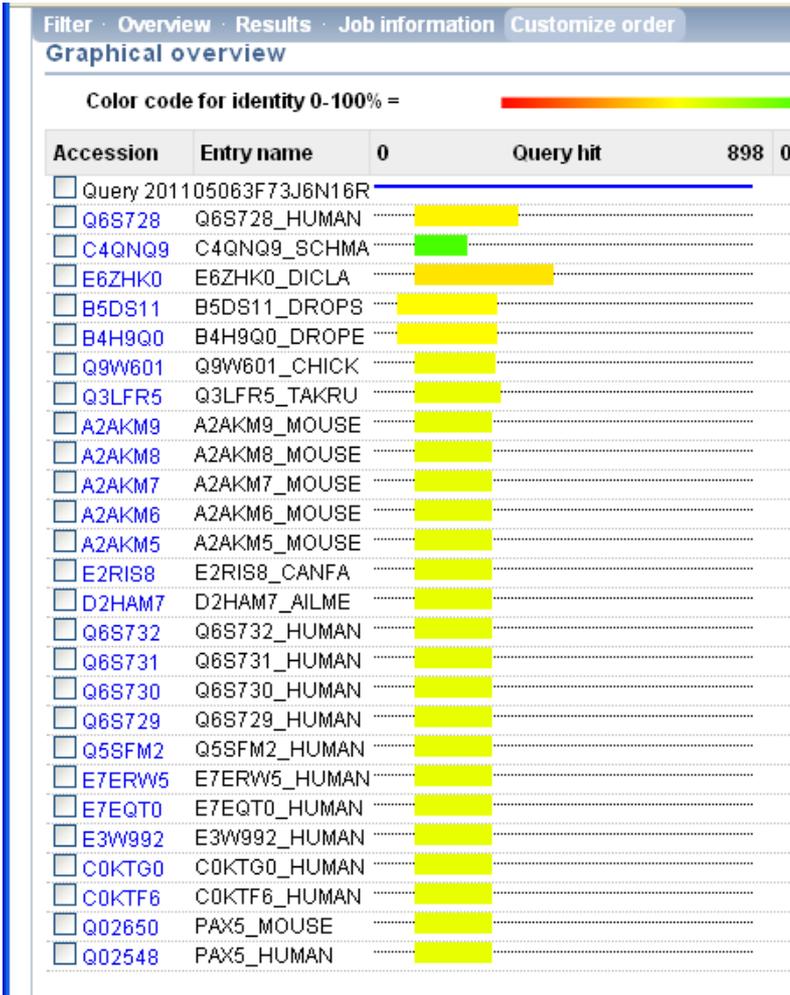
- Building the table
  - For every  $d(i,j)$ , we need to access three other cells and make some (constantly many) additions and comparisons
  - There are  $(m+1)*(n+1)$  cells
  - Thus:  $\sim 3*m*n = O(m*n)$  operations
- Finding **one optimal** alignment
  - We must walk from  $(n,m)$  to  $(1,1)$
  - Such a path can have at most length  $m+n$ 
    - We cannot go wrong!
  - Together: approximately  **$m+n$  operations**
- Together:  **$O(m*n)$**  (for  $m*n > m+n$ )

# This Lecture

---

- Approximate String Matching
- Edit distance and alignment
- Computing global alignments
- Local alignment

# Eyeless Again – a Closer Look



- The **similar regions** in the different homologues are not distributed randomly
- Actually, a single stretch of 128 AA, the **PAX domain**, is virtually unchanged in all homologues
  - Controls binding to DNA and hence regulatory effects
- Typical: Only some **parts of a gene are conserved**, and these carry function

# Example

---

ACCCTATCGATAGCTAGGAAGCTCGAAAATACCGACCAGTAT  
AGGAGTCGATAATACATATAAGAGATAGAATATATTGATG

Zufall?

ACCCTATCTATA--GC-TAGGAAGCTCGATAATACCGACCAGTAT-  
|           ||   ||           |   ||   ||           ||   |   |           |           |   ||  
A-GGAGTCGATCATACATATAAG-A-GATAGAATATA-TTG-ACG

Kein Zufall!

ACCCTATCGATAGCTAGGAAGCTCGAAAATACCGACCAGTAT  
                  |   |   |   |   |   |   |  
AGGAGTCGATAATACATATAAGAGATAGAATATATTGATG

# Distance or Similarity

---

- Until now, we computed a **global distance**
  - The higher  $e(A,B)$ , the less similar are A and B
  - The **longer A and B**, the higher their distance in general
  - **Different lengths** are punished:  $e(A,B) \geq ||A|-|B||$
- Often, we want a **local similarity** instead
  - If we don't compare two entities, but two strings presumably **containing each one entity**
  - If we have a sequence and don't know exactly where the genes are
  - If a function is associated to a motif in a protein
- We need to search for **substrings**  $A' \in A$ ,  $B' \in B$  which are very similar to each other
  - $A'$  and  $B'$  should have a certain length to be interesting
  - $e(A',B')$  does not help – optimal distance is 0 for  $A'=B'=""$

# Sequence Similarity

---

- A *scoring function* is a function  $s: \Sigma \times \Sigma \rightarrow \text{Integer}$ 
  - We also call  $s$  a *substitution matrix*
- The *ungapped similarity*  $sim'$  of  $A, B$  wrt.  $s$  with  $|A|=|B|=n$  is defined as

$$sim'(A, B) = \sum_{i=1}^n s(A[i], B[i])$$

- The *similarity*  $sim$  of  $A, B$  (wrt.  $s$ ) is the highest ungapped similarity score *over all alignments of  $A$  and  $B$* 
  - Higher = better; maximal similarity is  $n \cdot \max(s)$
- We are not yet there: This still is a global similarity score

# Example

---

$$\Sigma' = \{A, C, G, T, _\}$$

	A	C	G	T	_
A	4	-2	-2	-1	-3
C		4	-1	-2	-3
G			4	-2	-3
T				4	-3

$$\begin{array}{l} \text{AC\_GTC} \\ \text{AGGT\_C} \end{array} = -1$$

$$\begin{array}{l} \text{ACGTC} \\ \text{AGGTC} \end{array} = 15$$

$$\begin{array}{l} \text{A\_CGTC} \\ \text{AG\_GTC} \end{array} = 10$$

# Computing $\text{sim}(A, B)$

---

- Same ideas as for edit distance
- But: We want a **high similarity**, not a low distance
- Thus, we can compute  $\text{sim}(|A|, |B|)$  with

$$\text{sim}(i, 0) = \sum_{k=1..i} s(A[k], \_) \quad \text{sim}(0, j) = \sum_{k=1..j} s(\_, B[k])$$

$$\text{sim}(i, j) = \max \begin{cases} \text{sim}(i, j - 1) + s(\_, B[j]) \\ \text{sim}(i - 1, j) + s(A[i], \_) \\ \text{sim}(i - 1, j - 1) + s(A[i], B[j]) \end{cases}$$

# Example

	A	G	T	C
A	4	-1	-1	-1
G		4	-1	-1
T			4	-1
C				4
-	-3	-3	-3	-3

## Edit Distance

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

## Similarity

		A	G	G	T	C
	0	-3	-6	-9	-12	-15
A	-3	4	1	-2	-5	-8
G	-6	1	8	5		
T	-9					
C	-12					
C	-15					

# Lokal Similarity = Local Alignment

---

- Definition

- The *local similarity score*  $sim^*$  of  $A, B$  is defined as

$$sim^*(A, B) = \max_{\forall A' \text{ substringOf } A, B' \text{ substringOf } B} (sim(A', B'))$$

- Remark

- Inequality in length of  $A$  and  $B$  does not matter any more
- Sounds terribly complex, but there is a *neat trick*

ACCCTATCGATAGCTAGAAAGCTCGAAAATACCGACCAGTAT

| | | | | | |

AGGAGTCGATAATAACATATAAGAGATAGAATATATTGATG

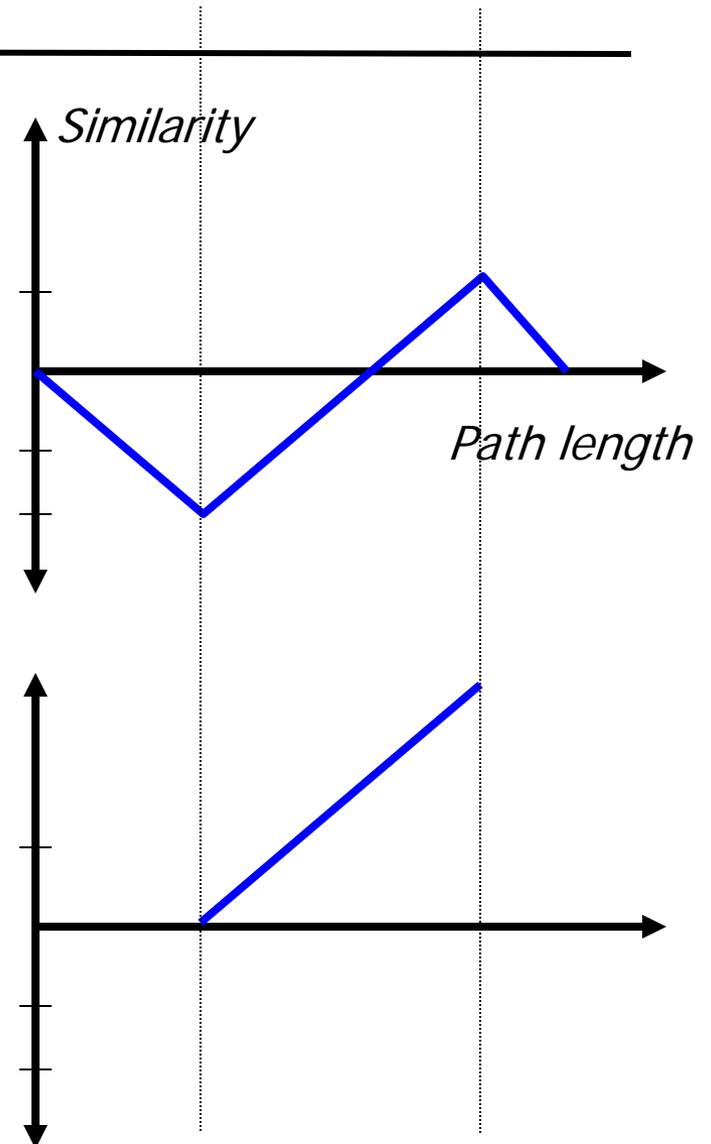
# Example

Match: +1

I/R/D: -1

		A	T	G	T	G	G
	0	-1	-2	-3	-4	-5	-6
G				-1			
T					0		
G						1	
A							0

		A	T	G	T	G	G
	0	0	0				
G				1			
T					2		
G						3	
A							2



# Smith-Waterman Algorithm

---

- Smith, Waterman: „Identification of common molecular subsequences“, J. Mol. Bio 147, 1981
- Idea
  - Note: **Local paths** need not span the entire strings
  - Look at a single path
  - A series of matches (positive values for scoring function  $s$ ) creates a **series of increasing similarity values**
  - Any step with  $s < 0$  lowers the score
  - Whenever the cumulative score falls below 0, we drop this prefix
  - Instead of carrying on, we conceptually **start a new local path**
  - To this end, we simply set  $\text{true\_score} = \max(0, \text{score})$
  - The **highest value in the matrix** is the end of the best local path

# Computation

---

- The same ideas as before
- We compute  $\text{sim}^*(A,B)$  with
  - Assume  $\forall X: s(X, \_) < 0$  and  $s(\_, X) < 0$

$$\text{sim}(i, 0) = \sum_{k=1..i} s(A[k], \_) \quad \text{sim}(0, j) = \sum_{k=1..j} s(\_, B[k])$$

$$\text{sim}(i, j) = \max \begin{cases} \text{sim}(i, j - 1) + s(\_, B[j]) \\ \text{sim}(i - 1, j) + s(A[i], \_) \\ \text{sim}(i - 1, j - 1) + s(A[i], B[j]) \\ 0 \end{cases}$$

# Example

Match: +1

I/R/D: -1

		A	T	G	T	C	G
	0	-1	-2	-3	-4	-5	-6
A	-1	1	0	-1	-2	-3	-4
T	-2	0	2	1	0	-1	-2
G	-3	-1	1	3	2	1	0

ATGTCG

ATG\_\_\_\_

ATGTCG

AT\_\_G

ATGTCG

A\_\_T\_G

		A	T	G	T	C	G
	0	0	0	0	0	0	0
A	0	1	0	0	0	0	0
T	0	0	2	1	1	0	0
G	0	0	1	3	2	1	1

ATGTCG

ATG\_\_\_\_

# Local versus global Alignment

---

- Global Alignment
  - Comparison of two entire sequences
  - Use when you think the entire sequences are related
  - Interest: The differences; assumption: Relatedness
  - Example: Proteins of the same family
- Local Alignment
  - Compare uncharacterized sequences
  - Use when comparing “randomly sampled” sequences
  - Interest: Similar regions; assumptions: None
  - Often a first step before global alignment
  - Example: Find similar genes in other species genomes

# Beware: Not all Events are Equal

**Wildtype**

CTTAGTGACTACGGTAAA

DNA

Leu Ser Asp Tyr Gly Lys

Protein

**Probably fatal**

CTTAGTGACTAGGGTAAA

DNA

Leu Ser Asp **Stop-Codon**

Protein

**Probably fatal**

CTTAGTGAACTACGGTAAA

DNA

Leu Ser **His Asp Leu Thr**

Protein

**Neutral**

CTTAGCGACTACGGTAAA

DNA

Leu Ser Asp Tyr Gly Lys

Protein

**Functional**

CTTAGTGAAATACGGTAAA

DNA

Leu Ser **Glu** Tyr Gly Lys

Protein

# Further Reading

---

- Everywhere
- Relaxed: Christianini & Hahn, Chapter 3
- Step by step: Waack, Chapter 9