Sequence Alignment

Ulf Leser

This Lecture

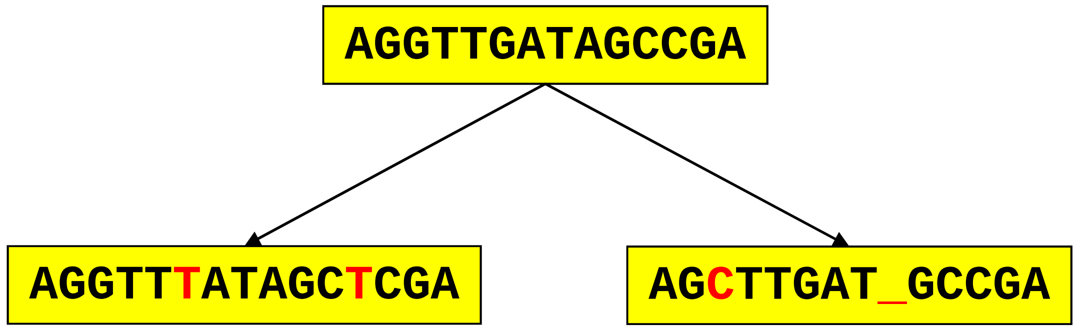
- Approximate String Matching
  - Edit distance and alignment
  - Computing a global alignment
  - Local alignment

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 hypothesis about their function

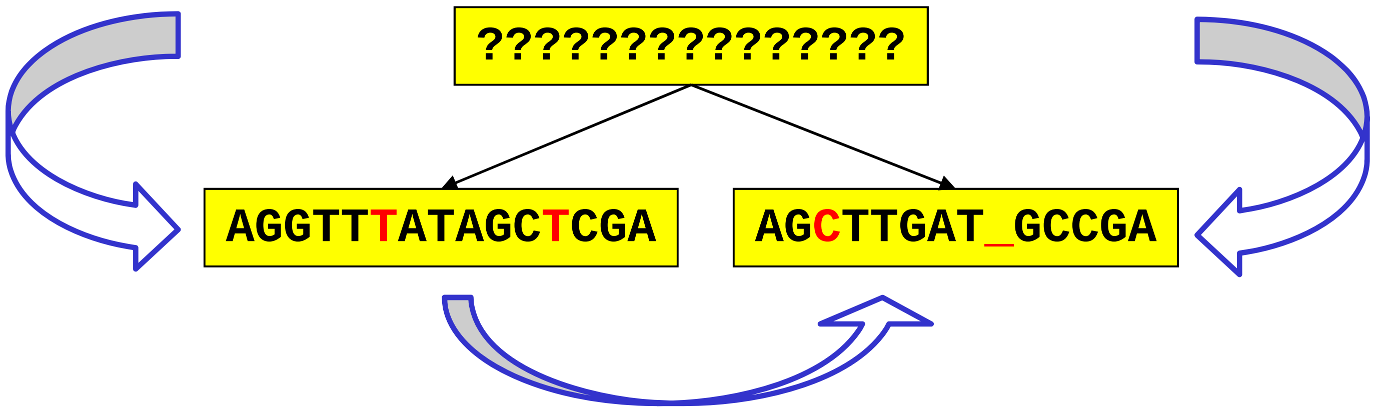
How Genes Evolve

- A simple model of gene evolution
  - 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)
  - How does evolution change gene sequences?



Basic Evolutionary Events

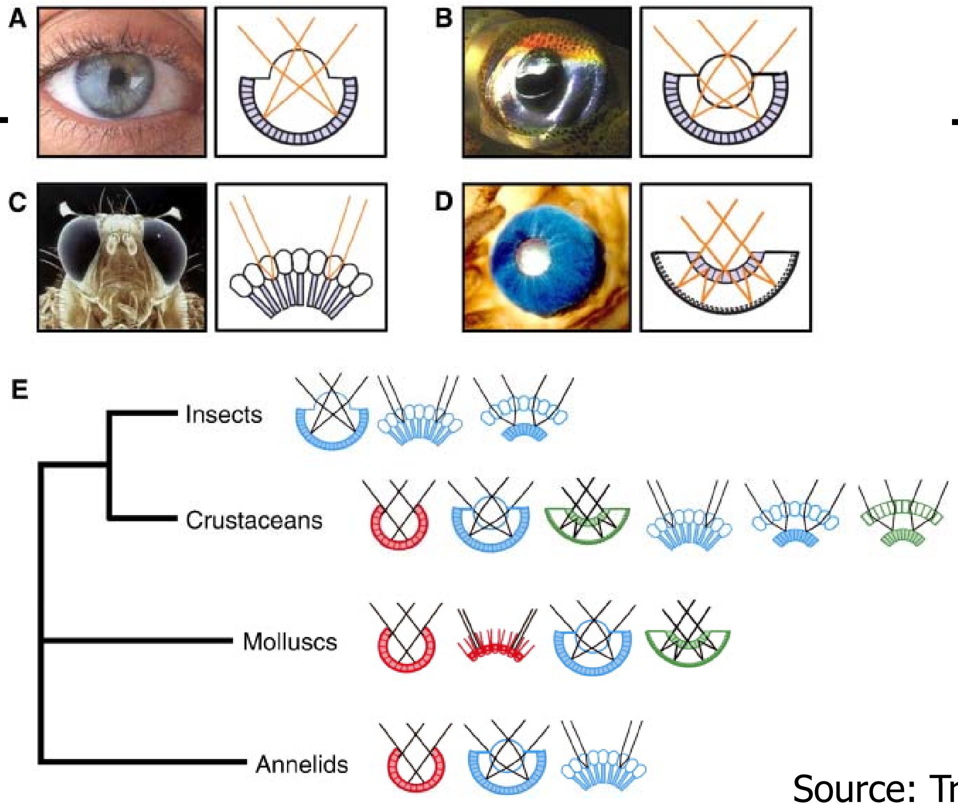
- The simplest model: Single bases can be replaced (R), inserted (I), or deleted (D) (or kept (M))
- Any changes must be explained by sequences of I, D, R
  - I.e., by single evolutionary events accumulating over time
  - We call this an edit script
- Very, very simplistic, but quite powerful model
- 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
- Gene is a “master” – controls a cascade of activations of other genes eventually leading to eye development
- Also inflicted with several other neural developments

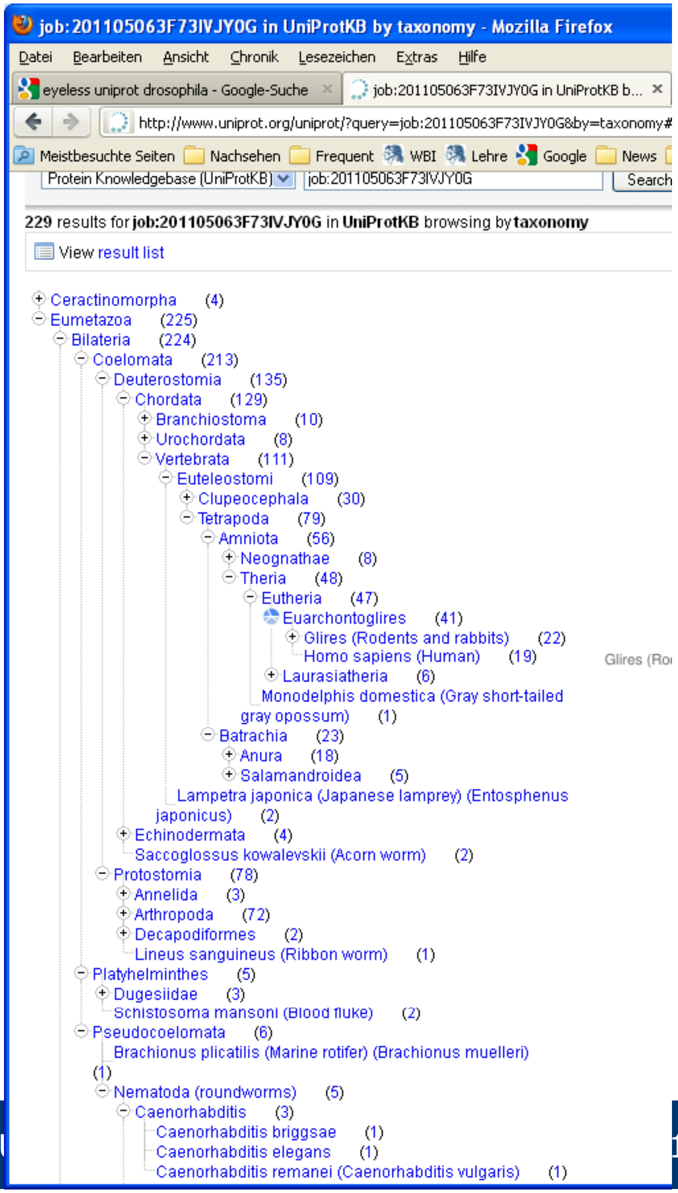
Eyes



Source: Treisman (2004).

- Eyes 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)


job:201105063F73IVJY0G in UniProtKB by taxonomy - Mozilla Firefox

Datei Bearbeiten Ansicht Chronik Lesezeichen Extras Hilfe

eyeless uniprot drosophila - Google-Suche | job:201105063F73IVJY0G in UniProtKB b...

http://www.uniprot.org/uniprot/?query=job:201105063F73IVJY0G&by=taxonomy#

Meistbesuchte Seiten | Nachsehen | Frequent | WBI | Lehre | Google | News

Protein Knowledgebase (UniProtKB) | job:201105063F73IVJY0G | Search

229 results for **job:201105063F73IVJY0G** in **UniProtKB** browsing by **taxonomy**

View result list

```
+ Ceractinomorpha (4)
- Eumetazoa (225)
  - Bilateria (224)
    - Coelomata (213)
      - Deuterostomia (135)
        - Chordata (129)
          + Branchiostoma (10)
          + Urochordata (8)
          - Vertebrata (111)
            - Euteleostomi (109)
              + Clupeocephala (30)
              - Tetrapoda (79)
                - Amniota (56)
                  + Neognathae (8)
                  - Theria (48)
                    - Eutheria (47)
                      Euarchontoglires (41)
                        + Glires (Rodents and rabbits) (22)
                          Homo sapiens (Human) (19)        Glires (Ro
                      + Laurasiatheria (6)
                      Monodelphis domestica (Gray short-tailed
                    gray opossum) (1)
                - Batrachia (23)
                  + Anura (18)
                  + Salamandroidea (5)
            Lampetra japonica (Japanese lamprey) (Entosphenus
          japonicus) (2)
        + Echinodermata (4)
          Saccoglossus kowalevskii (Acorn worm) (2)
      - Protostomia (78)
        + Annelida (3)
        + Arthropoda (72)
        + Decapodiformes (2)
          Lineus sanguineus (Ribbon worm) (1)
    - Platyhelminthes (5)
      + Dugesiidae (3)
        Schistosoma mansoni (Blood fluke) (2)
    - Pseudocoelomata (6)
        Brachionus plicatilis (Marine rotifer) (Brachionus muelleri)
      (1)
      - Nematoda (roundworms) (5)
        - Caenorhabditis (3)
            Caenorhabditis briggsae (1)
            Caenorhabditis elegans (1)
            Caenorhabditis remanei (Caenorhabditis vulgaris) (1)
```

- MFTLQPTPTAIGTVVPPWSAGTLIERLPSLEDMAHKDNVIAMRNLPCLGTAGG SGLGGIAGKPSPTMEAVEASTASHPHSTSSYFATTYYHLTDDECHSGVNQLGG VFVGGRPLPDSTRQKIVELAHSGARPCDISRILQVSNGCVSKILGRYYETGSIRP RAIGGSKPRVATAEVVSKISQYKRECPSIFAWEIRDRLLQENVCTNDNIPSVSSI NRVLRNLAAQKEQQSTGSGSSSTSAGNSISAKVSVSIGGNVSNVASGSRGTLS SSTDLMQTATPLNSSESGGASNSGEGSEQEAIYEKLRLLNTQHAAGPGPLEPA RAAPLVGQSPNHLGTRSSHPQLVHGNHQALQQHQQQSWPPRHYSGSWYPTS LSEIPISSAPNIASVTAYASGPSLAHSLSPPNDIESLASIGHQRNCPVATEDIHLK KELDGHQSDETGSGEGENSNGGASNIGNTEDDQARLILKRKLQRNRTSFTND QIDSLEKEFERTHYPDVFARERLAGKIGLPEARIQVWFSNRRAKWRREEKLRN QRRTPNSTGASATSSSTSATASLTDSPNSLSACSSLLSGSAGGPSVSTINGLSS PSTLSTNVNAPTLGAGIDSSESPTPIPHIRPSCTSDNDNGRQSEDCRRVCSPCP LGVGGHQNTHHIQSNGHAQGHALVPAISPRLNFNSGSFGAMYSNMHHTALS MSDSYGAVTPIPSFNHSAVGPLAPPSPIPQQGDLTPSSLYPCHMTLRPPPMAPA HHHIVPGDGGRPAGVGLGSGQSANLGASCSGSGYEVLSAYALPPPPMASSSAA DSSFSAASSASANVTPHHTIAQESCPSPCSSASHFGVAHSSGFSSDPISPAVS…
- 250 most similar protein sequences in UniProt
  - Sequence identities all >50%,
  - All p-Values < 1E-50

Edit Scripts and Edit Distances

- Definition
  - Let A, B $\in \Sigma^*$
  - *An edit script e is a sequence of operations I, D, R, M*
  - *e is an edit script for A and B iff e(A)=B*
    - *Slightly underdetermined – which replacement? Which base to insert?*
  - *The length of an edit script is the number of I,D,R it contains*
  - *The edit distance between A and B is the length of the shortest edit script for A and B*
- Remarks
  - If we know e(A)=B, determining e' with e'(B)=A is trivial
  - The shortest edit script is not unique, but its length is:
  -

```
MIMMMR                IRMMMDI
A_TGTA                _ATGTA_
AGTGTC                AGTGT_C
```

Alignment

- Edit scripts are intuitive from an evolutionary point-of-view, but not comfortable 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 identical
- Examples

–	**A_TGT_A**	**A_T_GTA**	**_AGAGAG**	**AGAGAG_**
	AGTGTC_	**_AGTGTC**	**GAGAGA_**	**_GAGAGA**
Score:	**3**	**5**	**2**	**2**

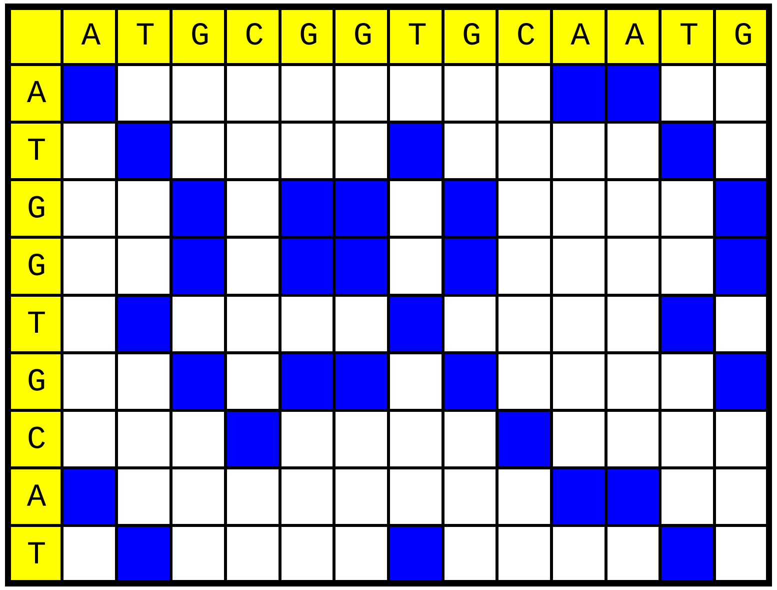
A Visual Approach: Dotplots

- *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]*

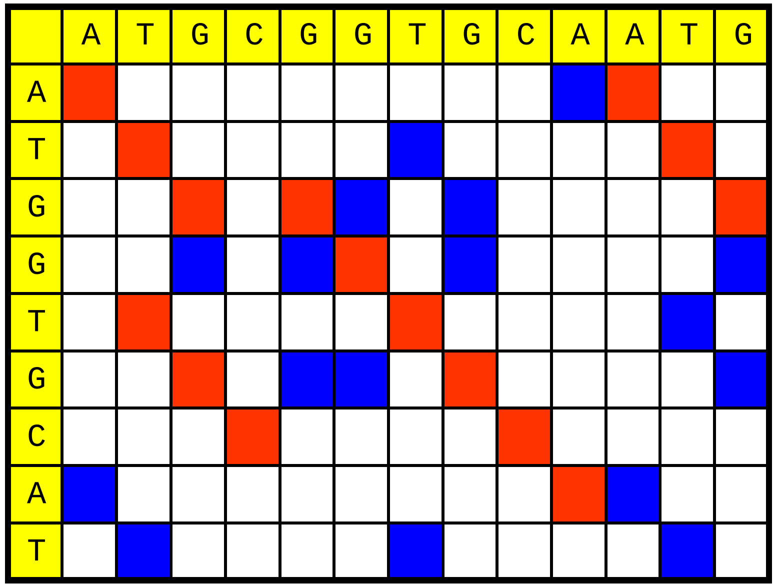
	A	T	G	C	G	G	T	G	C	A	A	T	G
A	🟦									🟦	🟦		
T		🟦					🟦					🟦	
G			🟦		🟦	🟦		🟦					🟦
G			🟦		🟦	🟦		🟦					🟦
T		🟦					🟦					🟦	
G			🟦		🟦	🟦		🟦					🟦
C				🟦					🟦				
A	🟦									🟦	🟦		
T		🟦					🟦					🟦	

Dotplot and Identical Substrings

- How do identical substrings look like in a dotplot?



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



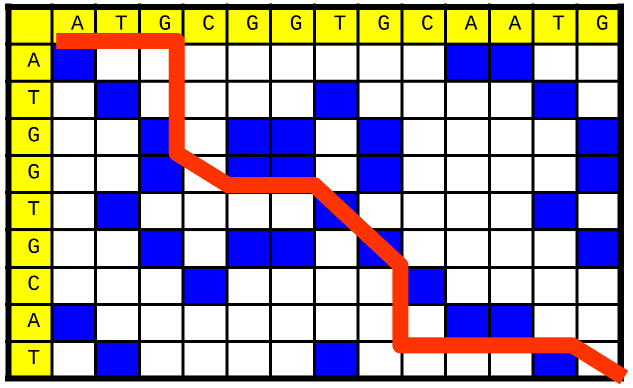
	A	T	G	C	G	G	T	G	C	A	A	T	G
A	🟥									🟦	🟥		
T		🟥					🟦					🟥	
G			🟥		🟥	🟦		🟦					🟥
G			🟦		🟦	🟥		🟦					🟦
T		🟥					🟥					🟦	
G			🟥		🟦	🟦		🟥					🟦
C				🟥					🟥				
A	🟦									🟥	🟦		
T		🟦					🟦					🟦	

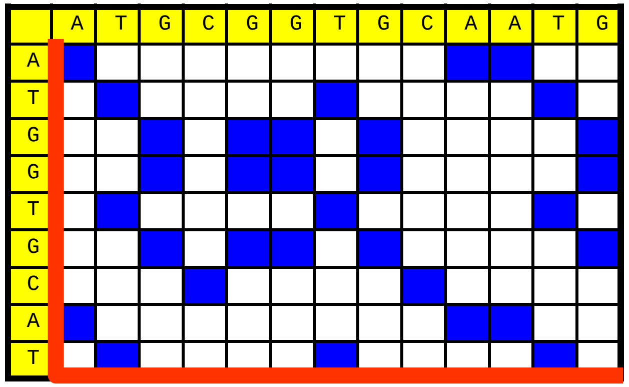
- Diagonals from up-left to down-right
  - Longest diagonal is the longest common substring

Alignments and Dotplots

- Let |A|=m, |B|=n and M be its dotplot matrix
- Every alignment of A, B can be uniquely mapped into a path through M
  - The path starts in the upper-left corner
  - Go through the alignment column by column
  - Next column is “X,_” – move to the left
  - Next column is “_, X” – move down
  - Next column is “X, Y” – move right-down

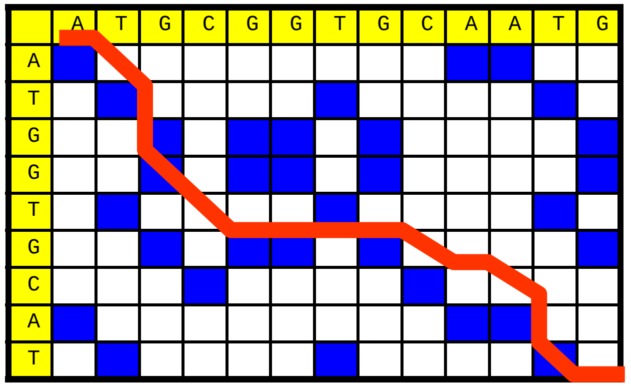
```
ATG___CGGTG__CAATG          _________ATGCGGTGCAATG
___ATGG__TGCA____T          ATGGTGCAT_____________
```



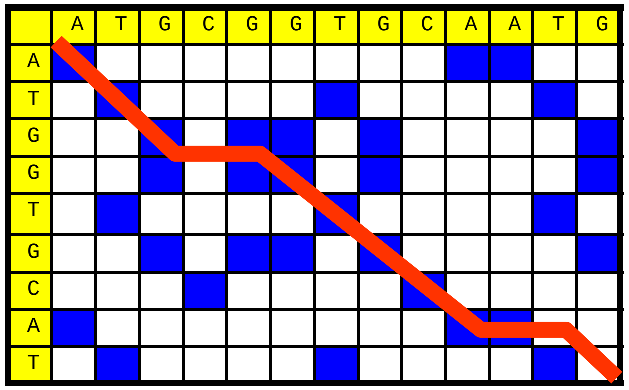


Examples

```
AT__GCGGTGCAA_TG
_ATGGT____G_CAT_
```



```
ATGCGGTGCAATG
ATG__GTGCA__T
```



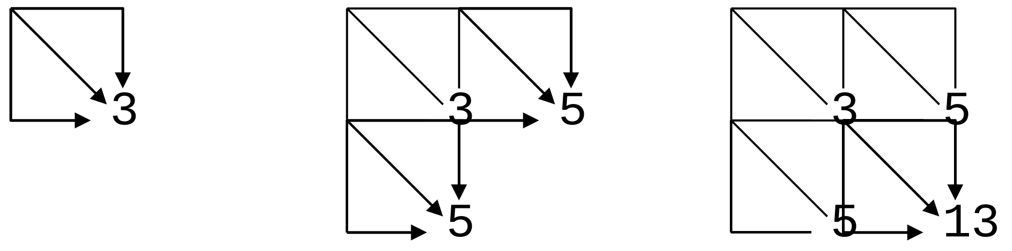
- Clearly, the number c(P) of 1's crossed by a path P is the same as |P|-e(A,B)
- Finding the path that minimizes |P|-c(P) also solves the problem of computing the edit distance

This Lecture

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

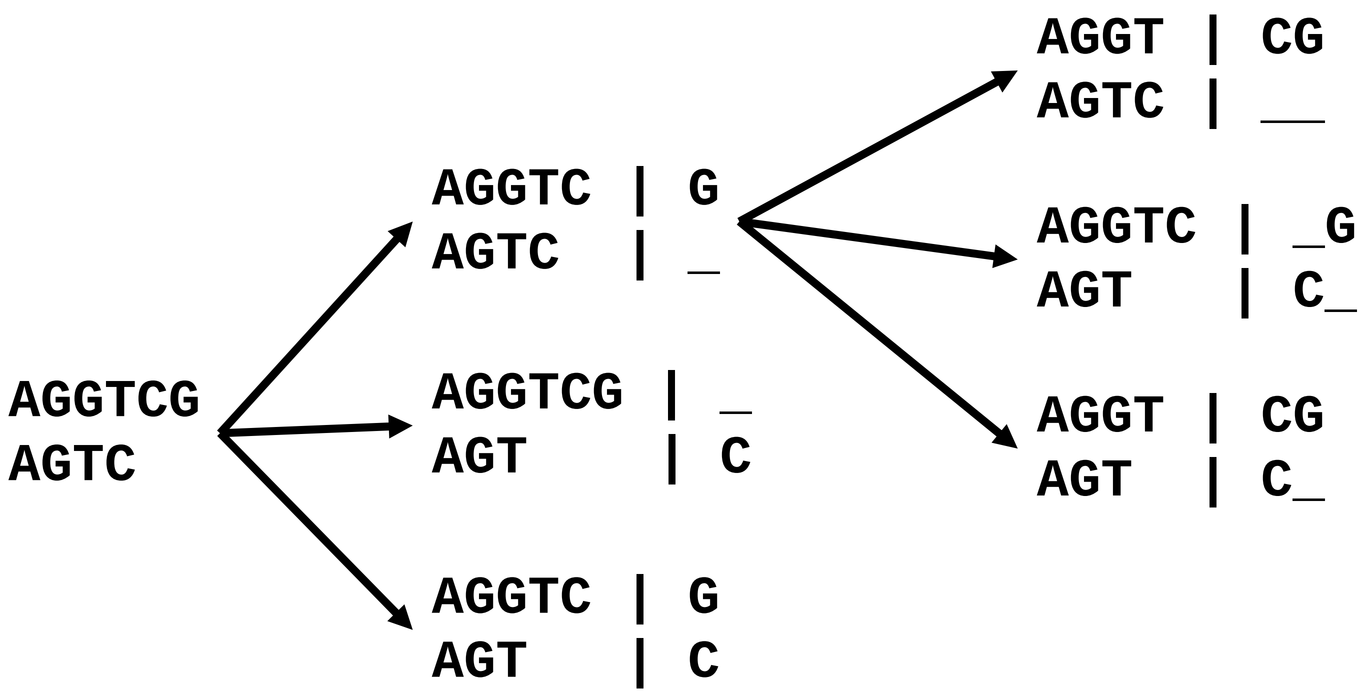
Algorithm

- How do we compute the edit distance of two strings?
- Naïve: 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 O(m*n) operations
  - Wait a second

Enumerating all Paths Recursively



- Extrem cases
  - d(i,0) = i: we need i operations to transform A[..i] into „"
  - d(0,j) = j: we need j operations to transform B[..i] into „"

The naïve (recursive) Way

- Observation
  - *Let |A|=n, |B|=m*
  - *Let d(i,j)=e(A[…i], B[…j]) for 0≤i≤n und 0≤j≤m with d(i, 0)=i and d(0,j)=j*
  - *We can compute e(A,B) = d(n,m) recursively as follows*

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

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

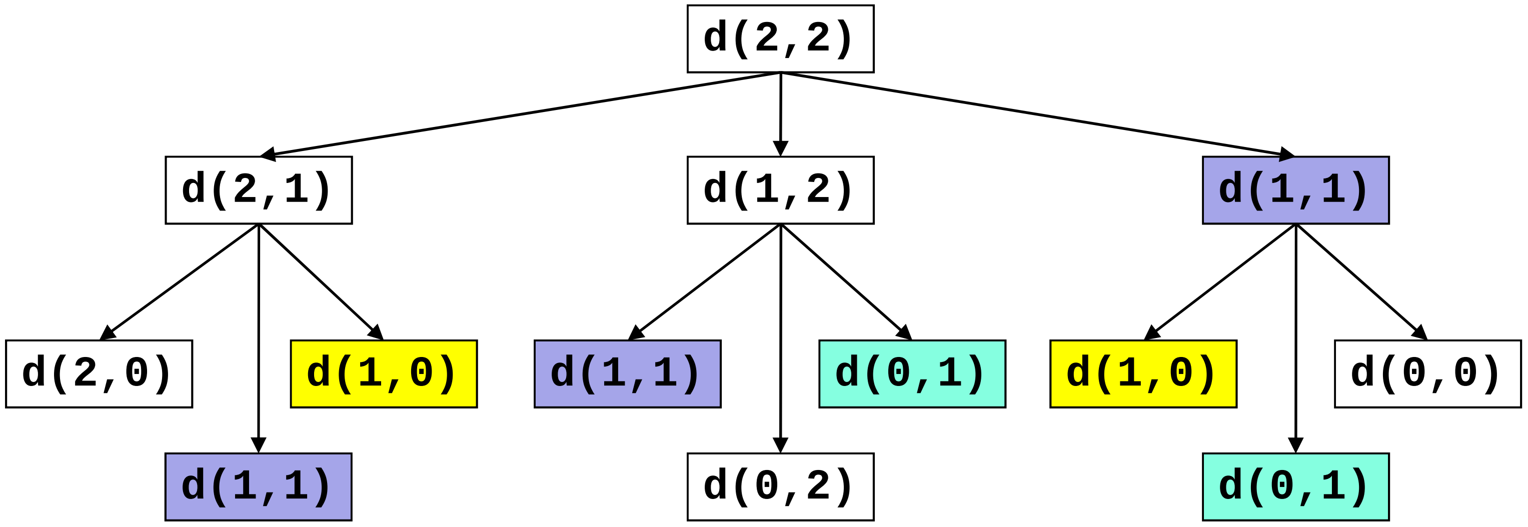
Algorithm

```
function d(i,j) {
        if (i = 0)                  return j;
        else if (j = 0)             return i;
        else
                return min (    d(i-1,j) + 1,
                                d(i,j-1) + 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?

$$\begin{tikzcd}
 & & & & d(2,2) \arrow[dlll] \arrow[d] \arrow[drrr] & & & & \\
 & d(2,1) \arrow[dl] \arrow[dd] \arrow[dr] & & & d(1,2) \arrow[dl] \arrow[dd] \arrow[dr] & & & d(1,1) \arrow[dl] \arrow[dd] \arrow[dr] & \\
d(2,0) & & d(1,0) & d(1,1) & & d(0,1) & d(1,0) & & d(0,0) \\
 & d(1,1) & & & d(0,2) & & & d(0,1) &
\end{tikzcd}$$

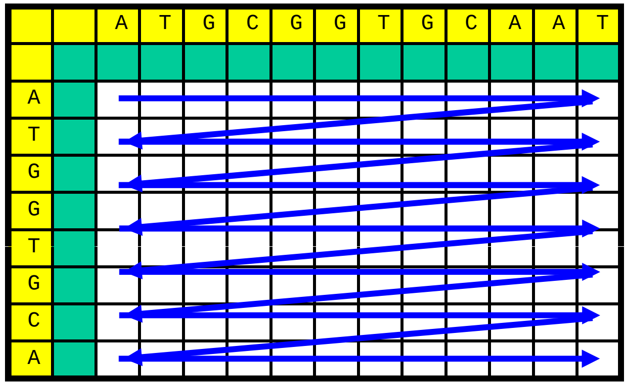
Much Redundant Computation

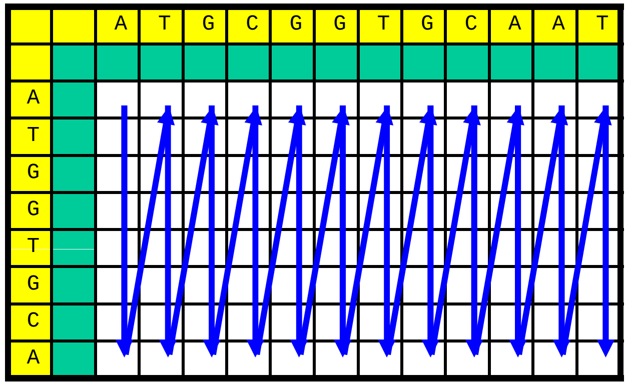


- There are only (n+1)*(m+1) different parameter combinations

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

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

		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							

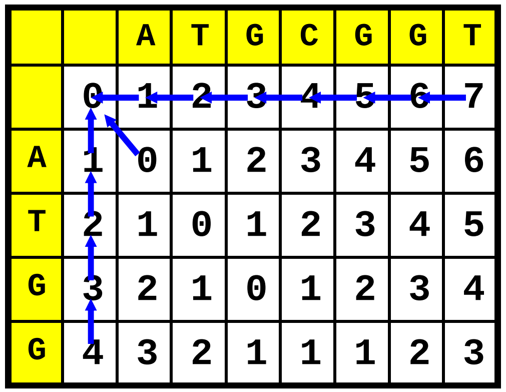
		A	T	G	C	G	G	T
	0	1	2	3	4	5	6	7
A	1	0	1	2	3	4	5	6
T	2	1	0	1	2	3	4	5
G	3							
G	4							

		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							

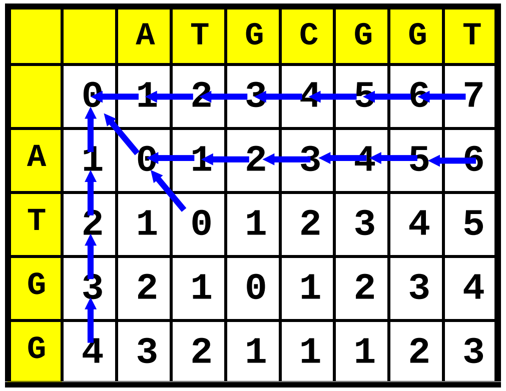
		A	T	G	C	G	G	T
	0	1	2	3	4	5	6	7
A	1	0	1	2	3	4	5	6
T	2	1	0	1	2	3	4	5
G	3	2	1	0	1	2	3	4
G	4	3	2	1	1	1	2	3

Finding the (an) optimal Alignment(s)

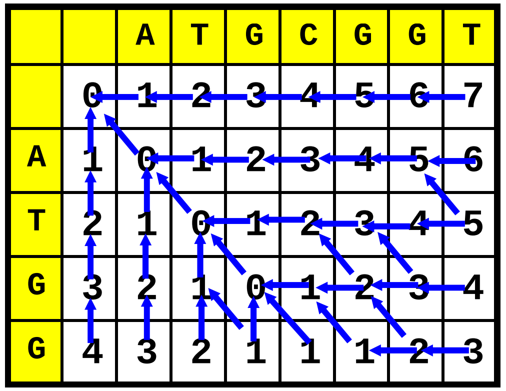
- Traceback
  - We find the path from back to front
  - Start at cell (n,m)
  - See which cells were used to compute d(n,m)
  - Walk any of these – finds one optimal path
  - Walking all means finding all optimal paths
- Alternative: Store provenance-pointers will filling the table



		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

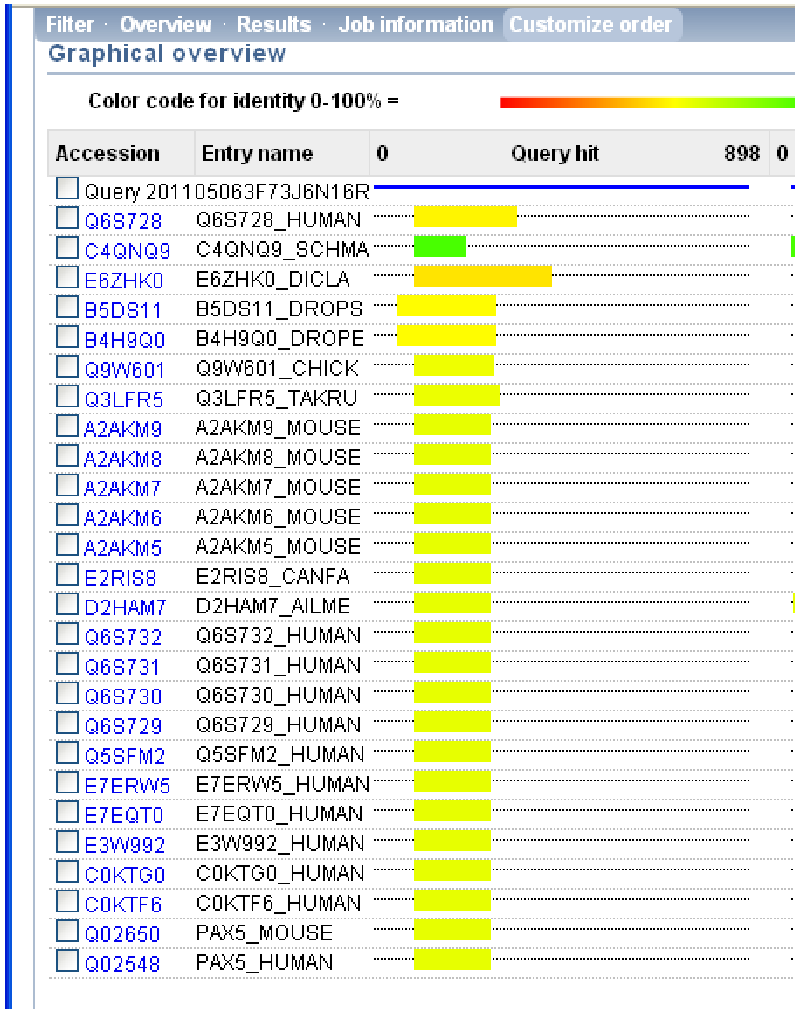


		A	T	G	C	G	G	T
	0	1	2	3	4	5	6	7
A	1	0	1	2	3	4	5	6
T	2	1	0	1	2	3	4	5
G	3	2	1	0	1	2	3	4
G	4	3	2	1	1	1	2	3

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*n cells
  - Thus: approximately 3*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) (für m*n > m+n)

Eyeless Again – a Closer Look


Filter · Overview · Results · Job information Customize order
Graphical overview
Color code for identity 0-100% =

Accession	Entry name	0 Query hit 898	0
Query 201105063F73J6N16R			
Q6S728	Q6S728_HUMAN		
C4QNQ9	C4QNQ9_SCHMA		
E6ZHK0	E6ZHK0_DICLA		
B5DS11	B5DS11_DROPS		
B4H9Q0	B4H9Q0_DROPE		
Q9W601	Q9W601_CHICK		
Q3LFR5	Q3LFR5_TAKRU		
A2AKM9	A2AKM9_MOUSE		
A2AKM8	A2AKM8_MOUSE		
A2AKM7	A2AKM7_MOUSE		
A2AKM6	A2AKM6_MOUSE		
A2AKM5	A2AKM5_MOUSE		
E2RIS8	E2RIS8_CANFA		
D2HAM7	D2HAM7_AILME		
Q6S732	Q6S732_HUMAN		
Q6S731	Q6S731_HUMAN		
Q6S730	Q6S730_HUMAN		
Q6S729	Q6S729_HUMAN		
Q5SFM2	Q5SFM2_HUMAN		
E7ERW5	E7ERW5_HUMAN		
E7EQT0	E7EQT0_HUMAN		
E3W992	E3W992_HUMAN		
C0KTG0	C0KTG0_HUMAN		
C0KTF6	C0KTF6_HUMAN		
Q02650	PAX5_MOUSE		
Q02548	PAX5_HUMAN		

- 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, but these carry the function

This Lecture

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

Example

```
ACCCTATCGATAGCTAGAAGCTCGAAAATACCGACCAGTAT
AGGAGTCGATAATACATATAAGAGATAGAATATATTGATG
```

```
ACCCTATCGATA--GC-TAGAAGCTCGATAATACCGACCAGTAT-
|     ||||||   | || |||   || |  |   |   | ||
A-GGAGTCGATAATACATATAAG-A-GATAGAATATA-TTG-ATG
```

```
ACCCTATCGATAGCTAGAAGCTCGAAAATACCGACCAGTAT
                     |||| |||||
                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 is their distance (in general)
  - Different lengths are punished: $e(A,B) \geq | |A|-|B| |$
- Often, we want a local similarity instead
  - 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, i.e., a subsequence in the gene
- We need to search for substrings $A' \in A$, $B' \in B$ which are very similar to each other
  - Further, 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

- *Let |A|=|B|=n*
- *A scoring function is a function s: $\Sigma' x \Sigma' \rightarrow$ Integer*
  - *We also call s a substitution matrix*
- *The direct similarity sim′ of A, B wrt. s is defined as*

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

- *The similarity sim of A, B (wrt. s) is the highest direct similarity score over all alignments of A and B*
  - Higher = better; the maximal similarity is not 0
- Remarks
  - 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

```
AC_GTC
AGGT_C
```
= -1

```
ACGTC
AGGTC
```
= 15

```
A_CGTC
AG_GTC
```
= 10

Computation

- The same ideas as for edit distance applies
- But now, we want to have a high similarity, not a low distance

$$d(i,0)=\sum_{k=1}^{i} s\left(A[k],\_\right) \qquad d(0,j)=\sum_{k=1}^{j} s\left(\_,B[k]\right)$$

$$d(i,j)=\max\left\{\begin{array}{c} d(i,j-1)+s(\_,B[j]) \\ d(i-1,j)+s(A[i],\_) \\ d(i-1,j-1)+s(A[i],B[j]) \end{array}\right\}$$

Example

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

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

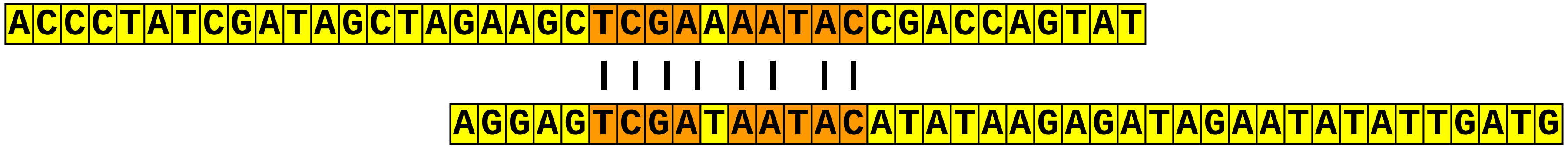
		A	G	G	T	C
	0	-3	-6	-9	-12	-15
A	-3	4	1	-2	-5	-8
G	-6	1	8	5		
T	-9					
C	-12					
C	-15					

Lokal Similarity = Local Alignment

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

$$sim^*(A,B) = \max_{\forall a' \subseteq A, b' \subseteq B}\big(sim(a',b')\big)$$

- Remark
  - Inequality in string length does not matter any more
  - Sounds terribly complex, but there is a neat trick

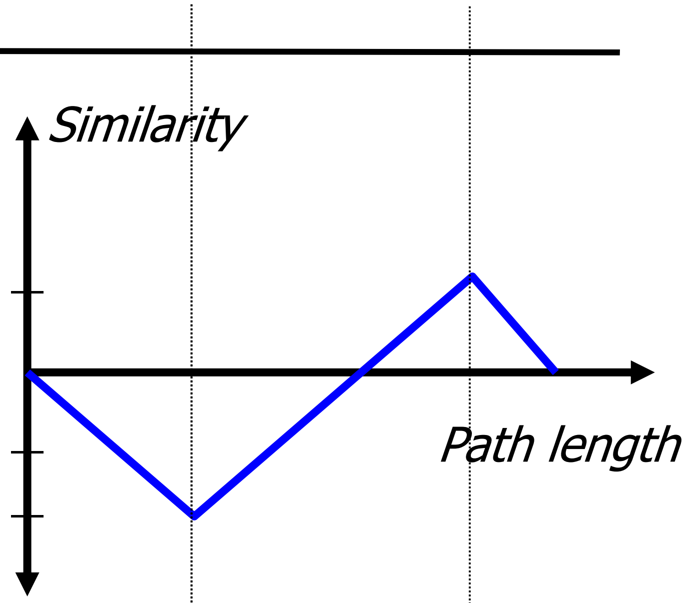

ACCCTATCGATAGCTAGAAGCTCGAAAATACCGACCAGTAT
AGGAGTCGATAATACATATAAGAGATAGAATATATTGATG

Example

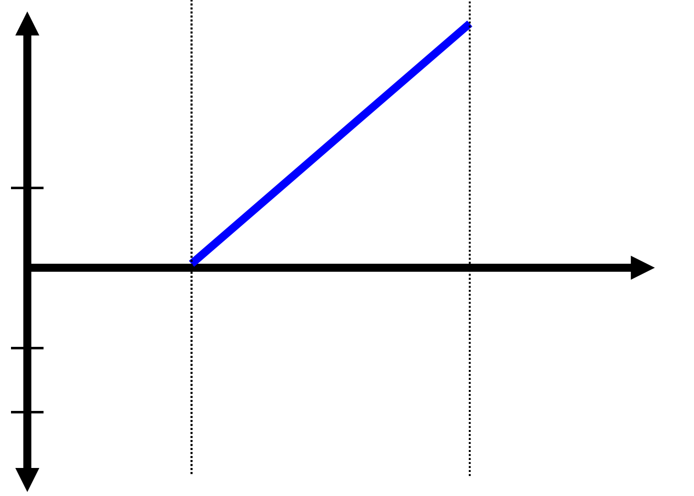
Match: +1

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 (global) 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 score gets below 0, we can forget this continuation of the path
  - Instead of carrying on, we conceptually start a new (local) path
  - To this end, we simply set d:=0 whenever it would be d<0
  - The highest value in the matrix is the end of the best local path

Computation

- The same ideas as before
- We compute sim*(A,B) as d(n,m) with
  - Assume ∀X: s(X,_)<0 and s(_,X)<0

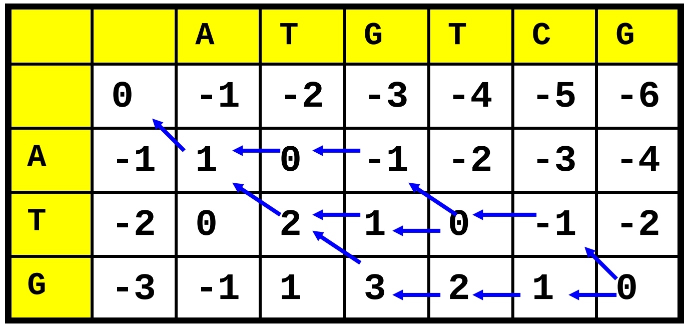
$$d(i,0) = 0 \qquad d(0,j) = 0$$

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

Example

Match: +1

I/R/D: -1

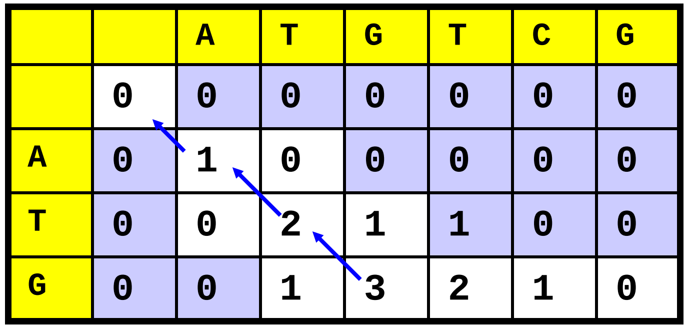


		A	T	G	T	C	G
	0	-1	-2	-3	-4	-5	-6
A	-1	1	0	-1	-2	-3	-4
T	-2	0	2	1	0	-1	-2
G	-3	-1	1	3	2	1	0

```
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	0

```
ATGTCG
ATG___
```

Local versus global Alignment

- Global Alignment
  - Comparison of two entire sequences
  - Use when you know the sequences are related
  - Interest: The differences
  - Example: Proteins of the same family
- Local Alignment
  - Finds interesting regions in yet uncharacterized sequences
  - Use when trying to relate a sequence to other (known) sequences
  - Interest: The similarities
  - Often a first step before global alignment
  - Example: Find similar genes in other species

Beware: Not all Events are Equal

Wildtype	C T T A G T G A C T A C G G T A A A	DNA
	Leu Ser Asp Tyr Gly Lys	Protein
Probably fatal	C T T A G T G A C T A G G G T A A A	DNA
	Leu Ser Asp **Stop-Codon**	Protein
Probably fatal	C T T A G T G A A C T A C G G T A A A	DNA
	Leu Ser His Asp Leu Thr	Protein
Neutral	C T T A G C G A C T A C G G T A A A	DNA
	Leu Ser Asp Tyr Gly Lys	Protein
Functional	C T T A G T G A A T A C G G T A A A	DNA
	Leu Ser Glu Tyr Gly Lys	Protein

Further Reading

- Everywhere

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