Fasta Format

>gi|18089116|gb|BC020718.1| Homo sapiens I factor
AAATTTCAAAAGAATACCTGGAGTGGAAAAGAGTTCTCAGCAGAGACAAAGACCCCGAACACCTCCAACA
TGAAGCTTCTTCATGTTTTCCTGTTATTTCTGTGCTTCCACTTAAGGTTTTGCAAGGTCACTTATACATC
TCAAGAGGATCTGGTGGAGAAAAAGTGCTTAGCAAAAAAATATACTCACCTCTCCTGCGATAAAGTCTTC
TGCCAGCCATGGCAGAGATGCATTGAGGGCACCTGTGTTTGTAAACTACCGTATCAGTGCCCAAAGAATG
GCACTGCAGTGTGTGCAACTAACAGGAGAAGCTTCCCAACATACTGTCAACAAAAGAGTTTGGAATGTCT
TCATCCAGGGACAAAGTTTTTAAATAACGGAACATGCACAGCCGAAGGAAAGTTTAGTGTTTCCTTGAAG
CATGGAAATACAGATTCAGAGGGAATAGTTGAAGTAAAACTTGTGGACCAAGATAAGACAATGTTCATAT
GCAAAAGCAGCTGGAGCATGAGGGAAGCCAACGTGGCCTGCCTTGACCTTGGGTTTCAACAAGGTGCTGA
TACTCAAAGAAGGTTTAAGTTGTCTGATCTCTCTATAAATTCCACTGAATGTCTACATGTGCATTGCCGA
GGATTAGAGACCAGTTTGGCTGAATGTACTTTTACTAAGAGAAGAACTATGGGTTACCAGGATTTCGCTG
ATGTGGTTTGTTATACACAGAAAGCAGATTCTCCAATGGATGACTTCTTTCAGTGTGTGAATGGGAAATA
CATTTCTCAGATGAAAGCCTGTGATGGTATCAATGATTGTGGAGACCAAAGTGATGAACTGTGTTGTAAA
GCATGCCAAGGCAAAGGCTTCCATTGCAAATCGGGTGTTTGCATTCCAAGCCAGTATCAATGCAATGGTG
AGGTGGACTGCATTACAGGGGAAGATGAAGTTGGCTGTGCAGGCTTTGCATCTGTGGCTCAAGAAGAAAC
AGAAATTTTGACTGCTGACATGGATGCAGAAAGAAGACGGATAAAATCATTATTACCTAAACTATCTTGT
GGAGTTAAAAACAGAATGCACATTCGAAGGAAACGAATTGTGGGAGGAAAGCGAGCACAACTGGGAAAAA
TGAAGCAAATCTCATTGGATATTTTTAAAGGTCTCCACAGAGTTTATGCCATATTGGAATTTTGTTGTAT
AATTCTCAAATAAATATTTTGGTGAAGCCAAAAAAAAAAAAAAAAAAAAAAAAAAAA

GenBank Format

```
1:  BC020718. Reports  Homo sapiens I fa...[gi:18089116]  Links
LOCUS       BC020718              1249 bp    mRNA    linear   PRI 06-OCT-2003
DEFINITION  Homo sapiens I factor (complement), mRNA (cDNA clone MGC:22501
VERSION     BC020718.1  GI:18089116
KEYWORDS    MGC.
SOURCE      Homo sapiens (human)
FEATURES             Location/Qualifiers
    source          1..1249
                    /organism="Homo sapiens"
                    /mol_type="mRNA"
                    /db_xref="taxon:9606"
                    /clone="MGC:22501 IMAGE:4716122"
                    /tissue_type="Liver"
                    /clone_lib="NIH_MGC_76"
                    /lab_host="DH10B"
                    /note="Vector: pDNR-LIB"
    gene            1..1249
                    /gene="IF"
                    /note="synonym: FI"
                    /db_xref="GeneID:3426"
                    /db_xref="MIM:217030"
    CDS             70..1203
                    /gene="IF"
                    /codon_start=1
                    /product="IF protein"
                    /protein_id="AAH20718.1"
                    /db_xref="GI:18089117"
                    /db_xref="GeneID:3426"
                    /db_xref="MIM:217030"
                    /translation="MKLLHVFLLFLCFHLRFCKVTYTSQEDLVEKKCLAKKYTHLSCD
                    KVFCQPWQRCIEGTCVCKLPYQCPKNGTAVCATNRRSFPTYCQQKSLECLHPGTKFLN
                    NGTCTAEGKFSVSLKHGNTDSEGIVEVKLVDQDKTMFICKSSWSMREANVACLDLGFQ
                    QGADTQRRFKLSDLSINSTECLHVHCRGLETSLAECTFTKRRTMGYQDFADVVCYTQK
                    ADSPMDDFFQCVNGKYISQMKACDGINDCGDQSDELCCKACQGKGFHCKSGVCIPSQY
                    QCNGEVDCITGEDEVGCAGFASVAQEETEILTADMDAERRRIKSLLPKLSCGVKNRMH
                    IRRKRIVGGKRAQLGKMKQISLDIFKGLHRVYAILEFCCIILK"
```

misc_feature 193..393
/gene="IF"
/note="FIMAC; Region: factor I membrane attack complex"
/db_xref="CDD:smart00057"
misc_feature 418..714
/gene="IF"
/note="SRCR; Region: Scavenger receptor cysteine-rich domain. These domains are disulphide rich extracellular domains. These domains are found in several extracellular receptors and may be involved in protein-protein interactions"
/db_xref="CDD:pfam00530"
misc_feature 838..954
/gene="IF"
/note="ldl_recept_a; Region: Low-density lipoprotein receptor domain class A"
/db_xref="CDD:pfam00057"
misc_difference 967
/gene="IF"
/note="'G' in cDNA is 'A' in the human genome; amino acid difference: 'A' in cDNA, 'T' in the human genome. The chimpanzee genome agrees with the cDNA sequence, suggesting that this difference is unlikely to be due to an artifact; Differences found between this sequence and the human reference genome (build 35) are described in misc_difference features below and these differences were also compared to chimpanzee genomic seqeunces available as of 09/15/2004 00:00:00"
misc_difference 1135
/gene="IF"
/note="'T' in cDNA is 'C' in the human genome; no amino acid change. The chimpanzee genome agrees with the cDNA sequence, suggesting that this difference is unlikely to be due to an artifact; Differences found between this sequence and the human reference genome (build 35) are described in misc_difference features below and these differences were also compared to chimpanzee genomic

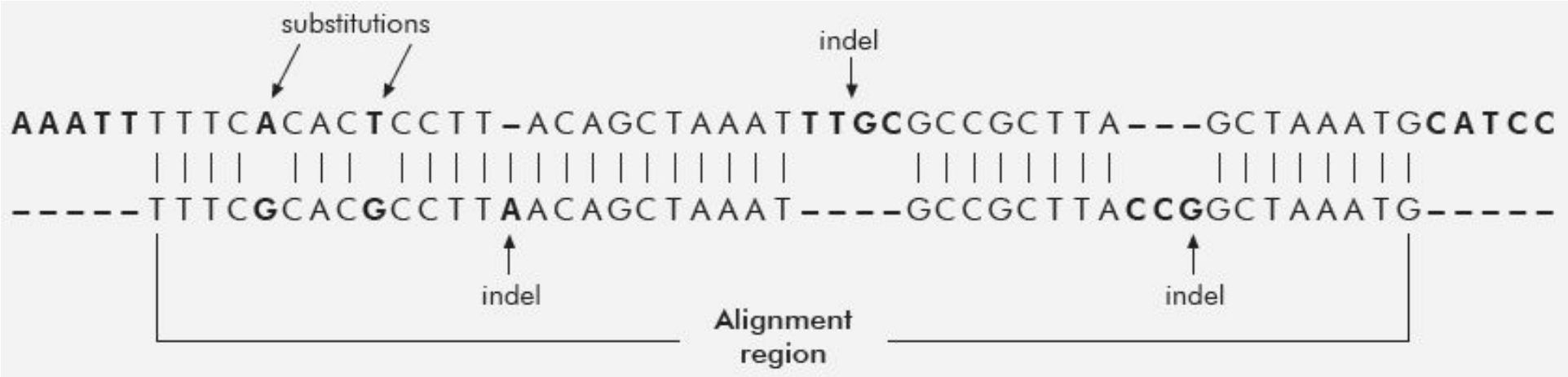
```
   1 aaatttcaaa agaatacctg gagtggaaaa gagttctcag cagagacaaa gaccccgaac
  61 acctccaaca tgaagcttct tcatgttttc ctgttatttc tgtgcttcca cttaaggttt
 121 tgcaaggtca cttatacatc tcaagaggat ctggtggaga aaaagtgctt agcaaaaaaa
 181 tatactcacc tctcctgcga taaagtcttc tgccagccat ggcagagatg cattgagggc
 241 acctgtgttt gtaaactacc gtatcagtgc ccaaagaatg gcactgcagt gtgtgcaact
 301 aacaggagaa gcttcccaac atactgtcaa caaaagagtt tggaatgtct tcatccaggg
 361 acaaagtttt taaataacgg aacatgcaca gccgaaggaa agtttagtgt ttccttgaag
 421 catggaaata cagattcaga gggaatagtt gaagtaaaac ttgtggacca agataagaca
 481 atgttcatat gcaaaagcag ctggagcatg agggaagcca acgtggcctg ccttgacctt
 541 gggtttcaac aaggtgctga tactcaaaga aggtttaagt tgtctgatct ctctataaat
 601 tccactgaat gtctacatgt gcattgccga ggattagaga ccagtttggc tgaatgtact
 661 tttactaaga gaagaactat gggttaccag gatttcgctg atgtggtttg ttatacacag
 721 aaagcagatt ctccaatgga tgacttcttt cagtgtgtga atgggaaata catttctcag
 781 atgaaagcct gtgatggtat caatgattgt ggagaccaaa gtgatgaact gtgttgtaaa
 841 gcatgccaag gcaaaggctt ccattgcaaa tcgggtgttt gcattccaag ccagtatcaa
 901 tgcaatggtg aggtggactg cattacaggg gaagatgaag ttggctgtgc aggctttgca
 961 tctgtggctc aagaagaaac agaaattttg actgctgaca tggatgcaga aagaagacgg
1021 ataaaatcat tattacctaa actatcttgt ggagttaaaa acagaatgca cattcgaagg
1081 aaacgaattg tgggaggaaa gcgagcacaa ctgggaaaaa tgaagcaaat ctcattggat
1141 atttttaaag gtctccacag agtttatgcc atattggaat tttgttgtat aattctcaaa
1201 taaatatttt ggtgaagcca aaaaaaaaaa aaaaaaaaaa aaaaaaaaa
//
```

Phylip Format

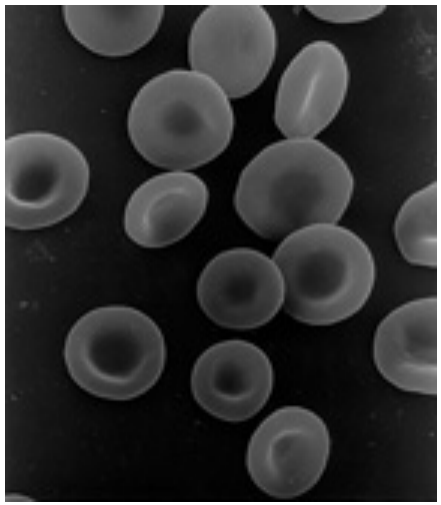
```
 number of species   length
Specie1    RWLFSTNHKD IGTLYFIFGI WSGMVGTALS LLIRAELG-Q SGSLIGD--F
specie2    RWFYSTNHKD IGTLYFIFGA WAGMVGTSLS MLIRAELG-Q PGSLIGD---
specie3    RWLYSTNHKD IGTLYLIFGA WAGMVGTALS LLIRAELGAQ PGSLIGD---
specie4    KWSGATNHKD IGTLYLIAGA WAGFVGAAMS VLIRLELG-Q CGSFLGNNN-

           KDQIYNVIVT AHAFVMIFFM VMPIMIGGFG NWLVPLMLGA PDMAFPRLNN
           -EQVYNVIVT AHAFIMIFFM VMPILIGGFG NWLVPIMLGA PDMAFPRLNN
           -DQIYNVVVT AHAFIMIFFM VMPILIGGFG NWLVPLMLGA PDMAFPRLNN
           -DQLYNVLVT AHAFVMIFFM VMPVLIGGFG NWLVPLMLGA PDMAFPRLNN

           LSFWFLPPAL TLLLVGGAVE SGAGTGWTVY PPLSAGIAHA GASVDLSIFS
           LSFWMLPPSL TLLLASSMVE SGAGTGWTVY PPLSSAIAHA GPSVDLAIFS
           MSFWLLPPAL TLLLSGGAVE SGAGTGWTVY PPLSSGIAHA GASVDLSIFS
           LSFWFLIPSL LLLLLSGAVE AGAGTGWTVY PPLS-AVGHA GSSVDFAIFS
```

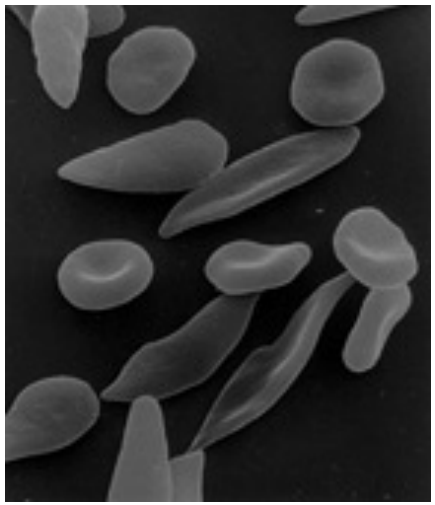
- gap

Example of alignment



Example of deleterious single base mutation: Sickle Cell Anemia





Due to 1 swapping an A for a T, causing inserted amino acid to be valine instead of glutamine in hemoglobin

Healthy Individual

>gi|28302128|ref|NM_000518.4| Homo sapiens hemoglobin, beta (HBB), mRNA

ACATTTGCTTCTGACACAACTGTGTTCACTAGCAACCTCAAACAGACACCATGGTGCATCTGACTCCTGA

GGAGAAGTCTGCCGTTACTGCCCTGTGGGGCAAGGTGAACGTGGATGAAGTTGGTGGTGAGGCCCTGGGC

AGGCTGCTGGTGGTCTACCCTTGGACCCAGAGGTTCTTTGAGTCCTTTGGGGATCTGTCCACTCCTGATG

CTGTTATGGGCAACCCTAAGGTGAAGGCTCATGGCAAGAAAGTGCTCGGTGCCTTTAGTGATGGCCTGGC

TCACCTGGACAACCTCAAGGGCACCTTTGCCACACTGAGTGAGCTGCACTGTGACAAGCTGCACGTGGAT

CCTGAGAACTTCAGGCTCCTGGGCAACGTGCTGGTCTGTGTGCTGGCCCATCACTTTGGCAAAGAATTCA

CCCCACCAGTGCAGGCTGCCTATCAGAAAGTGGTGGCTGGTGTGGCTAATGCCCTGGCCCACAAGTATCA

CTAAGCTCGCTTTCTTGCTGTCCAATTTCTATTAAAGGTTCCTTTGTTCCCTAAGTCCAACTACTAAACT

GGGGGATATTATGAAGGGCCTTGAGCATCTGGATTCTGCCTAATAAAAAACATTTATTTTCATTGC

>gi|4504349|ref|NP_000509.1| beta globin [Homo sapiens]

MVHLTPEEKSAVTALWGKVNVDEVGGEALGRLLVVYPWTQRFFESFGDLSTPDAVMGNPKVKAHGKKVLG

AFSDGLAHLDNLKGTFATLSELHCDKLHVDPENFRLLGNVLVCVLAHHFGKEFTPPVQAAYQKVVAGVAN

ALAHKYH

Diseased Individual

>gi|28302128|ref|NM_000518.4| Homo sapiens hemoglobin, beta (HBB), mRNA

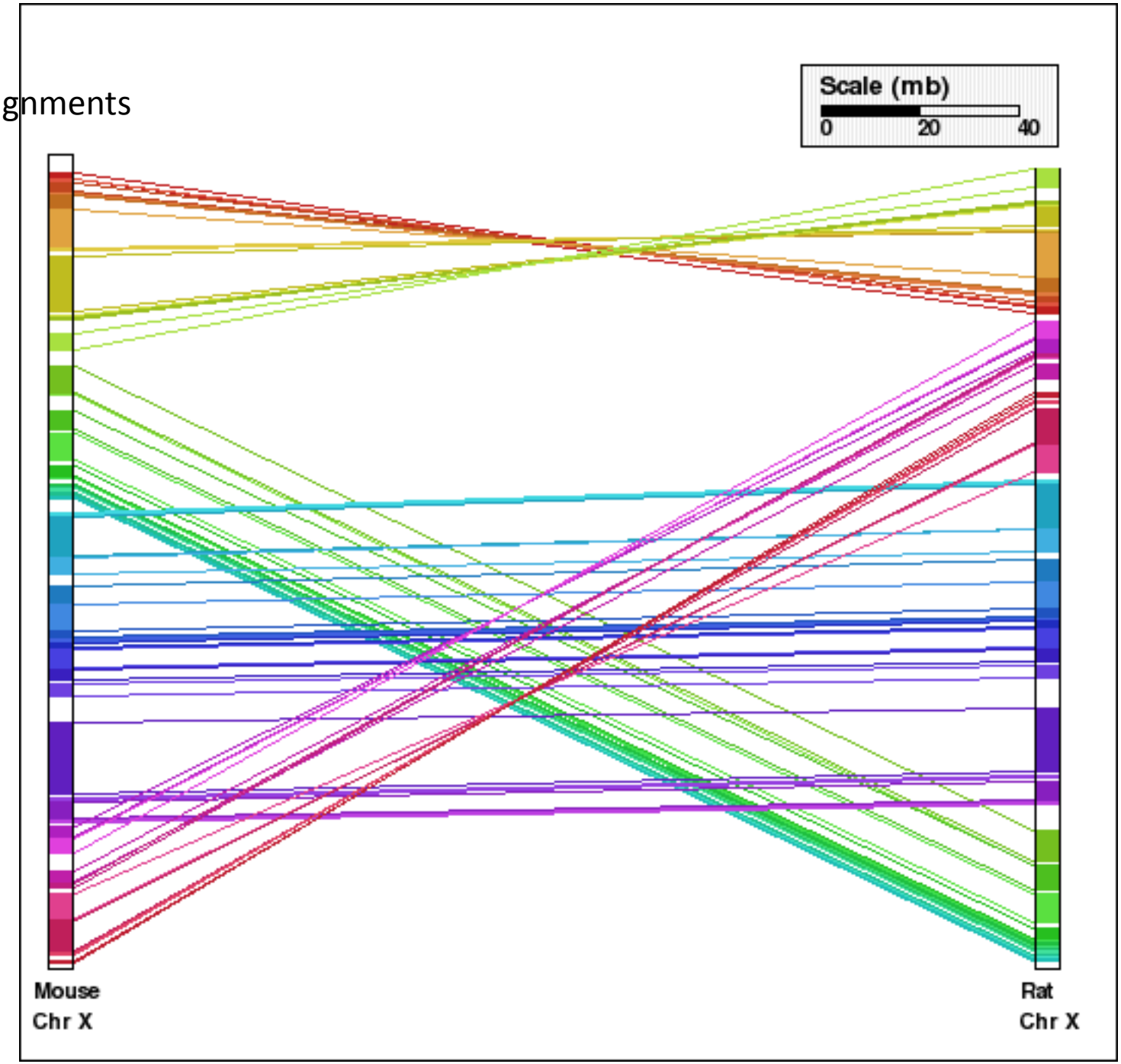
ACATTTGCTTCTGACACAACTGTGTTCACTAGCAACCTCAAACAGACACCATGGTGCATCTGACTCCTGA

GGTGAAGTCTGCCGTTACTGCCCTGTGGGGCAAGGTGAACGTGGATGAAGTTGGTGGTGAGGCCCTGGGC
AGGCTGCTGGTGGTCTACCCTTGGACCCAGAGGTTCTTTGAGTCCTTTGGGGATCTGTCCACTCCTGATG
CTGTTATGGGCAACCCTAAGGTGAAGGCTCATGGCAAGAAAGTGCTCGGTGCCTTTAGTGATGGCCTGGC
TCACCTGGACAACCTCAAGGGCACCTTTGCCACACTGAGTGAGCTGCACTGTGACAAGCTGCACGTGGAT
CCTGAGAACTTCAGGCTCCTGGGCAACGTGCTGGTCTGTGTGCTGGCCCATCACTTTGGCAAAGAATTCA
CCCCACCAGTGCAGGCTGCCTATCAGAAAGTGGTGGCTGGTGTGGCTAATGCCCTGGCCCACAAGTATCA
CTAAGCTCGCTTTCTTGCTGTCCAATTTCTATTAAAGGTTCCTTTGTTCCCTAAGTCCAACTACTAAACT
GGGGGATATTATGAAGGGCCTTGAGCATCTGGATTCTGCCTAATAAAAAACATTTATTTTCATTGC

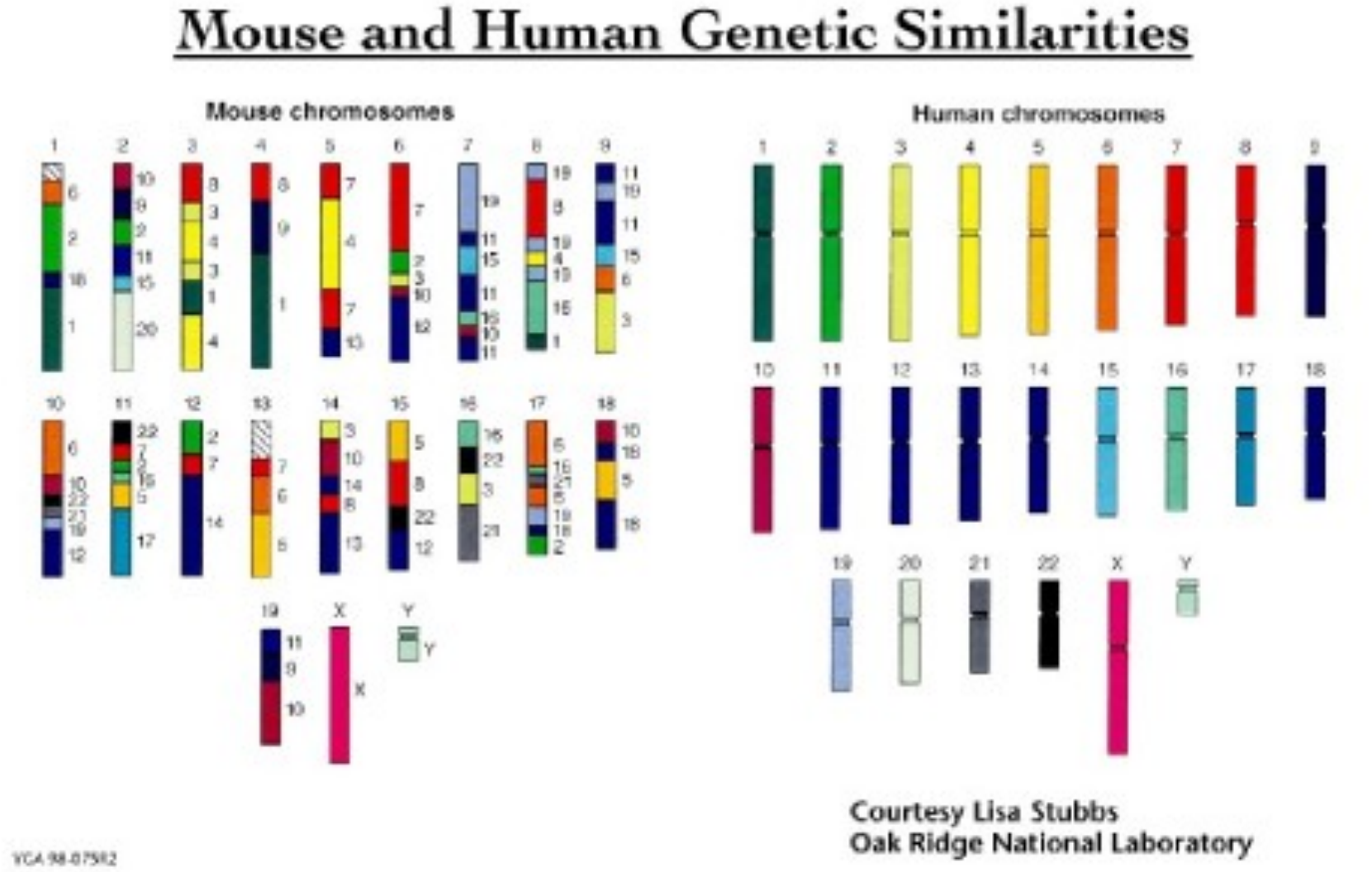
>gi|4504349|ref|NP_000509.1| beta globin [Homo sapiens]

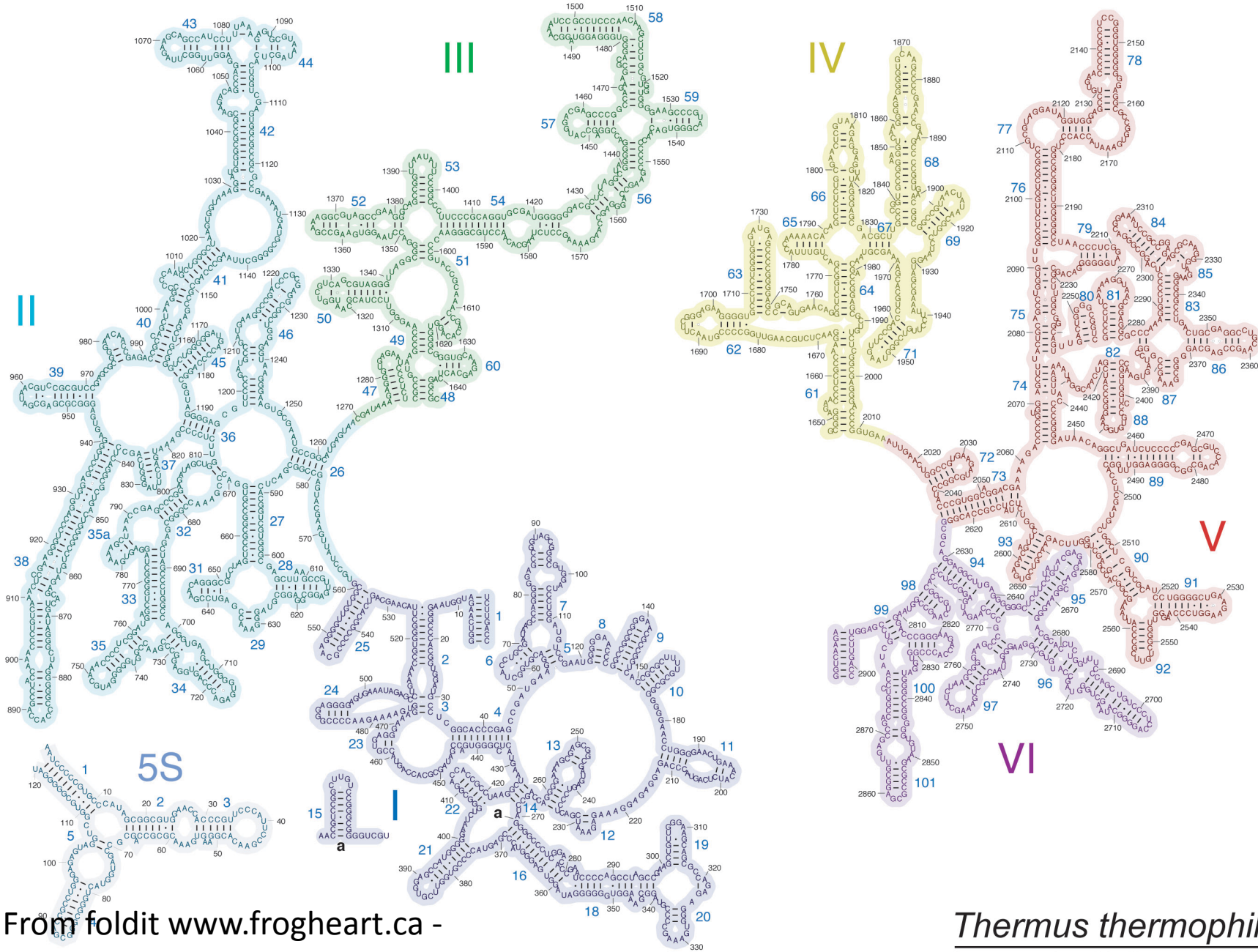
MVHLTPVEKSAVTALWGKVNVDEVGGEALGRLLVVYPWTQRFFESFGDLSTPDAVMGNPKVKAHGKKVLG
AFSDGLAHLDNLKGTFATLSELHCDKLHVDPENFRLLGNVLVCVLAHHFGKEFTPPVQAAYQKVVAGVAN
ALAHKYH

Big Alignments



Big Alignments





From foldit www.frogheart.ca -

Thermus thermophilus

large subunit ribosomal RNA

From Timothy L. Bailey

The Nussinov Folding Algorithm

Example: GGGAAAUCC

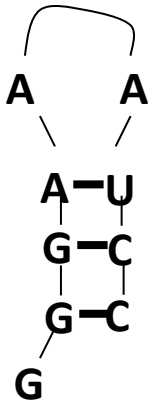
$\gamma(i,j)$ is the maximum number of base pairs in segment $[i,j]$

$$Initialisation\ \gamma(i,i-1) = 0\ \&\ \gamma(i,i) = 0$$

Starting with all subsequences of length 2, to length L:

$$\gamma(i,j) = \max \begin{cases} \gamma(i+1,j) \\ \gamma(i,j-1) \\ \gamma(i+1,j-1) + \delta(i,j) \\ \max_{i<k<j}[\gamma(i,k) + \gamma(k+1,j)] \end{cases}$$

Where $\delta(i,j) = 1$ if x_i and x_j are a complementary base pair, and $\delta(i,j) = 0$, otherwise.



j →

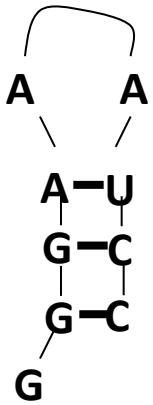
i ↓

	G	G	G	A	A	A	U	C	C
G	0								
G	0	0							
G		0	0						
A			0	0					
A				0	0				
U					0	0			
C						0	0		
C							0	0	
								0	0

From Timothy L. Bailey

Nussinov Folding Algorithm: After scores for subsequences of length 2

$$\gamma(i,j) = \max \begin{cases} \gamma(i+1,j) \\ \gamma(i,j-1) \\ \gamma(i+1,j-1) + \delta(i,j) \\ \max_{i<k<j}[\gamma(i,k) + \gamma(k+1,j)] \end{cases}$$



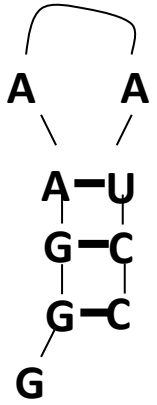
j →

i ↓	G	G	G	A	A	A	U	C	C
G	0	**0**							
G	0	0	**0**						
G		0	0	**0**					
A			0	0	**0**				
A				0	0	**0**			
U					0	0	**1**		
C						0	0	**0**	
							0	0	**0**
								0	0

From Timothy L. Bailey

Nussinov Folding Algorithm: After scores for subsequences of length 3

$$\gamma(i,j) = \max \begin{cases} \gamma(i+1,j) \\ \gamma(i,j-1) \\ \gamma(i+1,j-1)+\delta(i,j) \\ \max_{i<k<j}[\gamma(i,k)+\gamma(k+1,j)] \end{cases}$$



j →

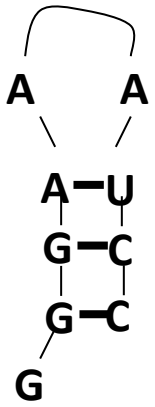
i ↓

	G	G	G	A	A	A	U	C	C
G	0	0	**0**						
G	0	0	0	**0**					
G		0	0	0	**0**				
A			0	0	0	**0**			
A				0	0	0	**1**		
U					0	0	1	**0**	
C						0	0	0	**0**
							0	0	0
								0	0

From Timothy L. Bailey

Nussinov Folding Algorithm
After scores for subsequences of length 4

$$\gamma(i,j) = \max \begin{cases} \gamma(i+1,j) \\ \gamma(i,j-1) \\ \gamma(i+1,j-1)+\delta(i,j) \\ \max_{i<k<j}[\gamma(i,k)+\gamma(k+1,j)] \end{cases}$$



j →

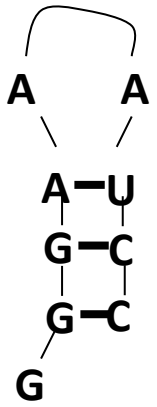
i		G	G	G	A	A	A	U	C	C
G		0	0	0	**0**					
G		0	0	0	0	**0**				
G			0	0	0	0	**0**			
G				0	0	0	0	**1**		
A					0	0	**0**	**1**	**1**	
A						0	0	1	1	**1**
U							0	0	0	0
C								0	0	0
									0	0

Two optimal substructures for same subsequence

From Timothy L. Bailey

Nussinov Folding Algorithm
After scores for subsequences of length 5

$$\gamma(i,j) = \max \begin{cases} \gamma(i+1,j) \\ \gamma(i,j-1) \\ \gamma(i+1,j-1)+\delta(i,j) \\ \max_{i<k<j}[\gamma(i,k)+\gamma(k+1,j)] \end{cases}$$

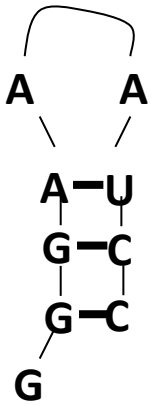


		G	G	G	A	A	A	U	C	C
		0	0	0	0	**0**				
G		0	0	0	0	0	**0**			
G			0	0	0	0	0	**1**		
G				0	0	0	0	1	**1**	
A					0	0	0	1	1	**1**
A						0	0	1	1	1
U							0	0	0	0
C								0	0	0
									0	0

From Timothy L. Bailey

Nussinov Folding Algorithm
After scores for subsequences of length 6

$$\gamma(i,j) = \max \begin{cases} \gamma(i+1,j) \\ \gamma(i,j-1) \\ \gamma(i+1,j-1) + \delta(i,j) \\ \max_{i<k<j}[\gamma(i,k) + \gamma(k+1,j)] \end{cases}$$

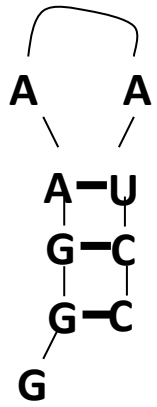


		G	G	G	A	A	A	U	C	C
		j →								
G	i ↓	0	0	0	0	0	**0**			
G		0	0	0	0	0	0	**1**		
G			0	0	0	0	0	1	**2**	
A				0	0	0	0	1	1	**1**
A					0	0	0	1	1	1
U						0	0	1	1	1
C							0	0	0	0
								0	0	0
									0	0

From Timothy L. Bailey

Nussinov Folding Algorithm
After scores for subsequences of length 7

$$\gamma(i,j) = \max \begin{cases} \gamma(i+1,j) \\ \gamma(i,j-1) \\ \gamma(i+1,j-1)+\delta(i,j) \\ \max_{i<k<j}[\gamma(i,k)+\gamma(k+1,j)] \end{cases}$$

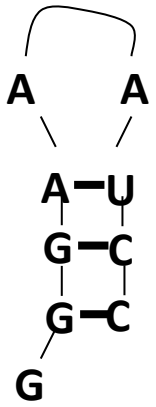


i \ j	G	G	G	A	A	A	U	C	C
G	0	0	0	0	0	0	**1**		
G	0	0	0	0	0	0	1	**2**	
G		0	0	0	0	0	1	2	**2**
A			0	0	0	0	1	1	1
A				0	0	0	1	1	1
U					0	0	1	1	1
C						0	0	0	0
							0	0	0
								0	0

From Timothy L. Bailey

Nussinov Folding Algorithm
After scores for subsequences of length 8

$$\gamma(i,j) = \max \begin{cases} \gamma(i+1,j) \\ \gamma(i,j-1) \\ \gamma(i+1,j-1)+\delta(i,j) \\ \max_{i<k<j}[\gamma(i,k)+\gamma(k+1,j)] \end{cases}$$



j →

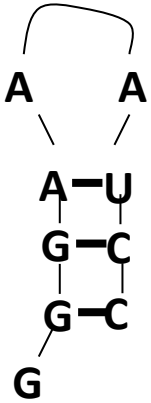
i ↓

	G	G	G	A	A	A	U	C	C
G	0	0	0	0	0	0	1	**2**	
G	0	0	0	0	0	0	1	2	**3**
G		0	0	0	0	0	1	2	2
A			0	0	0	0	1	1	1
A				0	0	0	1	1	1
A					0	0	1	1	1
U						0	0	0	0
C							0	0	0
								0	0

From Timothy L. Bailey

Nussinov Folding Algorithm
After scores for subsequences of length 9

$$\gamma(i,j) = \max \begin{cases} \gamma(i+1,j) \\ \gamma(i,j-1) \\ \gamma(i+1,j-1) + \delta(i,j) \\ \max_{i<k<j}[\gamma(i,k) + \gamma(k+1,j)] \end{cases}$$

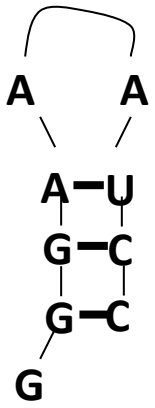


j →

i		G	G	G	A	A	A	U	C	C
		0	0	0	0	0	0	1	2	**3**
G	0	0	0	0	0	0	0	1	2	3
G		0	0	0	0	0	0	1	2	2
G			0	0	0	0	0	1	1	1
A				0	0	0	0	1	1	1
A					0	0	0	1	1	1
U						0	0	0	0	0
C								0	0	0
									0	0

From Timothy L. Bailey

Nussinov Folding Algorithm Traceback



		G	G	G	A	A	A	U	C	C
		j →								
G		0	0	0	0	0	0	1	2	3
G		0	0	0	0	0	0	1	2	3
G			0	0	0	0	0	1	2	2
A				0	0	0	0	1	1	1
A					0	0	0	1	1	1
U	i ↓					0	0	1	1	1
C							0	0	0	0
								0	0	0
									0	0

Algorithm: Nussinov RNA folding, fill stage

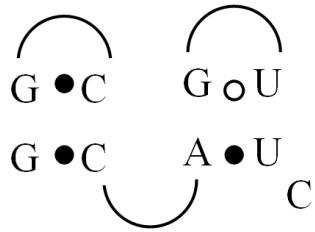
Initialisation:

$$\begin{aligned} \gamma(i, i-1) &= 0 \qquad \text{for } i = 2 \text{ to } L; \\ \gamma(i, i) &= 0 \qquad \text{for } i = 1 \text{ to } L. \end{aligned}$$

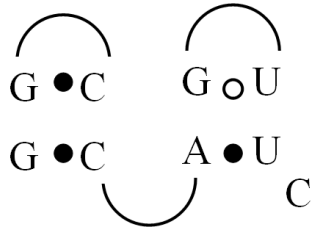
Recursion: starting with all subsequences of length 2, to length L:

$$\gamma(i,j) = \max \begin{cases} \gamma(i+1, j), \\ \gamma(i, j-1), \\ \gamma(i+1, j-1) + \delta(i, j), \\ \max_{i<k<j} \left[\gamma(i,k) + \gamma(k+1, j)\right]. \end{cases}$$

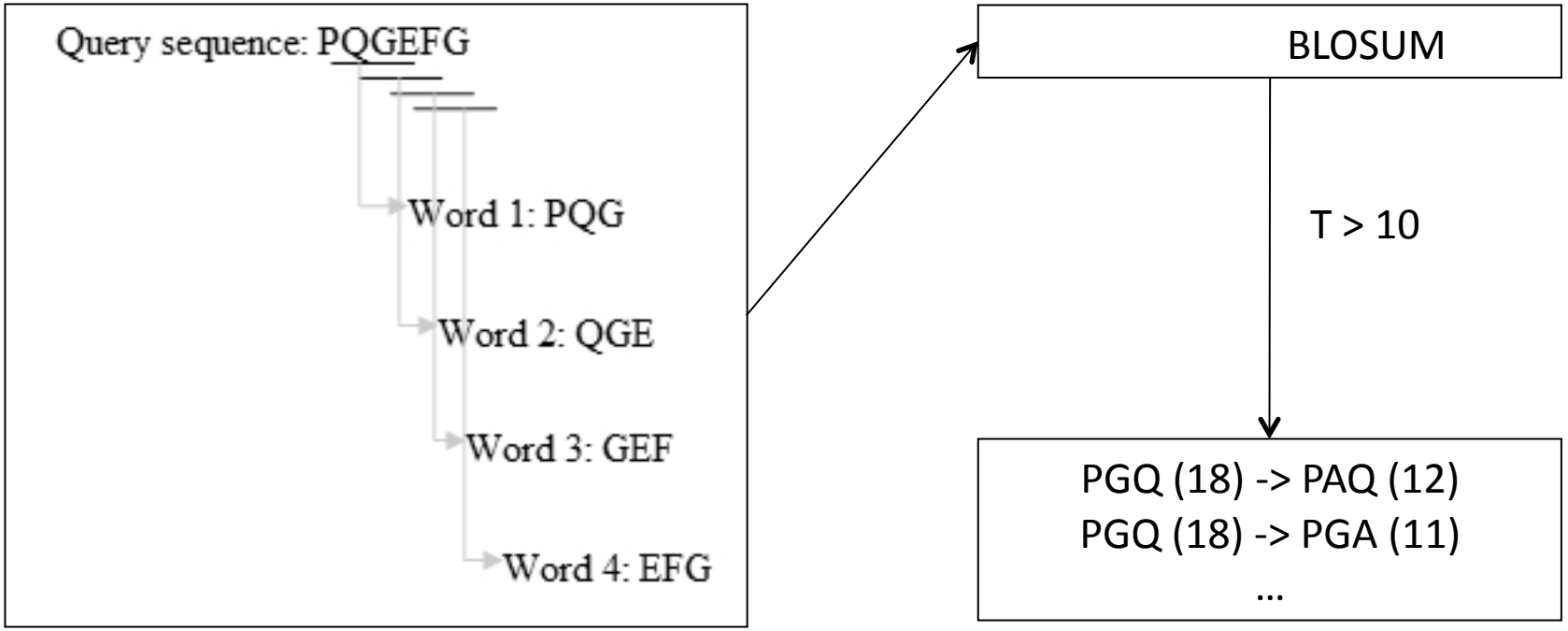
		G	G	C	C	A	G	U	U	C
		1	2	3	4	5	6	7	8	9
G	1	0	0	1	2	2	2	3	4	4
G	2	0	0	1	1	1	2	2	3	3
C	3		0	0	0	0	1	1	2	2
C	4			0	0	0	1	1	2	2
A	5				0	0	0	1	2	2
G	6					0	0	1	1	1
U	7						0	0	0	0
U	8							0	0	0
C	9								0	0

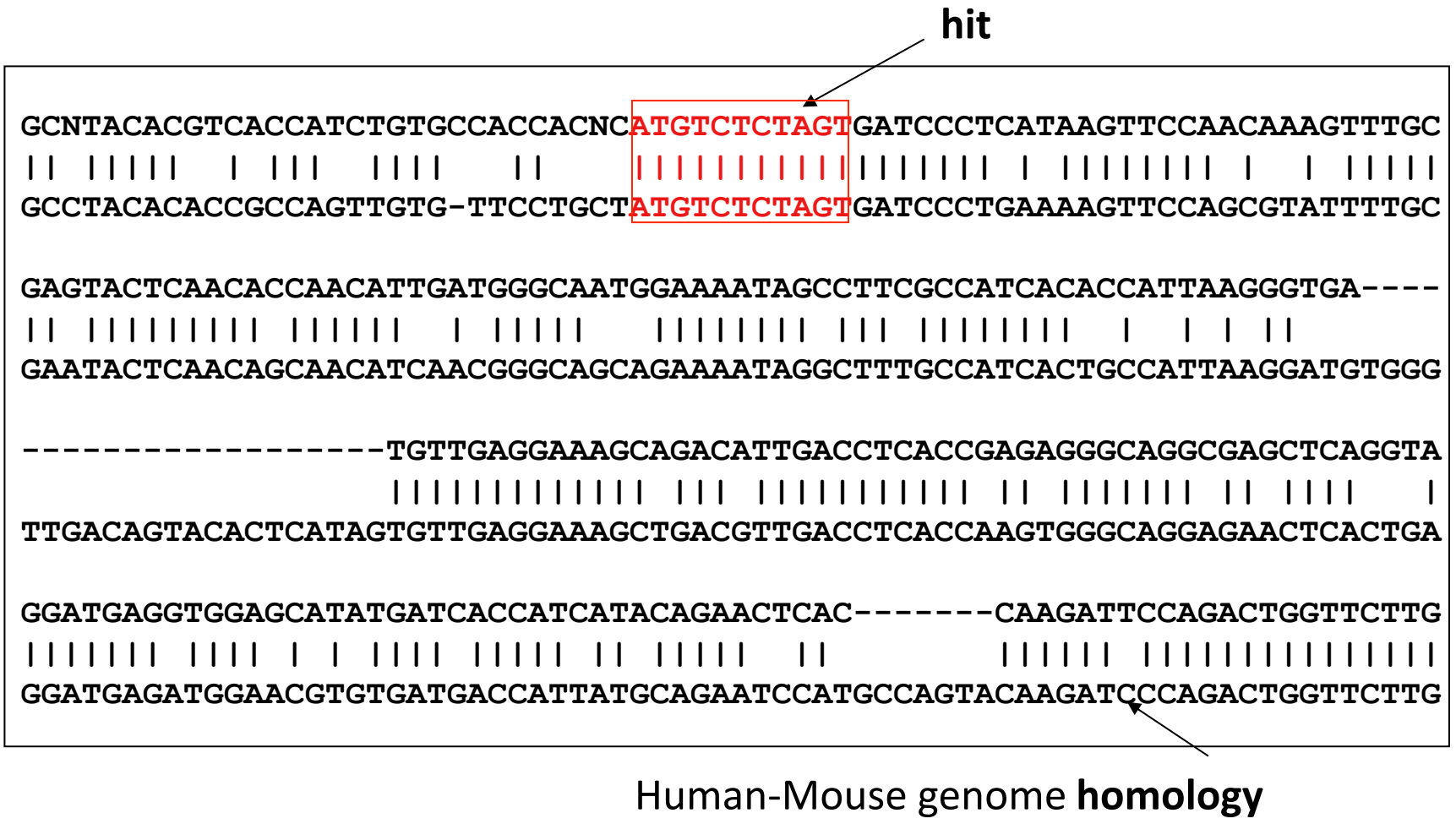


			G	G	C	C	A	G	U	U	C
			1	2	3	4	5	6	7	8	9
G	1		0	0	1	2	2	2	3	4	4
G	2		0	0	1	1	1	2	2	3	3
C	3			0	0	0	0	1	1	2	2
C	4				0	0	0	1	1	2	2
A	5					0	0	0	1	2	2
G	6						0	0	1	1	1
U	7							0	0	0	0
U	8								0	0	0
C	9									0	0

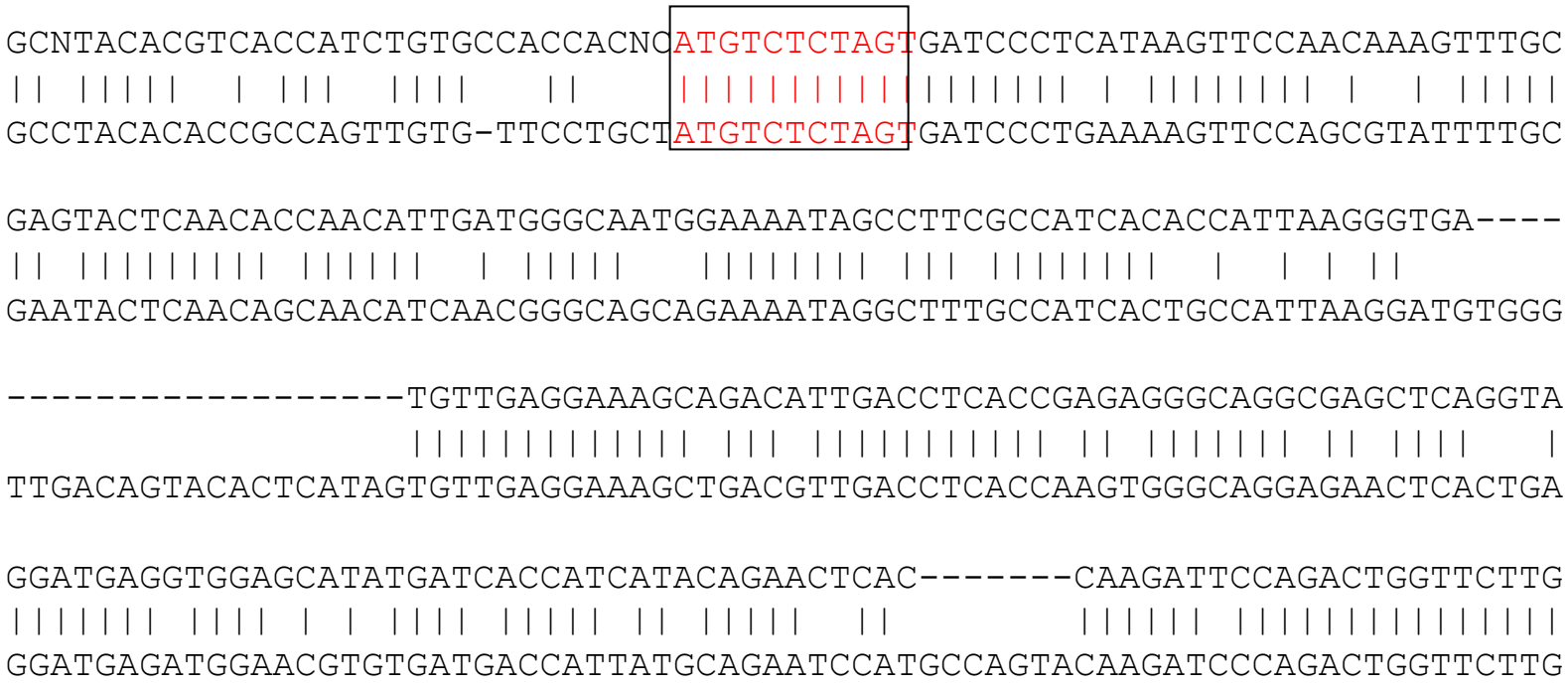


BLAST: Seeding




hit
GCNTACACGTCACCATCTGTGCCACCACNCATGTCTCTAGTGATCCCTCATAAGTTCCAACAAAGTTTGC
GCCTACACACCGCCAGTTGTG-TTCCTGCTATGTCTCTAGTGATCCCTGAAAAGTTCCAGCGTATTTTGC
GAGTACTCAACACCAACATTGATGGGCAATGGAAAATAGCCTTCGCCATCACACCATTAAGGGTGA----
GAATACTCAACAGCAACATCAACGGGCAGCAGAAAATAGGCTTTGCCATCACTGCCATTAAGGATGTGGG
------------------TGTTGAGGAAAGCAGACATTGACCTCACCGAGAGGGCAGGCGAGCTCAGGTA
TTGACAGTACACTCATAGTGTTGAGGAAAGCTGACGTTGACCTCACCAAGTGGGCAGGAGAACTCACTGA
GGATGAGGTGGAGCATATGATCACCATCATACAGAACTCAC-------CAAGATTCCAGACTGGTTCTTG
GGATGAGATGGAACGTGTGATGACCATTATGCAGAATCCATGCCAGTACAAGATCCCAGACTGGTTCTTG
Human-Mouse genome **homology**

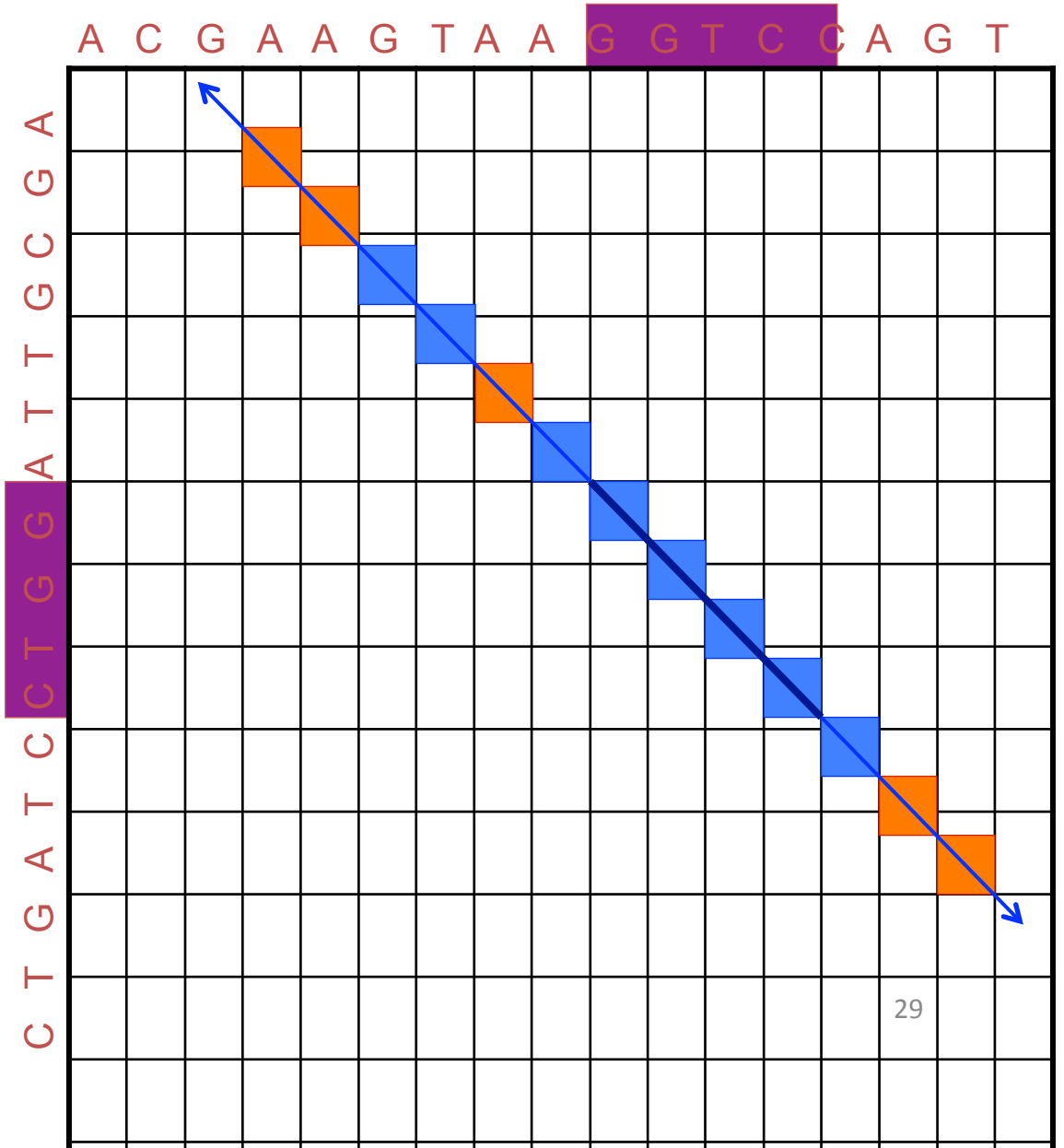
BLAST finds a “hit” and then extends



Seed match = hit

Original BLAST: Example

- $w = 4$
- Exact keyword match of GGTC
- Extend diagonals with mismatches until score is under 50%
- Output result
  GTAAGGTCC
  GTTAGGTCC

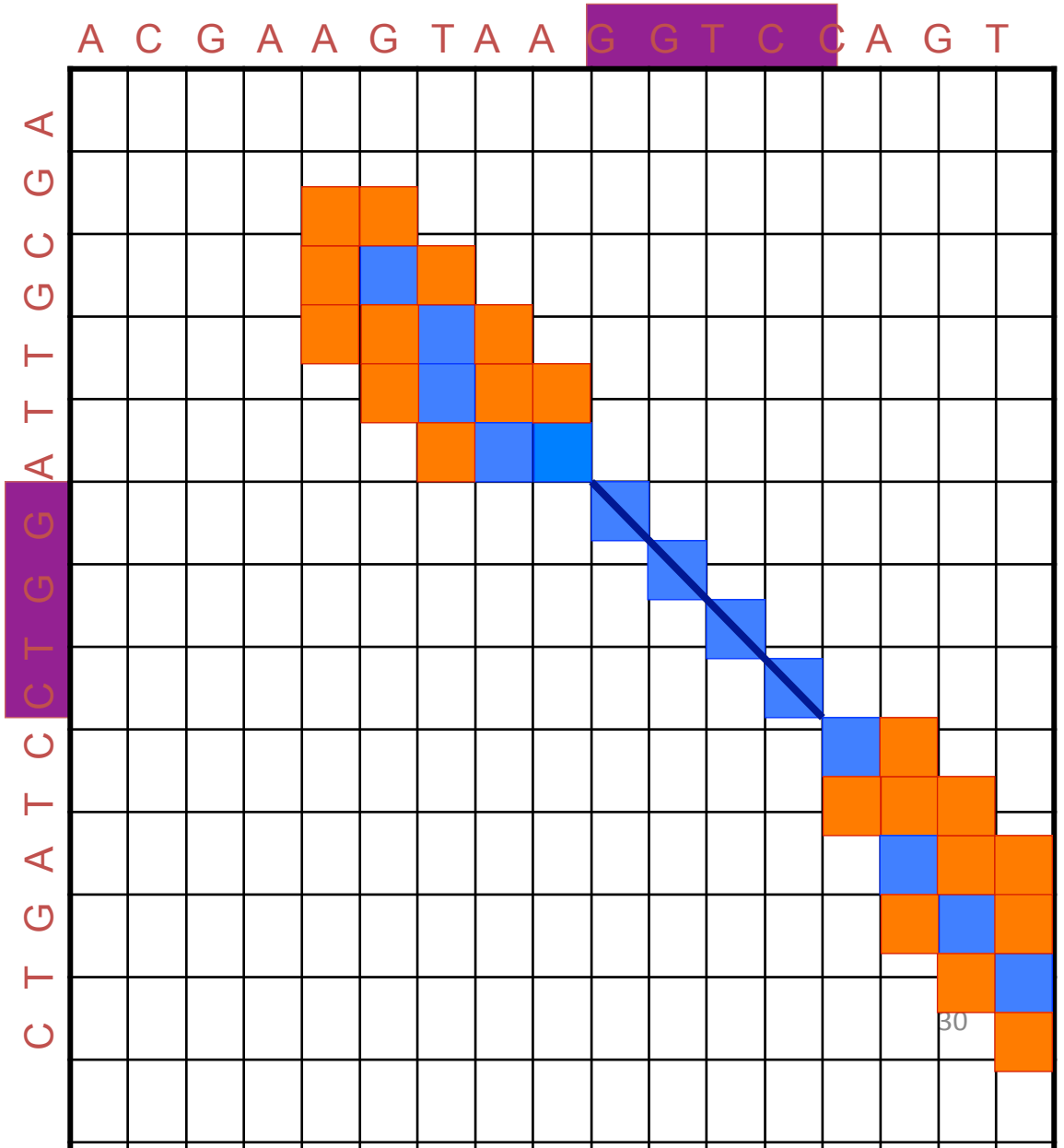


From lectures by Serafim Batzoglou (Stanford)

- Original BLAST exact keyword search, THEN:
- Extend with gaps around ends of exact match until score < *threshold*
- Output result

GTAAGGTCCAGT

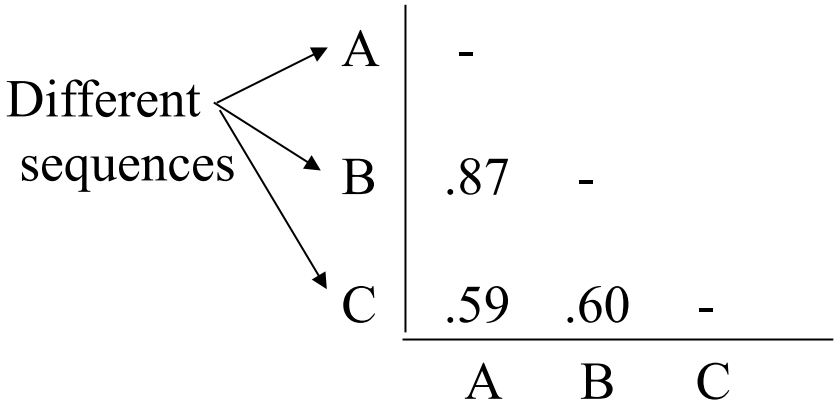
GTTAGGTC–AGT



From lectures by Serafim Batzoglou (Stanford)

Step 1-pairwise alignments

Compare each sequence with each other and calculate a Similarity matrix.

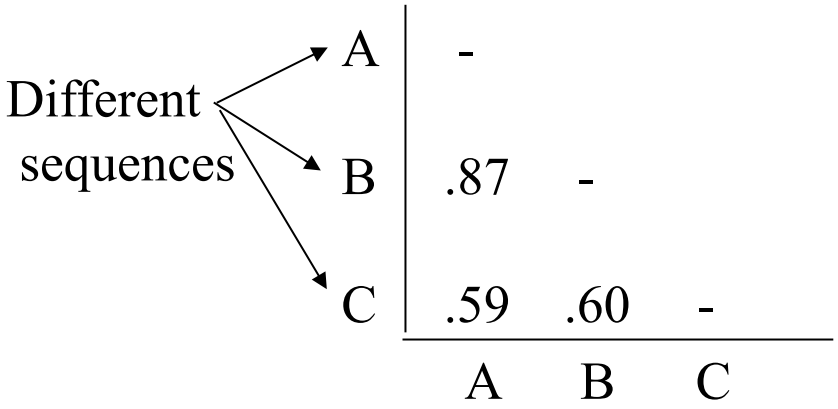


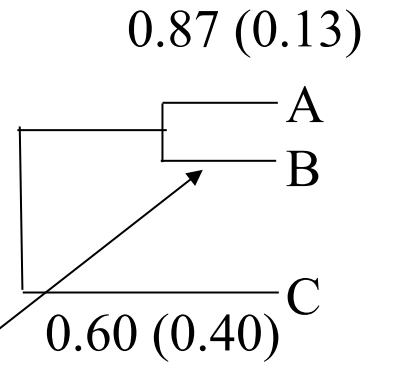
Each number represents the number of exact matches divided by the sequence length (ignoring gaps). Thus, the higher the number the more closely related the two sequences are.

In this similarity (distance) matrix sequence A is 87% identical to sequence B

Step 2-Create Guide Tree

Use the Similarity Matrix to create a Guide Tree to determine the "order" of the sequences.

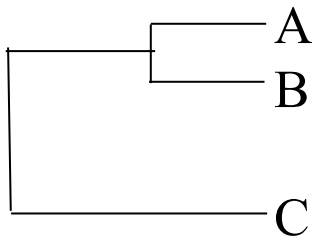




Guide Tree

Branch length proportional to estimated divergence between A and B (0.13)

Step 3-Progressive Alignment



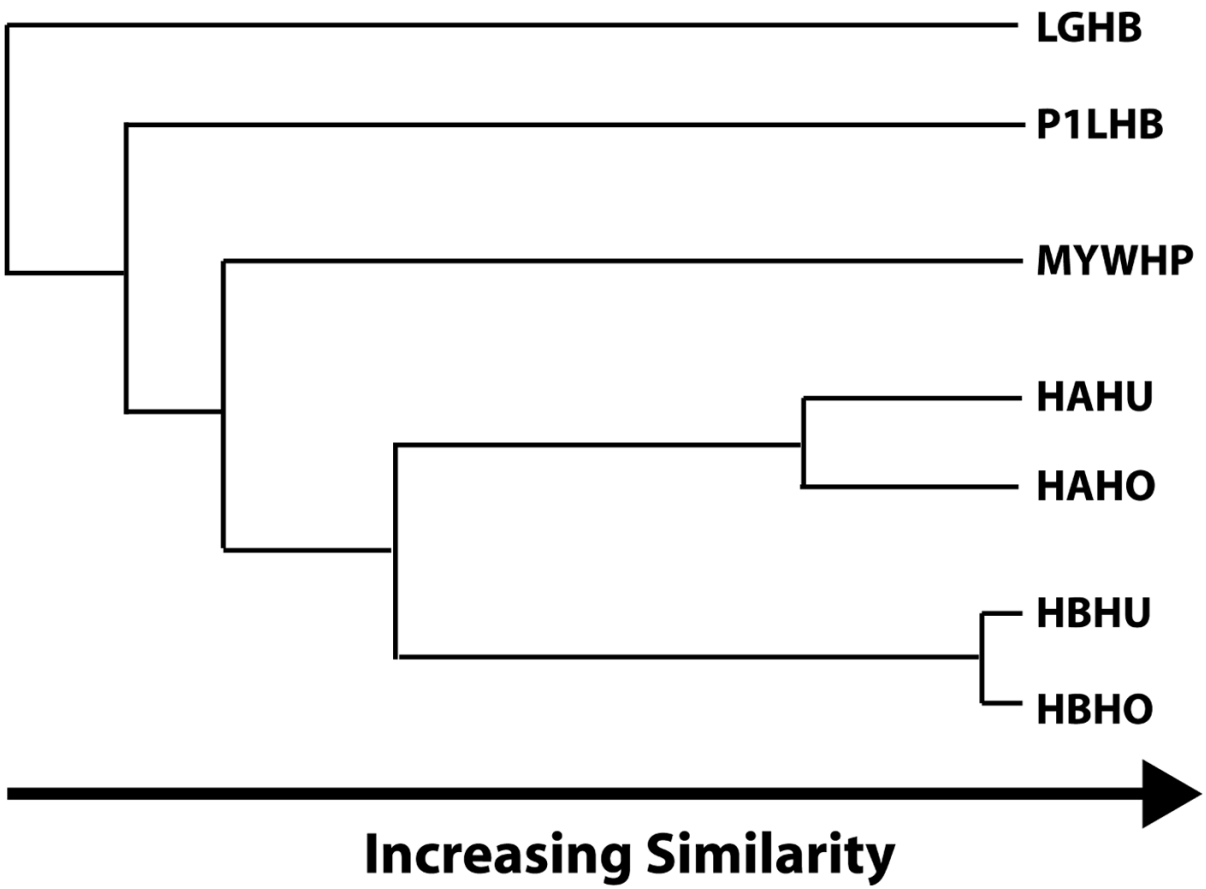
Guide Tree

Align A and B first. Then add sequence C to the previous alignment. In the closely aligned sequences and gaps are given a heavier weight than more divergent sequences.

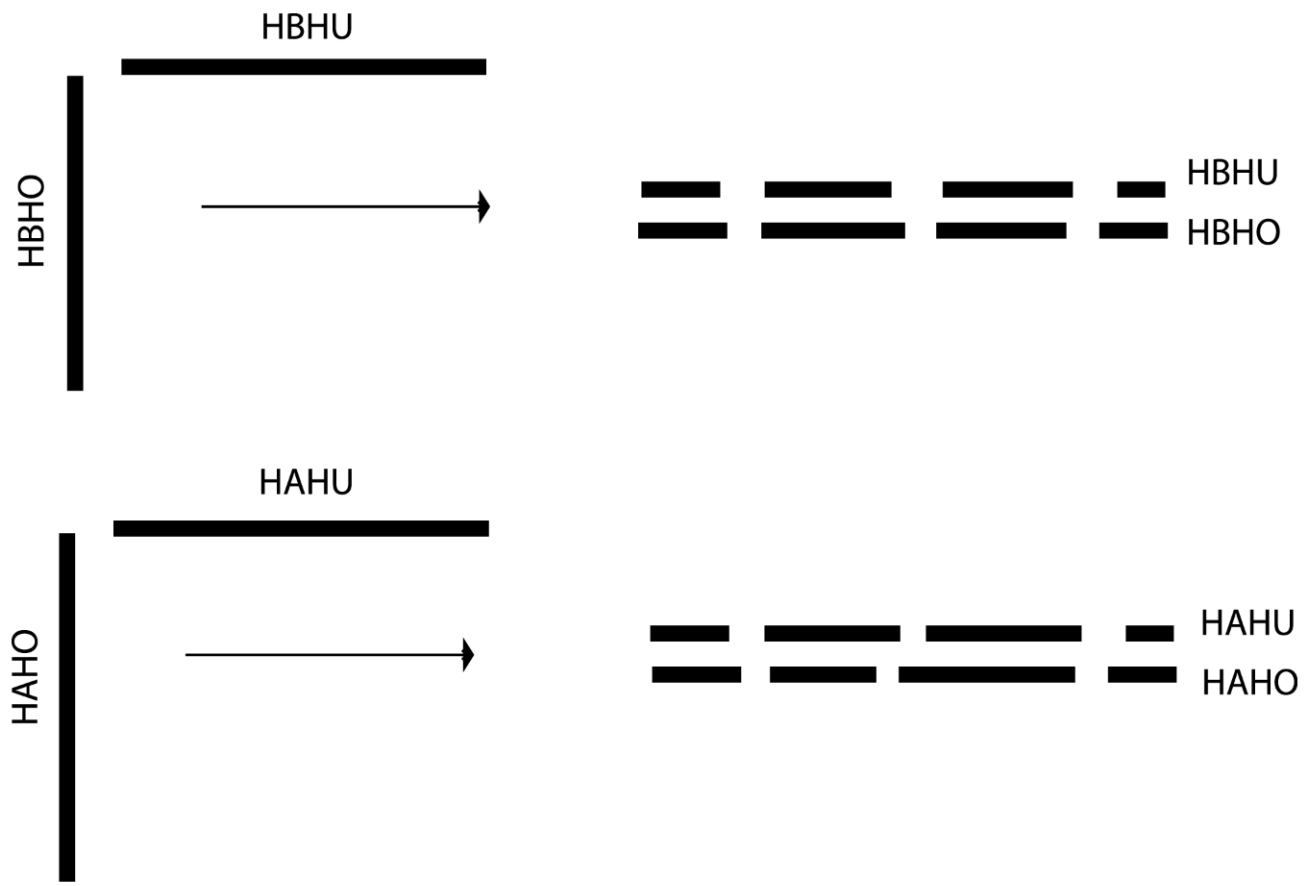
How does it work?

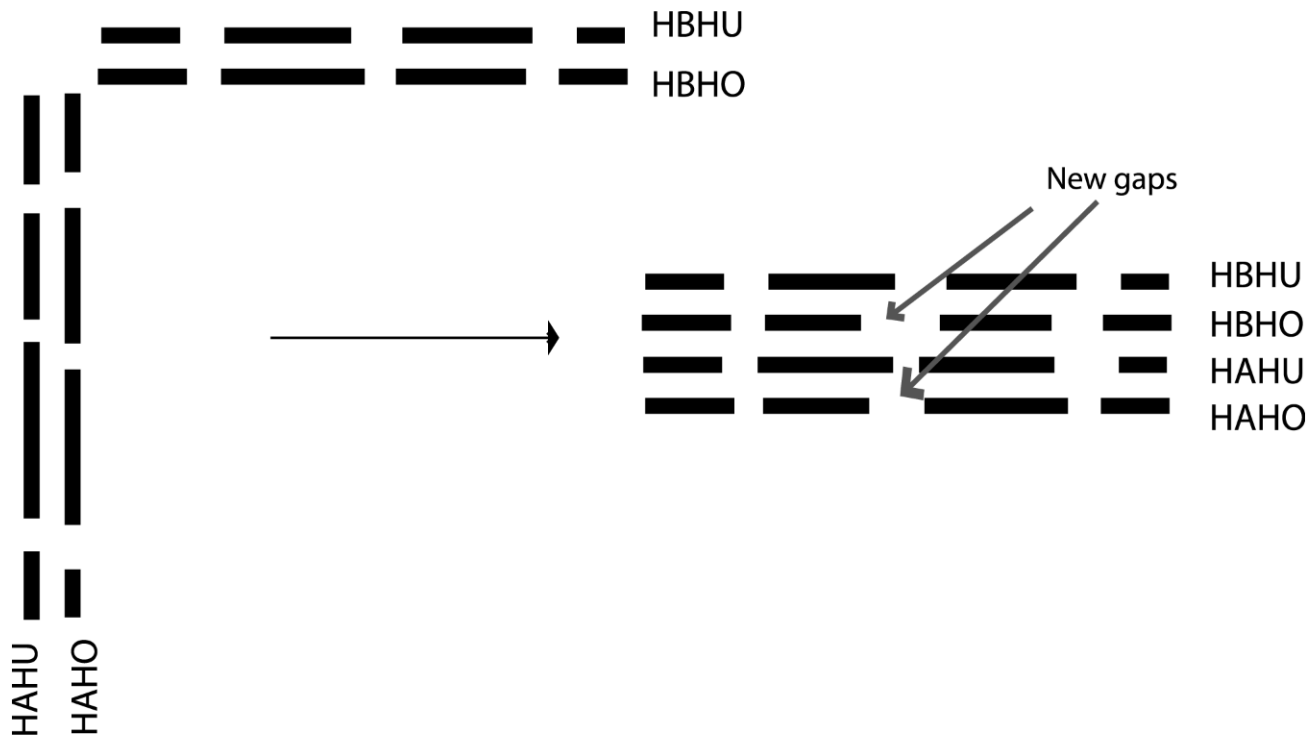
- Starting with a group of 7 sequences from different species
- Do pairwise alignments between all 7 sequences
- Score given for similarity, higher score indicates more similar

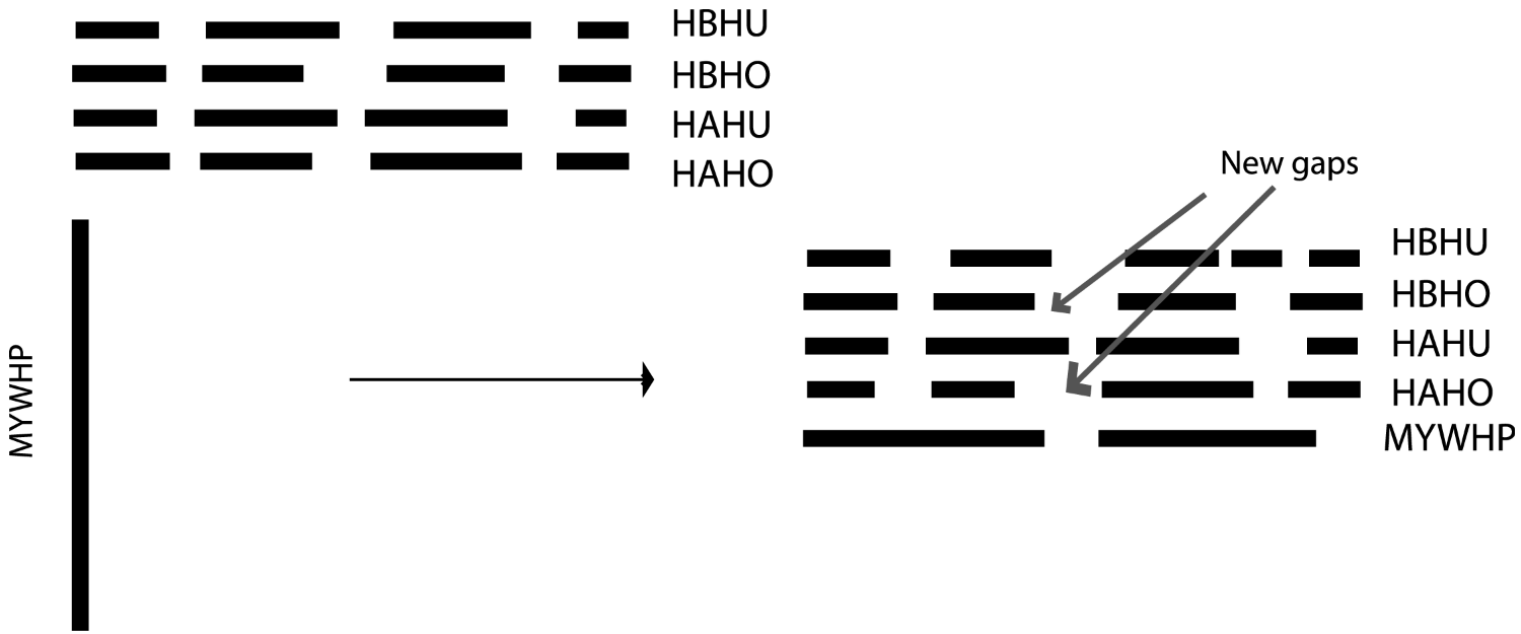
	HAHU	HBHU	HAHO	HBHO	MYWHP	P1LHB	LGHB
HAHU							
HBHU	21.1						
HAHO	32.9	19.7					
HBHO	20.7	**39.0**	20.4				
MYWHP	11.0	9.8	10.3	9.7			
P1LHB	9.3	8.6	9.6	8.4	7.0		
LGHB	7.1	7.3	7.5	7.4	7.3	4.3	



- Cluster the sequences by similarity to create a guide tree
- Branch length is proportional to estimated divergence between the two sequences


HBHU
HBHO
HBHU
HBHO
HAHU
HAHO
HAHU
HAHO


HBHU
HBHO
HAHU
HAHO
New gaps
HBHU
HBHO
HAHU
HAHO


HBHU
HBHO
HAHU
HAHO
MYWHP
New gaps
HBHU
HBHO
HAHU
HAHO
MYWHP

Michael Burrows

From Wikipedia, the free encyclopedia

This article is about the computer scientist. For the Church of Ireland (Anglican) bishop, see Michael Burrows (bishop).

Michael Burrows, FRS (b. 1963) is a British computer scientist and the creator of the Burrows–Wheeler transform.

Born in Britain, he now lives in the United States, although remaining a British citizen.

Burrows did his first degree in Electronic Engineering with Computer Science at University College London and then completed his PhD in the University of Cambridge Computer Laboratory at the University of Cambridge, where he was a member of Churchill College.

Upon leaving Cambridge, he worked at the Systems Research Center (SRC) at Digital Equipment Corporation (DEC) where, with Louis Monier, he was one of the two main creators of AltaVista.[1]

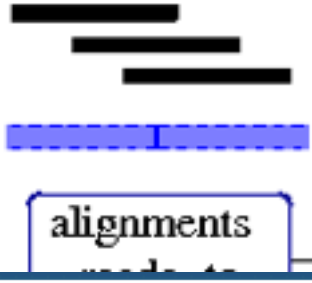
Following Compaq's acquisition of DEC, Burrows worked briefly for Microsoft.[2] Shortly thereafter he went to Google.[3]

After his early work at Cambridge University, where he researched micro-kernels and basic matters of security, he went on to enlarge upon that work as systems were deployed at large scale on the Internet.

During his employment at Google, Burrows has studied concurrency and synchronisation, and for programming in the large[*clarification needed*] – especially in respect the C++ language.[*citation needed*]

Burrows was elected a Fellow of the Royal Society in May 2013.[4]

http://bowtie-bio.sourceforge.net/bowtie2/index.shtml



Bowtie 2

Fast and sensitive read alignment

Bowtie 2 is an ultrafast and memory-efficient tool for aligning sequencing reads to long reference sequences. It is particularly good at aligning reads of about 50 up to 100s or 1,000s of characters, and particularly good at aligning to relatively long (e.g. mammalian) genomes. Bowtie 2 indexes the genome with an FM Index to keep its memory footprint small: for the human genome, its memory footprint is typically around 3.2 GB. Bowtie 2 supports gapped, local, and paired-end alignment modes.

OSI certified

Version 2.1.0 - February 21, 2013

- Improved multithreading support so that Bowtie 2 now uses native Windows threads when compiled on Windows and uses a faster mutex. Threading performance should improve on all platforms.
- Improved support for building 64-bit binaries for Windows x64 platforms.
- Bowtie 2 uses a lightweight mutex by default.
- Test option `--nospin` is no longer available. However bowtie2 can always be recompiled with `EXTRA_FLAGS="-DNO_SPINLOCK"` in order to drop the default spinlock usage.

Version 2.0.6 - January 27, 2013

- Fixed issue whereby spurious output would be written in `--no-unal` mode.
- Fixed issue whereby multiple input files combined with `--reorder` would cause truncated output and a memory spike.
- Fixed spinlock datatype for Win64 API (LLP64 data model) which made it crash when compiled under Windows 7 x64.
- Fixed bowtie2 wrapper to handle filename/paths operations in a more platform independent manner.
- Added pthread as a default library option under cygwin, and pthreadGC for MinGW.
- Fixed some minor issues that made MinGW compilation fail.

Version 2.0.5 - January 4, 2013

Site Map

Home
News archive
Manual
Getting started
Frequently Asked Questions
Tools that use Bowtie

Latest Release

Bowtie2 2.1.0 2/21/13

Please cite: Langmead B, Salzberg S. Fast gapped-read alignment with Bowtie 2. *Nature Methods*. 2012, 9:357-359.

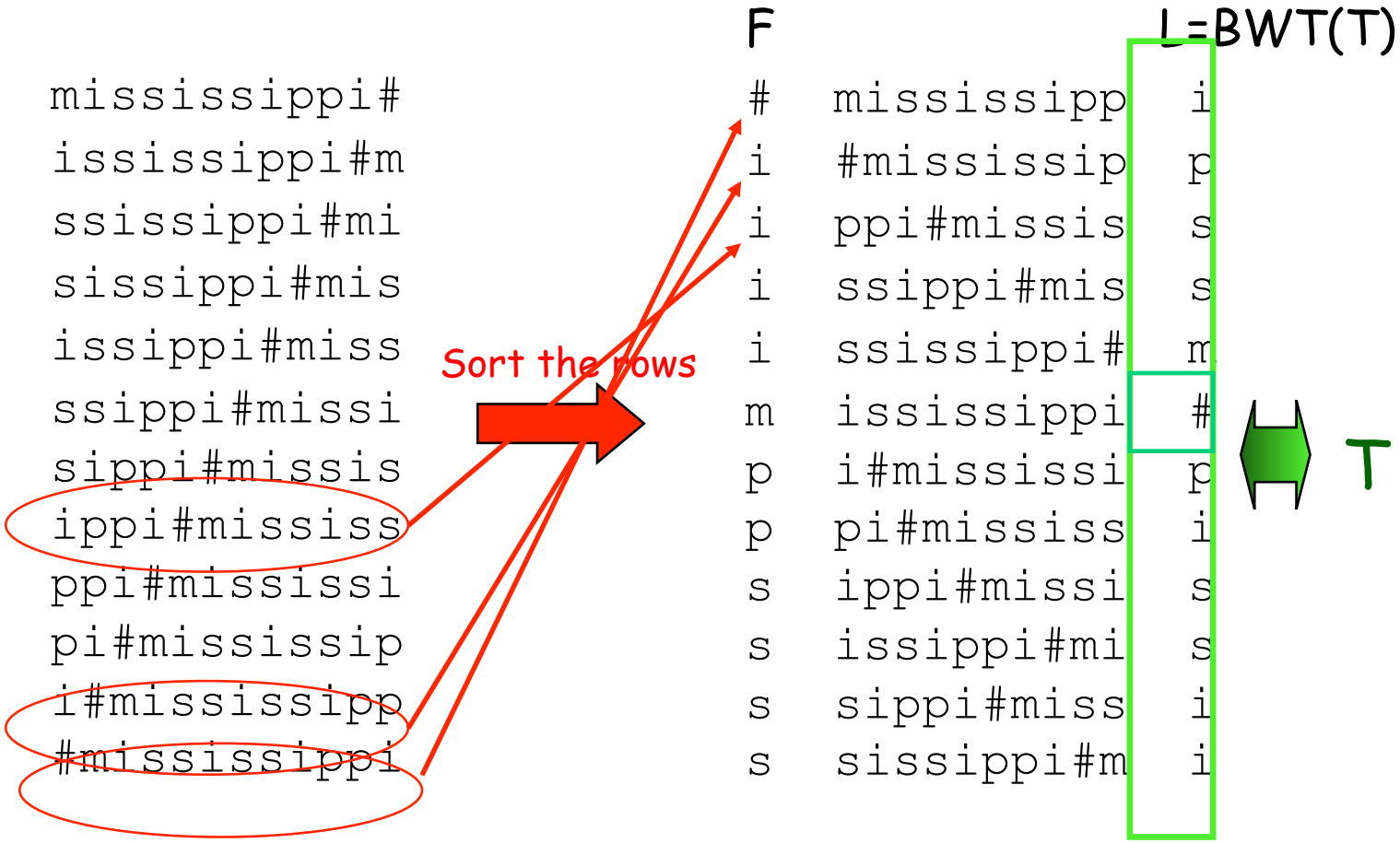
Related Tools

Bowtie: Ultrafast short read alignment
Crossbow: Genotyping, cloud computing
Myrna: Cloud, differential gene expression

Slide from Haim Kaplan and Elad Verbin
Tel-Aviv University

The BWT

T = mississippi#



BWT sorts the characters by their post-context

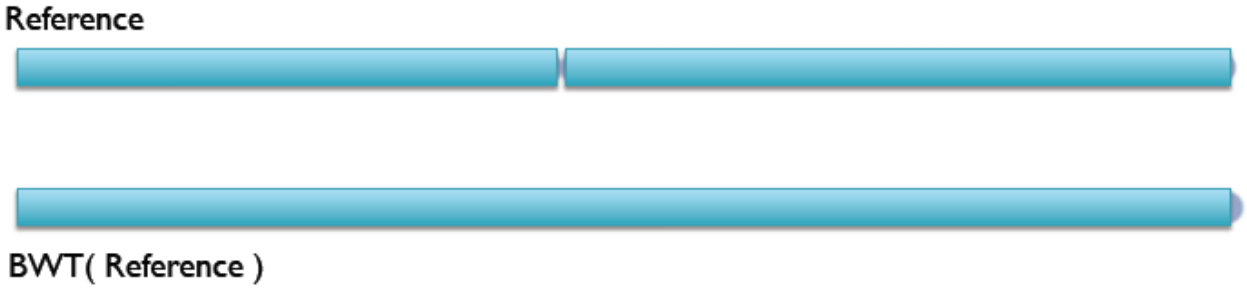
More Burrows-Wheeler

Input	SIX.MIXED.PIXIES.SIFT.SIXTY.PIXIE.DUST.BOXES
Burrows-Wheeler Output	TEXYDST.E.IXIXIXXSSMPPS.B..E.S.EUSFXDIIOIIIT

Repeated characters mean that it is easier to compress

From Dr Konrad Paszkiewicz

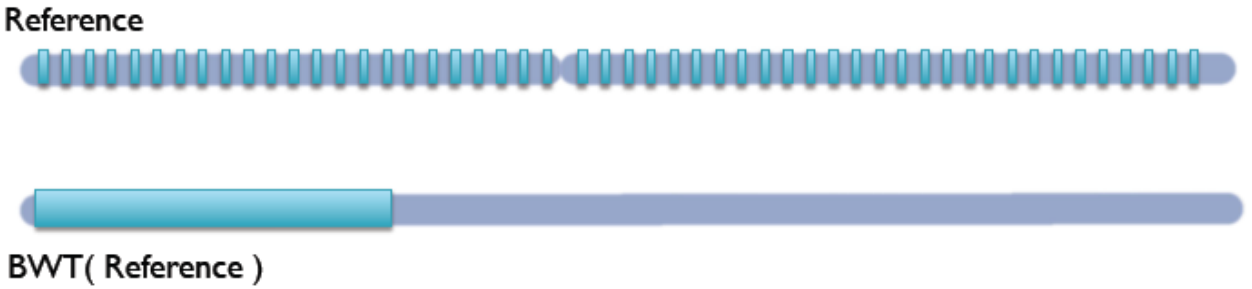
Bowtie/Soap2 example



Query:
AATGATACGGCGACCACCGAGATCTA

From Dr Konrad Paszkiewicz

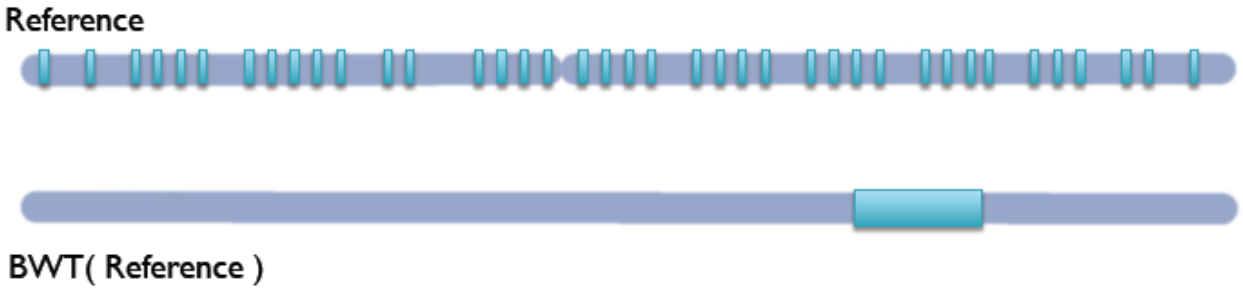
Bowtie/Soap2 example



Query:
AATGATACGGCGACCACCGAGATCTA

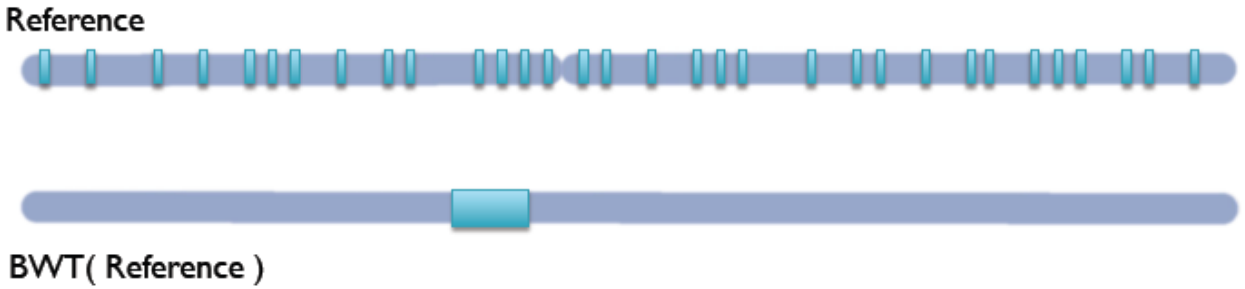
From Dr Konrad Paszkiewicz

Bowtie/Soap2 example



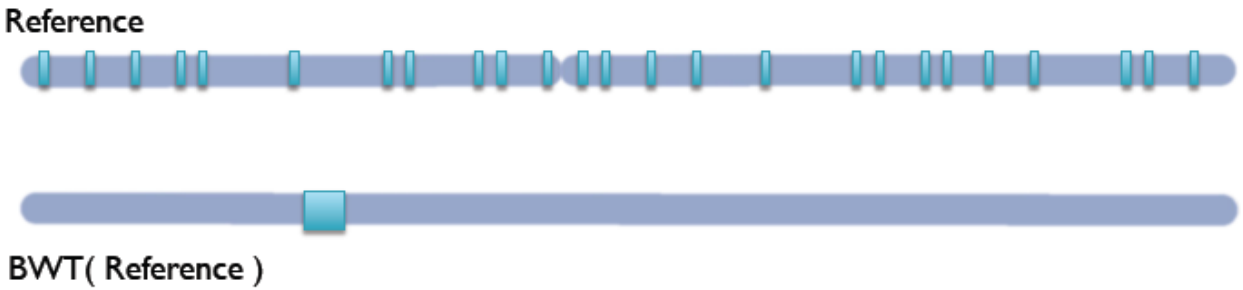
From Dr Konrad Paszkiewicz

Bowtie/Soap2 example



From Dr Konrad Paszkiewicz

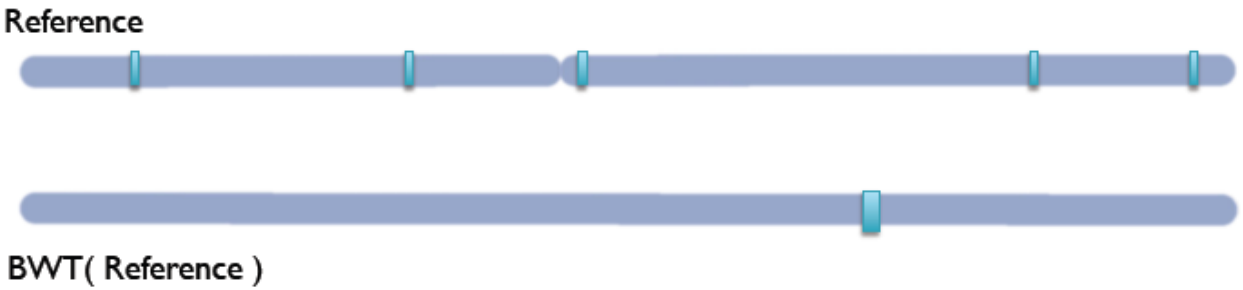
Bowtie/Soap2 example



Query:
AATGATACGGCGACCACCGAGATCTA

From Dr Konrad Paszkiewicz

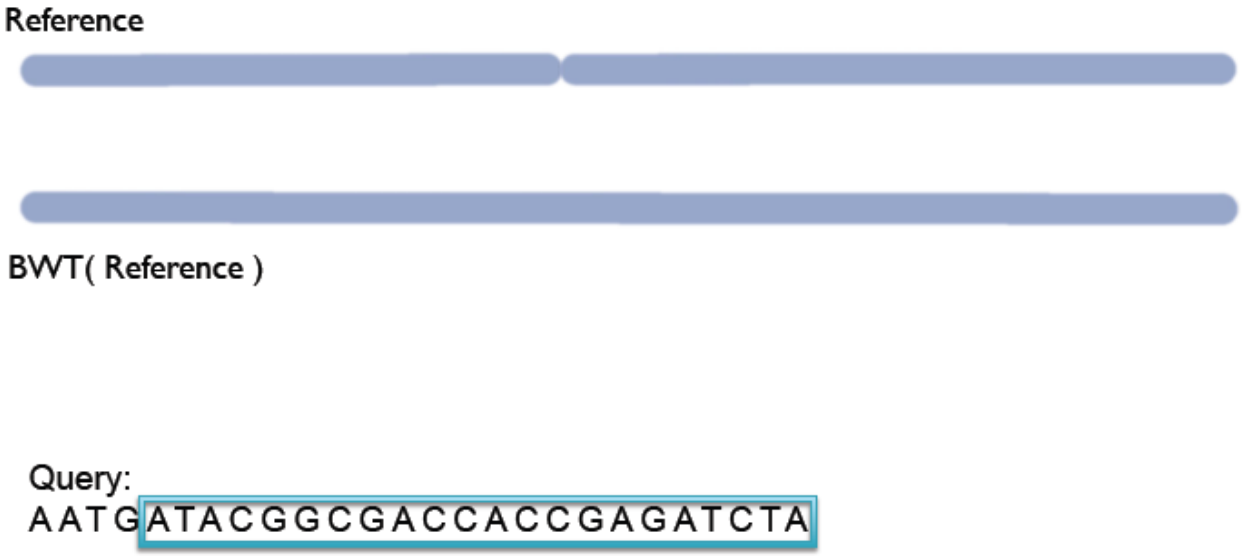
Bowtie/Soap2 example



Query:
AATGATACGGCGACCACCGAGATCTA

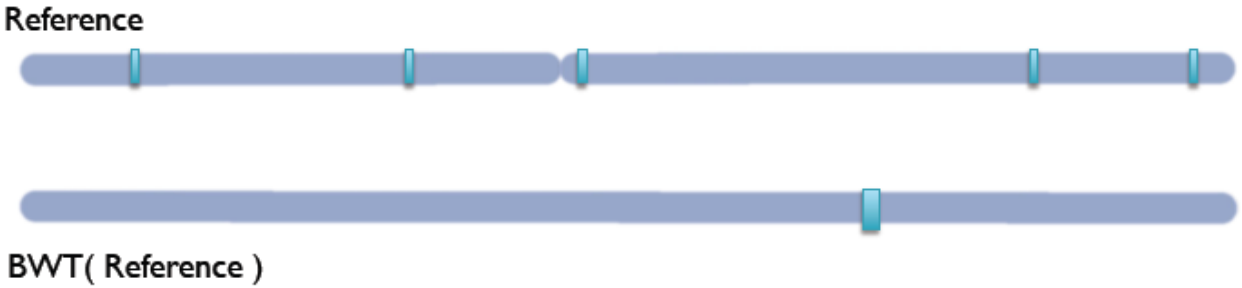
From Dr Konrad Paszkiewicz

Bowtie/Soap2 example



From Dr Konrad Paszkiewicz

Bowtie/Soap2 example



Query:
AATGTTACGGCGACCACCGAGATCTA

From Dr Konrad Paszkiewicz

Bowtie/Soap2 example

Query:

AATGTTACGGCGACCACCGAGATCTA

From Dr Konrad Paszkiewicz

A Euler path is a path that crosses every edge exactly once without repeating,
if it ends at the initial vertex then it is a Euler cycle.

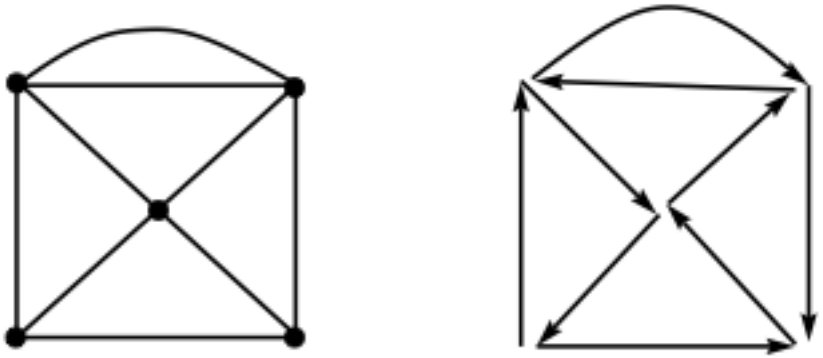
A Hamiltonian path passes through each vertex (note not each edge), exactly once,
if it ends at the initial vertex then it is a Hamiltonian cycle (or circuit).

In a Euler path you might pass through a vertex more than once.

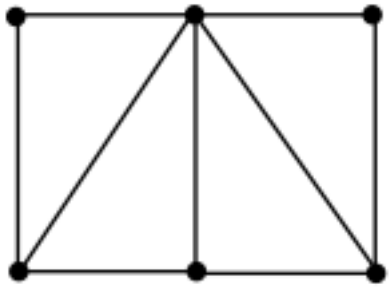
In a Hamiltonian path you may not pass though all edges.

An Eulerian circuit traverses every edge in a graph exactly once, but may repeat vertices,
while a Hamiltonian circuit visits each vertex in a graph exactly once but may repeat edges

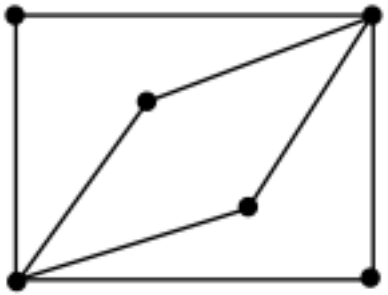
de Bruijn graph is a graph representing overlaps between kmers



Example of an Eulerian graph (left); possible solution trail on the right, starting bottom left corner



Hamiltonian and non-Eulerian



Eulerian and non-Hamiltonian

de Bruijn graph assemblers

- Oases,
- ABySS,
- SOAPdenovo,
- Velvet, MetaVelvet,
- MetaIDBA,
- Trinity

CSH PRESS GENOME RESEARCH

MOLECULAR CLONING 4

Velvet: Algorithms for de novo short read assembly using de Bruijn graphs

Daniel R. Zerbino and Ewan Birney[1]

[+] Author Affiliations

Abstract

We have developed a new set of algorithms, collectively called "Velvet," to manipulate de Bruijn graphs for genomic sequence assembly. A de Bruijn graph is a compact representation based on short words (*k*-mers) that is ideal for high coverage, very short read (25–50 bp) data sets. Applying Velvet to very short reads and paired-ends information only, one can produce contigs of significant length, up to 50-kb N50 length in simulations of prokaryotic data and 3-kb N50 on simulated mammalian BACs. When applied to real Solexa data sets without read pairs, Velvet generated contigs of ~8 kb in a prokaryote and 2 kb in a mammalian BAC, in close agreement with our simulated results without read-pair information. Velvet represents a new approach to assembly that can leverage very short reads in combination with read pairs to produce useful assemblies.

« Previous | Next Article »
Table of Contents

This Article

Published in Advance March 18, 2008, doi: 10.1101/gr.074492.107

Genome Res. 2008. 18: 821–829

- Abstract ***Free***
- » Full Text ***Free***
- Full Text (PDF) ***Free***
- Supplemental Research Data

All Versions of this Article:
- gr.074492.107v1
- gr.074492.107v2
- 18/5/821 ***most recent***

[−] **Article Category**

RESOURCE

[+] **Services**

[+] **Citing Articles**

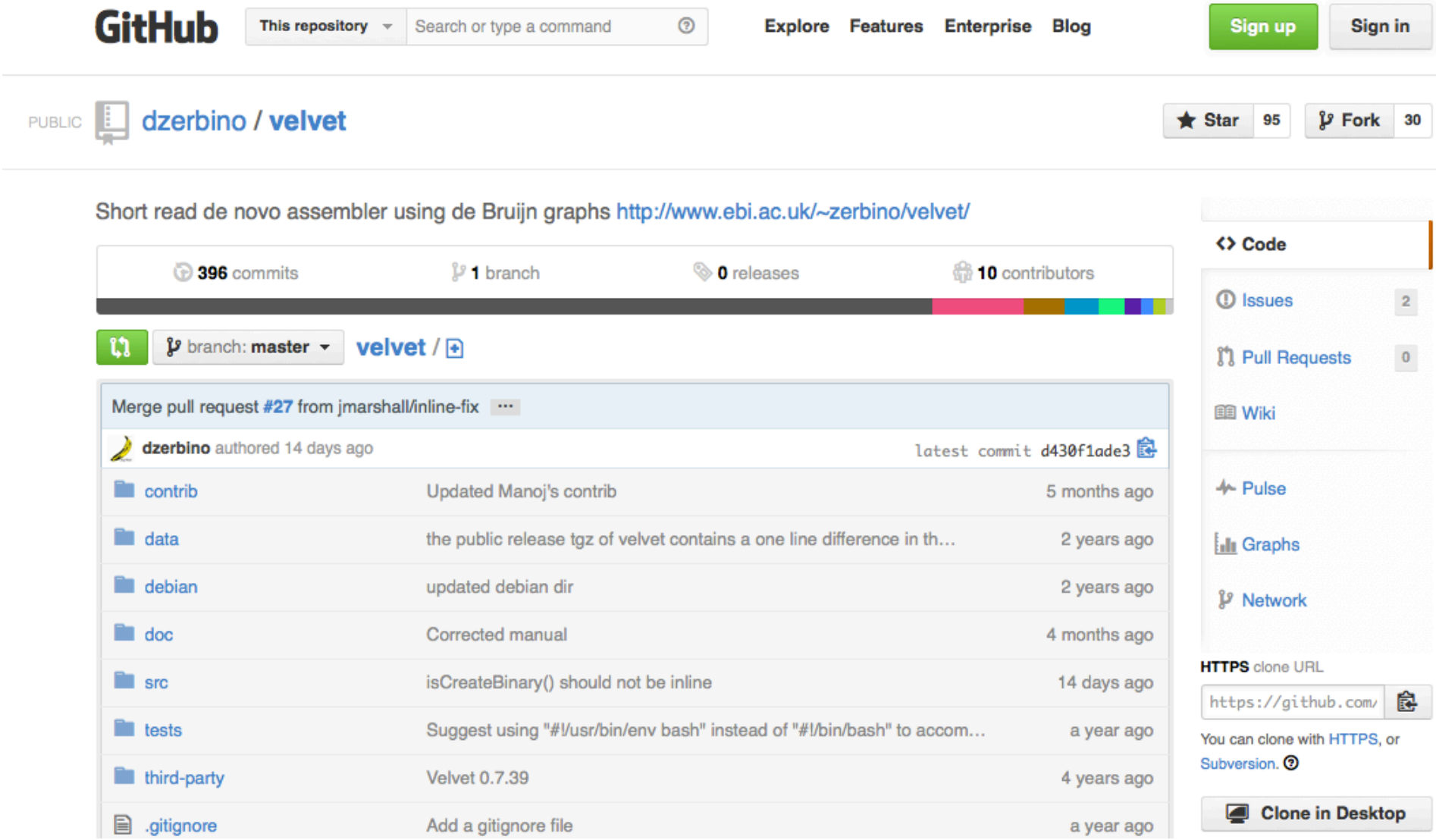
Current Issue

October 2013, 23 (10)

[+] **From the Cover**

Alert me to new issues of *Genome Research*

- **Advance Online Articles**
- **Submit a Manuscript**
- **GR in the News**
- **Editorial Board**
- **Permissions**
- **E-mail Alerts & RSS Feeds**


GitHub
This repository
Search or type a command
Explore
Features
Enterprise
Blog
Sign up
Sign in
PUBLIC
dzerbino / velvet
Star 95
Fork 30
Short read de novo assembler using de Bruijn graphs http://www.ebi.ac.uk/~zerbino/velvet/
396 commits
1 branch
0 releases
10 contributors
branch: master
velvet /
Merge pull request #27 from jmarshall/inline-fix
dzerbino authored 14 days ago
latest commit d430f1ade3
contrib Updated Manoj's contrib 5 months ago
data the public release tgz of velvet contains a one line difference in th… 2 years ago
debian updated debian dir 2 years ago
doc Corrected manual 4 months ago
src isCreateBinary() should not be inline 14 days ago
tests Suggest using "#!/usr/bin/env bash" instead of "#!/bin/bash" to accom… a year ago
third-party Velvet 0.7.39 4 years ago
.gitignore Add a gitignore file a year ago
Code
Issues 2
Pull Requests 0
Wiki
Pulse
Graphs
Network
HTTPS clone URL
https://github.com/
You can clone with HTTPS, or Subversion.
Clone in Desktop

```
def de_bruijn(k, n):
    """
    De Bruijn sequence for alphabet size k
    and subsequences of length n.
    """
    a = [0] * k * n
    sequence = []
    def db(t, p):
        if t > n:
            if n % p == 0:
                for j in range(1, p + 1):
                    sequence.append(a[j])
        else:
            a[t] = a[t - p]
            db(t + 1, p)
            for j in range(a[t - p] + 1, k):
                a[t] = j
                db(t + 1, t)
    db(1, 1)
    return sequence

print(de_bruijn(2, 3))
```

which prints

```
[0, 0, 0, 1, 0, 1, 1, 1]
```

Parsimony Example

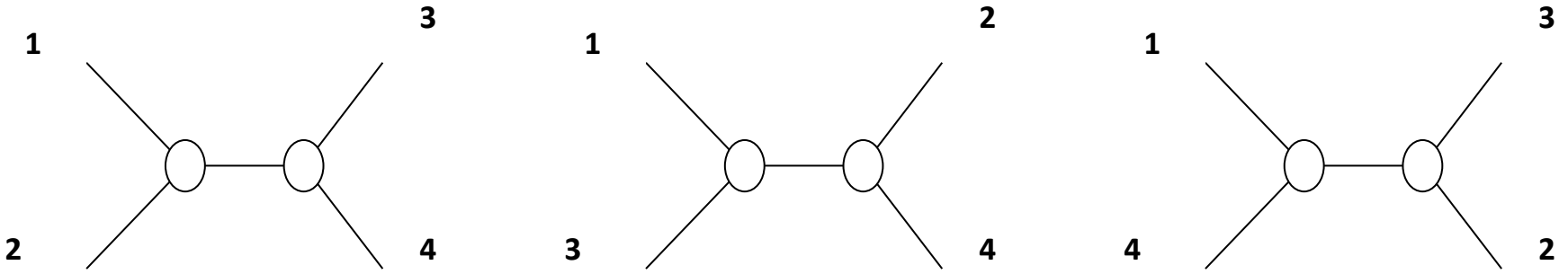
1 A A G A G T G C A

2 A G C C G T G C G

3 A G A T A T C C A

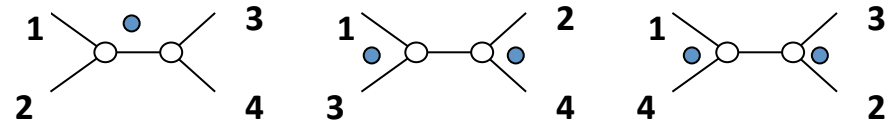
4 A G A G A T C C G

- four sequences, three possible unrooted trees

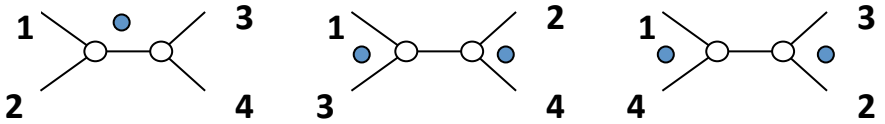


Maximum Parsimony Example

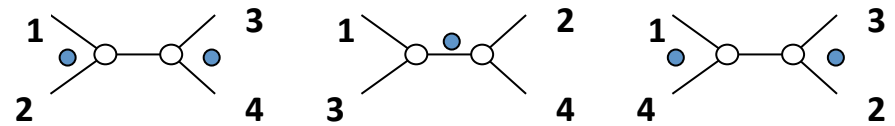
1	G	G	A
2	G	G	G
3	A	C	A
4	A	C	G



Column 1



Column 2



Column 3

Tree 1: 4

Tree 2: 5

Tree 3: 6

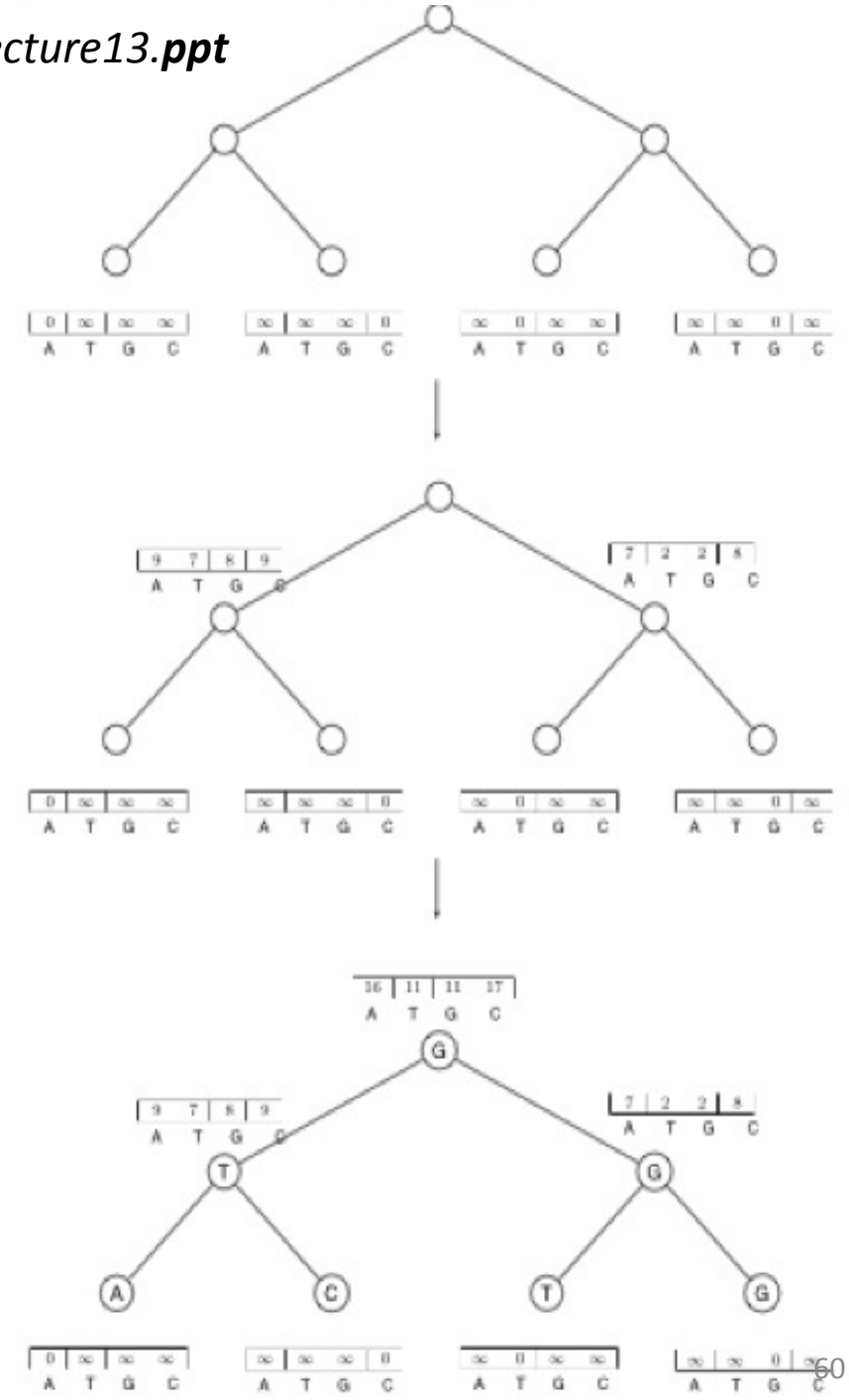
- Is a substitution

*www.cs.uiuc.edu/class/fa07/cs498ss/lectures/lecture13.**ppt***

Sankoff's Algorithm

- Check children's every vertex and determine the minimum between them
- An example

δ	A	T	G	C
A	0	3	4	9
T	3	0	2	4
G	4	2	0	4
C	9	4	4	0

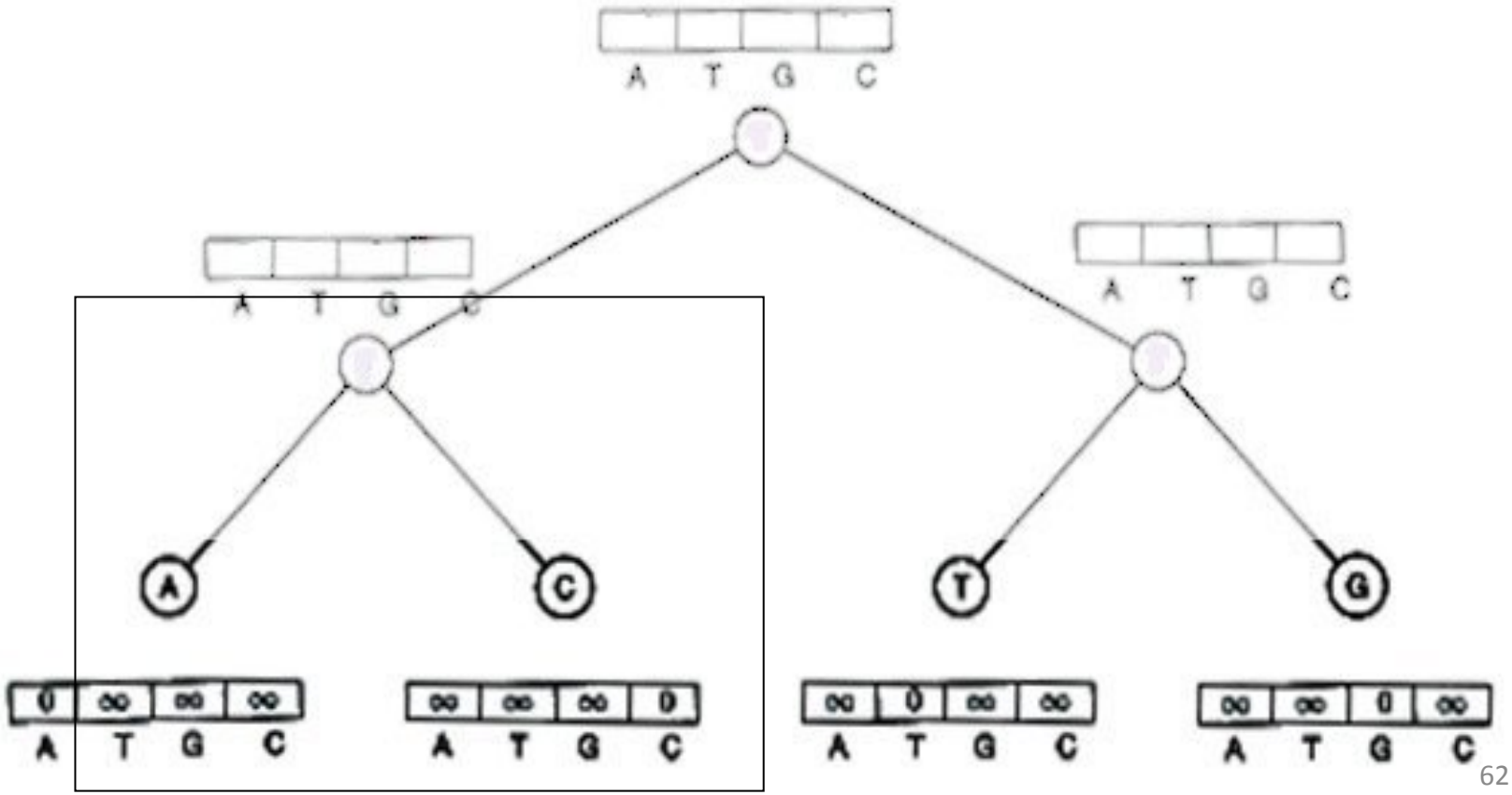


Sankoff Algorithm: Dynamic Programming

- Calculate and keep track of a score for every possible label at each vertex
  - $s_t(v)$ = minimum parsimony score of the **subtree** rooted at vertex *v* if *v* has character *t*
- The score at each vertex is based on scores of its children:
  - $s_t(\textbf{\textit{parent}}) = \min_i \{s_i(\textbf{\textit{ left child }}) + \delta_{i,t}\} + \min_j \{s_j(\textbf{\textit{ right child }}) + \delta_{j,t}\}$

Sankoff Algorithm (cont.)

- Begin at leaves:
  - If leaf has the character in question, score is 0
  - Else, score is ∞

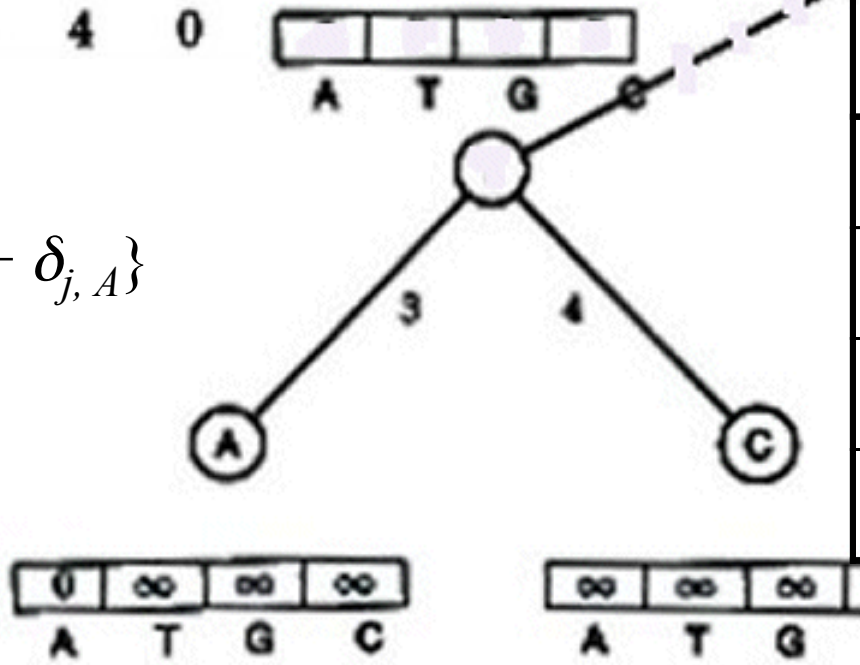


Sankoff Algorithm (cont.)

δ	A	T	G	C
A	0	3	4	9
T	3	0	2	4
G	4	2	0	4
C	9	4	4	0

$$s_t(v) = \min_i \{s_i(u) + \delta_{i,\,t}\} + \min_j \{s_j(w) + \delta_{j,\,t}\}$$

$$s_A(v) = 0 + \min_j \{s_j(w) + \delta_{j,\,A}\}$$



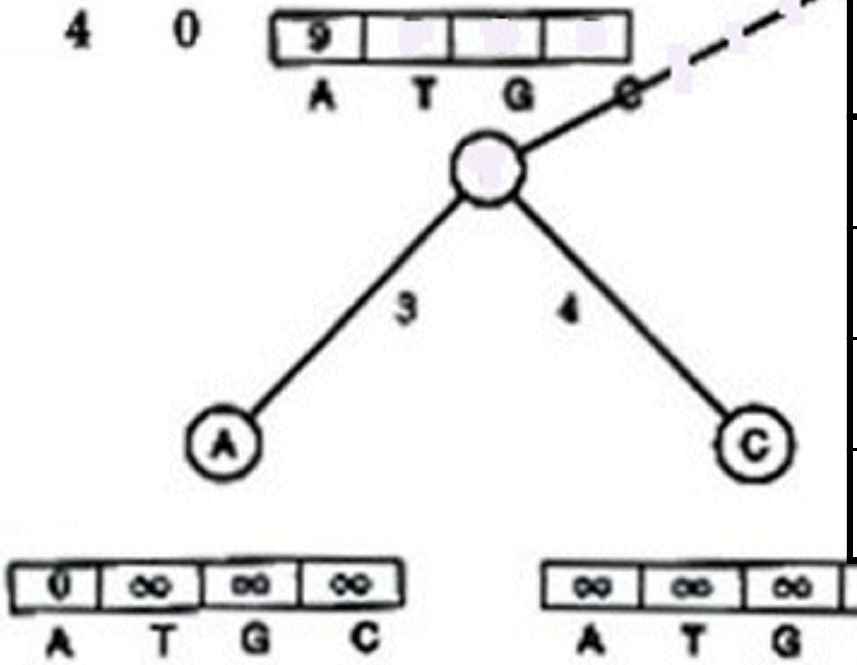
	$s_i(u)$	$\delta_{i,A}$	sum
A	0	0	**0**
T	∞	3	∞
G	∞	4	∞
C	∞	9	∞

Sankoff Algorithm (cont.)

δ	A	T	G	C
A	0	3	4	9
T	3	0	2	4
G	4	2	0	4
C	9	4	4	0

$$s_t(v) = \min_i \{s_i(u) + \delta_{i,\,t}\} + \min_j \{s_j(w) + \delta_{j,\,t}\}$$

$s_A(v) = 0$
$+ 9 = 9$



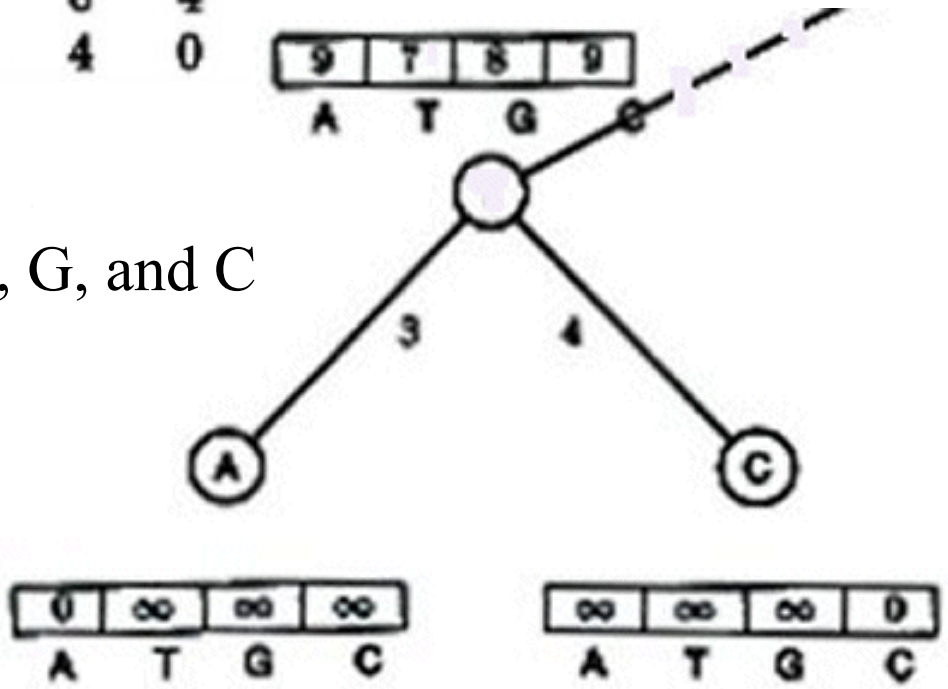
	$s_j(u)$	$\delta_{j,\,A}$	sum
A	∞	0	∞
T	∞	3	∞
G	∞	4	∞
C	0	9	**9**

Sankoff Algorithm (cont.)

δ	A	T	G	C
A	0	3	4	9
T	3	0	2	4
G	4	2	0	4
C	9	4	4	0

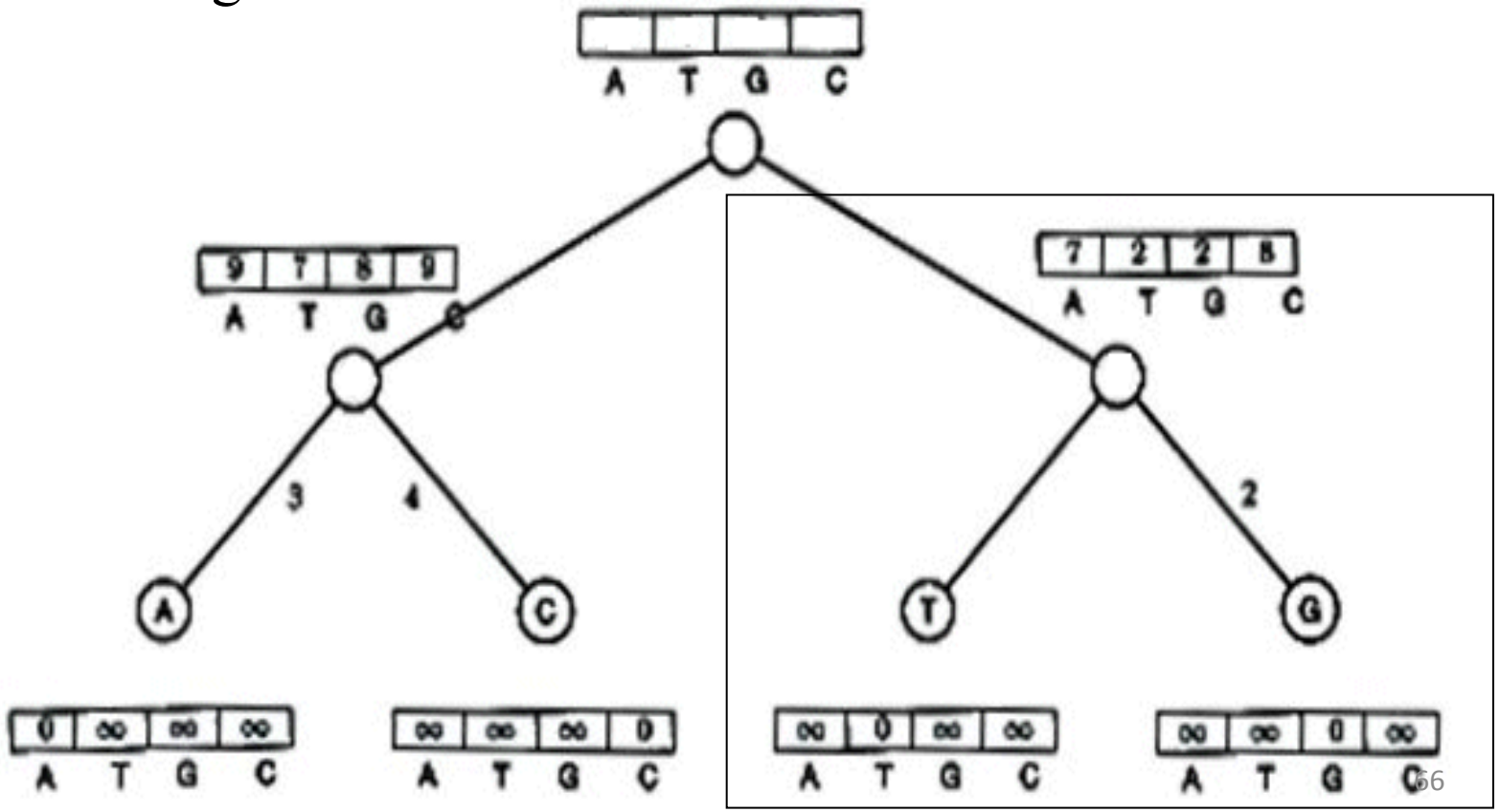
$$s_t(v) = \min_i \{s_i(u) + \delta_{i,t}\} + \min_j \{s_j(w) + \delta_{j,t}\}$$

Repeat for T, G, and C



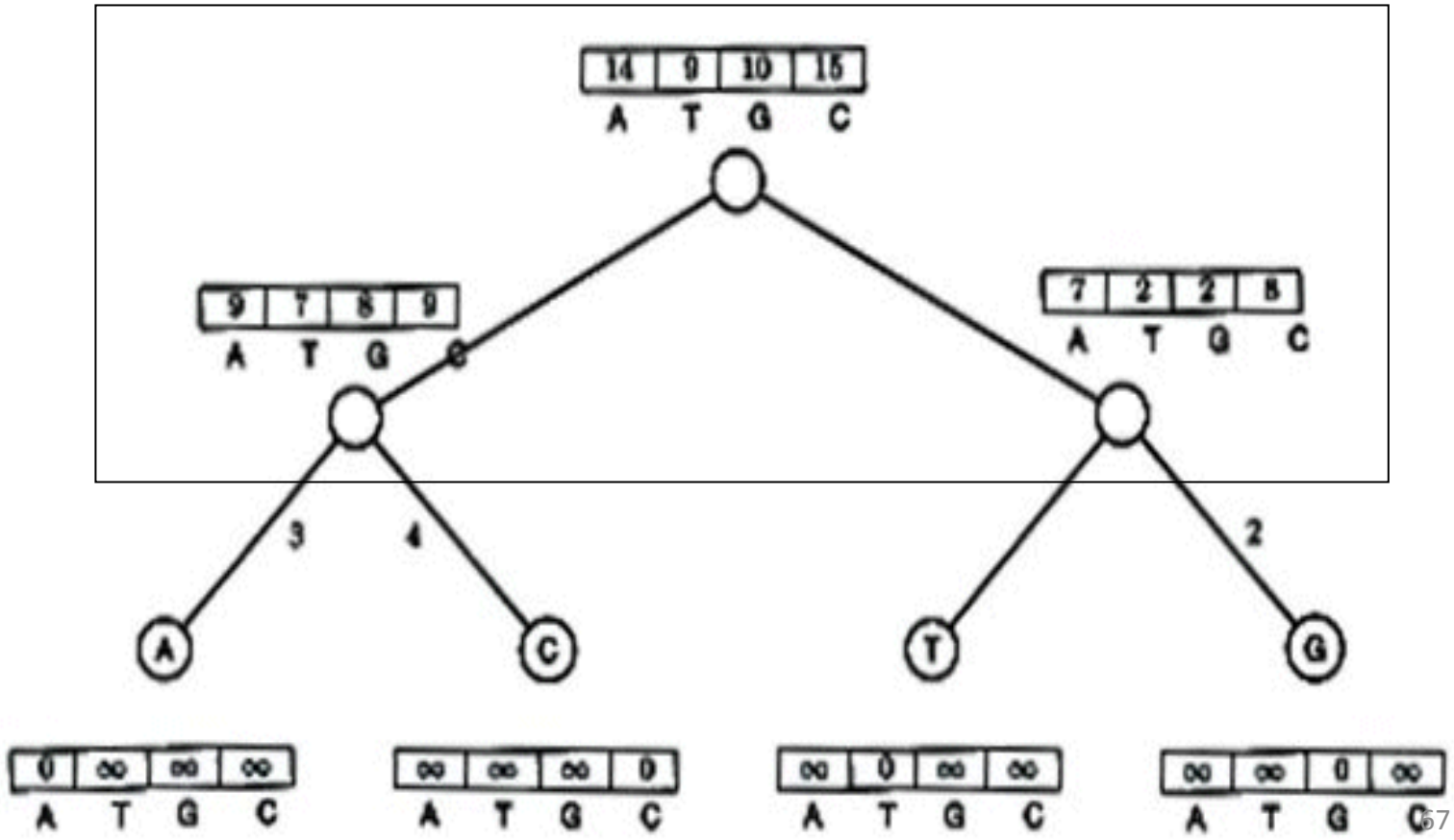
Sankoff Algorithm (cont.)

Repeat for right subtree



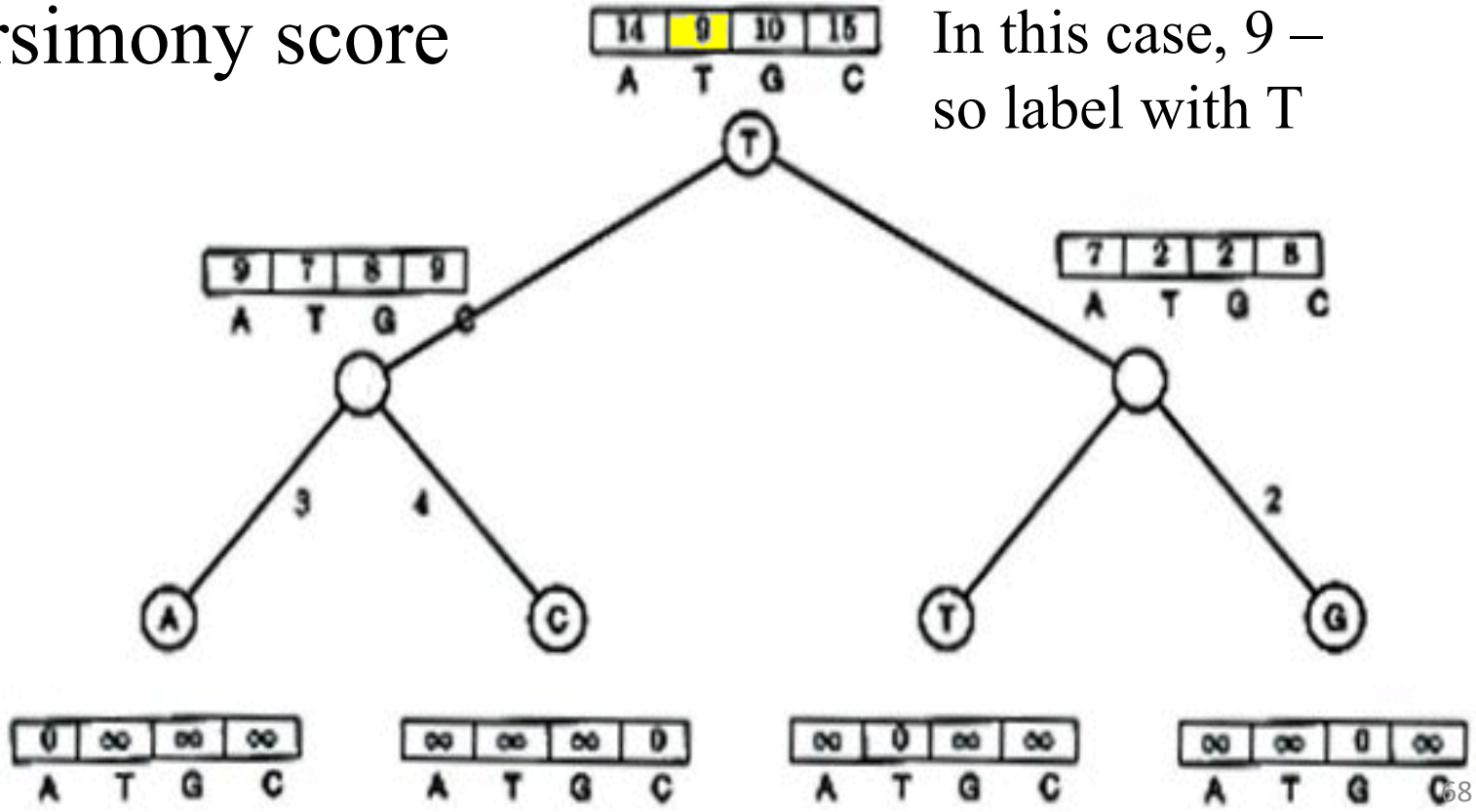
Sankoff Algorithm (cont.)

Repeat for root



Sankoff Algorithm (cont.)

Smallest score at root is minimum weighted parsimony score



In this case, 9 – so label with T

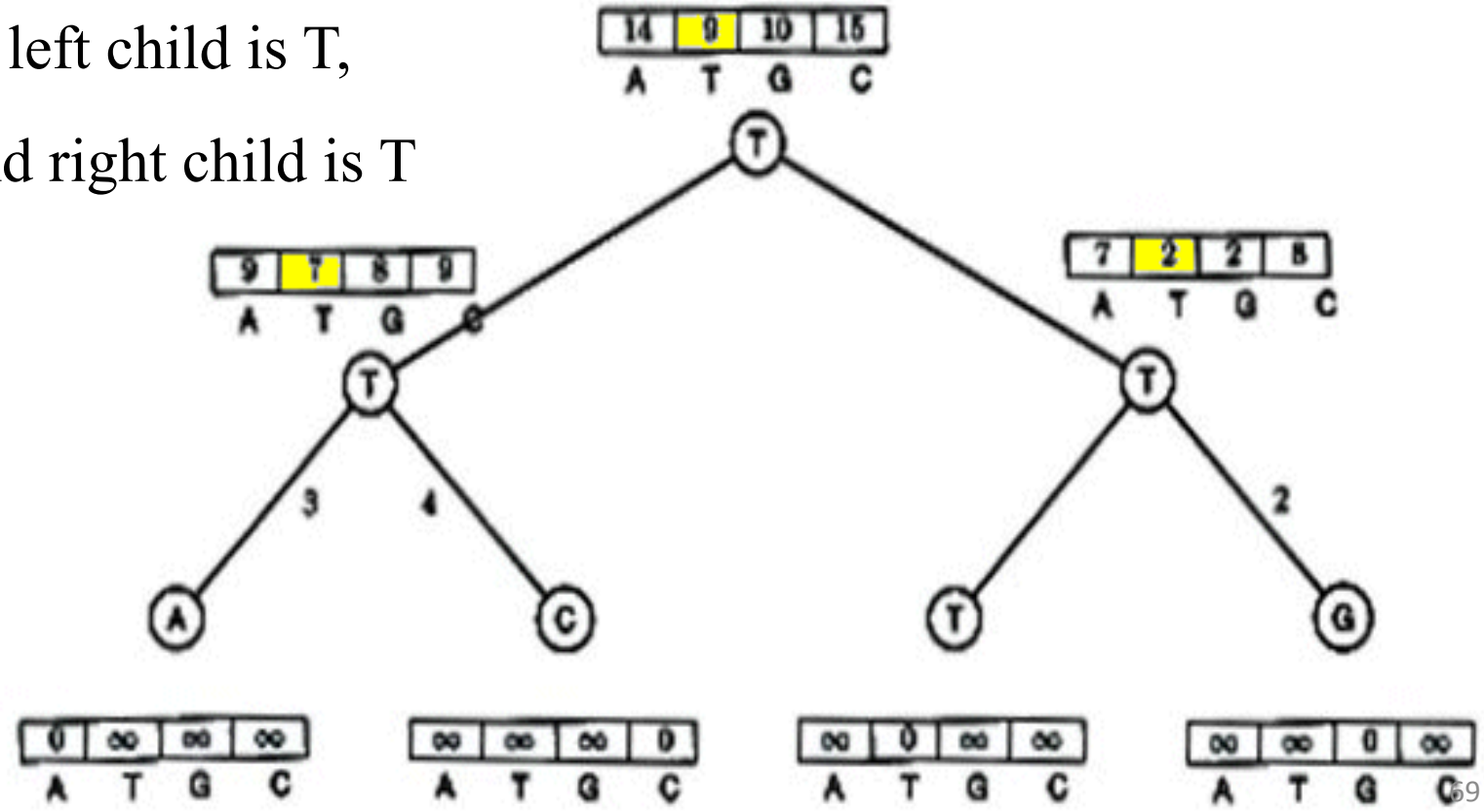
Sankoff Algorithm: Traveling down the Tree

The scores at the root vertex have been computed by going up the tree
After the scores at root vertex are computed the Sankoff algorithm moves down the tree and assign each vertex with optimal character.

9 is derived from 7 + 2

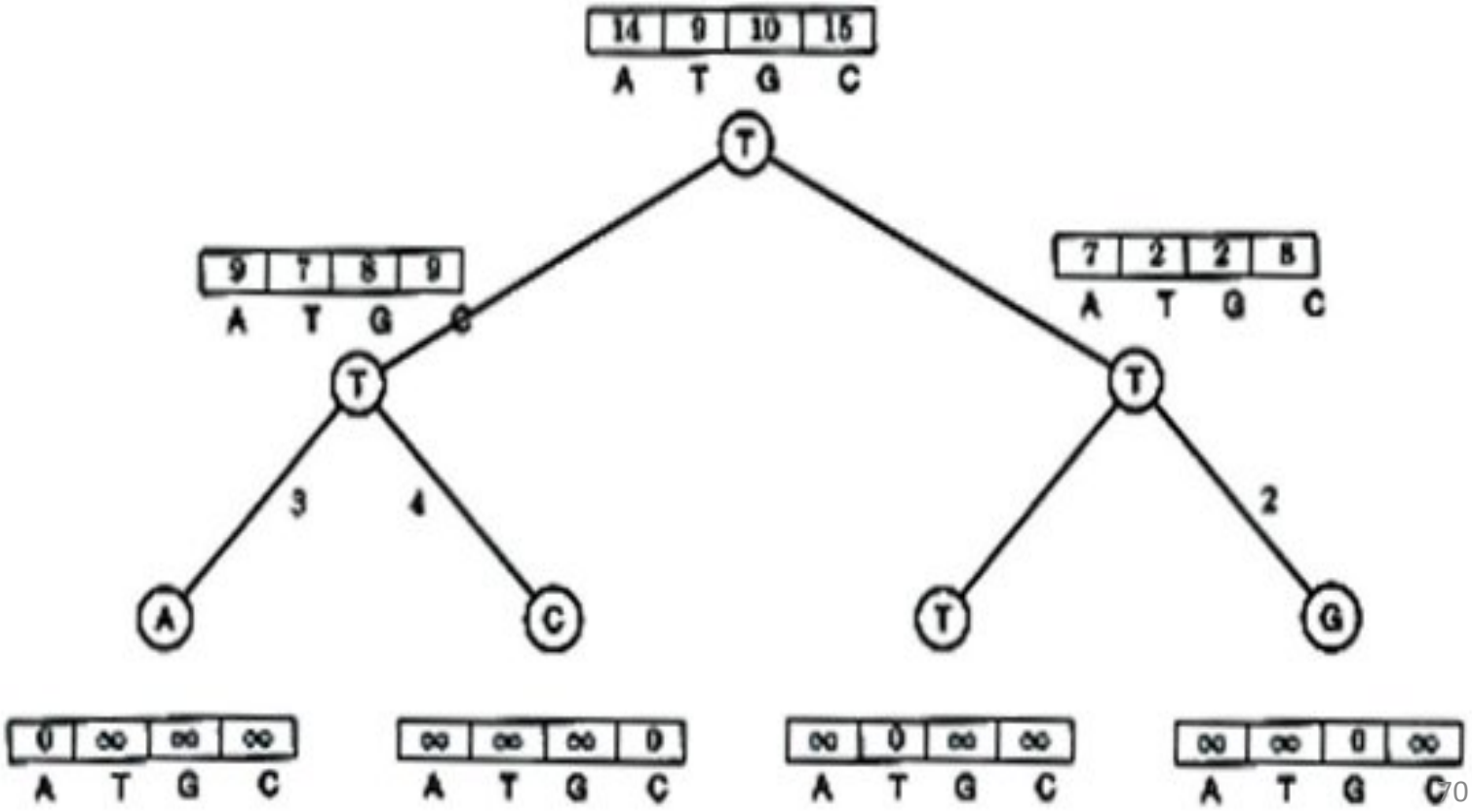
So left child is T,

And right child is T



Sankoff Algorithm (cont.)

And the tree is thus labeled…

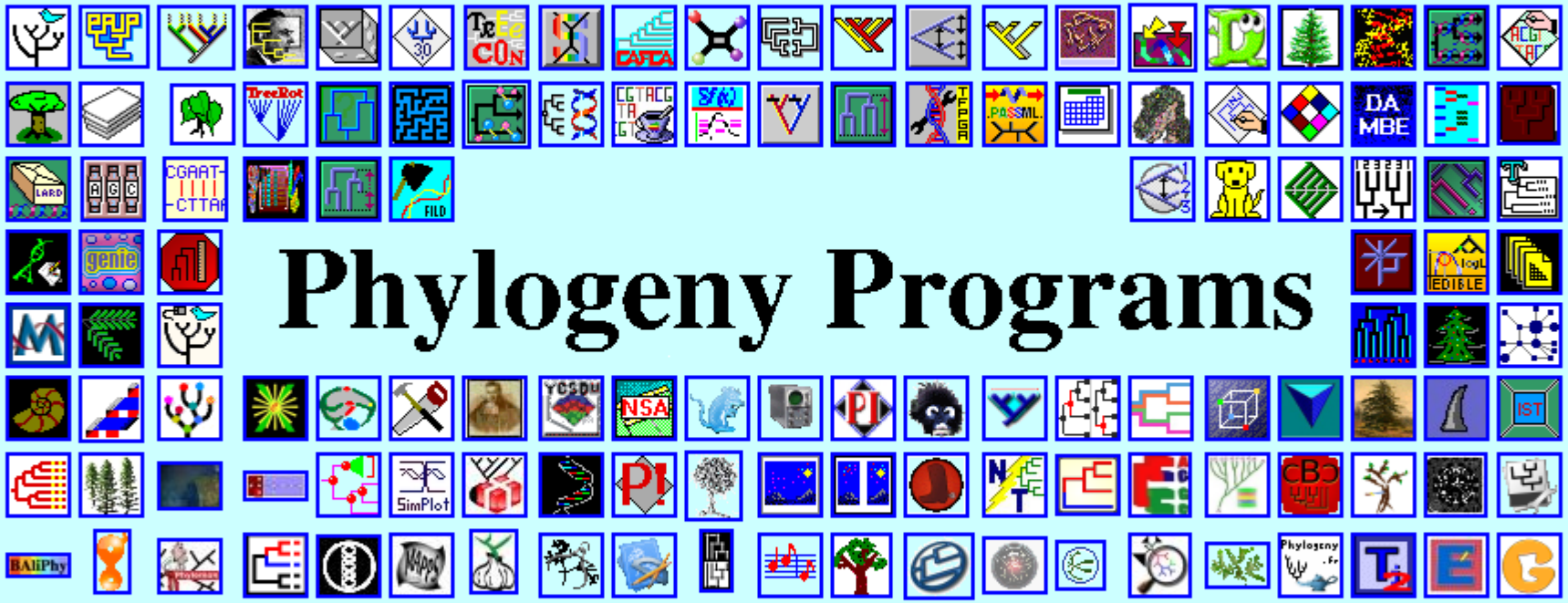


File Edit View History Bookmarks Tools Help

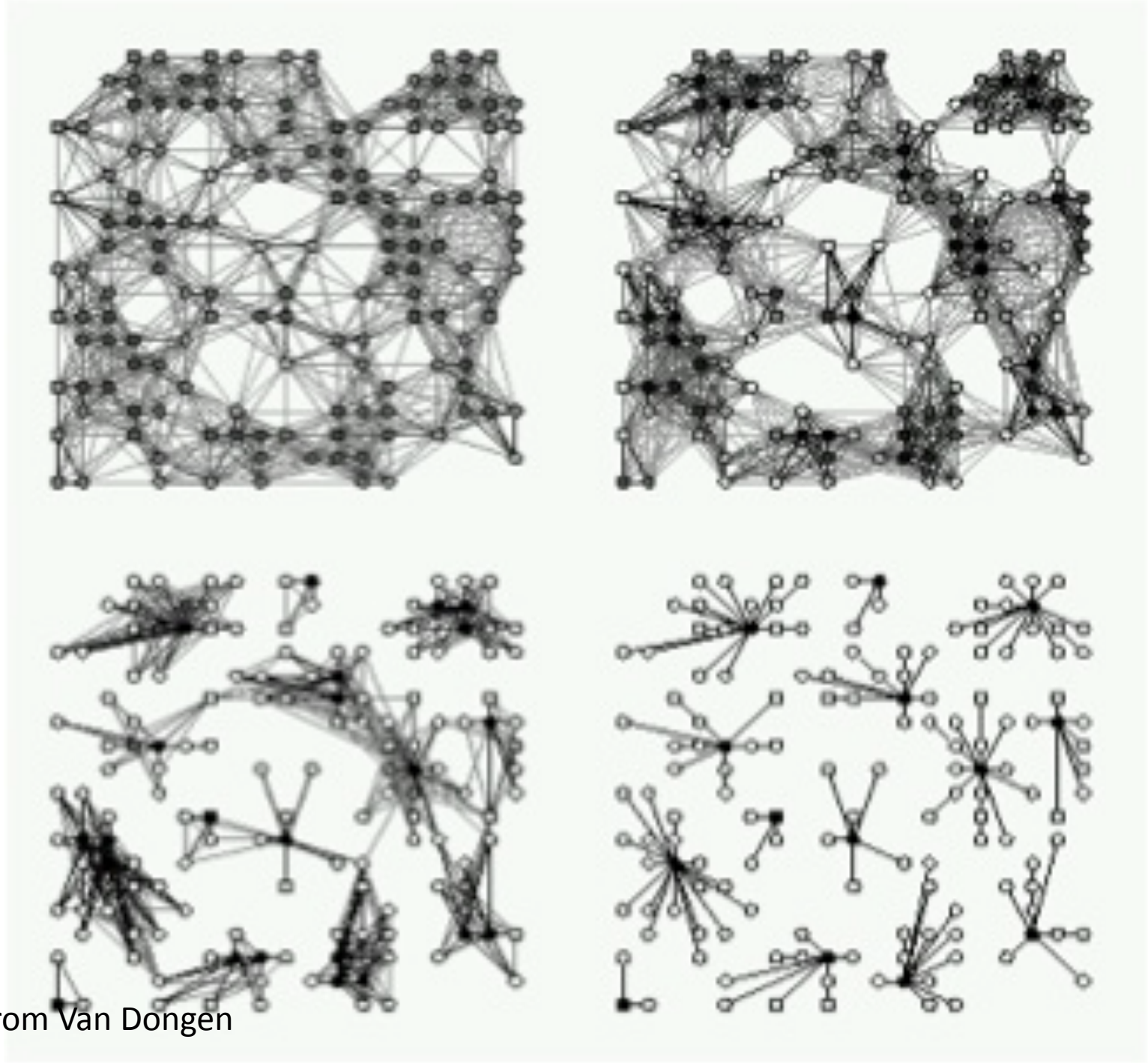
http://evolution.genetics.washington.edu/phylip/software.html

Customize Links Windows Free Hotmail Windows Marketplace Windows Media

Google phylip Search Bookmarks Check AutoLink AutoFill Send to phylip

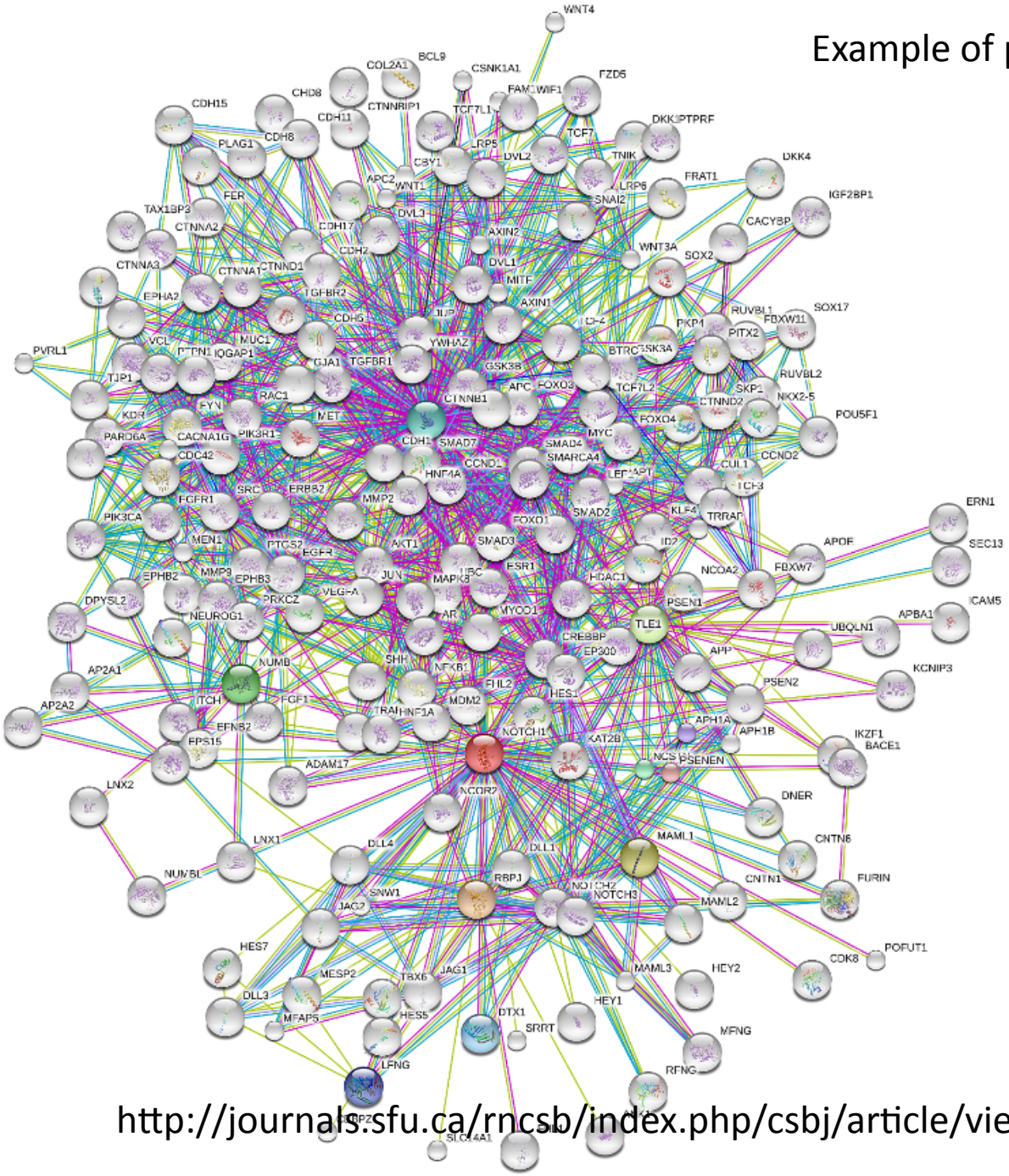


Here are 376 phylogeny packages and 50 free servers, all that I know about. It is an attempt to be completely comprehensive. I have not made any attempt to exclude programs that do not meet some standard of quality or importance. Updates to these pages are made roughly weekly. Here is a "waiting list" of new programs waiting to have their full entries constructed. Many of the programs in these pages are available on the web, and some of the older ones are also available from ftp server machines.

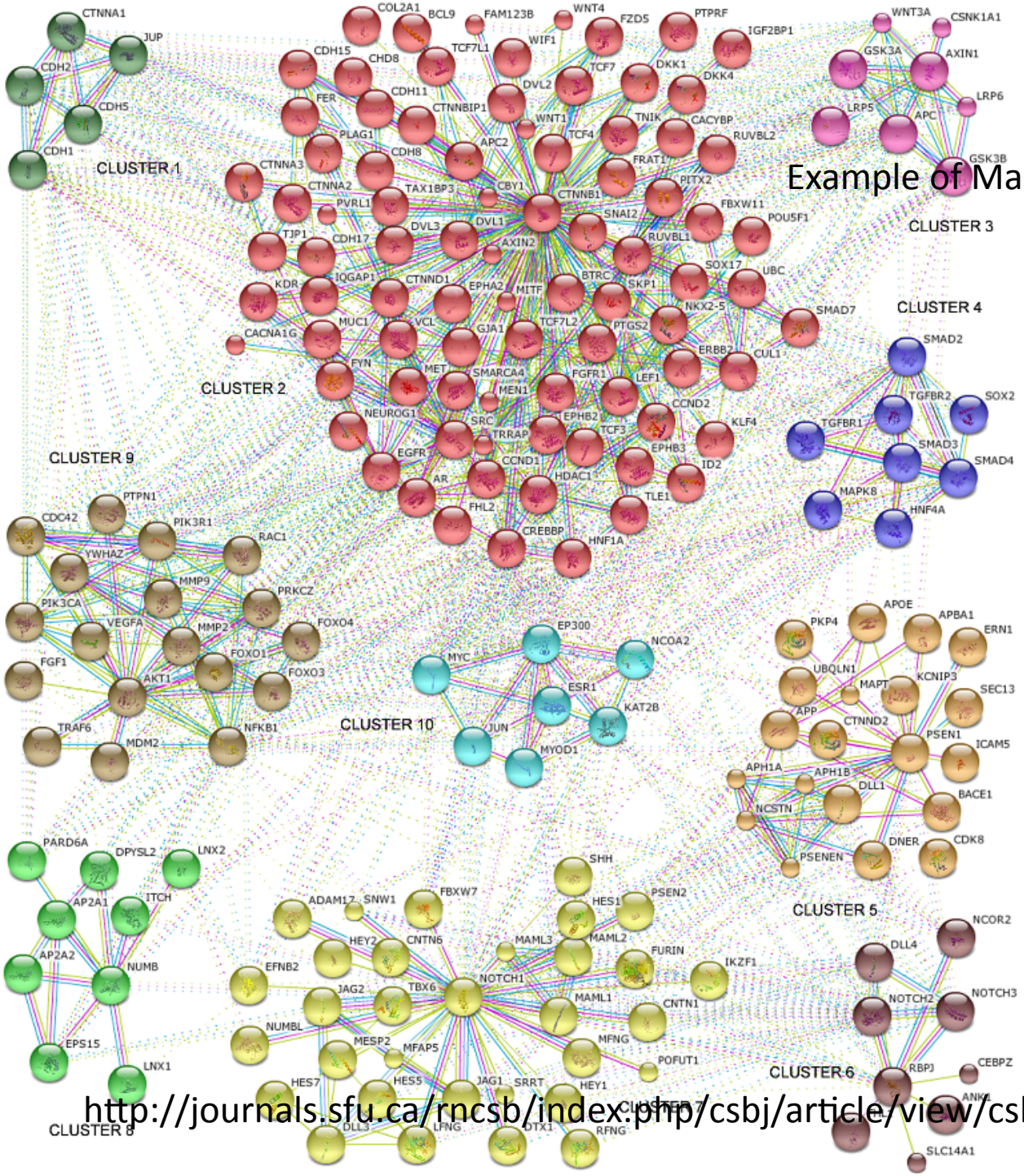


From Van Dongen

Example of protein network



http://journals.sfu.ca/rncsb/index.php/csbj/article/view/csbj.201207005/168

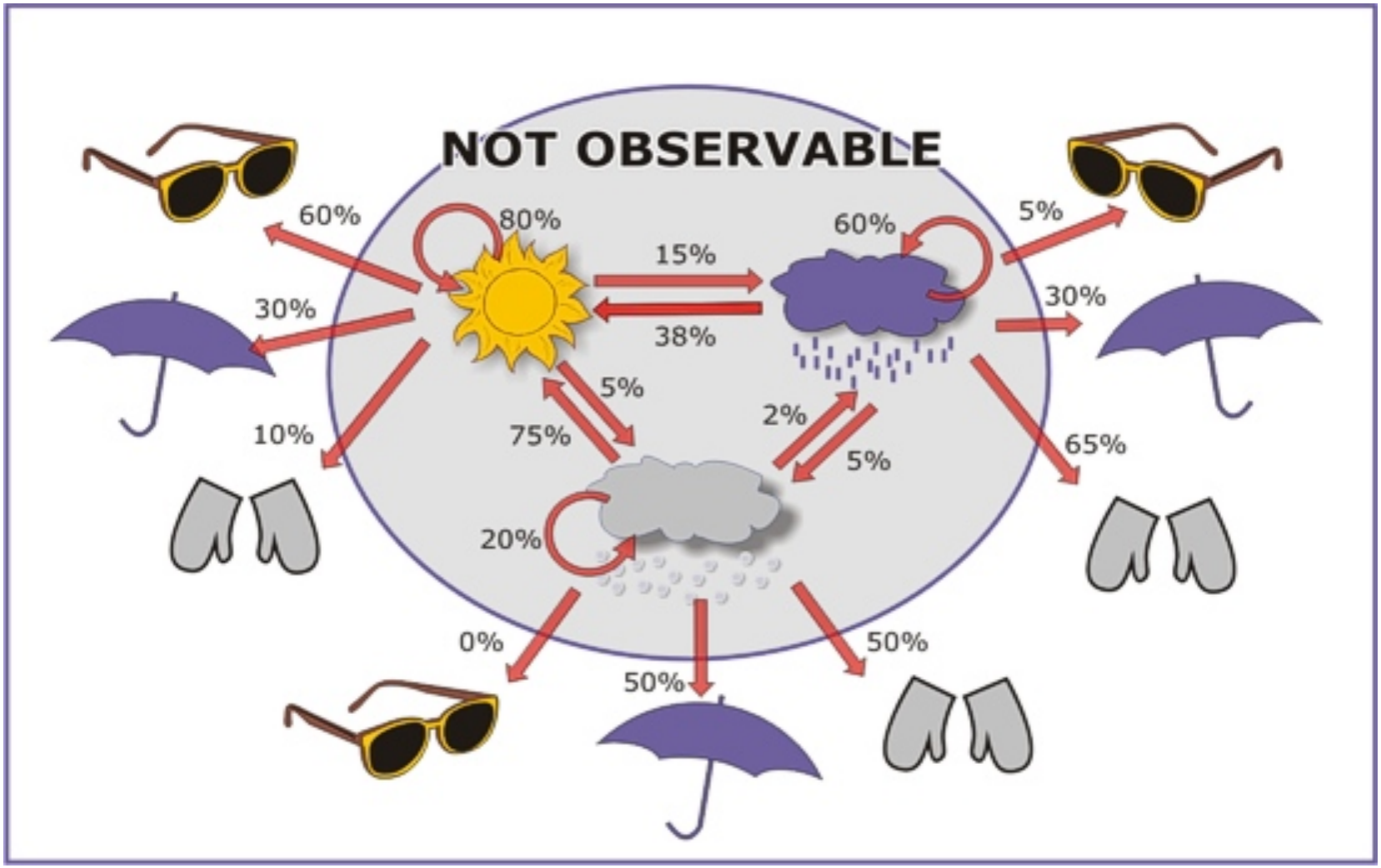


Example of Markov clustering

http://journals.sfu.ca/rncsb/index.php/csbj/article/view/csbj.201207005/168

From CSCE555 Bioinformatics

Hidden Markov Models



From CSCE555 Bioinformatics

Probability of a Sequence of Events

Refresh: definition of a HMM

Definition: A hidden Markov model (HMM)

- Alphabet $\Sigma = \{ b_1, b_2, \ldots, b_M \}$
- Set of states $Q = \{ 1, \ldots, K \}$
- Transition probabilities between any two states

  a_{ij} = transition prob from state i to state j

  $a_{i1} + \ldots + a_{iK} = 1$, for all states i = 1…K

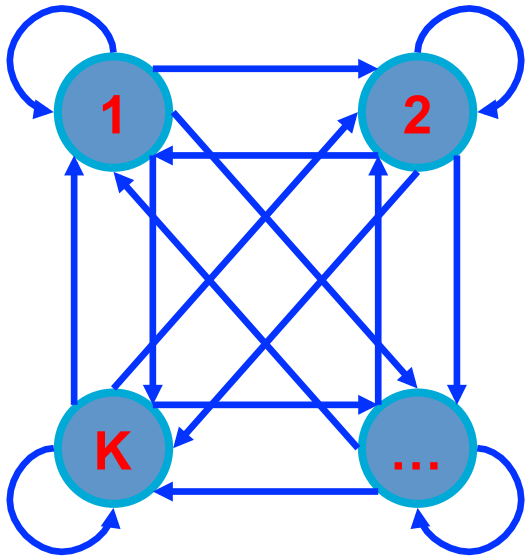
- Start probabilities a_{0i}

  $a_{01} + \ldots + a_{0K} = 1$

- Emission probabilities within each state

  $e_i(b) = P(x_i = b \mid \pi_i = k)$

  $e_i(b_1) + \ldots + e_i(b_M) = 1$, for all states i = 1…K



Example: The Dishonest Casino

A casino has two dice:

- Fair die

  P(1) = P(2) = P(3) = P(5) = P(6) = 1/6
- Loaded die

  P(1) = P(2) = P(3) = P(5) = 1/10

  P(6) = 1/2

Casino player switches back-&-forth between fair and loaded die once every 20 turns

Game:

1. You bet $1
2. You roll (always with a fair die)
3. Casino player rolls (maybe with fair die, maybe with loaded die)
4. Highest number wins $2

Question # 1 – Evaluation

GIVEN

A sequence of rolls by the casino player

12455264621461461361366616646616366163661636165156151151461235623 44

QUESTION

How likely is this sequence, given our model of how the casino works?

This is the **EVALUATION** problem in HMMs

Question # 2 – Decoding

GIVEN

A sequence of rolls by the casino player

1245526462146146136136661664661636616366163616515615115146123562344

QUESTION

What portion of the sequence was generated with the fair die, and what portion with the loaded die?

This is the **DECODING** question in HMMs

Question # 3 – Learning

GIVEN

A sequence of rolls by the casino player

12455264621461461361366616646616366163661636165156151151461235623 44

QUESTION

How "loaded" is the loaded die? How "fair" is the fair die? How often does the casino player change from fair to loaded, and back?

This is the **LEARNING** question in HMMs

The three main questions on HMMs

1. Evaluation

   GIVEN a HMM M, and a sequence x,

   FIND Prob[x | M]

2. Decoding

   GIVEN a HMM M, and a sequence x,

   FIND the sequence π of states that maximizes P[x, π | M]

3. Learning

   GIVEN a HMM M, with unspecified transition/emission probs.,
   and a sequence x,

   FIND parameters $\theta = (e_i(.), a_{ij})$ that maximize P[x | θ]

Example: the dishonest casino

Let the sequence of rolls be:

x = 1, 2, 1, 5, 6, 2, 1, 6, 2, 4

Then, what is the likelihood of

π = Fair, Fair, Fair, Fair, Fair, Fair, Fair, Fair, Fair, Fair?

(say initial probs $a_{0Fair} = ½$, $a_{oLoaded} = ½$)

½ × P(1 | Fair) P(Fair | Fair) P(2 | Fair) P(Fair | Fair) … P(4 | Fair) =

$½ \times (1/6)^{10} \times (0.95)^{9} = .00000005211586472 11 = 0.5 \times 10^{-9}$

Example: the dishonest casino

So, the likelihood the die is fair in all this run

is just 0.521×10^{-9}

OK, but what is the likelihood of

= Loaded, Loaded, Loaded, Loaded, Loaded, Loaded, Loaded, Loaded, Loaded, Loaded?

$\frac{1}{2} \times P(1 \mid \text{Loaded})\ P(\text{Loaded}, \text{Loaded}) \ldots P(4 \mid \text{Loaded}) =$

$\frac{1}{2} \times (1/10)^8 \times (1/2)^2\ (0.95)^9 = .000000000781176215 = 7.9 \times 10^{-10}$

Therefore, it is after all 6.59 times more likely that the die is fair all the way, than that it is loaded all the way.

Example: the dishonest casino

Let the sequence of rolls be:

x = 1, 6, 6, 5, 6, 2, 6, 6, 3, 6

Now, what is the likelihood π = F, F, …, F?

$\frac{1}{2} \times (1/6)^{10} \times (0.95)^{9} = 0.5 \times 10^{-9}$, same as before

What is the likelihood

π = L, L, …, L?

$\frac{1}{2} \times (1/10)^{4} \times (1/2)^{6} (0.95)^{9} = .00000049238235134735 = 0.5 \times 10^{-7}$

So, it is 100 times more likely the die is loaded

Example

Let x be a sequence with a portion of ~ 1/6 6's, followed by a portion of ~ ½ 6's…

x = 123456123456…12345 6626364656…1626364656

Then, it is not hard to show that optimal parse is:

FFF……F LLL……L

6 nucleotides "123456" parsed as F, contribute $.95^6 \times (1/6)^6 = 1.6 \times 10^{-5}$

parsed as L, contribute $.95^6 \times (1/2)^1 \times (1/10)^5 = 0.4 \times 10^{-5}$

"162636" parsed as F, contribute $.95^6 \times (1/6)^6 = 1.6 \times 10^{-5}$

parsed as L, contribute $.95^6 \times (1/2)^3 \times (1/10)^3 = 9.0 \times 10^{-5}$