Elements of Bioinformatics Autumn 2010

VELI MÄKINEN

HTTP://WWW.CS.HELSINKI.FI/EN/COURSES/582606/2010/S/K/1

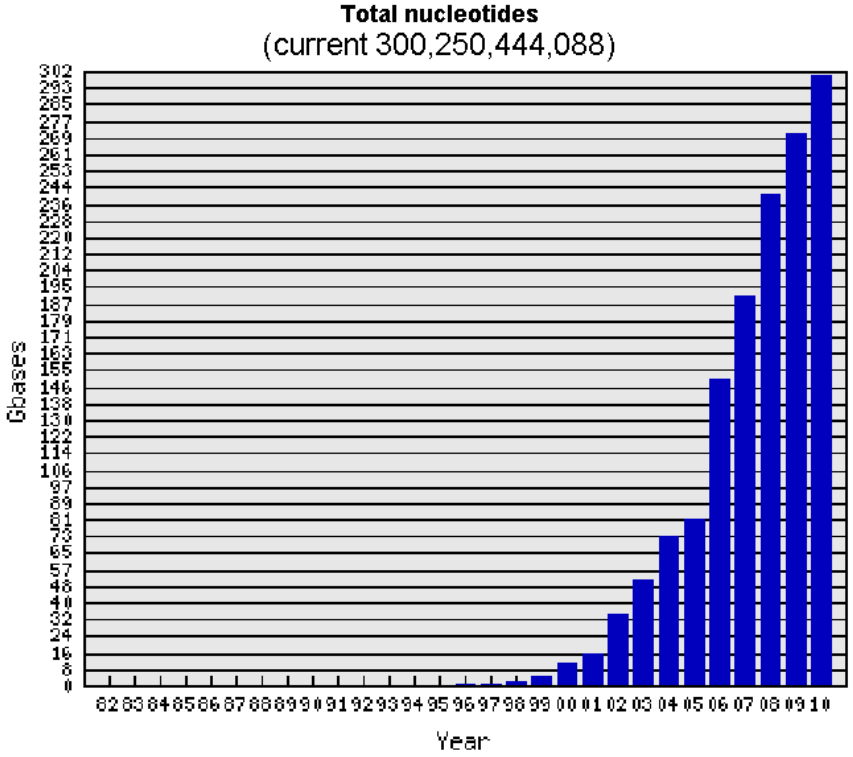
Lecture Mon 8.11.

RAPID ALIGNMENT METHODS: FASTA AND BLAST

GENOME-WIDE COMPARISON: SUFFIX TREE, MUMMER

The biological problem

- Global and local alignment algoritms are slow in practice
- Consider the scenario of aligning a *query sequence* against a large database of sequences
  - New sequence with unknown function



http://www.ebi.ac.uk/embl/Services/DBStats/

Problem with large amount of sequences

- Exponential growth in both number and total length of sequences
- Possible solution: Compare against model organisms only
- With large amount of sequences, chances are that matches occur by random
  - Need for statistical analysis

First solution: FASTA

- FASTA is a multistep algorithm for sequence alignment (Wilbur and Lipman, 1983)
- The sequence file format used by the FASTA software is widely used by other sequence analysis software
- Main idea:
  - Choose regions of the two sequences (query and database) that look promising (have some degree of similarity)
  - Compute local alignment using dynamic programming in these regions

FASTA outline

- FASTA algorithm has five steps:
  - *1. Identify common k-mers between I and J*
  - 2. Score diagonals with k-mer matches, identify 10 best diagonals
  - 3. Rescore initial regions with a substitution score matrix
  - 4. Join initial regions using gaps, penalise for gaps
  - 5. Perform dynamic programming to find final alignments

Analyzing the k-mer content

- Example query string I: TGATGATGAAGACATCAG
- For k = 8, the set of k-mers of I is

```
TGATGATG
 GATGATGA
  ATGATGAA
   TGATGAAG
      ...
          GACATCAG
```

Analyzing the k-mer content

- There are n-k+1 k-mers in a string of length n
- If at least one k-mer of I is not found from another string J, we know that I differs from J
- Need to consider statistical significance: I and J might share k-mers by chance only
- Let m=|I| and n=|J|

Word lists and comparison by content

- The k-mers of I can be arranged into a table of k-mer occurences $L_w(I)$
- Consider the k-mers when k=2 and I=GCATCGGC:

GC, CA, AT, TC, CG, GG, GC

AT: 3
CA: 2
CG: 5
GC: 1, 7 ← Start indecies of k-mer GC in I
GG: 6
TC: 4

Building $L_w(I)$ takes O(n) time

Common k-mers

- Number of common k-mers in I and J can be computed using $L_w(I)$ and $L_w(J)$
- For each k-mer w in I, there are $|L_w(J)|$ occurrences in J
- Therefore I and J have $\sum_w |L_w(I)||L_w(J)|$ common k-mer pairs
- This can be computed in $O(m + n + 4^k)$ time
  - $O(m + n + 4^k)$ time to build the lists
  - $O(4^k)$ time to multiply the corresponding list entry sizes (in DNA strings)

Common k-mers

- I = GCATCGGC
- J = CCATCGCCATCG

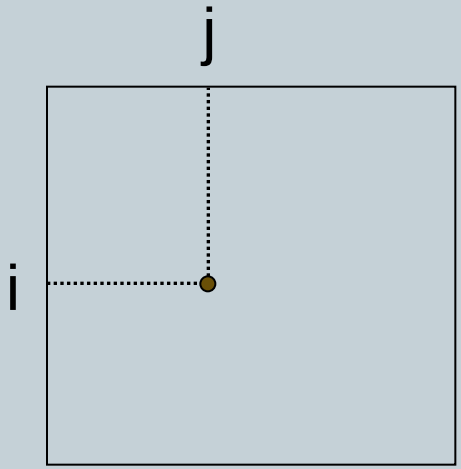
L_w(I)	L_w(J)	Common k-mers
AT: 3	AT: 3, 9	2
CA: 2	CA: 2, 8	2
	CC: 1, 7	0
CG: 5	CG: 5, 11	2
GC: 1, 7	GC: 6	2
GG: 6		0
TC: 4	TC: 4, 10	2
		10 in total

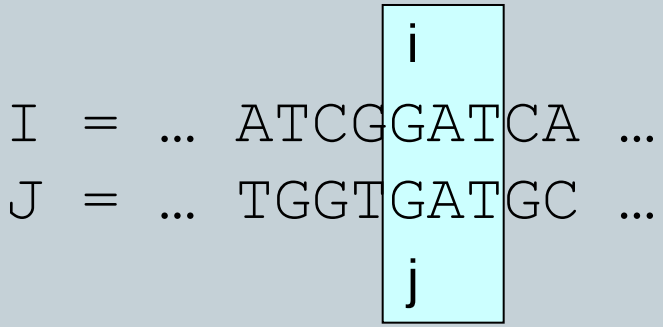
FASTA outline

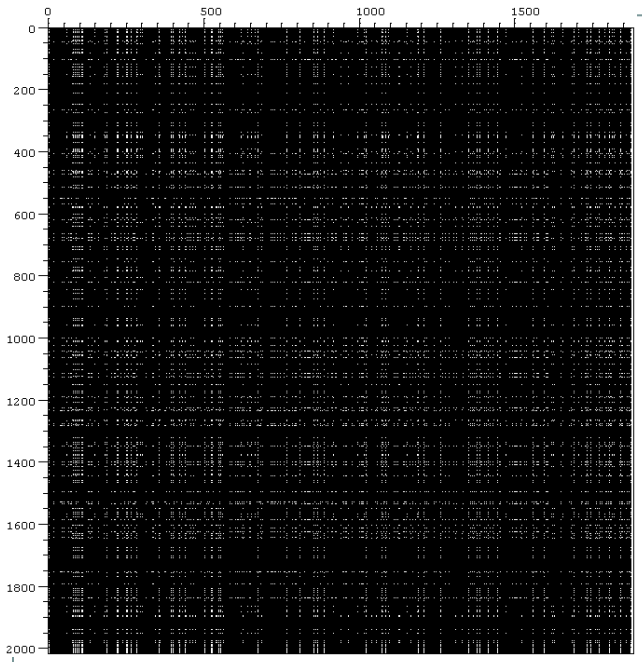
- FASTA algorithm has five steps:
  - 1. Identify common k-mers between I and J
  - *2. Score diagonals with k-mer matches, identify 10 best diagonals*
  - 3. Rescore initial regions with a substitution score matrix
  - 4. Join initial regions using gaps, penalise for gaps
  - 5. Perform dynamic programming to find final alignments

Dot matrix comparisons

- k-mer matches in two sequences I and J can be represented as a *dot matrix*
- Dot matrix element (i, j) has ”a dot”, if the k-mer starting at position i in I is identical to the k-mer starting at position j in J
- The dot matrix can be plotted for various k

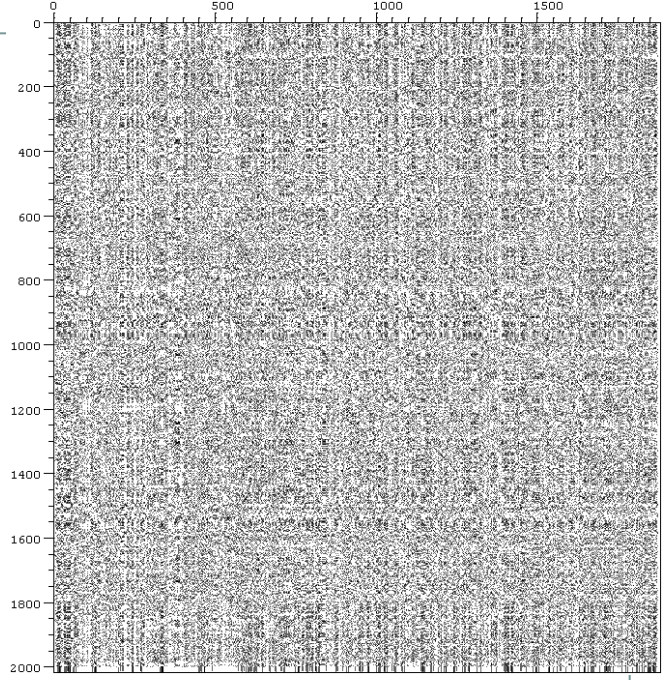




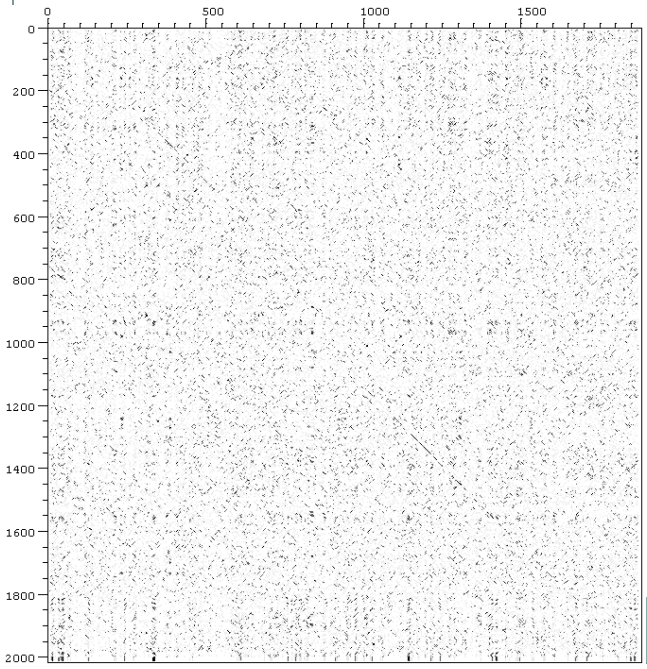


k=1

k=4

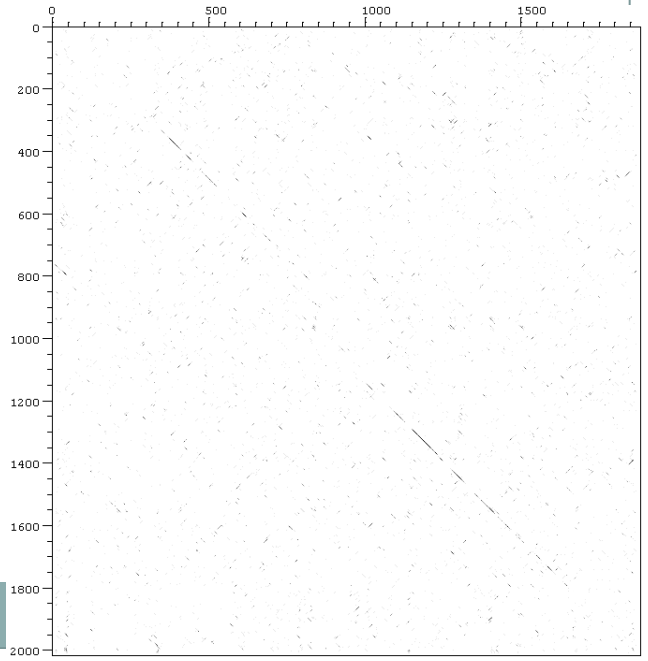


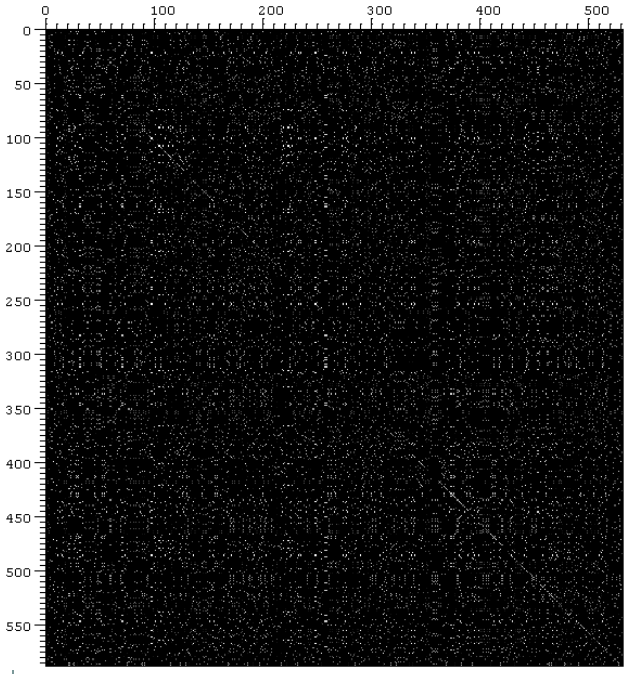
Dot matrix (k=1,4,8,16)
for two **DNA** sequences
X85973.1 (1875 bp)
Y11931.1 (2013 bp)



k=8

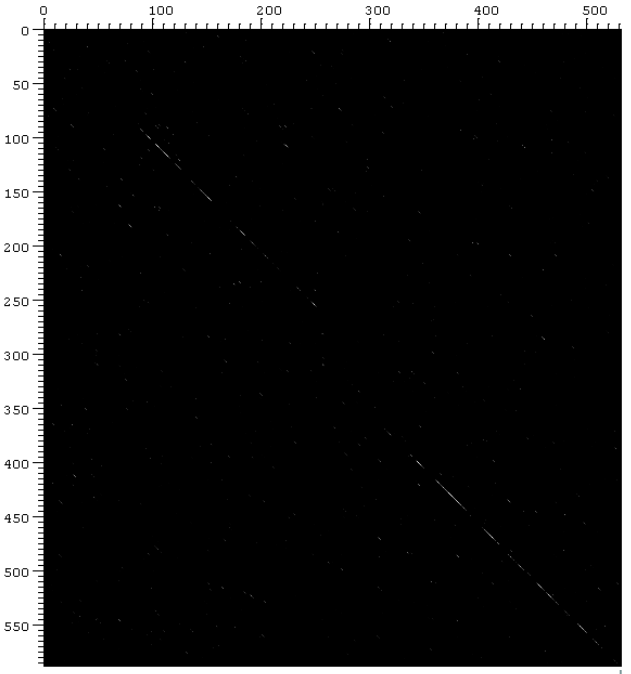
k=16



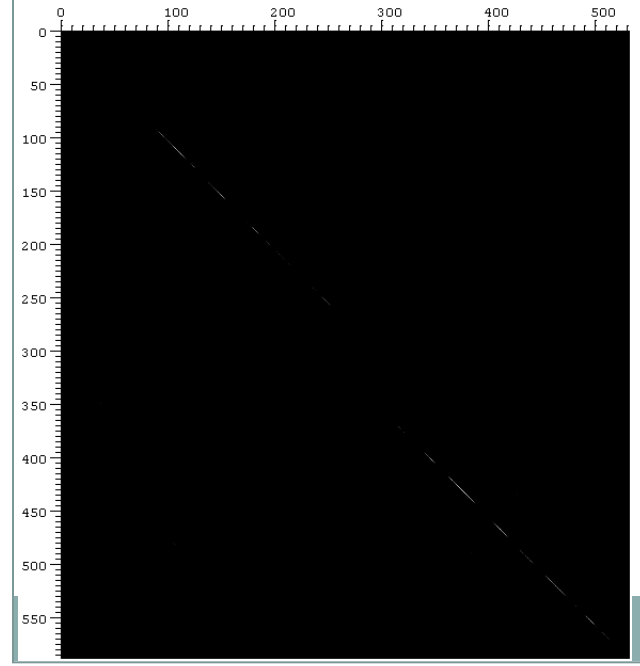


k=1

k=4

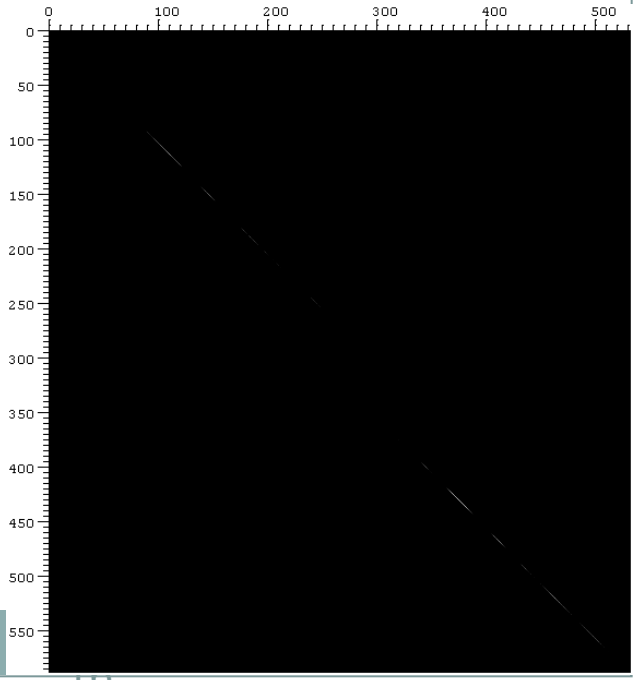


Dot matrix (k=1,4,8,16) for two **protein** sequences
CAB51201.1 (531 aa)
CAA72681.1 (588 aa)



k=8

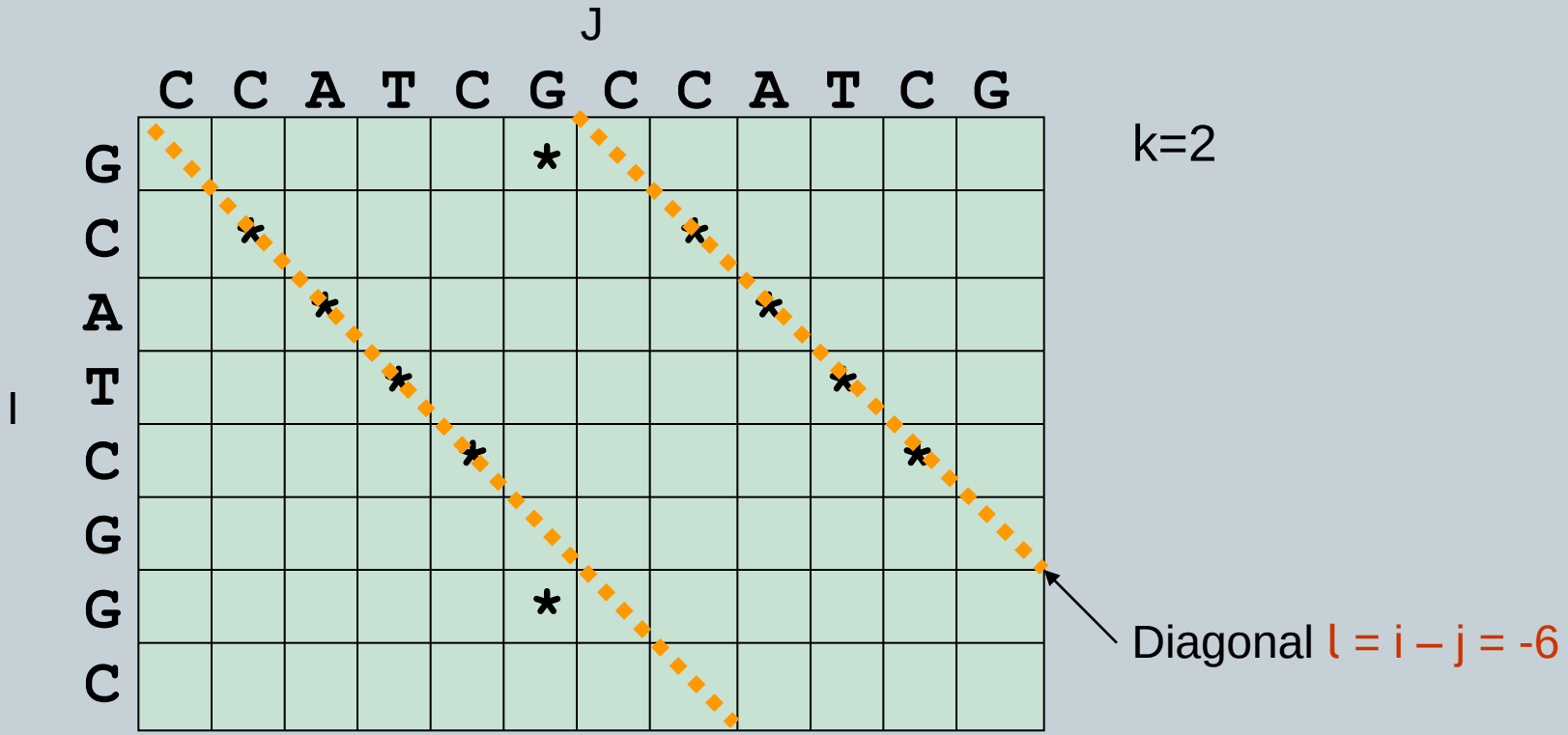
k=16



Shading indicates now the match score according to a score matrix (Blosum62 here)

Computing diagonal sums

- We would like to find high scoring diagonals of the dot matrix
- Lets index diagonals by the offset, l = i - j



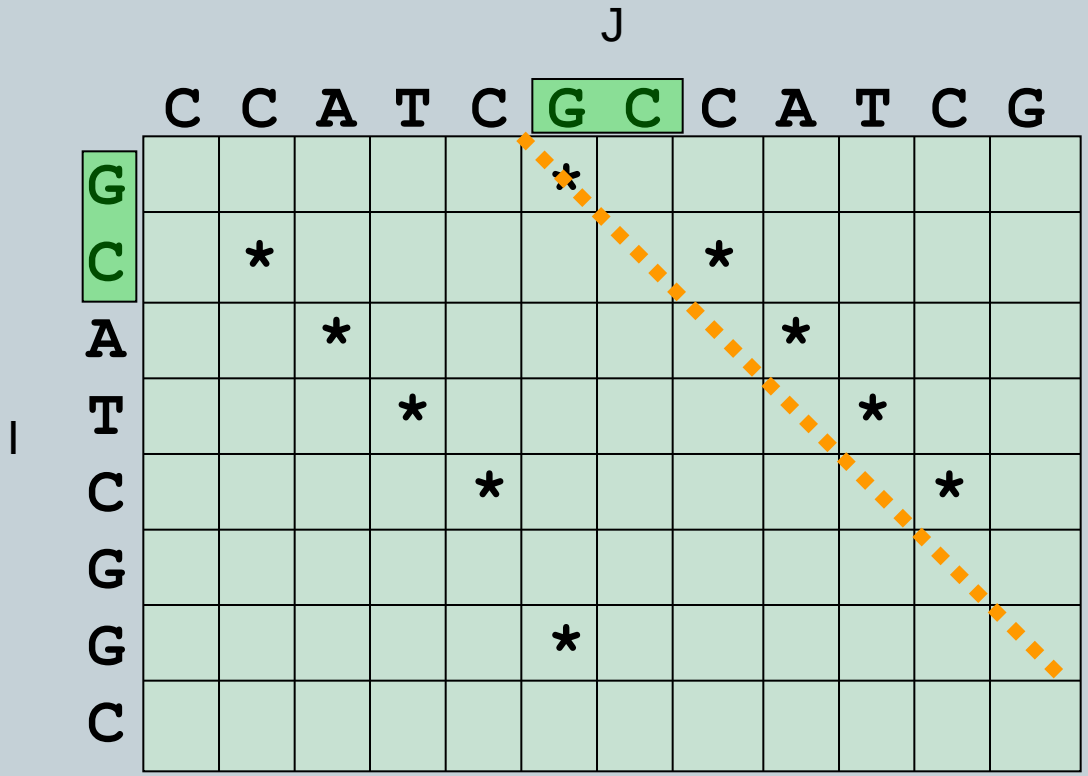
Computing diagonal sums

- As an example, lets compute diagonal sums for I = GCATCGGC, J = CCATCGCCATCG, k = 2
- 1. Construct k-mer list $L_w(J)$
- 2. Diagonal sums S_l are computed into a table, indexed with the offset and initialised to zero

l	-10	-9	-8	-7	-6	-5	-4	-3	-2	-1	0	1	2	3	4	5	6
S_l	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Computing diagonal sums

- 3. Go through k-mers of I, look for matches in $L_w(J)$ and update diagonal sums

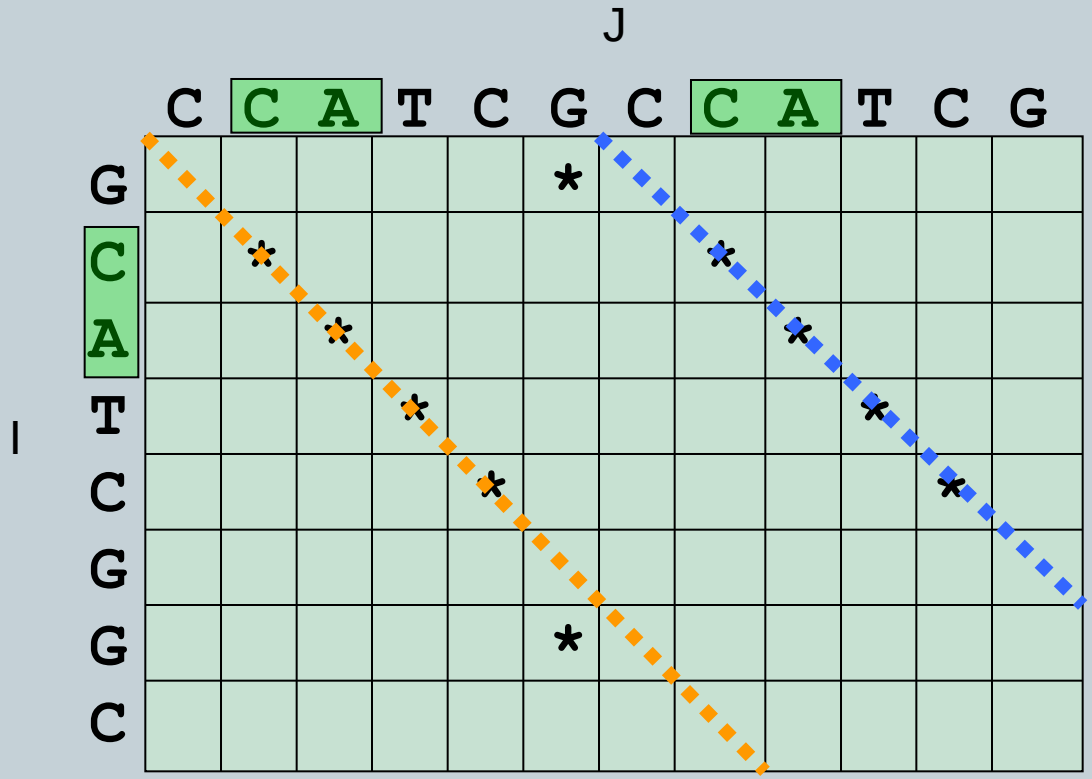


For the first 2-mer in I, GC, $L_{GC}(J) = \{6\}$.

We can then update the sum of diagonal $l = i - j = 1 - 6 = -5$ to $S_{-5} := S_{-5} + 1 = 0 + 1 = 1$

Computing diagonal sums

- 3. Go through k-mers of I, look for matches in $L_w(J)$ and update diagonal sums



Next 2-mer in I is CA, for which $L_{CA}(J) = \{2, 8\}$.

Two diagonal sums are updated:

$I = i - j = 2 - 2 = 0$

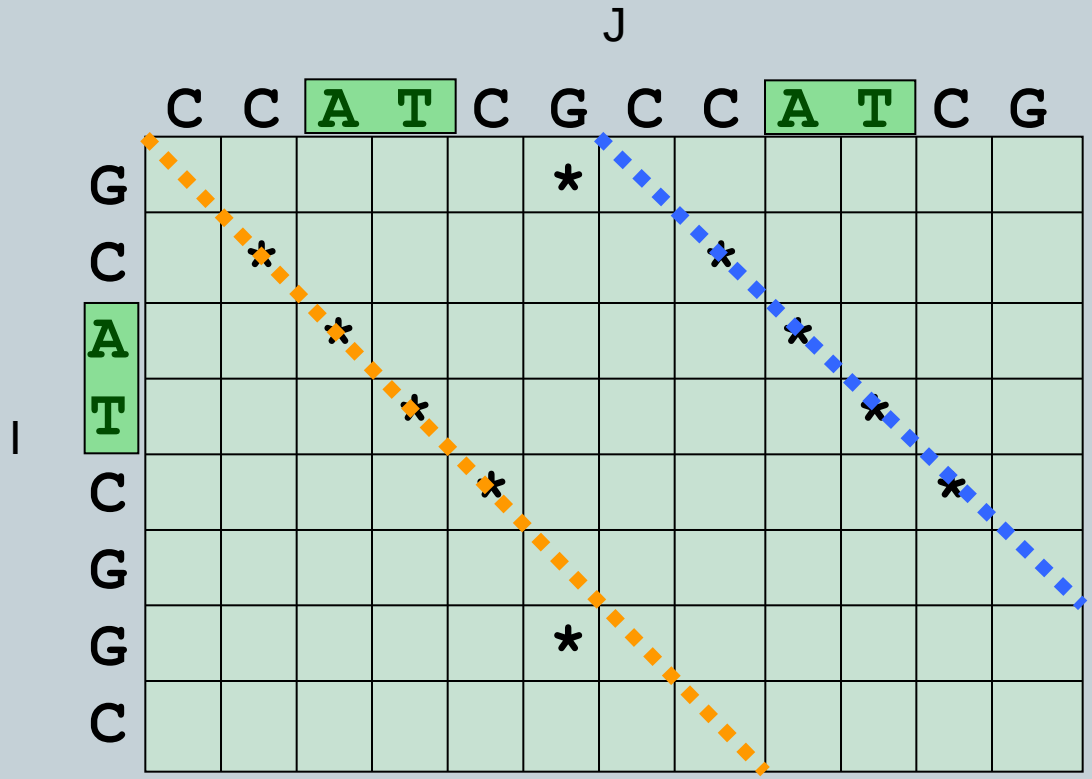
$S_0 := S_0 + 1 = 0 + 1 = 1$

$I = i - j = 2 - 8 = -6$

$S_{-6} := S_{-6} + 1 = 0 + 1 = 1$

Computing diagonal sums

- 3. Go through k-mers of I, look for matches in $L_w(J)$ and update diagonal sums



Next 2-mer in I is AT, for which $L_{AT}(J) = \{3, 9\}$.

Two diagonal sums are updated:

$I = i - j = 3 - 3 = 0$

$S_0 := S_0 + 1 = 1 + 1 = 2$

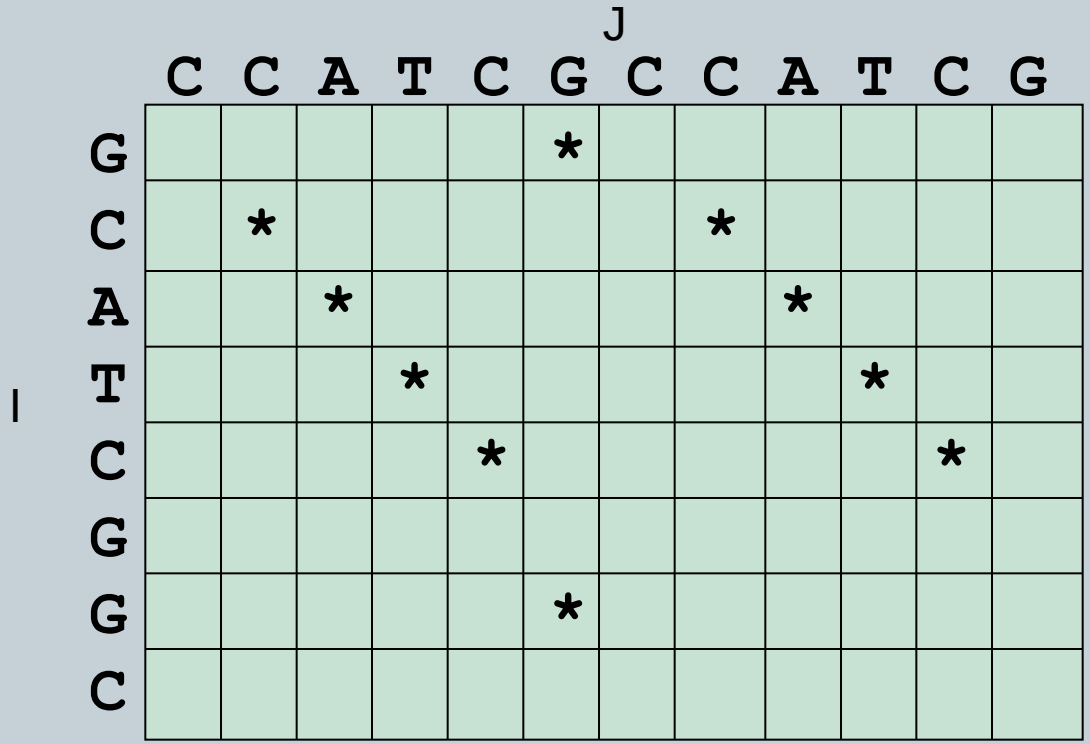
$I = i - j = 3 - 9 = -6$

$S_{-6} := S_{-6} + 1 = 1 + 1 = 2$

Computing diagonal sums

After going through the k-mers of I, the result is:

l	-10	-9	-8	-7	-6	-5	-4	-3	-2	-1	0	1	2	3	4	5	6
S_l	0	0	0	0	4	1	0	0	0	0	4	1	0	0	0	0	0



Algorithm for computing diagonal sum of scores

$S_l := 0$ for all $1 - n < l \leq m - 1$

Compute $L_w(J)$ for all k-mers w

for i := 1 to m – k +1 do

  w := $I_i I_{i+1} \ldots I_{i+k-1}$

  for $j \in L_w(J)$ do

    l := i – j

    $S_l := S_l + 1$ ← Match score is here 1

  end

end

FASTA outline

- FASTA algorithm has five steps:
  - 1. Identify common k-mers between I and J
  - 2. Score diagonals with k-mer matches, identify 10 best diagonals
  - *3. Rescore initial regions with a substitution score matrix*
  - *4. Join initial regions using gaps, penalise for gaps*
  - 5. Perform dynamic programming to find final alignments

Rescoring initial regions

- Each high-scoring diagonal chosen in the previous step is rescored according to a score matrix
- This is done to find subregions with identities shorter than k
- Non-matching ends of the diagonal are trimmed

```
I: C C A T C G C C A T C G
J: C C A A C G C A A T C A
```

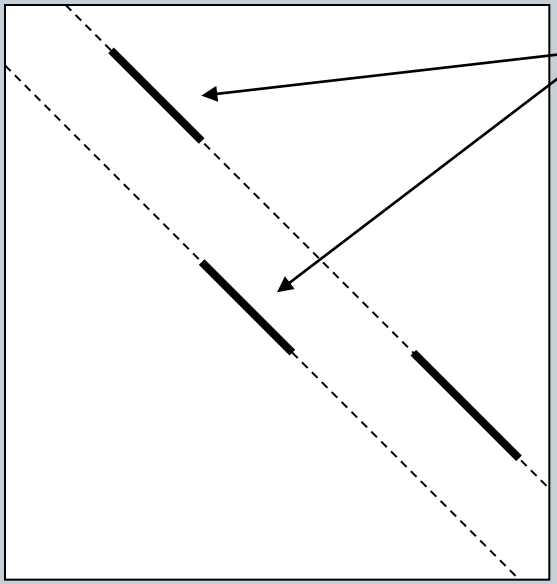
75% identity, no 4-mer identities

```
I': C C A T C G C C A T C G
J': A C A T C A A A T A A A
```

33% identity, one 4-mer identity

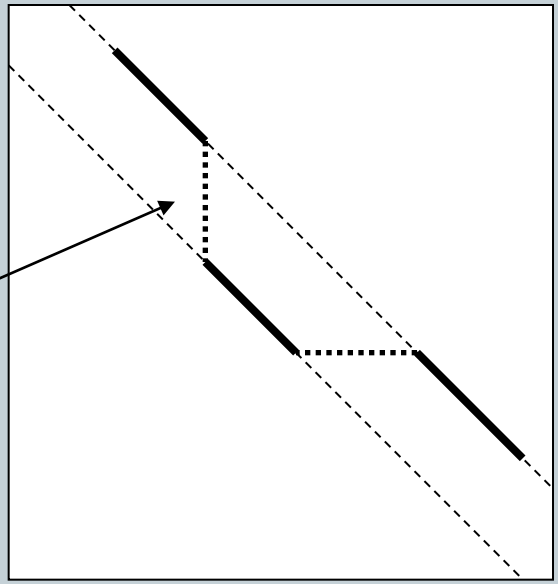
Joining diagonals

- Two offset diagonals can be joined with a gap, if the resulting alignment has a higher score
- Separate gap open and extension are used
- Find the best-scoring combination of diagonals



High-scoring diagonals

Two diagonals joined by a gap

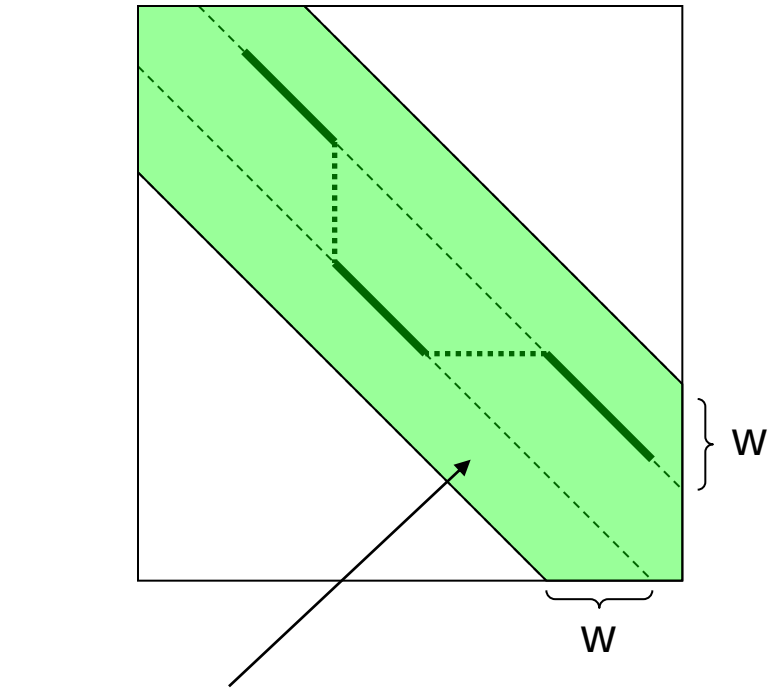


FASTA outline

- FASTA algorithm has five steps:
  - 1. Identify common k-mers between I and J
  - 2. Score diagonals with k-mer matches, identify 10 best diagonals
  - 3. Rescore initial regions with a substitution score matrix
  - 4. Join initial regions using gaps, penalise for gaps
  - *5. Perform dynamic programming to find final alignments*

Local alignment in the highest-scoring region

- Last step of FASTA: perform local alignment using dynamic programming around the highest-scoring diagonals
- Region to be aligned covers –w and +w offset diagonal to the highest-scoring diagonals
- With long sequences, this region is typically very small compared to the whole m x n matrix



Dynamic programming matrix
M filled only for the green region

Properties of FASTA

- Fast compared to local alignment using dynamic programming only
  - Only a narrow region of the full matrix is aligned
- *Lossy filter* : may fail to find some high scoring local alignments
- Increasing parameter k decreases the number of hits:
  - Increases specificity
  - Decreases sensitivity
  - Decreases running time

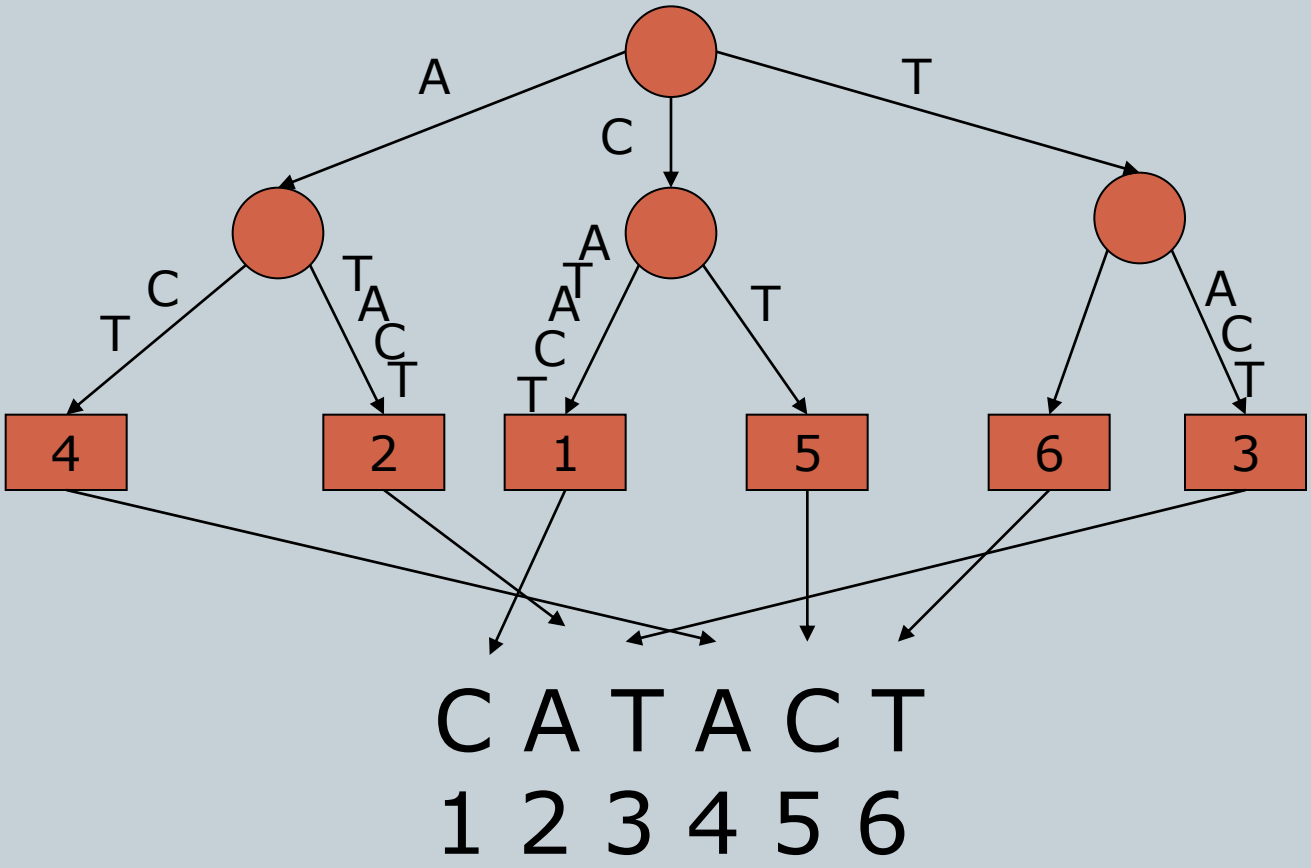
Properties of FASTA

- FASTA looks for initial exact matches to query sequence
  - Two proteins can have very different amino acid sequences and still be biologically similar
  - This may lead into a lack of sensitivity with diverged sequences
- Demonstration of FASTA at EBI
  - http://www.ebi.ac.uk/fasta/

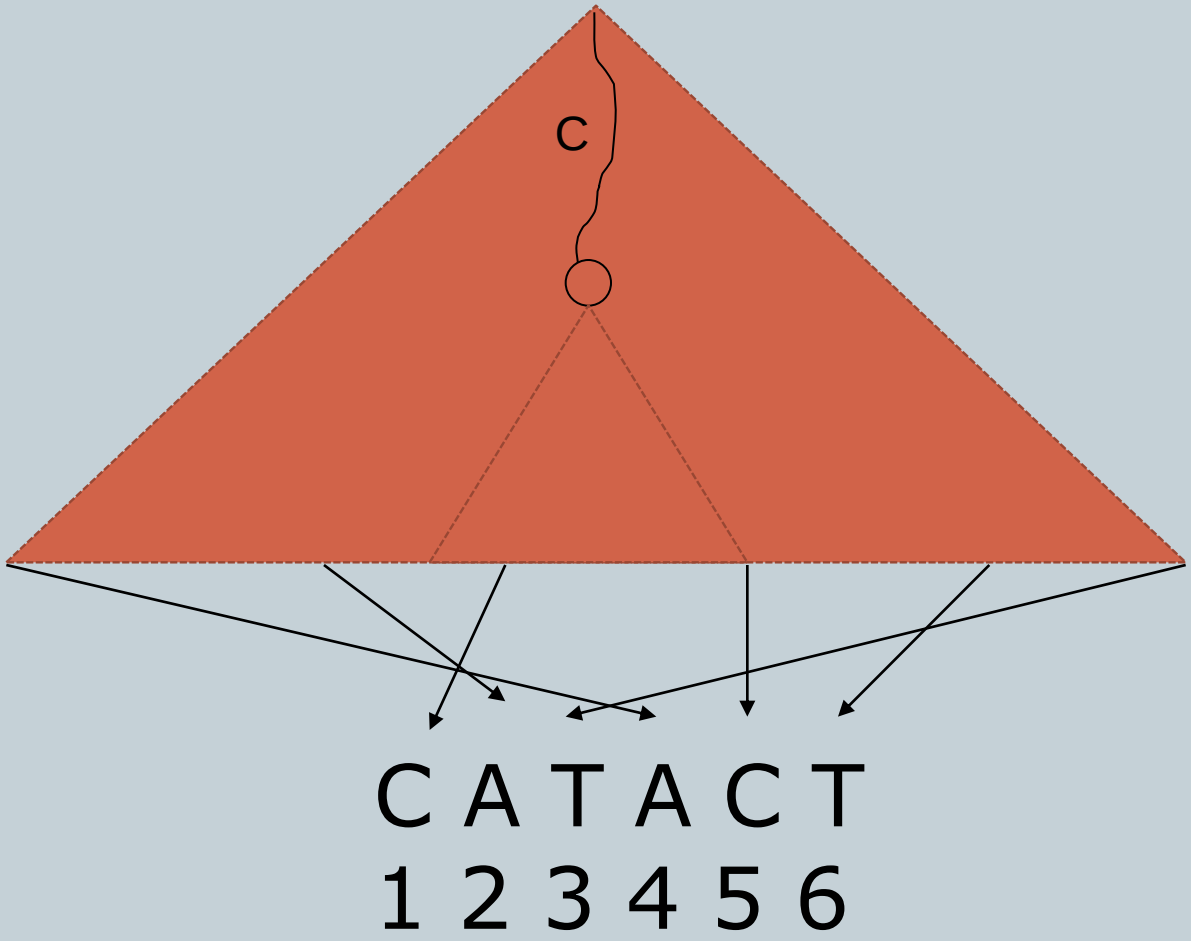
Note on alternative implementations

- *Generalized suffix tree* can be used for counting the common k-mer pairs in optimal time and space (see exercise 5.2 at Algorithms for Bioinformatics course)
- Generalized suffix tree with some additional data structures can also be used for directly computing all *maximal matches*, *i.e.*, tuples {(i',i),(j',j)} such that $a_{i'}...a_i = b_{j'}...b_j$ and the ranges cannot be extended left or right (see Gusfield's book Algorithms on Strings, Trees, and Sequences: Computer Science and Computational Biology).
- *Descending suffix walk* with the query on the suffix tree of the database can also be modified to solve the maximal matches problem.
- MUMMER software (http://mummer.sourceforge.net/) implements these kind of ideas.
  - Exercise: Try Mummer and learn what are MUM, MAM, and MEM, and for what purposes they can be used.

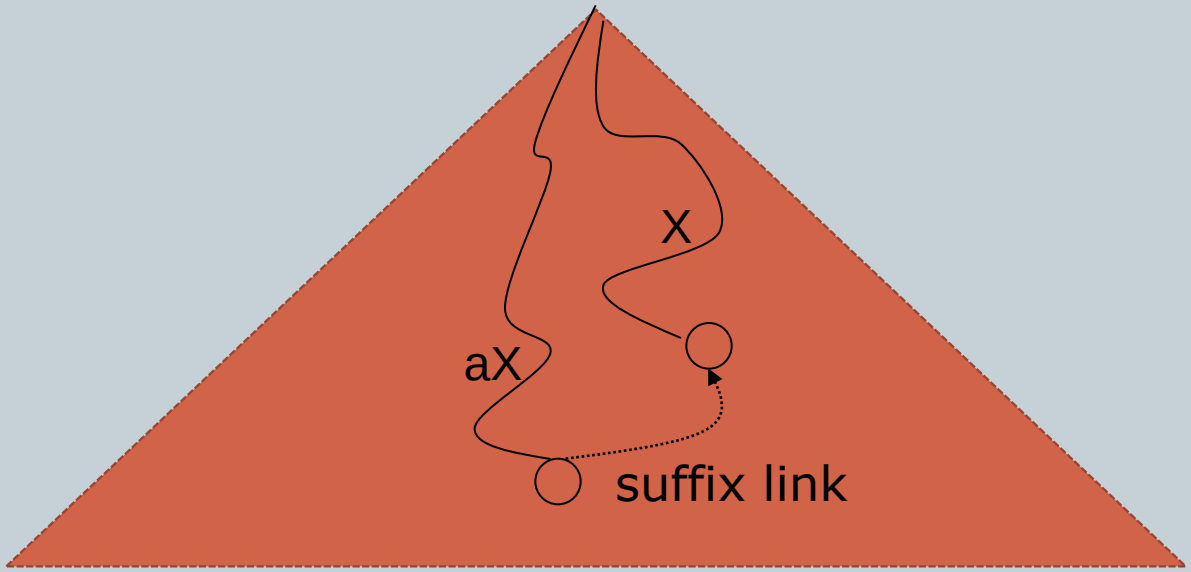
Suffix tree



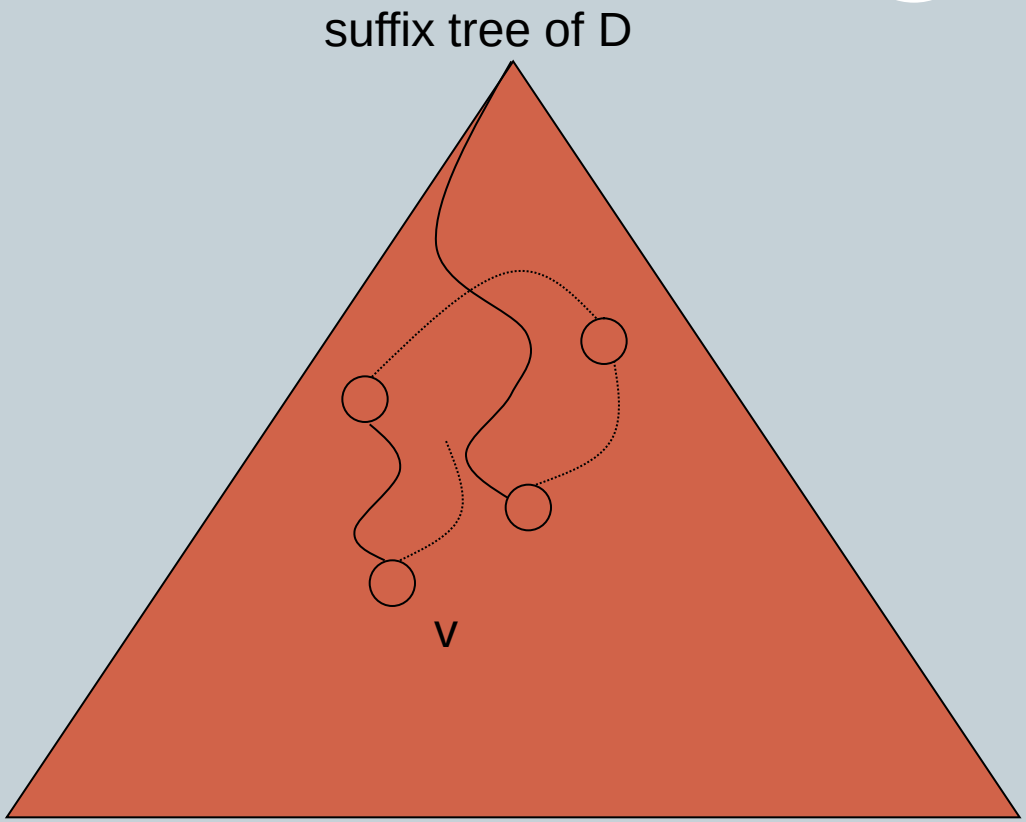
Abstract representation of suffix tree



Suffix link



Descending suffix walk



Set l=1. Read Q[1,m] left-to-right, always going down in the tree when possible. If the next symbol of Q does not match any edge label on current position, take suffix link (l++), and try again. (Suffix link in the root to itself emits a symbol). Let v be a node visited after reading a symbol Q[r] just before taking a suffix link. Then Q[l,r] is a *maximal match* with substrings of D (whose occurrences can be found from the subtree of v), and e.g. the *longest common substring* of Q and D is Q[l,r] with largest r-l. Listing all maximal matches is more complicated but doable.

BLAST: Basic Local Alignment Search Tool

- BLAST (Altschul et al., 1990) and its variants are some of the most common sequence search tools in use
- Roughly, the basic BLAST has three parts:
  - 1. Find *segment pairs* between the query sequence and a database sequence above score threshold (”seed hits”)
  - 2. Extend seed hits into *locally maximal segment pairs*
  - 3. Calculate p-values and a rank ordering of the local alignments
- Gapped BLAST introduced in 1997 allows for gaps in alignments

Finding seed hits

- First, we generate a set of *neighborhood sequences* for given k, *match score matrix* and *threshold* T
- Neighborhood sequences of a k-mer w include all strings of length k that, when aligned against w, have the alignment score at least T
- For instance, let I = GCATCGGC, J = CCATCGCCATCG and k = 5, match score be 1, mismatch score be 0 and T = 4

Finding seed hits

- I = GCATCGGC, J = CCATCGCCATCG, k = 5, match score 1, mismatch score 0, T = 4
- This allows for one mismatch in each k-mer
- The neighborhood of the first k-mer of I, GCATC, is GCATC and the 15 sequences

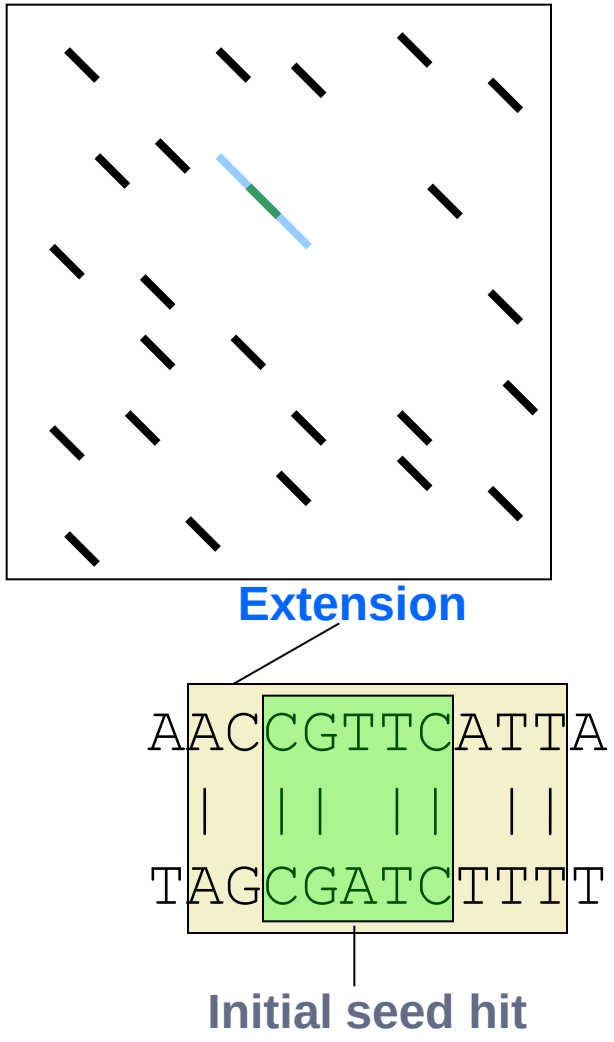
{A, CCATC, T}, {A, GATC, GC, T}, {C, GTC, GCA, T}, {A, CC, GCAT, G}, {A, G, T}

Finding seed hits

- I = GCATCGGC has 4 k-mers and thus 4x16 = 64 5-mer patterns to locate in J
  - Occurrences of patterns in J are called ***seed hits***
- Patterns can be found using exact search in time proportional to the sum of pattern lengths + length of J + number of matches (Aho-Corasick algorithm)
  - Attend 58093 String processing algorithms to learn Aho-Corasick and alike algorithms.
- Compare this approach to FASTA

Extending seed hits: original BLAST

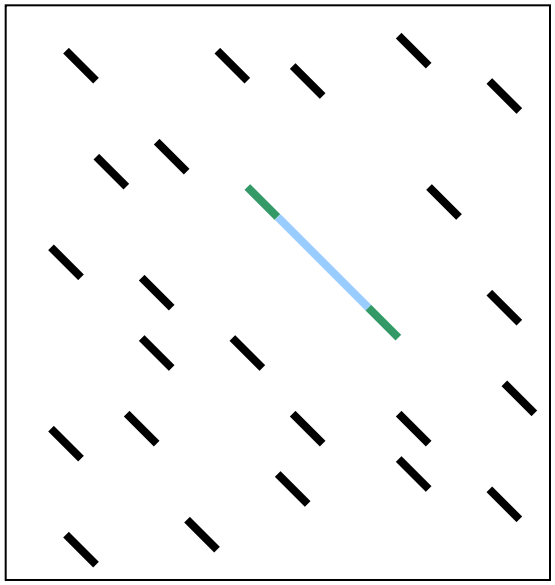
- Initial seed hits are extended into **locally maximal segment pairs** or **High-scoring Segment Pairs** (HSP)
- Extensions do not add gaps to the alignment
- Sequence is extended until the alignment score drops below the maximum attained score minus a threshold parameter value
- All statistically significant HSPs reported



Altschul, S.F., Gish, W., Miller, W., Myers, E. W. and Lipman, D. J., *J. Mol. Biol.*, 215, 403-410, 1990

Extending seed hits: gapped BLAST

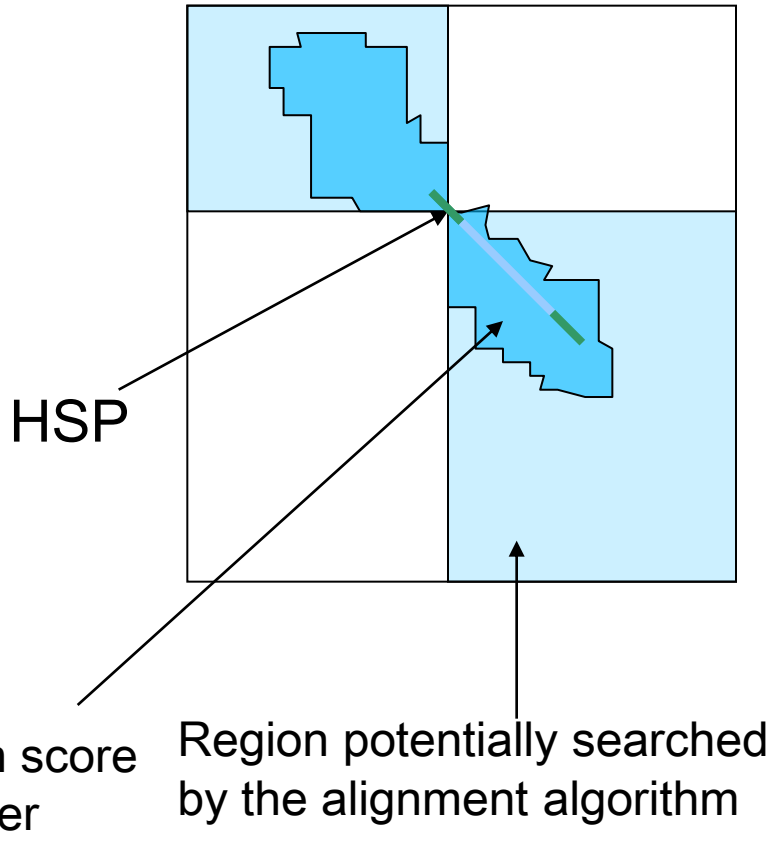
- In a later version of BLAST, two seed hits have to be found on the same diagonal
  - Hits have to be non-overlapping
  - If the hits are closer than A (additional parameter), then they are joined into a HSP
- Threshold value T is lowered to achieve comparable sensitivity
- If the resulting HSP achieves a score at least S_g, a *gapped extension* is triggered



Altschul SF, Madden TL, Schäffer AA, Zhang J, Zhang Z, Miller W, and Lipman DJ, *Nucleic Acids Res*. 1;25(17), 3389-402, 1997

Gapped extensions of HSPs

- Local alignment is performed starting from the HSP
- Dynamic programming matrix filled in ”forward” and ”backward” directions (see figure)
- Skip cells where value would be X_g below the best alignment score found so far



Estimating the significance of results

- In general, we have a score $S(D, X) = s$ for a sequence X found in database D
- BLAST rank-orders the sequences found by p-values
- The p-value for this hit is $P(S(D, Y) \geq s)$ where Y is a random sequence with the same charasteristics as X
  - Measures the amount of ”surprise” of finding sequence X
- A smaller p-value indicates more significant hit
  - A p-value of 0.1 means that one-tenth of random sequences would have as large score as our result

Estimating the significance of results

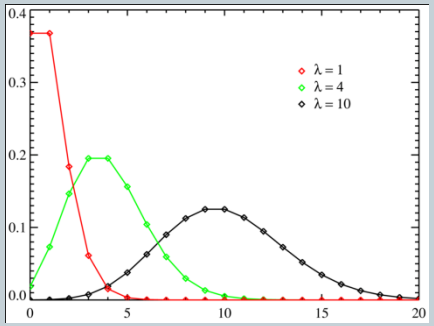
- In BLAST, p-values are computed roughly as follows
- There are mn places to begin an optimal alignment in the m x n alignment matrix
- Optimal alignment is preceded by a mismatch and has t matching (identical) letters
  - (Assume match score 1 and mismatch/indel score -∞)
- Let p = P(two random letters are equal)
- The probability of having a mismatch and then t matches is $(1-p)p^t$

Estimating the significance of results

- We model this event by a Poisson distribution (why?) with mean $\lambda = nm(1-p)p^t$
- P(there is local alignment t or longer)

  $\approx 1 - P(\text{no such event})$

  $= 1 - e^{-\lambda} = 1 - \exp(-nm(1-p)p^t)$
- An equation of the same form is used in Blast:
- E-value = $P(S(D, Y) \geq s) \approx 1 - \exp(-mn\gamma\xi^t)$ where $\gamma > 0$ and $0 < \xi < 1$
- Parameters γ and ξ are estimated from data
- For better analysis, see
  - Chapter 10 in Evens & Grant: *Statistical Methods in Bioinformatics*, Springer 2005 (you may need to read Chapters 1-9 as well to fully understand the theory), or
  - Durbin et al. page 39 (similar as above, but derived with score matrices)



Properties of BLAST

- Better sensitivity than in FASTA
- Still a lossy filter
- Has become *the standard* in Bioinformatics:
  - This is due to the p-value computation and ranking of results
    - However, these computations apply to any alignment algorithm not just to BLAST
    - BLAST may fail to find real occurrences, even those with smallest p-values

Alternatives to BLAST

- Gapped seeds & other advanced filtering mechanisms
  - Burkhardt & Kärkkäinen: Gapped q-Grams (CPM 2001)
  - Li et al.: PatternHunter (Bioinformatics 2002)
- Compressed indexing & search space pruning
  - Lam et at.: Compressed indexing and local alignment of DNA, *Bioinformatics*, 25:1754-1760, 2008.
    - Many short read alignment software extending the idea (Bowtie, BWA, SOAP2, readaligner)
  - Russo et al.: Indexed Hierarchical Approximate String Matching (SPIRE 2008)
- Will be covered in the *Biological Sequence Analysis* course, Spring 2011