19 BLAST, FASTA and BLAT

This lecture is based on the following, which are all recommended reading:

- D.J. Lipman and W.R. Pearson, Rapid and Sensitive Protein Similarity Searches. Science 227, 1435-1441 (1985).
- Pearson, W. R. and Lipman, D. J. Improved tools for biological sequence comparison. Proc. Natl. Acad. Sci USA 85, 2444-2448 (1988).
- S.F. Altschul, W. Gish, W. Miller, E.W. Myers and D.J. Lipman. *Basic local alignment search tool*, J. Molecular Biology, 215:403-410 (1990).
- `http://www.ncbi.nlm.nih.gov/BLAST/tutorial/Altschul-1.html`
- W. James Kent. *BLAT- the BLAST-like alignment tool*, Genome Research 12, 2002.
- D. Gusfield, Algorithms on strings, trees and sequences, pg. 376–381, 1997.

19.1 BLAST and FASTA

Pairwise alignment is used to detect homologies between different protein or DNA sequences, either as global or local alignments.

This can be solved using dynamic programming in time proportional to the product of the lengths of the two sequences being compared.

However, this is too slow for searching current databases and in practice algorithms are used that run much faster, at the expense of possibly missing some significant hits due to the heuristics employed.

Such algorithms usually *seed and extend* approaches in which first small exact matches are found, which are then extended to obtain long inexact ones.

19.2 BLAST terminology

BLAST, the Basic Local Alignment Search Tool, is perhaps the most widely used bioinformatics tool ever written. It is an alignment heuristic that determines "local alignments" between a *query* and a *database*.

Let q be the query and d the database. A *segment* is simply a substring s of q or d.

A *segment-pair* (s, t) consists of two segments, one in q and one d, of the same length.

```
V A L L A R
P A M M A R
```

We think of s and t as being aligned without gaps and *score* this alignment using a substitution score matrix, e.g. BLOSUM or PAM.

The alignment score for (s, t) is denoted by $\sigma(s, t)$.

A *maximum segment pair (MSP)* is any segment pair (s, t) of maximum alignment score $\sigma(s, t)$

A *locally maximal segment pair (LMSP)* is any segment pair (s, t) whose score cannot be improved by shortening or extending the segment pair.

Given a cutoff score C, a segment pair (s, t) is called a *high-scoring segment pair (HSP)*, if it is locally maximal and $\sigma(s, t) \geq C$.

Finally, a *word* is simply a short substring.

Given C, the goal of BLAST is to compute all HSPs.

19.3 The BLAST algorithm

Given three parameters, i.e. a word size K, a word similarity threshold T and a minimum match score C.

For *protein* sequences, BLAST operates as follows:

1. The list of all words of length K that have similarity $\geq T$ to some word in the query sequence q is generated.
2. The database sequence d is scanned for all hits t of words s in the list.
3. Each such *seed* (s, t) is extended until its score $\sigma(s, t)$ falls a certain distance below the best score found for shorter extensions and then all best extensions are reported that have score $\geq C$.

In practice, K is around 4 for proteins.

With care, the list of all words of length K that have similarity $\geq T$ to some word in the query sequence q can be produced in time proportional to the number of words in the list.

These are placed in a "keyword tree" and then, for each word in the tree, all exact locations of the word in the database d are detected in time linear to the length of d.

As BLAST does not allow indels, hit extension is very fast.

Note that the use of seeds of length K and the termination of extensions with fading scores are both steps that speed up the algorithm, but also imply that BLAST is not guaranteed to find all HSPs.

19.4 The BLAST algorithm

For *DNA* sequences, BLAST operates as follows:

- The list of all words of length K in the query sequence a is generated.
- The database d is scanned for all hits of words in this list. Blast uses a two-bit encoding for DNA. This saves space and also search time, as four bases are encoded per byte.

In practice, K is around 12 for DNA.

19.5 The BLAST family

There are a number of different variants of the BLAST program:

- BLASTN: compares a DNA query sequence to a DNA sequence database,
- BLASTP: compares a protein query sequence to a protein sequence database,
- TBLASTN: compares a protein query sequence to a DNA sequence database (6 frames translation),
- BLASTX: compares a DNA query sequence (6 frames translation) to a protein sequence database, and
- TBLASTX: compares a DNA query sequence (6 frames translation) to a DNA sequence database (6 frames translation).

19.6 Statistical significance of an HSP

Given an HSP (s, t) with score $\sigma(s, t)$. How significant is this match? Assume that the length m and n of the query and database are sufficiently large.

HSP scores are characterized by two parameters, K and λ. The expected number of HSPs with score at least S is given by the *E-value*, defined as:

$$E := Kmne^{-\lambda S}.$$

The parameters K and λ depend on the background probabilities of the symbols and on the employed scoring matrix. Essentially, K and λ are scaling-factors for the search space and for the scoring scheme, respectively.

As the E-value depends on the choice of the parameters K and λ, one cannot compare E-values from different BLAST searches. We can obtain a parameter-free E-value as follows.

For a given HSP (s, t) we transform the *raw* score $S = \sigma(s, t)$ into a *bit-score* thus:

$$S' = \frac{\lambda S - \ln K}{\ln 2}.$$

Such bit-scores can be compared between different BLAST searches.

The statistical essence of the scoring system is subsumed in the bit-scores and thus to obtain the significance of a given score S' the only additional value required is the size of the search space: The E-value of a given bit-score S' is

$$E' := mn2^{-S'},$$

simply by solving for S in the previous equation and then plugging the result into the original E-value equation.

The number of random HSPs (s,t) with $\sigma(s,t) \geq C$ can be described by a Poisson distribution. Hence, the probability of finding exactly k HSPs with a score $\geq S$ is given by

$$P(k) = e^{-E}\frac{E^k}{k!},$$

where E is the E-value for S.

The probability of finding at least one HSP "by chance" is

$$1 - P(0) = 1 - e^{-E}.$$

BLAST reports E-values rather than P-values as it is easier, for example, to interpret the difference between an E-value of 5 and 10, than to interpret the difference between a P-value of 0.95 and 0.95. For small E-values < 0.01, the two values are nearly identical.

19.7 FASTA

FASTA (pronounced fast-ay) is a heuristic for finding significant matches between a query string q and a database string d.

FASTA's general strategy is to find the most significant diagonals in the dot-plot or dynamic programming matrix.

The performance of the algorithm is influenced by a *word-size* parameter k, usually 6 for DNA and 2 for amino acids.

The first step of the algorithm is to determine all exact matches of length k between the two sequences, called *hot-spots*.

To find these exact matches quickly, usually a hash table is built that consists of all words of length k that are contained in the query sequence. Exact matches are then found by look-up of each word of length k that is contained in the database.

A *hot-spot* is given by (i,j), where i and j are the *locations* (i.e., start positions) of an exact match of length k.

Any such hot-spot (i,j) lies on the diagonal $(i-j)$ of the dot-plot or dynamic programming matrix.

Using this scheme, the main diagonal has number 0, whereas diagonals above the main one have positive numbers, the ones below negative.

A *diagonal run* is a set of hot-spots that lie in a consecutive sequence on the same diagonal. It corresponds to a gapless local alignment.

A score is assigned to each diagonal run. This is done by giving a positive score to each match (using e.g. the PAM250 match score matrix in the case of proteins) and a negative score for gaps in the run.

The algorithm then locates the ten best diagonal runs.

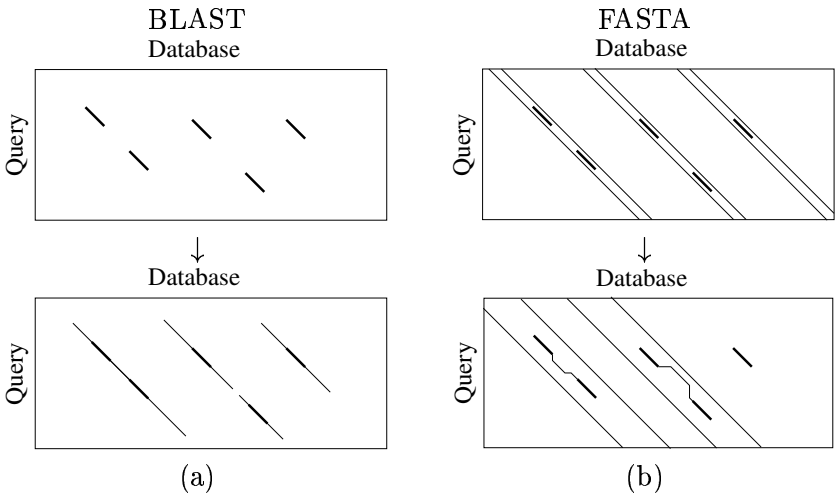
Each of the ten diagonal runs is re-scored using the match score matrix and the best-scoring sub-alignment of each is extracted.

The next step is to combine high scoring sub-alignments into a single larger alignment, allowing the introduction of gaps into the alignment.

Finally, a *banded* Smith-Waterman dynamic program is used to produce an optimal local alignment along the best matched regions.

In this way, FASTA determines a highest scoring region, not all high scoring alignments between two sequences. Hence, FASTA may miss instances of repeats or multiple domains shared by two proteins.

19.8 BLAST and FASTA



(a) In BLAST, individual seeds are found and then extended without indels. (b) IN FASTA, individual seeds contained in the same diagonal are merged and the resulting segments are then connected using a banded Smith-Waterman alignment.

19.9 BLAT- BLAST-Like Alignment Tool

Analyzing vertebrate genomes requires rapid mRNA/DNA and cross-species protein alignments. As the amount of data increases, faster tools are required for comparing sequences.

A new tool, BLAT, is supposedly more accurate and 500 times faster than popular existing tools for mRNA/DNA alignments and 50 times faster for protein alignments at sensitivity settings typically used when comparing vertebrate sequences.

BLAT's speed stems from an index of all non-overlapping K-mers in the genome. The program has several stages: It uses the index to find regions in the genome that are possibly homologous to the query sequence. It performs an alignment between such regions. It stitches together the aligned regions (often exons) into larger alignments (typically genes). Finally, BLAT revisits small internal exons and adjusts large gap boundaries that have canonical splice sites where feasible.

19.10 Mapping ESTs and Mouse reads

In the public assembly of the human genome, the problem arose to map 3 million ESTs to the human genome. Additionally, 13 million (and continually more) whole genome shotgun reads need

to be aligned to the human genome.

The human EST alignments compared 1.75 Gb in 3.72 million ESTs against 2.88 Gb bases of Human DNA and took 200 hours on a farm of 90 Linux boxes.

BLAT was used to align 2.5× of unassembled mouse reads to the masked human genome. This involved 7.51 Gb in 13.3 million reads and took 16,300 CPU hours.

As work continued to finish the human genome, these computations needed to be repeated on a monthly or bi-monthly basis. Hence, a comparison tool was needed that could do the job in a couple of weeks.

This was the motivation for the development of BLAT.

19.11 BLAT vs BLAST

BLAT is similar to BLAST: The program rapidly scans for relatively short matches (hits) and extends these into HSPs. However BLAT differs from BLAST in some important ways:

- BLAST builds an index of the query string and then scans linearly through the database – BLAT builds an index of the database and then scans linearly through the query,
- BLAST triggers an extension when one or two hits occur – BLAT can trigger extensions on any given number of perfect or near perfect matches,
- BLAST returns each area of homology as separate alignments – BLAT stitches them together into larger alignments,
- BLAST delivers a list of exons sorted by size, with alignments extending slightly beyond the edge of each exon – BLAT "unsplices" mRNA onto the genome, giving a single alignment that uses each base of the mRNA only once, with correctly positioned splice sites.

19.12 Seed-and-extend

Like all fast alignment programs, BLAT uses the two stage *seed-and-extend* approach:

- in the *seed stage*, the program detects regions of the two sequences that are likely to be homologous, and
- in the *extend stage*, these regions are examined in detail and alignments are produces for the regions that are indeed homologous according to some criterion.

BLAT provides three different methods for the seed stage:

- Single perfect K-mer matches,
- Multiple perfect K-mer matches, and
- Single near-perfect K-mer matches.

Given a long database sequence and a short query sequence, we will discuss the different seed strategies.

The simplest seed method is to look for subsequences of a given size K that are shared by the query and the database. In many applications, every K-mer in the query sequence is compared with all non-overlapping K-mers in the database sequence.

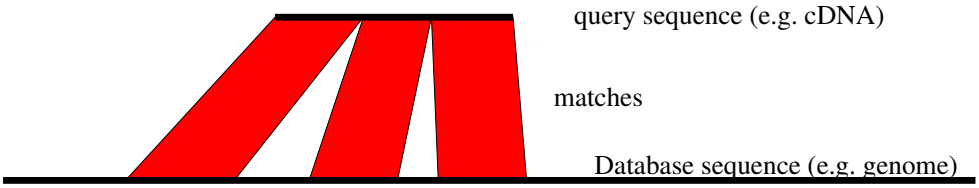
We want to analyze:

1. how many homologous regions are missed, and
2. how many non-homologous regions are passed to the extension stage, using this criteria.

Errors of type (1) will cause the application to miss true homologs, whereas errors of type (2) will increase the running time of the application.

19.13 Some definitions

K: The K-mer size, $8 - 16$ for nucleotides and $3 - 7$ for amino acids.
M: Match ratio between homologous areas, $\approx 98\%$ for cDNA/genomic alignments within the same species, $\approx 89\%$ for protein alignments between human and mouse.
H: The size of a homologous area. For a human exon this is typically $50 - 200$ bp.
G: Database size, e.g. 3 Gb for human.
Q: Query size.
A: Alphabet size, 20 for amino acids, 4 for nucleotides.



19.14 Single perfect matches

Assuming that each letter is independent of the previous letter, the probability that a specific K-mer in a homologous region of the database matches perfectly the corresponding K-mer in the query is:

$$p_1 = M^K.$$

Let $T = \lfloor \frac{H}{K} \rfloor$ denote the number of non-overlapping K-mers in a homologous region of length H.

The probability that at least one non-overlapping K-mer in the homologous region matches perfectly with the corresponding K-mer in the query is:

$$P = 1 - (1 - p_1)^T = 1 - (1 - M^K)^T.$$

The number of non-overlapping K-mers that are expected to match by chance, assuming all letters are equally likely, is:

$$F = (Q - K + 1) \cdot \frac{G}{K} \cdot \left(\frac{1}{A}\right)^K .$$

These formulas can be used to predict the sensitivity and specificity of single perfect nucleotide K-mer matches as a seed-search criterion:

Table 3. Sensitivity and Specificity of Single Perfect Nucleotide K-mer Matches as a Search Criterion

	7	8	9	10	11	12	13	14
A. 81%	0.974	0.915	0.833	0.726	0.607	0.486	0.373	0.314
83%	0.988	0.953	0.897	0.815	0.711	0.595	0.478	0.415
85%	0.996	0.978	0.945	0.888	0.808	0.707	0.594	0.532
87%	0.999	0.992	0.975	0.942	0.888	0.811	0.714	0.659
89%	1.000	0.998	0.991	0.976	0.946	0.897	0.824	0.782
91%	1.000	1.000	0.998	0.993	0.981	0.956	0.912	0.886
93%	1.000	1.000	1.000	0.999	0.995	0.987	0.968	0.957
95%	1.000	1.000	1.000	1.000	0.999	0.998	0.994	0.991
97%	1.000	1.000	1.000	1.000	1.000	1.000	1.000	0.999

B. K	7	8	9	10	11	12	13	14
F	1.3e+07	2.9e+06	635783	143051	32512	7451	1719	399

(*A*) Columns are for K sizes of 7–14. Rows represent various percentage identities between the homologous sequences. The table entries show the fraction of homologies detected as calculated from equation 3 assuming a homologous region of 100 bases. The larger the value of K, the fewer homologies are detected.
(*B*) K represents the size of the perfect match. F shows how many perfect matches of this size expected to occur by chance according to equation 4 in a genome of 3 billion bases using a query of 500 bases.

(Source: Kent 2002)

These formulas can be used to predict the sensitivity and specificity of single perfect amino acid K-mer matches as a seed-search criterion:

Table 4. Sensitivity and Specificity of Single Perfect Amino Acid K-mer Matches as a Search Criterion

K	3	4	5	6	7
A. 71%	0.992	0.904	0.697	0.496	0.317
73%	0.996	0.931	0.752	0.560	0.374
75%	0.998	0.952	0.803	0.625	0.436
77%	0.999	0.969	0.850	0.689	0.503
79%	0.999	0.981	0.890	0.752	0.574
81%	1.000	0.989	0.924	0.810	0.646
83%	1.000	0.994	0.950	0.862	0.718
85%	1.000	0.997	0.970	0.906	0.787
87%	1.000	0.999	0.984	0.942	0.850
89%	1.000	1.000	0.993	0.968	0.903
91%	1.000	1.000	0.997	0.985	0.945
93%	1.000	1.000	0.999	0.995	0.975

B. K	3	4	5	6	7
F	4.2e+07	1.6e+06	62625	2609	112

(*A*) Columns are for K sizes of 3–7. Rows represent various percentage identities between the homologous sequences. The table entries show the fraction of homologies detected as calculated from equation 3 assuming a homologous region of 33 amino acids. (*B*) K represents the size of the perfect match. F shows how many perfect matches of this size are expected to occur by chance according to equation 4 in a translated genome of 3 billion bases using a query of 167 amino acids (corresponding to 500 bases).

(Source: Kent 2002)

Examples

1. For EST alignments, we would like to find seeds for 99% of all homologous regions that have 5% or less sequencing noise. Table 3 indicates that $K = 14$ or less will work. For $K = 14$, we can expect that 399 random hits per query will be produced. A smaller value of K will produce significantly more random hits.

2. The mouse and human genomes average 89% identity at the amino acid level. To find true seeds for 99% of all translated mouse reads requires $K = 5$ or less. For $K = 5$, each read will generate ≈ 62625 random hits, see Table 4.

3. Comparing mouse and human at a nucleotide level, where there is only 86% identity is not feasible: Table 3 implies that $K = 7$ must be used to find 99% of all true hits, but this value generates ≈ 13 million random hits per query.

19.15 Single near-perfect matches

Now consider the case of near-perfect matches, that is, hits with one letter mismatch. The probability that a non-overlapping K-mer in a homologous region of the database matches near-perfectly the corresponding K-mer in the query is:

$$p_1 = K \cdot M^{K-1} \cdot (1 - M) + M^K.$$

Again, the probability that any non-overlapping K-mer in the homologous region matches near-perfectly with the corresponding K-mer in the query is:

$$P = 1 - (1 - p_1)^T.$$

The number of K-mers which match near-perfectly by chance is:

$$F = (Q - k + 1) \cdot \frac{G}{K} \cdot \left(K \cdot \left(\frac{1}{A} \right)^{K-1} \cdot \left(1 - \frac{1}{A} \right) + \left(\frac{1}{A} \right)^K \right).$$

These formulas can be used to predict the sensitivity and specificity of single near-perfect nucleotide K-mer matches as a seed-search criterion:

Table 5. Sensitivity and Specificity of Single Near-Perfect (One Mismatch Allowed) Nucleotide K-mer Matches as a Search Criterion

	12	13	14	15	16	17	18	19	20	21	22
A. 81%	0.945	0.880	0.831	0.721	0.657	0.526	0.465	0.408	0.356	0.255	0.218
83%	0.975	0.936	0.904	0.820	0.770	0.649	0.591	0.535	0.480	0.361	0.318
85%	0.991	0.971	0.954	0.900	0.865	0.767	0.719	0.669	0.619	0.490	0.445
87%	0.997	0.990	0.983	0.954	0.935	0.867	0.833	0.796	0.757	0.634	0.591
89%	1.000	0.997	0.995	0.984	0.976	0.939	0.920	0.897	0.872	0.775	0.741
91%	1.000	1.000	0.999	0.996	0.994	0.979	0.971	0.962	0.950	0.890	0.869
93%	1.000	1.000	1.000	0.999	0.999	0.996	0.994	0.991	0.988	0.963	0.954
95%	1.000	1.000	1.000	1.000	1.000	1.000	0.999	0.999	0.999	0.994	0.992
97%	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000

B. K	12	13	14	15	16	17	18	19	20	21	22
F	275671	68775	17163	4284	1070	267	67	17	4.2	1.0	0.3

(*A*) Columns are for K sizes of 12–22. Rows represent various percentage identities between the homologous sequences. The table entries show the fraction of homologies detected as calculated by equation 6 assuming a homologous region of 100 bases. (*B*) K represents the size of the near-perfect match. F shows how many perfect matches of this size expected to occur by chance according to equation 7 in a genome of 3 billion bases using a query of 500 bases.

(Source: Kent 2002)

These formulas can be used to predict the sensitivity and specificity of single near-perfect amino acid K-mer matches as a seed-search criterion:

Table 6. Sensitivity and Specificity of Single Near-Perfect (One Mismatch Allowed) Amino Acid K-mer Matches as a Search Criterion

	4	5	6	7	8	9
A. 71%	1.000	0.992	0.946	0.823	0.725	0.515
73%	1.000	0.995	0.965	0.867	0.785	0.586
75%	1.000	0.998	0.978	0.905	0.840	0.657
77%	1.000	0.999	0.987	0.935	0.886	0.727
79%	1.000	0.999	0.993	0.959	0.924	0.791
81%	1.000	1.000	0.997	0.976	0.952	0.849
83%	1.000	1.000	0.999	0.987	0.973	0.897
85%	1.000	1.000	0.999	0.994	0.986	0.936
87%	1.000	1.000	1.000	0.997	0.994	0.964
89%	1.000	1.000	1.000	0.999	0.998	0.982
91%	1.000	1.000	1.000	1.000	0.999	0.993
93%	1.000	1.000	1.000	1.000	1.000	0.998
B. K	4	5	6	7	8	9
F	1.2E+08	6.0E+06	300078	14985	749	37

(*A*) Columns are for K sizes of 4–9. Rows represent various percentage identities between the homologous sequences. The table entries show the fraction of homologies detected. (*B*) K represents the size of the near-perfect match. F shows how many perfect matches of this size expected to occur by chance in a translated genome of 3 billion bases using a query of 167 amino acids.

(Source: Kent 2002)

Examples

1. For the purposes of EST alignments, a K of 22 or less produce true seeds for 99% of all queries, while on average producing only one random hit, see Table 5.
2. For comparison of translated mouse reads and the human genome, Table 6 indicates that $K = 8$ would detect true seeds for 99% of all mouse reads, while only generating 374 random hits per read.
3. A comparison of mouse reads and the human genome (86% identity) on the nucleotide level would require $K = 13$ or $K = 12$ to detect true seeds for 99% of the reads, while generating 275671 random hits per read. Using a fast extension program, this computation is feasible.

BLAT implements near-perfect matches allowing one mismatch in a hit, as follows:

A non-overlapping index of all K-mers in the database is generated.

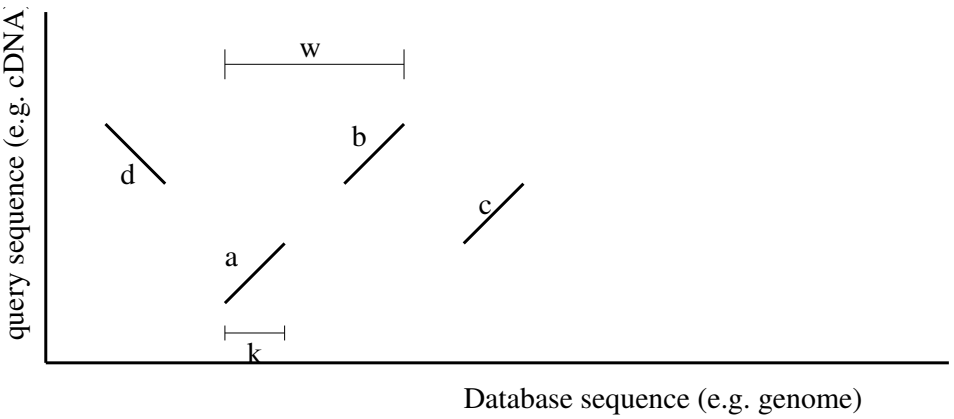
Every possible K-mer in the query sequence that matches in all but one, or in all, positions, is looked up. Hence, this means $K \cdot (A - 1) + 1$ lookups. For an amino-acid search with $K = 8$, for example, 153 lookups are required per occurring K-mer.

For a given level of sensitivity, the near-perfect match criterion runs 15× more slowly than the multiple-perfect match criterion and thus is not so useful in practice.

19.16 Multiple perfect matches

An alternative seeding strategy is to require multiple perfect matches that are constrained to be near each other.

For example, consider a situation where there are two hits between the query and the database sequences that "lie on the same diagonal" and are close to each other (within some given distance W), such as a and b here:



For $N = 1$, the probability that a non-overlapping K-mer in a homologous region of the database matches perfectly the corresponding K-mer in the query is (as discussed above):

$$p_1 = M^K \, .$$

The probability that there are exactly n matches within the homologous region is

$$P_n = p_1^n \cdot (1 - p_1)^{T-n} \cdot \frac{T!}{n! \cdot (T-n)!} \, ,$$

and the probability that there are N or more matches is the sum:

$$P = P_N + P_{N+1} + \ldots + P_T \, .$$

Again, we are interested in the number of matches generated by chance. The probability that such a chain is generated for $N = 1$ is simply:

$$F = (Q - K + 1) \cdot \frac{G}{K} \cdot \left(\frac{1}{A} \right)^K \, .$$

The probability of a second match occurring within W letters after the first is

$$S = 1 - \left(1 - \left(\frac{1}{A} \right)^K \right)^{\frac{W}{K}} \, ,$$

because the second match can occur within any of the $\frac{W}{K}$ non-overlapping K-mers in the database within W letters after the first match.

The number of size N chains of K-mers in which any two consecutive hits are not more than W apart is

$$F_N = F_1 \cdot S^{N-1} \, .$$

These formulas can be used to predict the sensitivity and specificity of multiple nucleotide (2 and 3) perfect K-mer matches as a seed-search criterion:

Table 7. Sensitivity and Specificity of Multiple (2 and 3) Perfect Nucleotide K-mer Matches as a Search Criterion

	2,8	2,9	2,10	2,11	2,12	3,8	3,9	3,10	3,11	3,12
A. 81%	0.681	0.508	0.348	0.220	0.129	0.389	0.221	0.112	0.051	0.021
83%	0.790	0.638	0.475	0.326	0.208	0.529	0.339	0.193	0.099	0.045
85%	0.879	0.762	0.615	0.460	0.318	0.676	0.487	0.313	0.180	0.093
87%	0.942	0.866	0.752	0.611	0.461	0.809	0.649	0.470	0.305	0.177
89%	0.978	0.940	0.868	0.761	0.625	0.910	0.801	0.648	0.476	0.314
91%	0.994	0.980	0.947	0.884	0.787	0.969	0.914	0.815	0.673	0.505
93%	0.999	0.996	0.986	0.962	0.912	0.993	0.976	0.933	0.851	0.722
95%	1.000	1.000	0.998	0.993	0.979	0.999	0.997	0.987	0.961	0.902
97%	1.000	1.000	1.000	1.000	0.999	1.000	1.000	0.999	0.997	0.987

B. N,K	2,8	2,9	2,10	2,11	2,12	3,8	3,9	3,10	3,11	3,12
F	524	27	1.4	0.1	0.0	0.1	0.0	0.0	0.0	0.0

(*A*) Columns are for N sizes of 2 and 3 and K sizes of 8–12. Rows represent various percentage identities between the homologous sequences. The table entries show the fraction of homologies detected as calculated by equation 10. (*B*) N and K represent the number and size of the near-perfect matches, respectively. F shows how many perfect clustered matches expected to occur by chance according to equation 14 in a translated genome of 3 billion bases using a query of 167 amino acids.

(Source: Kent 2002)

These formulas can be used to predict the sensitivity and specificity of multiple amino acid (2 and 3) perfect K-mer matches as a seed-search criterion:

Table 8. Sensitivity and Specificity of Multiple (2 and 3) Perfect Amino Acid K-mer Matches as a Search Criterion

	2,3	2,4	2,5	2,6	2,7	3,3	3,4	3,5	3,6	3,7
A. 71%	0.945	0.643	0.297	0.126	0.044	0.945	0.643	0.297	0.126	0.044
73%	0.965	0.712	0.363	0.167	0.063	0.965	0.712	0.363	0.167	0.063
75%	0.978	0.776	0.436	0.218	0.089	0.978	0.776	0.436	0.218	0.089
77%	0.987	0.833	0.514	0.280	0.123	0.987	0.833	0.514	0.280	0.123
79%	0.993	0.882	0.596	0.353	0.169	0.993	0.882	0.596	0.353	0.169
81%	0.997	0.922	0.678	0.435	0.226	0.997	0.922	0.678	0.435	0.226
83%	0.999	0.952	0.757	0.526	0.298	0.999	0.952	0.757	0.526	0.298
85%	0.999	0.973	0.829	0.622	0.385	0.999	0.973	0.829	0.622	0.385
87%	1.000	0.987	0.889	0.719	0.485	1.000	0.987	0.889	0.719	0.485
89%	1.000	0.995	0.936	0.809	0.596	1.000	0.995	0.936	0.809	0.596
91%	1.000	0.998	0.969	0.886	0.712	1.000	0.998	0.969	0.886	0.712
93%	1.000	1.000	0.988	0.944	0.823	1.000	1.000	0.988	0.944	0.823

B. N,K	2,3	2,4	2,5	2,6	2,7	3,3	3,4	3,5	3,6	3,7
F	171875	245	0.4	0.0	0.0	708	0.0	0.0	0.0	0.0

(*A*) Columns are for N sizes of 2 and 3 and K sizes of 3–7. Rows represent various percentage identities between the homologous sequences. The table entries show the fraction of homologies detected. (*B*) N and K represents the number and size of the perfect matches, respectively. F shows how many perfect clustered matches expected to occur by chance in a translated genome of 3 billion bases using a query of 167 amino acids.

(Source: Kent 2002)

19.17 Clumping hits

BLAT builds a non-overlapping index of all K-mers in the database, ignoring those K-mers that occur too often in the database, those containing ambiguity codes and optionally, those in lower case ("soft screened regions").

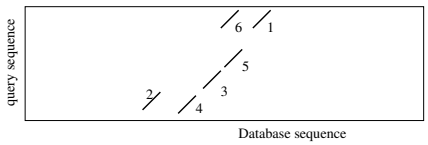
BLAT then looks up each overlapping K-mer of the query sequence in the index, obtaining a list L of *hits*. Each hit consists of a database position and a query position.

The next step is to form *clumps* of hits that represent regions in the database sequence that are homologous to the query sequence. Each such clump consists of a number of hits (that exceeds a given minimum number of hits) that form a chain in which two consecutive hits are not too far apart from each other and also in which the gap size in either sequence does not exceed a given threshold.

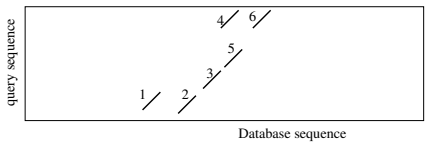
Multiple hits are clumped together as follows:

- The hit list L is sorted by database coordinate.
- The list L is split into buckets of size 64 kb each, based on the database coordinate.
- Each bucket is sorted along the diagonal, i.e. hits are sorted by the value of database position minus query position.
- Hits that are within the gap limit are grouped together into proto-clumps.
- Hits within proto-clumps are then sorted by their database coordinate and put into real clumps, if they are within the window limit on the database coordinate.
- Clumps within 300 bp or 100 amino acids of each other in the database are merged and then 500 bp are added to each end of a clump.

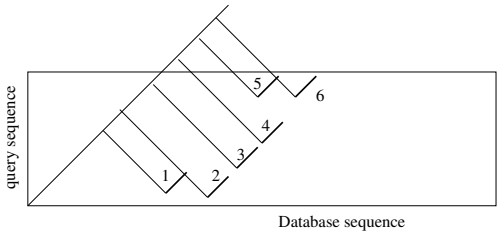
A list of hits:



Sorted by database coordinate:



Sorted along the diagonal:



19.18 Nucleotide alignments

Clumping is the first part of the extension stage. In the case of nucleotide alignments, each clump is then processed as follows.

- A hit list is generated between the query sequence q and the homologous region h in the database, looking for smaller, perfect K-mers.
- If a K-mer w in q matches multiple K-mers in h, then w is repeatedly extended by one until the match is unique or exceeds a certain size.
- The hits are extended as far as possible, without mismatches.
- Overlapping hits are merged.

- If there are gaps in the alignment in both the query and the database, then the algorithm recurses to fill in the gaps, using a smaller K.
- Then extensions using indels followed by matches are considered.
- Large gaps in the query sequence often correspond to introns and they are slid around to find the best GT/AG consensus sequence for the intron ends.

19.19 Protein alignments

In the case of amino acid sequences, each clump is processed as follows:

- All hits obtained in the seed stage is extended into maximally scoring ungapped alignments (HSPs) using a score function where a match is worth 2 and a mismatch is worth 1.
- A graph is build with HSPs as nodes.
- If HSP A starts before HSP B in both sequences, then an edge is put from A to B that is weighted by the score of B minus a gap penalty based on the distances between A and B.
- If A and B overlap, then an optimal *crossover* position x is determined that maximizes the sum of score of A up to x and B starting from x and the edge weight is set accordingly.
- A dynamic program then extracts the maximal scoring alignment by traversing the graph.
- The HSPs contained in the path are removed and if any HSPs are left then the dynamic program is run again.

19.20 Mouse/Human alignment choices

The similarity between the human and mouse genomes is 86% on the nucleotide level and 89% on the amino-acid level (for coding regions). The following table compares DNA vs amino acid alignments, and different seeding strategies:

Table 10. Mouse/Human Alignment Choices

	K	F	F Translated
1 Perfect-DNA	7	13,078,962	
1 Perfect-AA	5	62,625	187,875
Near-perfect-DNA	12	275,671	
Near-perfect-AA	8	749	2,247
2 Perfect DNA	6	237,983	
2 Perfect AA	4	245	734
3 Perfect DNA	5	109,707	
3 Perfect AA	3	708	2,123

Assuming 86% base identity and 89% amino acid identity, this table shows the maximum K sizes and number of chance matches passed to the alignment stage when searching for regions of 100 bases (or 33 amino acids) with at least a 99% chance of detecting the homology. These values reflect our targets for human/mouse alignments. For translated DNA sequences, the F value is multiplied by six to reflect three reading frames on both strands of the query. Even with this multiplication, the specificity for a given sensitivity is several orders of magnitude greater in the amino acid rather than the nucleotide domain.

(Source: Kent 2002)