CSE 397-497:
Computational Issues in Molecular Biology

Lecture 6

Spring 2004

Topics for today

Based on premise that algorithms we've studied are too slow:

- Faster method for global comparison when sequences are similar (§3.3.4).

Even this might not be fast enough. Also, biologists more commonly need local comparison. What they really use is:

- PAM matrices (§3.5.1).
- BLAST (§3.5.2 + Altschul et al. '90 paper).
- FAST (§3.5.3 + Lipman & Pearson '85 paper).

Note: we can understand a little bit of the "why," but finer details go beyond a computer scientist's level of understanding.

Speed

Recall that all of the dynamic programming algorithms for basic sequence comparison we've discussed so far take time $O(mn)$.

From a computer science standpoint, this would be considered reasonably efficient under most circumstances.

But this isn't fast enough when you consider the lengths of the sequences and the sizes of the databases in question.

> "GenBank continues to grow at an exponential rate with 5.4 million new sequences added over the past 12 months. As of Release 131 in August 2002, GenBank contained over 22.6 billion nucleotide bases from 18.2 million different sequences."
>
> "GenBank," D. A. Benson, I. Karsch-Mizrachi, D. J. Lipman, J. Ostell and D. L. Wheeler, *Nucleic Acids Research*, 2003, Vol. 31, No. 1, pp. 23-27.

Observations

In general, we only really care about the comparison when the sequences in question are in some way "similar." E.g.,

- To identifiy all homologous sequences in a database.*
- To identify "motifs" with a sequence-similarity significantly better than random chance.

Two "rules of thumb" we'll be using:

- If two sequences are similar, their optimal alignment in a global comparison can't include many deletions or insertions.
- If two sequences are similar, they're likely to contain short subsequences that match exactly.

* In the literature, the terms "similarity" and "homology" are often used interchangeably. To be accurate, *similarity* refers to sequences being similar according to certain statistical criteria, while *homology* refers to the molecules performing similar biological functions or having evolved from a common ancestor. Although similar sequences tend to perform similar functions, this is by no means guaranteed. The only way to establish homology is through experiment.

A faster method for comparing similar sequences

In global case, when two sequences are similar, it's possible to compare them more rapidly. What does “similar” imply?

That their lengths must be nearly the same. Why is this true?

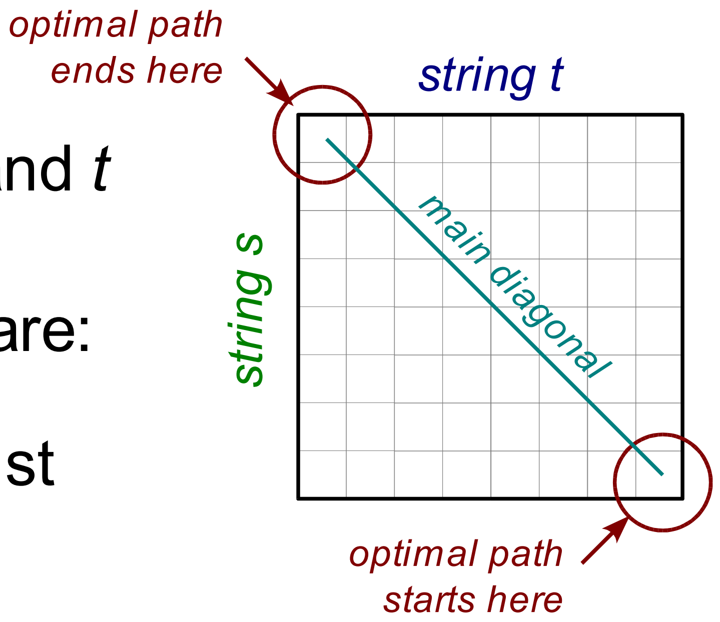
Say costs are constant: c_{match} (> 0) is cost of match and c_{indel} (< 0) is cost of insertion/deletion. What do we know?

$$\mathrm{sim}(s,t) \leq c_{match} \cdot \min(m,n) + c_{indel} \cdot |m-n|$$

In other words, maximum possible similarity drops when *m* is much different from *n*. This makes sense because we have to delete or insert additional symbols to make lengths correspond.

A faster method for comparing similar sequences

For simplicity, let's say lengths of *s* and *t* are identical. Hence, dynamic programming matrix is a perfect square:



We know where the optimal path must start and where it must end.

What could happen in between?

Every time the path takes a step away from the main diagonal (e.g., via a deletion), there must be a corresponding step towards the diagonal somewhere (e.g., an offsetting insertion).

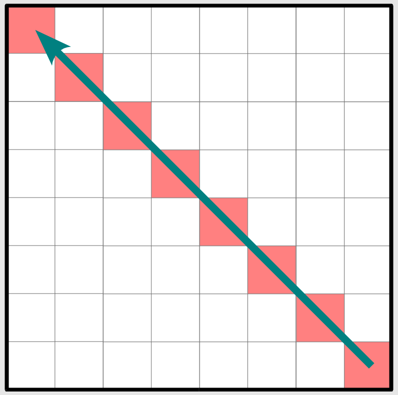
The total cost of this step is $2 \cdot c_{indel}$.

A faster method for comparing similar sequences

For simplicity, let's say that $c_{match} = 1$ and $c_{indel} = -1$.

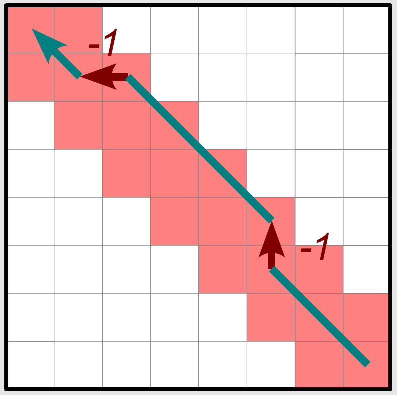
Then the maximum possible similarity between two sequences of length 8 is __8__.

If two sequences are identical (8), what must optimal path look like?



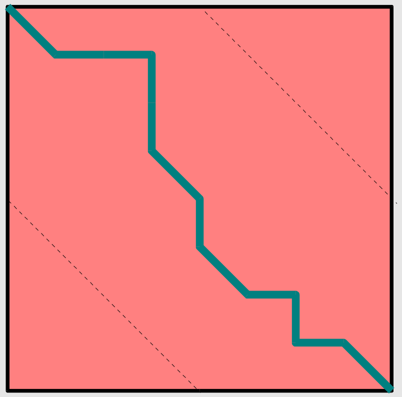
If two sequences are nearly identical (5), what might optimal path look like?

(There can be at most one indel pair; one possible path shown.)

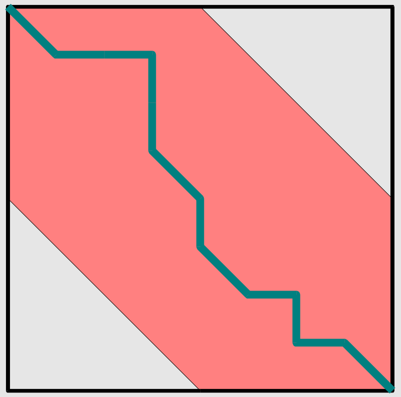


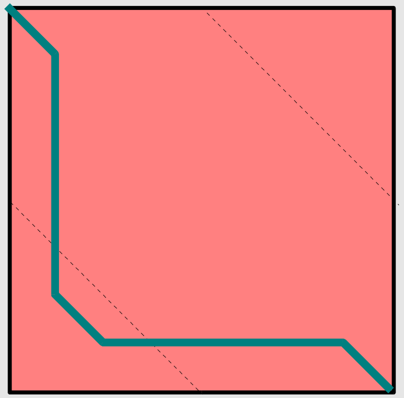
A faster method for comparing similar sequences

Notice what's happening: if optimal path takes route near main diagonal, we only need to compute shaded cells.

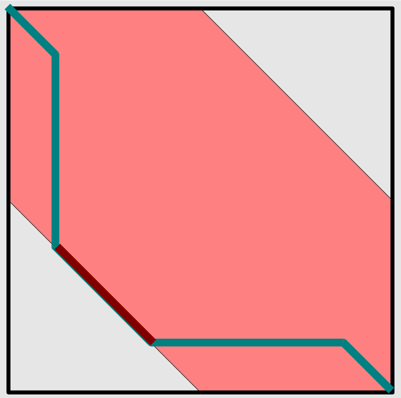


if optimal path stays near main diagonal, we get same optimal result

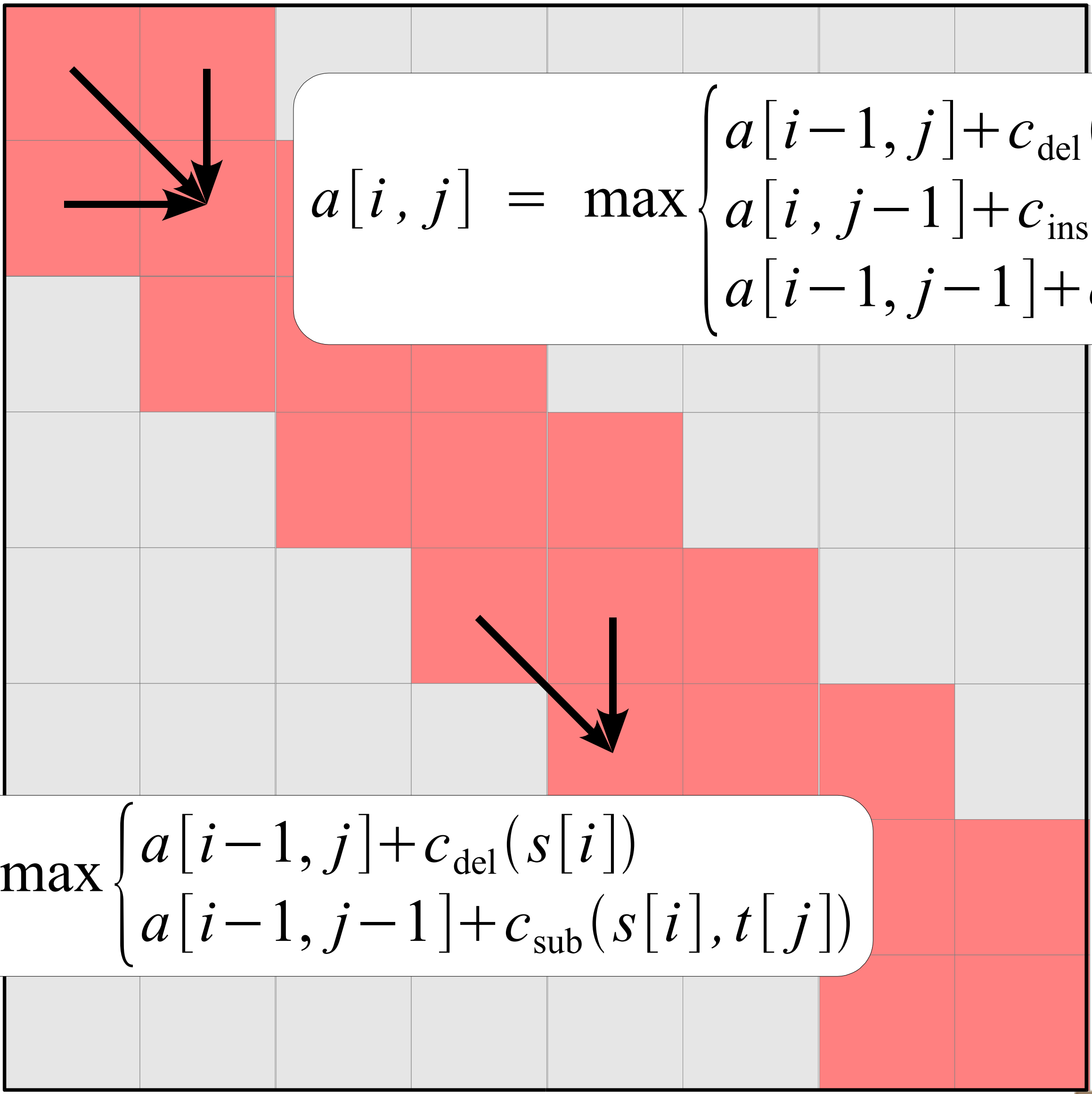




if optimal path strays far from main diagonal, we get suboptimal result



A faster method for comparing similar sequences

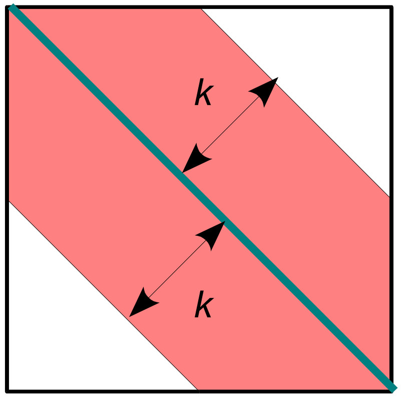


$$a[i,j] = \max\begin{cases} a[i-1,j]+c_{\text{del}}(s[i]) \\ a[i,j-1]+c_{\text{ins}}(t[j]) \\ a[i-1,j-1]+c_{\text{sub}}(s[i],t[j]) \end{cases}$$

$$a[i,j] = \max\begin{cases} a[i-1,j]+c_{\text{del}}(s[i]) \\ a[i-1,j-1]+c_{\text{sub}}(s[i],t[j]) \end{cases}$$

A faster method for comparing similar sequences

Say we compute in band ±*k* around main diagonal (inner loop of recurrence becomes $-k \leq i - j \leq k$), then time complexity is $O(kn)$.



How do we know if we actually found an optimal alignment?

Well, we know the algorithm is exact if the maximum number of indel pairs is at most *k*, so if the score we get is greater than any score we could get for (*k*+1) or more indel pairs, we win:

$$c_{match} \cdot (n - (k+1)) \; + \; c_{indel} \cdot 2(k+1)$$

Provided score is greater than this value, we know it's optimal.

A faster method for comparing similar sequences

How are we supposed to know right value of *k* to start off?

Try $k = 1$ and see if we found an optimal path ...

... if not, try $k = 2, 4, 8, 16$, etc.

Time complexity is $n + 2n + 4n + \ldots + kn \leq 2kn$.

I.e., it doesn't hurt not knowing final *k* to begin with.

Upper bound on *k* is difference between maximum possible score and true optimal score (i.e., distance off main diagonal).

Database search

As noted previously, biologists care more about local comparisons than global comparisons.

What can be done to accelerate this process?

BLAST = *Basic Local Alignment Search Tool*.

BLAST takes a query sequence and searches the database for sequences that contain a segment so similar to one in the query that such similarity is deemed statistically significant (i.e., unlikely to occur by chance).

Say query is "The quick brown fox jumps over the lazy dog."

Finding "the" in a collection of English text strings may not be significant.

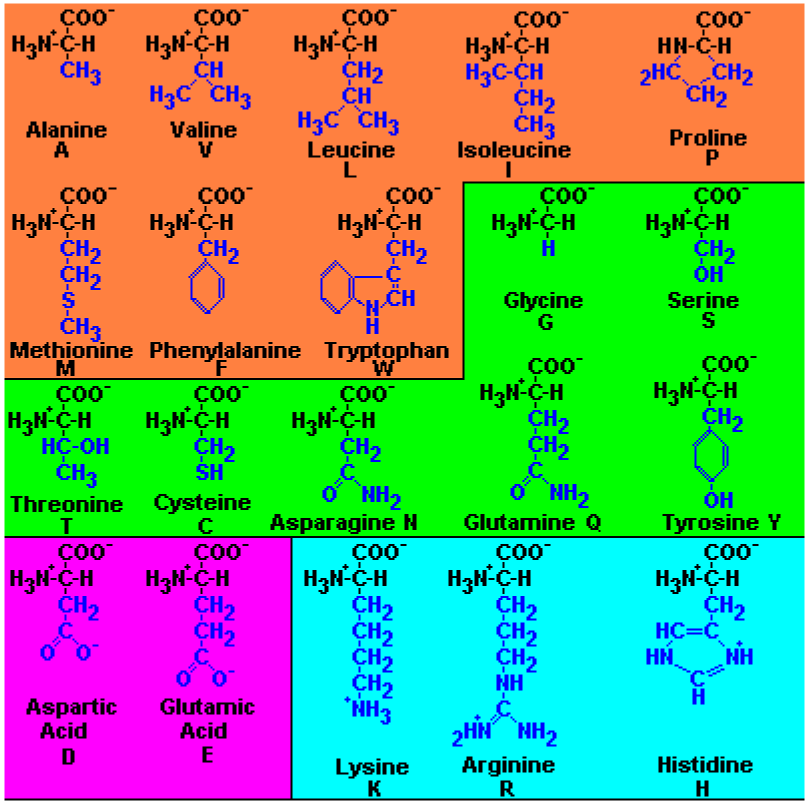
Finding "quick brown fox" in same collection may well be significant.

Note: original BLAST does not allow for indels or gaps.

Database search

Important questions to be answered:

- How do we assess the similarity between segments of the query sequence and the database sequences?
- How do we evaluate statistical significance?



Previously, we assumed constant costs for substitutions.

In reality, we need to determine whether mutations we see are random or have biological basis.

Case for amino acids is more complex than for nucelotides.

PAM matrices

One important measure of mutual substitution used by biologists is *Point Accepted Mutations* (or *Percent of Accepted Mutations*) matrices, often called *PAM* matrices.

PAM matrices are computed in terms of evolutionary "units." E.g., a 1-PAM matrix reflects evolution leading to an average of one mutation per hundred amino acids. Higher level PAM matrices are computed from this.

To build a 1-PAM matrix, M^1, we need:

- list of accepted mutations, $a \to b$,
- probability of occurrence for each amino acid, p_a.

This allows us to compute mutability of amino acid *a* into *b* as:

$$M^1_{ab} = Pr(a \to b) = Pr(a \to b \mid a \text{ changed}) \, Pr(a \text{ changed})$$

PAM and scoring matrices

Given the 1-PAM matrix, we can compute mutability of amino acid *a* into *b* though *k* units of evolution as a *k*-PAM matrix:

$$M^k_{ab} \text{ is the } (a, b) \text{ entry in } \underbrace{M^1 M^1 \ \ldots \ M^1}_{k \text{ times}}$$

PAM matrices are used to build scoring matrices whose entries measure the probability an amino acid is present due to a mutation versus a random event:

$$\frac{M_{ab}}{p_b}$$

For convenience, actual score is 10 times logarithm of ratio.

250-PAM matrix

	A	R	N	D	C	Q	E	G	H	I	L	K	M	F	P	S	T	W	Y	V
A	2																			
R	-2	6																		
N	0	0	2																	
D	0	-1	2	4																
C	-2	-4	-4	-5	12															
Q	0	1	1	2	-5	4														
E	0	-1	1	3	-5	2	4													
G	1	-3	0	1	-3	-1	0	5												
H	-1	2	2	1	-3	3	1	-2	6											
I	-1	-2	-2	-2	-2	-2	-2	-3	-2	5										
L	-2	-3	-3	-4	-6	-2	-3	-4	-2	2	6									
K	-1	3	1	0	-5	1	0	-2	0	-2	-3	5								
M	-1	0	-2	-3	-5	-1	-2	-3	-2	2	4	0	6							
F	-4	-4	-4	-6	-4	-5	-5	-5	-2	1	2	-5	0	9						
P	1	0	-1	-1	-3	0	-1	-1	0	-2	-3	-1	-2	-5	6					
S	1	0	1	0	0	-1	0	1	-1	-1	-3	0	-2	-3	1	2				
T	1	-1	0	0	-2	-1	0	0	-1	0	-2	0	-1	-3	0	1	3			
W	-6	2	-4	-7	-8	-5	-7	-7	-3	-5	-2	-3	-4	0	-6	-2	-5	17		
Y	-3	-4	-2	-4	0	-4	-4	-5	0	-1	-1	-4	-2	7	-5	-3	-3	0	10	
V	0	-2	-2	-2	-2	-2	-2	-1	-2	4	2	-2	2	-1	-1	-1	0	-6	-2	4

BLAST

With matrix such as 250-PAM, we can score protein segments:

```
Q   M   F   T   R
E   M   L   K   F      =  6
2   6   2   0  -4
```

Nucleotide segments can be scored using similar rules:

```
A   C   G   T   A
A   C   G   A   C      =  7
5   5   5  -4  -4
```

BLAST defines *maximal segment pair* (*MSP*) to be highest scoring pair of identical length segments from two sequences.

A segment pair is *locally maximal* if its score cannot be improved either by extending it or shortening it.

BLAST

Overall BLAST approach is to locate short, high scoring segment pairs between query and database sequence. It then extends these "seeds" in both directions until it seems unlikely further extensions will raise score (this is heuristic).

Basic steps in BLAST:

(1) Compile list of high-scoring "words" from query.

(2) Search for hits for these in database (call these "seeds").

(3) Extend seeds so long as score improves.

Scanning database (step 2) can be facilitated by using hash table or building finite state automaton (FSA).

Recommended word size is 4 for proteins, 12 for nucleic acids.

BLAST example: compiling word list

Say that the query protein sequence is **QLNFSAGW**.

Divide the query into all subsequences of length *w* = 2:

```
                Q L N F S A G W
           ┌    Q L
           │      L N
           │        N F
   yields  │          F S
           │            S A
           │              A G
           └                G W
```

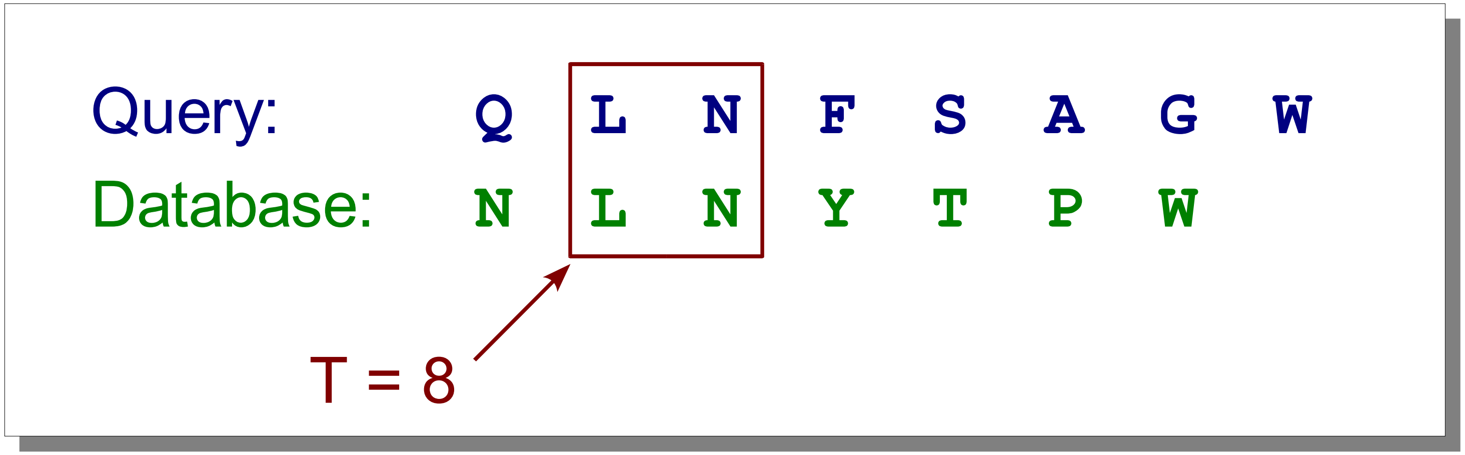
BLAST example: compiling word list

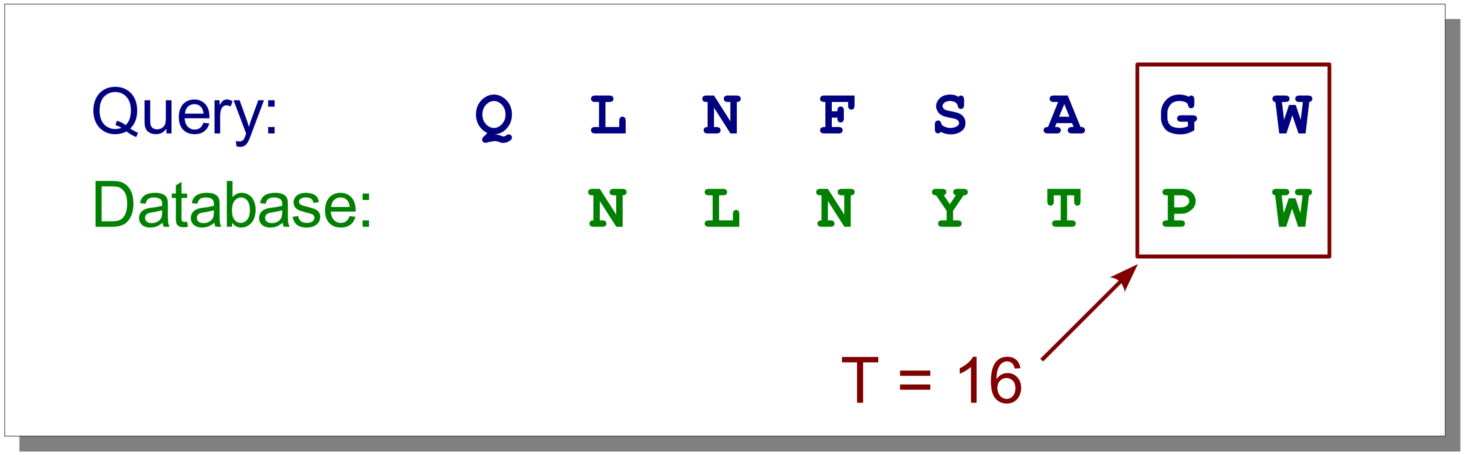
Generate all 2-letter combinations that score at least *T* = 8 when aligned with query subsequence using 250-PAM:

QL	**QL** (10), **QM** (10), **HL** (9)
LN	**LN** (8)
NF	**NF** (11), **AF** (9), **NY** (9), **DF** (11), **QF** (10), **EF** (10), **GF** (9), **HF** (11), **KF** (10), **SF** (10)
FS	**FS** (11), **FA** (10), **FN** (10), **FD** (9), **FG** (10), **FP** (10), **YS** (9)
SA	none
AG	none
GW	**AW** (18), **RW** (14), **NW** (17), **GW** (18), **QW** (16), **EW** (17), **GW** (22), **HW** (15), **IW** (14), **KW** (15), **MW** (14), **PW** (16), **SW** (18), **TW** (17), **VW** (10)

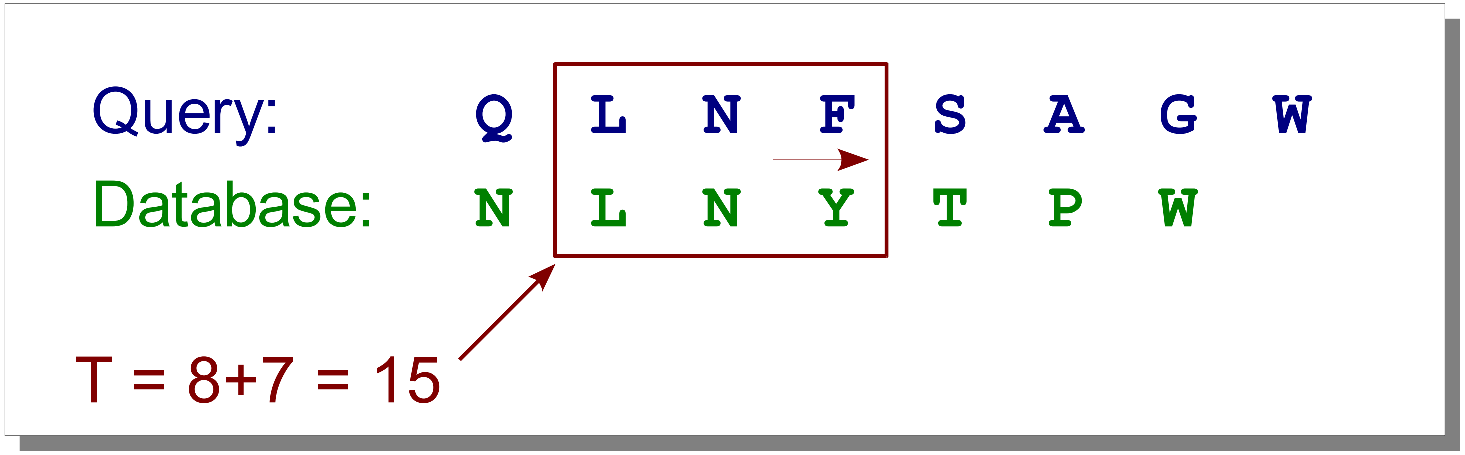
BLAST example: identifying high scoring segment pairs

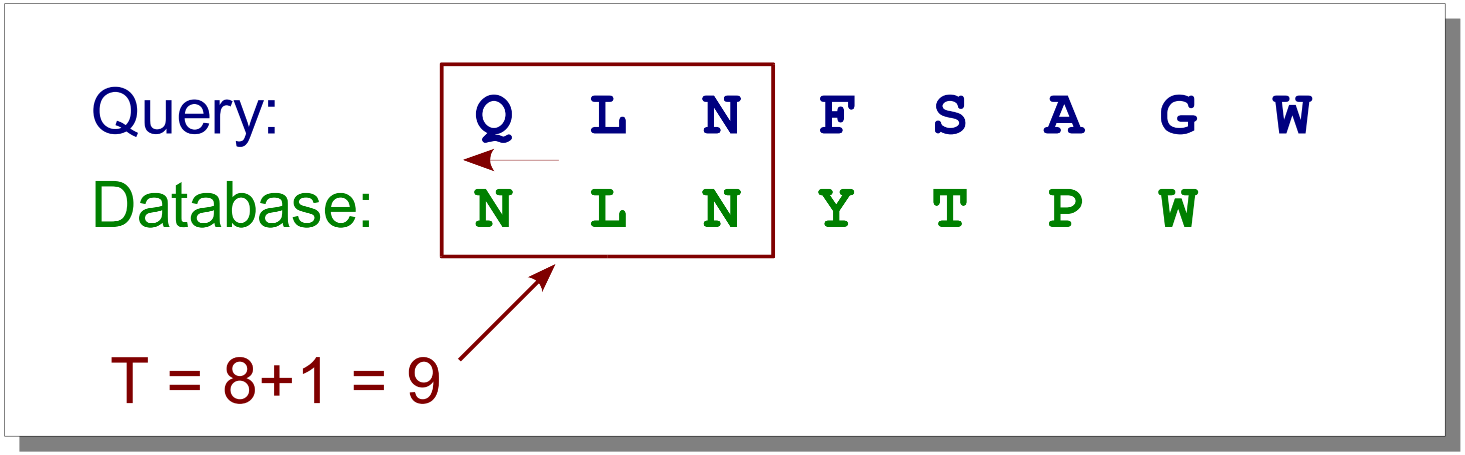
Say that the database protein sequence is **NLNYTPW**.





BLAST example: extending a hit





Continue trying to extend hit until score falls below threshold.

BLAST statistics

When is a maximal alignment score statistically significant?

Turn it around: what is probability it could arise by chance?

Consider two random sequences generated by rolling a 20-faced die an appropriate number of times. The die is loaded according to relative frequencies of amino acids in database.

When *m* and *n* are large, the expected number of distinct segment pairs between *s* and *t* with score above *S* is:

$$Kmne^{-\lambda S}$$

where *K* and λ are parameters obtained from the scoring matrix and the amino acid probabilities (distribution is Poisson).

BLAST statistics

From probability distribution of maximal alignment scores, we can determine probability of getting a random alignment as good as one observed. If this probability is small (say < 0.05), the alignment is deemed statistically significant.

In BLAST output, this probability *p* is converted to a bit score which is equal to -log 2 *p*. Smaller probability = larger bit score.

We can also calculate expected number of times, *E*, an alignment with such a score would occur in a database of the same size. BLAST lets you discard alignments expected to occur more than certain number of times (default 10).

Note: statistical significance must NOT be mistaken as biological truth.

BLAST variations

blastp:	compares an amino acid query sequence against a protein sequence database.
blastn:	compares a nucleotide query sequence against a nucleotide sequence database.
blastx:	compares the six-frame conceptual translation products of a nucleotide query sequence (both strands) against a protein sequence database.
tblastn:	compares a protein query sequence against a nucleotide sequence database dynamically translated in all six reading frames (both strands).
tblastx:	compares the six-frame translations of a nucleotide query sequence against the six-frame translations of a nucleotide sequence database.
PSI-Blast:	Position-Specific Iterated BLAST. This is potentially a very sensitive method to pull out significant hits in a protein-protein database search. The first search pass is conducted in exactly the same way as other BLAST searches, but chosen results are used to create a profile used as the query sequence in a second and subsequent iterations. These steps may be repeated until the set of hits being reported does not change (convergence).

FAST

Philosophically, FAST is similar to BLAST (FAST came first):

- short matching segments between query and database sequences are identified,
- these matches are extended and refined (including running dynamic programming on likely candidates),
- matches that are statistically significant are reported.

FAST builds a lookup table of all *k*-tuples in query (*k* = 1 or 2).

It then scans database sequence and records offsets of matches it finds. E.g.,

> if a *k*-tuple at *s*[*i*] matches one at *t*[*i*], offset = 0
>
> if a *k*-tuple at *s*[*i*] matches one at *t*[*i*+3], offset = -3
>
> if a *k*-tuple at *s*[*i*] matches one at *t*[*i*-2], offset = +2

FAST

Keeping track of *k*-tuple matches at each offset allows FAST to identify strongest diagonals for further anaysis.

	1	2	3	4	5	6	7	8	9	10	11
Query (s):	H	A	R	F	Y	A	A	Q	I	V	L

A	2, 6, 7
F	4
H	1
I	9
L	11
Q	8
R	3
V	10
Y	5

	1	2	3	4	5	6	7	8
Database (t):	V	D	M	A	A	Q	I	A
Offsets:	+9			−2	−3	+2	+2	−6
				+2	+1			−2
				+3	+2			−1

-7	-6	-5	-4	-3	-2	-1	0	+1	+2	+3		+9
	1			1	2	1		1	4	1	...	1

good choice

GenBank

"GenBank," D. A. Benson, I. Karsch-Mizrachi, D. J. Lipman, J. Ostell and D. L. Wheeler, *Nucleic Acids Research*, 2003, Vol. 31, No. 1, pp. 23-27.

One point you may have found confusing:

> "ESTs continue to be the major source of new sequence records and gene sequences. Last year there were 8.6 million ESTs in GenBank. Over the past year, the number of ESTs has increased by over 45% to the current total of 12.5 million sequences representing over 430 different organisms."

So what is an EST?

ESTs

EST = *Expressed Sequence Tag*

As we know, when a gene is expressed, its unique sequence of bases is transcribed onto messenger RNA (mRNA). These molecules, which serve as templates for protein synthesis, can be captured from the various tissues in which they are found. Although fragile and transient, mRNA molecules can be translated into sturdier complementary DNA (cDNA).

ESTs are cloned segments of cDNA molecules that have been partly sequenced – usually several hundred bases from both ends. These sequenced ends could provide information about the location and function of the entire gene they represent.

http://www.genomenewsnetwork.org/timeline/1991_Venter.shtml

Wrap-up

Reading for next time:

- "Cross-Domain Approximate String Matching," D. Lopresti and G. Wilfong, *Proceedings of the Sixth International Symposium on String Processing and Information Retrieval*, September 1999, Cancún, Mexico, pp. 120-127.

Remember:

- Come to class prepared to discuss what you have read.
- Check Blackboard regularly for updates.