We have dynamic programming, what else do we need?

Running time: $O(n^2)$ on two sequences of length n

Memory: basic implementation $O(n^2)$, but can be done in $O(n)$

We need a more efficient algorithm, particularly for comparative genomics

Heuristic alignment

- Trade sensitivity for speed (some alignments not found)
- Reduce the search to “promising” parts of the matrix

Heuristic local alignment

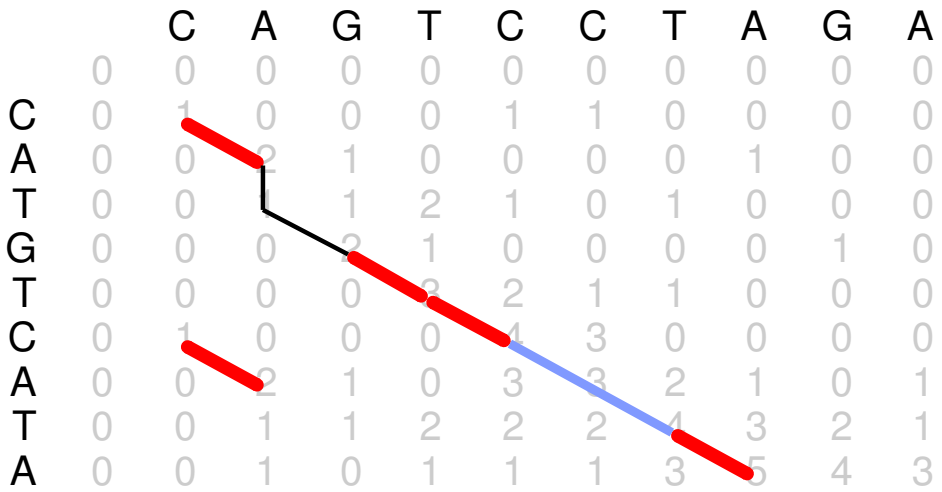
BLASTN [Altschul et al 1990], FASTA [Pearson 1988]

- Find short exact matches of length k (**seeds**)
- Extend hits along diagonals to ungapped alignments
- Connect alignments on nearby diagonals to gapped alignment
- Possibly optimize by dynamic programming

Heuristic local alignment

Example: start from **seeds** of length $k = 2$

(in practice we would use $k = 11$ or more)



1. find hits
2. ungapped
3. gapped

Running time of heuristic local alignment

Algorithm

- Find seeds (short exact matches of length k)
- **Expensive step:** extend/connect seeds to longer alignments

Random seeds of length k: not part of any high-scoring alignment.

These are filtered in the extension step, but they slow down the program

How many random hits?

Two unrelated nucleotides match with probability $1/4$

We have k matches in a row with probability 4^{-k}

Expected number of false positives roughly $nm4^{-k}$

Increase of k by 1 means cca 4-fold decrease of spurious seeds

Sensitivity of heuristic local alignment

Algorithm

- Find seeds (short exact matches of length k)
- **Expensive step:** extend/connect seeds to longer alignments

Some alignments not found: high score but no seed of length k

Example: `CA-GTCCTA` no seed of length $k \geq 4$

`CATGTCATA`

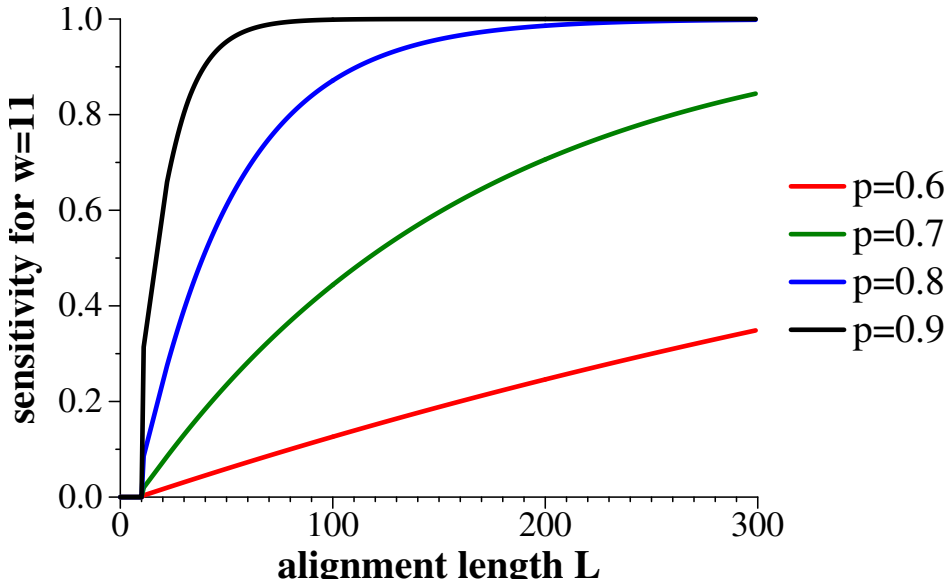
Sensitivity: fraction of real alignments containing a seed of length k

Can we estimate the sensitivity?

Assume random ungapped alignment of length L

Every position match with probability p

Sensitivity $f(L, p) = \Pr(\text{alignment contains } k \text{ consecutive matches})$



(human-mouse: $p \approx 0.7$)

a_n = number of binary sequences of length n that do not contain k adjacent 1s ($k = 2$)

0	00	000	0000
1	01	001	0001
	10	010	0010
	11	011	0011
		100	0100
		101	0101
		110	0110
		111	0111
			1000
			1001
			1010
			1011
			1100
			1101
			1110
			1111

$a_0 = 2^0 = 1$, since $n < k$ (empty string)

$a_1 = 2^1 = 2$, since $n < k$

$a_2 = 2 + 1 = 3$, *i.e, #(".0") + #(".1")*

$a_3 = 3 + 2 + 0 = 5$

i.e, #("..0") + #("..1") + #(".11")

$a_4 = 5 + 3 = 8$

i.e, #("...0") + #("...1") + #("..11")

$a_n = a_{n-1} + a_{n-2}$

$a_n = a_{n-1} + a_{n-2} + a_{n-3} + \ldots + a_{n-k}$

a_n = number of binary sequences of length n that do not contain k adjacent 1s ($k = 3$)

0	00	000	0000
1	01	001	0001
	10	010	0010
	11	011	0011
		100	0100
		101	0101
		110	0110
		111	0111
			1000
			1001
			1010
			1011
			1100
			1101
			1110
			1111

$a_0 = 2^0 = 1$, since $n < k$ (empty string)

$a_1 = 2^1 = 2$, since $n < k$

$a_2 = 2^2 = 4$, since $n < k$

$a_3 = 4 + 2 + 1 = 7$

i.e, #("..0") + #("..1") + #("..11")

$a_4 = 7 + 4 + 2 = 13$

i.e, #("...0") + #("...1") + #("..11")

$a_n = a_{n-1} + a_{n-2} + a_{n-3}$

$a_n = a_{n-1} + a_{n-2} + a_{n-3} + \ldots + a_{n-k}$

How to find short exact matches?

- Create a **dictionary** of k-mers (short substrings of length k) from the first sequence.
- Search for all k-mers from the second sequence in the dictionary

Exmple: CAGTCCTAGA vs CATGTCATA

Dictionary:

```
AG 2, 8
CA 1
CC 5
CT 6
GA 9
GT 3
TA 7
TC 4
```

Search for:

```
CA → 1
AT → -
TG → -
GT → 3
TC → 4
CA → 1
AT → -
TA → 7
```