BLAST

text: Philip Clausen
Basic Local Alignment Search Tool
Number of possible alignments

\[
\binom{m+n}{n} = \frac{(m+n)!}{m! \ n!}
\]
What if we made the two sequences form a matrix?

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>C</th>
<th>G</th>
<th>T</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>G</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>T</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>T</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Simplified Needleman-Wunsch

\[
D(i, j) = \max \left\{ \begin{array}{l}
D(i + 1, j + 1) + s(x_i, y_j) \\
D(i + 1, j) - g \\
D(i, j + 1) - g
\end{array} \right. \\
g = 3, \ s(x, y) = \begin{cases}
1 & x = y \\
-1 & x \neq y
\end{cases}
\]
What if we made the two sequences form a matrix?

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>C</th>
<th>G</th>
<th>T</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>0</td>
<td>-2</td>
<td>-4</td>
<td>-10</td>
</tr>
<tr>
<td>G</td>
<td>-4</td>
<td>-1</td>
<td>-1</td>
<td>-7</td>
</tr>
<tr>
<td>T</td>
<td>-6</td>
<td>-3</td>
<td>0</td>
<td>-2</td>
</tr>
<tr>
<td>T</td>
<td>-10</td>
<td>-7</td>
<td>-4</td>
<td>1</td>
</tr>
<tr>
<td>-12</td>
<td>-9</td>
<td>-6</td>
<td>-3</td>
<td>0</td>
</tr>
</tbody>
</table>
Drastic improvement on computation speed

\[
\frac{(m + n)!}{m! \cdot n!} \gg m \cdot n
\]

<table>
<thead>
<tr>
<th>Sequence lengths</th>
<th>Needleman-Wunsch</th>
<th>Exhaustive search</th>
<th>Number of e· in the universe</th>
</tr>
</thead>
<tbody>
<tr>
<td>150 X 150</td>
<td>22 500</td>
<td>( \sim 10^{89} )</td>
<td>( \sim 10^{80} )</td>
</tr>
</tbody>
</table>
Is a run time of “m n” good enough?
Basic Local Alignment Search Tool

1. Uses seed and extend armed with the Smith Waterman algorithm to speed up comparison of query sequences to a database.

2. Too slow for raw data, and therefore it requires assembly.

3. Guarantied to give the optimal alignments, given a correct seeding length.

4. Due to indexing of query sequences on the fly it is memory efficient, while maintaining $O(n)$ running time.
Seed and extend

Simple example, $k = 5$:

\begin{align*}
Q: & \text{AAGACTCCGACTGGGACTTTGATGTTCGAAAGA} \\
T: & \text{GACTGGGACTTTTGATG}
\end{align*}
Seed and extend

Simple example, $k = 5$:

Q: AAGACTCCGACTGGGACTTTGATGTTCGAAAGA

T: GACTGGGACTTTGATG
Seed and extend

Simple example, $k = 5$:

Q: AAGACTCCGACTGGGACTTTGATGTTCGAAAGA

T: GACTGGGACTTTGATG
Seed and extend

Simple example, $k = 5$:

Q: AAGACTCCGACTGGGACTTTGATGTTCGAAAGA

T: GACTGGGACTTTGATG
Seed and extend

Simple example, $k = 5$:

Seed

Q: AAGACTCCC\textcolor{green}{GACTG}GGAGTTTTGATG\textcolor{orange}{TTCGAAAGA}

Extend

T: \textcolor{green}{GACTGGGAGTTTTGATG}
Seed and extend, impact on computation

\[ mn \gg m + n \]
BLAST overview

Find all Seeds

Extend and refine

Final alignments
Learning objectives

1. Illustrate alignment in form of a matrix.

2. Understand how seed and extend improves alignment time.
Don’t worry! BLAST have been invented for you. So you can skip the details.
Questions and exercises