Comparing Genes
INTRODUCTION TO SEQUENCE ALIGNMENT
Comparing Genes is a Fundamental Problem in Biology

Comparing Genes Problem:
- **Input:** Two genes.
- **Output:** How “similar” these genes are.

**Goal:** Convert this important biological question into a well-defined computational problem.

© 2022 Phillip Compeau
Try 1: Hamming Distance

Hamming Distance Problem:
• **Input:** Two strings.
• **Output:** The number of “mismatched” symbols in the two strings.
Try 1: Hamming Distance

Hamming Distance Problem:
• **Input:** Two strings.
• **Output:** The number of “mismatched” symbols in the two strings.

ATGCATGCA
TGCATGCA

Hamming distance = 8
Try 1: Hamming Distance

Hamming Distance Problem:
• **Input:** Two strings.
• **Output:** The number of “mismatched” symbols in the two strings.

ATGCATGC
TGCATGCA

Hamming distance = 8

**STOP:** What are the issues with this approach?
Try 1: Hamming Distance

Hamming Distance Problem:
• Input: Two strings.
• Output: The number of “mismatched” symbols in the two strings.

ATGCATGC
TGCATGCA

Hamming distance = 8

Note: these strings have a long shared substring, it just doesn’t line up perfectly.
Try 2: Longest Substring

Longest Shared Substring Problem:
- **Input:** Two strings.
- **Output:** The longest substring shared by both strings.

STOP: What are the weaknesses of using the length of a longest shared substring to represent the similarity between two strings?
Try 2: Longest Substring

Longest Shared Substring Problem:

• **Input:** Two strings.
• **Output:** The longest substring shared by both strings.

Consider the strings AAACAAACAAACAAACAAACAAA and AAAGAAAGAAAGAAAGAAAGAAAGAAA. These strings are very similar, but they don’t have a long shared substring in common.
Try 3: Counting Shared $k$-Mers

Instead of finding a longest shared substring of two strings, we will count the number of shared substrings.

For simplicity, we restrict to substrings of the same length; recall that a $k$-mer is the term we use in comp bio for a string of length $k$. 
Try 3: Counting Shared $k$-Mers

$s1 = \text{ACGTATACACGTAT}$

<table>
<thead>
<tr>
<th>String</th>
<th>Count</th>
</tr>
</thead>
<tbody>
<tr>
<td>ACA</td>
<td>1</td>
</tr>
<tr>
<td>ACG</td>
<td>2</td>
</tr>
<tr>
<td>ATA</td>
<td>1</td>
</tr>
<tr>
<td>CAC</td>
<td>1</td>
</tr>
<tr>
<td>CGT</td>
<td>2</td>
</tr>
<tr>
<td>GTA</td>
<td>2</td>
</tr>
<tr>
<td>TAC</td>
<td>1</td>
</tr>
<tr>
<td>TAT</td>
<td>2</td>
</tr>
</tbody>
</table>

$s2 = \text{TATCGGTATATCCTAC}$

<table>
<thead>
<tr>
<th>String</th>
<th>Count</th>
</tr>
</thead>
<tbody>
<tr>
<td>ATA</td>
<td>1</td>
</tr>
<tr>
<td>ATC</td>
<td>2</td>
</tr>
<tr>
<td>CCT</td>
<td>1</td>
</tr>
<tr>
<td>CGG</td>
<td>1</td>
</tr>
<tr>
<td>CTA</td>
<td>1</td>
</tr>
<tr>
<td>GGT</td>
<td>1</td>
</tr>
<tr>
<td>GTA</td>
<td>1</td>
</tr>
<tr>
<td>TAC</td>
<td>1</td>
</tr>
<tr>
<td>TAT</td>
<td>3</td>
</tr>
<tr>
<td>TCC</td>
<td>1</td>
</tr>
<tr>
<td>TCG</td>
<td>1</td>
</tr>
</tbody>
</table>

STOP: How should we count the # of shared 3-mers of two strings?
Try 3: Counting Shared $k$-Mers

$$s1 = \text{ACGTATACACGTAT}$$

<table>
<thead>
<tr>
<th>String</th>
<th>Count</th>
</tr>
</thead>
<tbody>
<tr>
<td>ACA</td>
<td>1</td>
</tr>
<tr>
<td>ACG</td>
<td>2</td>
</tr>
<tr>
<td>ATA</td>
<td>1</td>
</tr>
<tr>
<td>CAC</td>
<td>1</td>
</tr>
<tr>
<td>CGT</td>
<td>2</td>
</tr>
<tr>
<td>GTA</td>
<td>2</td>
</tr>
<tr>
<td>TAC</td>
<td>1</td>
</tr>
<tr>
<td>TAT</td>
<td>2</td>
</tr>
</tbody>
</table>

$$s2 = \text{TATCGGTATATCCTAC}$$

<table>
<thead>
<tr>
<th>String</th>
<th>Count</th>
</tr>
</thead>
<tbody>
<tr>
<td>ATA</td>
<td>1</td>
</tr>
<tr>
<td>ATC</td>
<td>2</td>
</tr>
<tr>
<td>CCT</td>
<td>1</td>
</tr>
<tr>
<td>CGG</td>
<td>1</td>
</tr>
<tr>
<td>CTA</td>
<td>1</td>
</tr>
<tr>
<td>GGT</td>
<td>1</td>
</tr>
<tr>
<td>GTA</td>
<td>1</td>
</tr>
<tr>
<td>TAC</td>
<td>1</td>
</tr>
<tr>
<td>TAT</td>
<td>3</td>
</tr>
<tr>
<td>TCC</td>
<td>1</td>
</tr>
<tr>
<td>TCG</td>
<td>1</td>
</tr>
</tbody>
</table>

Take minimum counts for each shared $k$-mer:

$$1 + 1 + 1 + 2 = 5$$
Toward a Better Approach

STOP: What similarities do you see in these strings?

ATGCTTA
TGCATTTAA
Toward a Better Approach

STOP: What similarities do you see in these strings?

ATGCTTTA
TGCATTTAA

Key Point: we can find similarities if we “slide” the strings, letting symbols shift (but stay in same order).

ATGC–TTA–
–TGCATTTAA

© 2022 Phillip Compeau
Symbol Matching Problem:

- **Input:** Two strings.
- **Output:** The greatest number of matched symbols in any “alignment” of the two strings.
Toward a More Accurate Problem

Symbol Matching Problem:
- **Input:** Two strings.
- **Output:** The greatest number of matched symbols in any “alignment” of the two strings.

**Exercise:** How many matches can you find if the strings are ATGTTATA and ATCGTCC? What algorithm did you use?
Matching Symbols as a Game

<table>
<thead>
<tr>
<th>Growing alignment</th>
<th>Remaining symbols</th>
<th>Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>ATGTTATA</td>
<td>ATCGTCC</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>+1</td>
</tr>
<tr>
<td></td>
<td>A</td>
<td>+1</td>
</tr>
<tr>
<td></td>
<td>TGTTATA</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>AT</td>
<td>+1</td>
</tr>
<tr>
<td></td>
<td>G</td>
<td></td>
</tr>
<tr>
<td></td>
<td>T</td>
<td></td>
</tr>
<tr>
<td></td>
<td>TATA</td>
<td>+1</td>
</tr>
<tr>
<td></td>
<td>A</td>
<td></td>
</tr>
<tr>
<td></td>
<td>TC</td>
<td></td>
</tr>
<tr>
<td></td>
<td>GT</td>
<td></td>
</tr>
<tr>
<td></td>
<td>CC</td>
<td>+1</td>
</tr>
<tr>
<td></td>
<td>AT</td>
<td></td>
</tr>
<tr>
<td></td>
<td>G</td>
<td></td>
</tr>
<tr>
<td></td>
<td>TT</td>
<td></td>
</tr>
<tr>
<td></td>
<td>ATA</td>
<td>+1</td>
</tr>
<tr>
<td></td>
<td>A</td>
<td></td>
</tr>
<tr>
<td></td>
<td>TG</td>
<td></td>
</tr>
<tr>
<td></td>
<td>TT</td>
<td></td>
</tr>
<tr>
<td></td>
<td>TA</td>
<td></td>
</tr>
<tr>
<td></td>
<td>A</td>
<td></td>
</tr>
<tr>
<td></td>
<td>T</td>
<td></td>
</tr>
<tr>
<td></td>
<td>A</td>
<td></td>
</tr>
<tr>
<td></td>
<td>G</td>
<td></td>
</tr>
<tr>
<td></td>
<td>T</td>
<td></td>
</tr>
<tr>
<td></td>
<td>C</td>
<td></td>
</tr>
<tr>
<td></td>
<td>T</td>
<td></td>
</tr>
<tr>
<td></td>
<td>C</td>
<td>+1</td>
</tr>
</tbody>
</table>

FIGURE 5.1 One way of playing the alignment game for the strings ATGTTATA and ATCGTCC, with score 4. At each step, we choose to remove either one or both symbols from the left of the two sequences in the “remaining symbols” column. If we remove both symbols, then we align them in the “growing alignment”. If we remove only one symbol, then we align this symbol with a space symbol in the growing alignment. Matched symbols are shown in red (and receive score 1). Mismatched symbols are shown in purple; symbols aligned against space symbols are shown in blue or green depending on which sequence they were removed from.

© 2022 Phillip Compeau
Matching Symbols as a Game

<table>
<thead>
<tr>
<th>Growing alignment</th>
<th>Remaining symbols</th>
<th>Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>ATGTTATA</td>
<td>ATCGTCC</td>
<td></td>
</tr>
<tr>
<td>A</td>
<td>TGTATA</td>
<td>+1</td>
</tr>
<tr>
<td>A</td>
<td>TCGTCC</td>
<td></td>
</tr>
<tr>
<td>AT-</td>
<td>GTTATA</td>
<td></td>
</tr>
<tr>
<td>ATC</td>
<td>GTCC</td>
<td></td>
</tr>
<tr>
<td>AT-</td>
<td>GTATA</td>
<td></td>
</tr>
<tr>
<td>ATC</td>
<td>GTCC</td>
<td></td>
</tr>
<tr>
<td>AT-</td>
<td>GTTATA</td>
<td></td>
</tr>
<tr>
<td>ATC</td>
<td>GT-CC</td>
<td></td>
</tr>
<tr>
<td>AT-</td>
<td>GTTATA</td>
<td></td>
</tr>
<tr>
<td>ATC</td>
<td>GT-CC</td>
<td></td>
</tr>
</tbody>
</table>

FIGURE 5.1 One way of playing the alignment game for the strings ATGTTATA and ATCGTCC, with score 4. At each step, we choose to remove either one or both symbols from the left of the two sequences in the “remaining symbols” column. If we remove both symbols, then we align them in the “growing alignment.” If we remove only one symbol, then we align this symbol with a space symbol in the growing alignment. Matched symbols are shown in red (and receive score 1). Mismatched symbols are shown in purple; symbols aligned against space symbols are shown in blue or green depending on which sequence they were removed from.

© 2022 Phillip Compeau
## Matching Symbols as a Game

<table>
<thead>
<tr>
<th>Growing alignment</th>
<th>Remaining symbols</th>
<th>Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>ATGGTTATA</td>
<td>ATCGTCC</td>
<td></td>
</tr>
<tr>
<td>ATCGTCC</td>
<td>ATGGTTATA</td>
<td>+1</td>
</tr>
<tr>
<td>ATGGTTATA</td>
<td>ATCGTCC</td>
<td></td>
</tr>
<tr>
<td>ATGGTTATA</td>
<td>ATCGTCC</td>
<td>+1</td>
</tr>
<tr>
<td>ATGGTTATA</td>
<td>ATCGTCC</td>
<td></td>
</tr>
<tr>
<td>ATGGTTATA</td>
<td>ATCGTCC</td>
<td>+1</td>
</tr>
<tr>
<td>ATGGTTATA</td>
<td>ATCGTCC</td>
<td></td>
</tr>
<tr>
<td>ATGGTTATA</td>
<td>ATCGTCC</td>
<td>+1</td>
</tr>
</tbody>
</table>

FIGURE 5.1 One way of playing the alignment game for the strings \textit{ATGTTATA} and \textit{ATCGTCC}, with score 4. At each step, we choose to remove either one or both symbols from the left of the two sequences in the “remaining symbols” column. If we remove both symbols, then we align them in the “growing alignment”. If we remove only one symbol, then we align this symbol with a space symbol in the growing alignment. Matched symbols are shown in red (and receive score 1). Mismatched symbols are shown in purple; symbols aligned against space symbols are shown in blue or green depending on which sequence they were removed from.
### Matching Symbols as a Game

<table>
<thead>
<tr>
<th>Growing alignment</th>
<th>Remaining symbols</th>
<th>Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>ATGTTATA</td>
<td>ATCGTCC</td>
<td></td>
</tr>
<tr>
<td>+1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ATGTTATA</td>
<td>ATCGTCC</td>
<td>+1</td>
</tr>
<tr>
<td>+1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>AT-GTTATA</td>
<td>ATCGTCC</td>
<td>+1</td>
</tr>
<tr>
<td>+1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>AT-GTTAT</td>
<td>ATCGTCC</td>
<td>+1</td>
</tr>
<tr>
<td>+1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>AT-GTTATA</td>
<td>ATCGTCC</td>
<td>+1</td>
</tr>
<tr>
<td>+1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>AT-GTTATA</td>
<td>ATCGTCC</td>
<td>+1</td>
</tr>
<tr>
<td>+1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>AT-GTTATA</td>
<td>ATCGTCC</td>
<td>+1</td>
</tr>
<tr>
<td>+1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>AT-GTTATA</td>
<td>ATCGTCC</td>
<td>+1</td>
</tr>
<tr>
<td>+1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>AT-GTTATA</td>
<td>ATCGTCC</td>
<td>+1</td>
</tr>
<tr>
<td>+1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>AT-GTTATA</td>
<td>ATCGTCC</td>
<td>+1</td>
</tr>
<tr>
<td>+1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>AT-GTTATA</td>
<td>ATCGTCC</td>
<td>+1</td>
</tr>
<tr>
<td>+1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>AT-GTTATA</td>
<td>ATCGTCC</td>
<td>+1</td>
</tr>
<tr>
<td>+1</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

FIGURE 5.1 One way of playing the alignment game for the strings ATGTTATA and ATCGTCC, with score 4. At each step, we choose to remove either one or both symbols from the left of the two sequences in the “remaining symbols” column. If we remove both symbols, then we align them in the “growing alignment”. If we remove only one symbol, then we align this symbol with a space symbol in the growing alignment. Matched symbols are shown in red (and receive score 1). Mismatched symbols are shown in purple; symbols aligned against space symbols are shown in blue or green depending on which sequence they were removed from.

© 2022 Phillip Compeau
### Matching Symbols as a Game

#### CHAPTER 5

<table>
<thead>
<tr>
<th>Growing alignment</th>
<th>Remaining symbols</th>
<th>Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>A T G T T A T A</td>
<td>A T C G T C C</td>
<td></td>
</tr>
<tr>
<td>A T G T T A T A</td>
<td>T C G T C C</td>
<td>+1</td>
</tr>
<tr>
<td>A T G T T A T A</td>
<td>G T T A T A</td>
<td>+1</td>
</tr>
<tr>
<td>A T G T T A T A</td>
<td>C G T C C</td>
<td>+1</td>
</tr>
<tr>
<td>A T G T T A T A</td>
<td>G T T A T A</td>
<td></td>
</tr>
<tr>
<td>A T G T T A T A</td>
<td>G T C C</td>
<td></td>
</tr>
<tr>
<td>A T G T T A T A</td>
<td>T T A T A</td>
<td>+1</td>
</tr>
<tr>
<td>A T G T T A T A</td>
<td>T C C</td>
<td></td>
</tr>
</tbody>
</table>

FIGURE 5.1 One way of playing the alignment game for the strings `ATGTTATA` and `ATCGTCC`, with score 4. At each step, we choose to remove either one or both symbols from the left of the two sequences in the “remaining symbols” column. If we remove both symbols, then we align them in the “growing alignment”. If we remove only one symbol, then we align this symbol with a space symbol in the growing alignment. Matched symbols are shown in red (and receive score 1). Mismatched symbols are shown in purple; symbols aligned against space symbols are shown in blue or green depending on which sequence they were removed from.
### FIGURE 5.1

One way of playing the alignment game for the strings $ATGTTATA$ and $ATCGTCC$, with score 4. At each step, we choose to remove either one or both symbols from the left of the two sequences in the “remaining symbols” column. If we remove both symbols, then we align them in the “growing alignment”. If we remove only one symbol, then we align this symbol with a space symbol in the growing alignment. Matched symbols are shown in red (and receive score 1). Mismatched symbols are shown in purple; symbols aligned against space symbols are shown in blue or green depending on which sequence they were removed from.
### Matching Symbols as a Game

**FIGURE 5.1** One way of playing the alignment game for the strings `ATGTTATA` and `ATCGTCC`, with score 4. At each step, we choose to remove either one or both symbols from the left of the two sequences in the "remaining symbols" column. If we remove both symbols, then we align them in the "growing alignment". If we remove only one symbol, then we align this symbol with a space symbol in the growing alignment. Matched symbols are shown in red (and receive score 1). Mismatched symbols are shown in purple; symbols aligned against space symbols are shown in blue or green depending on which sequence they were removed from.

<table>
<thead>
<tr>
<th>Growing alignment</th>
<th>Remaining symbols</th>
<th>Score</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>ATGTTATA</code></td>
<td><code>ATCGTCC</code></td>
<td></td>
</tr>
<tr>
<td><code>A</code></td>
<td><code>GTTATA</code></td>
<td>+1</td>
</tr>
<tr>
<td><code>A</code></td>
<td><code>TCGTCC</code></td>
<td></td>
</tr>
<tr>
<td><code>AT</code></td>
<td><code>TGTTATA</code></td>
<td>+1</td>
</tr>
<tr>
<td><code>AT</code></td>
<td><code>CTCGTCC</code></td>
<td></td>
</tr>
<tr>
<td><code>AT</code></td>
<td><code>TGTTATA</code></td>
<td>+1</td>
</tr>
<tr>
<td><code>AT</code></td>
<td><code>CTCGTCC</code></td>
<td></td>
</tr>
<tr>
<td><code>AT</code></td>
<td><code>TGTTATA</code></td>
<td>+1</td>
</tr>
<tr>
<td><code>AT</code></td>
<td><code>CTCGTCC</code></td>
<td></td>
</tr>
<tr>
<td><code>AT</code></td>
<td><code>TGTTATA</code></td>
<td>+1</td>
</tr>
<tr>
<td><code>AT</code></td>
<td><code>CTCGTCC</code></td>
<td></td>
</tr>
<tr>
<td><code>AT</code></td>
<td><code>TGTTATA</code></td>
<td>+1</td>
</tr>
<tr>
<td><code>AT</code></td>
<td><code>CTCGTCC</code></td>
<td></td>
</tr>
<tr>
<td><code>AT</code></td>
<td><code>TGTTATA</code></td>
<td>+1</td>
</tr>
<tr>
<td><code>AT</code></td>
<td><code>CTCGTCC</code></td>
<td></td>
</tr>
<tr>
<td><code>AT</code></td>
<td><code>TGTTATA</code></td>
<td>+1</td>
</tr>
<tr>
<td><code>AT</code></td>
<td><code>CTCGTCC</code></td>
<td></td>
</tr>
<tr>
<td><code>AT</code></td>
<td><code>TGTTATA</code></td>
<td>+1</td>
</tr>
<tr>
<td><code>AT</code></td>
<td><code>CTCGTCC</code></td>
<td></td>
</tr>
<tr>
<td><code>AT</code></td>
<td><code>TGTTATA</code></td>
<td>+1</td>
</tr>
<tr>
<td><code>AT</code></td>
<td><code>CTCGTCC</code></td>
<td></td>
</tr>
<tr>
<td><code>AT</code></td>
<td><code>TGTTATA</code></td>
<td>+1</td>
</tr>
<tr>
<td><code>AT</code></td>
<td><code>CTCGTCC</code></td>
<td></td>
</tr>
<tr>
<td><code>AT</code></td>
<td><code>TGTTATA</code></td>
<td>+1</td>
</tr>
<tr>
<td><code>AT</code></td>
<td><code>CTCGTCC</code></td>
<td></td>
</tr>
<tr>
<td><code>AT</code></td>
<td><code>TGTTATA</code></td>
<td>+1</td>
</tr>
<tr>
<td><code>AT</code></td>
<td><code>CTCGTCC</code></td>
<td></td>
</tr>
<tr>
<td><code>AT</code></td>
<td><code>TGTTATA</code></td>
<td>+1</td>
</tr>
<tr>
<td><code>AT</code></td>
<td><code>CTCGTCC</code></td>
<td></td>
</tr>
<tr>
<td><code>AT</code></td>
<td><code>TGTTATA</code></td>
<td>+1</td>
</tr>
<tr>
<td><code>AT</code></td>
<td><code>CTCGTCC</code></td>
<td></td>
</tr>
<tr>
<td><code>AT</code></td>
<td><code>TGTTATA</code></td>
<td>+1</td>
</tr>
<tr>
<td><code>AT</code></td>
<td><code>CTCGTCC</code></td>
<td></td>
</tr>
<tr>
<td><code>AT</code></td>
<td><code>TGTTATA</code></td>
<td>+1</td>
</tr>
<tr>
<td><code>AT</code></td>
<td><code>CTCGTCC</code></td>
<td></td>
</tr>
<tr>
<td><code>AT</code></td>
<td><code>TGTTATA</code></td>
<td>+1</td>
</tr>
<tr>
<td><code>AT</code></td>
<td><code>CTCGTCC</code></td>
<td></td>
</tr>
<tr>
<td><code>AT</code></td>
<td><code>TGTTATA</code></td>
<td>+1</td>
</tr>
<tr>
<td><code>AT</code></td>
<td><code>CTCGTCC</code></td>
<td></td>
</tr>
<tr>
<td><code>AT</code></td>
<td><code>TGTTATA</code></td>
<td>+1</td>
</tr>
<tr>
<td><code>AT</code></td>
<td><code>CTCGTCC</code></td>
<td></td>
</tr>
<tr>
<td><code>AT</code></td>
<td><code>TGTTATA</code></td>
<td>+1</td>
</tr>
<tr>
<td><code>AT</code></td>
<td><code>CTCGTCC</code></td>
<td></td>
</tr>
<tr>
<td><code>AT</code></td>
<td><code>TGTTATA</code></td>
<td>+1</td>
</tr>
<tr>
<td><code>AT</code></td>
<td><code>CTCGTCC</code></td>
<td></td>
</tr>
<tr>
<td><code>AT</code></td>
<td><code>TGTTATA</code></td>
<td>+1</td>
</tr>
<tr>
<td><code>AT</code></td>
<td><code>CTCGTCC</code></td>
<td></td>
</tr>
<tr>
<td><code>AT</code></td>
<td><code>TGTTATA</code></td>
<td>+1</td>
</tr>
<tr>
<td><code>AT</code></td>
<td><code>CTCGTCC</code></td>
<td></td>
</tr>
<tr>
<td><code>AT</code></td>
<td><code>TGTTATA</code></td>
<td>+1</td>
</tr>
<tr>
<td><code>AT</code></td>
<td><code>CTCGTCC</code></td>
<td></td>
</tr>
<tr>
<td><code>AT</code></td>
<td><code>TGTTATA</code></td>
<td>+1</td>
</tr>
<tr>
<td><code>AT</code></td>
<td><code>CTCGTCC</code></td>
<td></td>
</tr>
<tr>
<td><code>AT</code></td>
<td><code>TGTTATA</code></td>
<td>+1</td>
</tr>
<tr>
<td><code>AT</code></td>
<td><code>CTCGTCC</code></td>
<td></td>
</tr>
<tr>
<td><code>AT</code></td>
<td><code>TGTTATA</code></td>
<td>+1</td>
</tr>
<tr>
<td><code>AT</code></td>
<td><code>CTCGTCC</code></td>
<td></td>
</tr>
<tr>
<td><code>AT</code></td>
<td><code>TGTTATA</code></td>
<td>+1</td>
</tr>
<tr>
<td><code>AT</code></td>
<td><code>CTCGTCC</code></td>
<td></td>
</tr>
<tr>
<td><code>AT</code></td>
<td><code>TGTTATA</code></td>
<td>+1</td>
</tr>
<tr>
<td><code>AT</code></td>
<td><code>CTCGTCC</code></td>
<td></td>
</tr>
<tr>
<td><code>AT</code></td>
<td><code>TGTTATA</code></td>
<td>+1</td>
</tr>
<tr>
<td><code>AT</code></td>
<td><code>CTCGTCC</code></td>
<td></td>
</tr>
<tr>
<td><code>AT</code></td>
<td><code>TGTTATA</code></td>
<td>+1</td>
</tr>
<tr>
<td><code>AT</code></td>
<td><code>CTCGTCC</code></td>
<td></td>
</tr>
<tr>
<td><code>AT</code></td>
<td><code>TGTTATA</code></td>
<td>+1</td>
</tr>
<tr>
<td><code>AT</code></td>
<td><code>CTCGTCC</code></td>
<td></td>
</tr>
<tr>
<td><code>AT</code></td>
<td><code>TGTTATA</code></td>
<td>+1</td>
</tr>
<tr>
<td><code>AT</code></td>
<td><code>CTCGTCC</code></td>
<td></td>
</tr>
</tbody>
</table>

© 2022 Phillip Compeau
### Matching Symbols as a Game

<table>
<thead>
<tr>
<th>Growing alignment</th>
<th>Remaining symbols</th>
<th>Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>ATGTTATA</td>
<td>ATCGTCC</td>
<td></td>
</tr>
<tr>
<td>A</td>
<td>TGGTATATTA</td>
<td>+1</td>
</tr>
<tr>
<td>A</td>
<td>TCCTCC</td>
<td></td>
</tr>
<tr>
<td>AT</td>
<td>GTATATTA</td>
<td>+1</td>
</tr>
<tr>
<td>AT</td>
<td>CTTCC</td>
<td></td>
</tr>
<tr>
<td>AT−</td>
<td>GTATTA</td>
<td>+1</td>
</tr>
<tr>
<td>ATC</td>
<td>GTTCC</td>
<td></td>
</tr>
<tr>
<td>AT−G</td>
<td>TTTATA</td>
<td>+1</td>
</tr>
<tr>
<td>ATCG</td>
<td>TCC</td>
<td></td>
</tr>
<tr>
<td>AT−GT</td>
<td>TTATA</td>
<td>+1</td>
</tr>
<tr>
<td>ATCGT</td>
<td>TCC</td>
<td></td>
</tr>
<tr>
<td>AT−GTT</td>
<td>TATA</td>
<td>+1</td>
</tr>
<tr>
<td>ATCGT−</td>
<td>AT</td>
<td></td>
</tr>
<tr>
<td>AT−GTTA</td>
<td>TA</td>
<td></td>
</tr>
<tr>
<td>ATCGT−C</td>
<td>C</td>
<td></td>
</tr>
</tbody>
</table>

Figure 5.1: One way of playing the alignment game for the strings ATGTTATA and ATCGTCC, with score 4. At each step, we choose to remove either one or both symbols from the left of the two sequences in the “remaining symbols” column. If we remove both symbols, then we align them in the “growing alignment”. If we remove only one symbol, then we align this symbol with a space symbol in the growing alignment. Matched symbols are shown in red (and receive score 1). Mismatched symbols are shown in purple; symbols aligned against space symbols are shown in blue or green depending on which sequence they were removed from.

© 2022 Phillip Compeau
### Matching Symbols as a Game

#### FIGURE 5.1

One way of playing the alignment game for the strings `ATGTTATA` and `ATCGTCC`, with score 4. At each step, we choose to remove either one or both symbols from the left of the two sequences in the "remaining symbols" column. If we remove both symbols, then we align them in the "growing alignment". If we remove only one symbol, then we align this symbol with a space symbol in the growing alignment. Matched symbols are shown in red (and receive score 1). Mismatched symbols are shown in purple; symbols aligned against space symbols are shown in blue or green depending on which sequence they were removed from.

<table>
<thead>
<tr>
<th>Growing alignment</th>
<th>Remaining symbols</th>
<th>Score</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>ATGTTATA</code></td>
<td><code>ATCGTCC</code></td>
<td></td>
</tr>
<tr>
<td><code>ATGTTATA</code></td>
<td><code>ATCGTCC</code></td>
<td></td>
</tr>
<tr>
<td><code>AT</code></td>
<td><code>TGTATATA</code></td>
<td>+1</td>
</tr>
<tr>
<td><code>AT</code></td>
<td><code>TCGTCC</code></td>
<td></td>
</tr>
<tr>
<td><code>AT -</code></td>
<td><code>GTATATA</code></td>
<td></td>
</tr>
<tr>
<td><code>AT -</code></td>
<td><code>CTGTCC</code></td>
<td></td>
</tr>
<tr>
<td><code>AT - G</code></td>
<td><code>TTATATA</code></td>
<td>+1</td>
</tr>
<tr>
<td><code>AT - G</code></td>
<td><code>CTCC</code></td>
<td></td>
</tr>
<tr>
<td><code>AT - GT</code></td>
<td><code>TATATA</code></td>
<td>+1</td>
</tr>
<tr>
<td><code>AT - GT</code></td>
<td><code>CTCC</code></td>
<td></td>
</tr>
<tr>
<td><code>AT - GT T</code></td>
<td><code>ATATA</code></td>
<td>+1</td>
</tr>
<tr>
<td><code>AT - GT T</code></td>
<td><code>CTCC</code></td>
<td></td>
</tr>
<tr>
<td><code>AT - GT T A</code></td>
<td><code>ATA</code></td>
<td></td>
</tr>
<tr>
<td><code>AT - GT T A</code></td>
<td><code>CTCC</code></td>
<td></td>
</tr>
<tr>
<td><code>AT - GT T A T</code></td>
<td><code>ATA</code></td>
<td></td>
</tr>
<tr>
<td><code>AT - GT T A T</code></td>
<td><code>CTCC</code></td>
<td></td>
</tr>
<tr>
<td><code>AT - GT T A T A</code></td>
<td></td>
<td>+1</td>
</tr>
<tr>
<td><code>AT - GT T A T A</code></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

© 2022 Phillip Compeau
## Matching Symbols as a Game

<table>
<thead>
<tr>
<th>Growing alignment</th>
<th>Remaining symbols</th>
<th>Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>ATGTTATA</td>
<td>ATCGTCC</td>
<td></td>
</tr>
<tr>
<td>A</td>
<td>GTTTATA</td>
<td>+1</td>
</tr>
<tr>
<td>A</td>
<td>TCGTCC</td>
<td></td>
</tr>
<tr>
<td>AT</td>
<td>GTTTATA</td>
<td>+1</td>
</tr>
<tr>
<td>AT</td>
<td>CGTCC</td>
<td></td>
</tr>
<tr>
<td>AT-</td>
<td>GTTTATA</td>
<td>+1</td>
</tr>
<tr>
<td>ATC</td>
<td>GTCC</td>
<td></td>
</tr>
<tr>
<td>AT-</td>
<td>GTTATA</td>
<td>+1</td>
</tr>
<tr>
<td>ATC</td>
<td>GTCC</td>
<td></td>
</tr>
<tr>
<td>AT-</td>
<td>GTTATA</td>
<td>+1</td>
</tr>
<tr>
<td>ATC</td>
<td>GTCC</td>
<td></td>
</tr>
<tr>
<td>AT-</td>
<td>GTTATA</td>
<td>+1</td>
</tr>
<tr>
<td>ATC</td>
<td>GTCC</td>
<td></td>
</tr>
<tr>
<td>AT-</td>
<td>GTTATA</td>
<td>+1</td>
</tr>
<tr>
<td>ATC</td>
<td>GTCC</td>
<td></td>
</tr>
<tr>
<td>AT-</td>
<td>GTTATA</td>
<td>+1</td>
</tr>
<tr>
<td>ATC</td>
<td>GTCC</td>
<td></td>
</tr>
</tbody>
</table>

**FIGURE 5.1** One way of playing the alignment game for the strings ATGTTATA and ATCGTCC, with score 4. At each step, we choose to remove either one or both symbols from the left of the two sequences in the “remaining symbols” column. If we remove both symbols, then we align them in the “growing alignment”. If we remove only one symbol, then we align this symbol with a space symbol in the growing alignment. Matched symbols are shown in red (and receive score 1). Mismatched symbols are shown in purple; symbols aligned against space symbols are shown in blue or green depending on which sequence they were removed from.
Given two strings $v$ and $w$, an alignment of $v$ and $w$ is a two-row matrix such that:

- the first row contains symbols of $v$
- the second row contains symbols of $w$
- each row may also contain gap symbols ("-")
- no column has two gap symbols

\[
\begin{array}{c}
AT\ T\ T\ T\ A\ T\ A \\
ATCGT\ T\ C\ G\ T\ T\ \ \ \ C\ \ \ \ C
\end{array}
\]
Given two strings \( v \) and \( w \), an **alignment** of \( v \) and \( w \) is a two-row matrix such that:

- the first row contains symbols of \( v \)
- the second row contains symbols of \( w \)
- each row may also contain **gap symbols** ("-")
- no column has two gap symbols

**Matches**
From a Game to a Definition

Given two strings $v$ and $w$, an alignment of $v$ and $w$ is a two-row matrix such that:

- the first row contains symbols of $v$
- the second row contains symbols of $w$
- each row may also contain gap symbols ("-")
- no column has two gap symbols
Given two strings $v$ and $w$, an alignment of $v$ and $w$ is a two-row matrix such that:

- the first row contains symbols of $v$
- the second row contains symbols of $w$
- each row may also contain gap symbols ("-")
- no column has two gap symbols

AT − GTTATA
AT C GT − C − C

Insertions

© 2022 Phillip Compeau
Given two strings \( v \) and \( w \), an **alignment** of \( v \) and \( w \) is a two-row matrix such that:

- the first row contains symbols of \( v \)
- the second row contains symbols of \( w \)
- each row may also contain **gap symbols** ("-")
- no column has two gap symbols
A common subsequence of $v$ and $w$ is a sequence of symbols occurring (not necessarily contiguously) in both $v$ and $w$. 
A common subsequence of $v$ and $w$ is a sequence of symbols occurring (not necessarily contiguously) in both $v$ and $w$.

The matches in an alignment of $v$ and $w$ form a common subsequence of $v$ and $w$.
The Problems are the Same!

Longest Common Subsequence Length Problem:
• **Input:** Two strings.
• **Output:** The length of a longest common subsequence of these strings.

Symbol Matching Problem:
• **Input:** Two strings.
• **Output:** The greatest number of matched symbols in any “alignment” of the two strings.
THE MANHATTAN TOURIST PROBLEM
Manhattan Tourist Problem

STOP: How can we see the most sites if we move from 59th and 8th to 42nd and 3rd, moving south or east at each step? (And what algorithm did you use?)
Manhattan Tourist as a Network

FIGURE 5.2 (Left) A simplification of Midtown Manhattan. You start at the intersection of 59th Street and 8th Avenue in the northwest corner and end at the intersection of 42nd Street and 3rd Avenue in the southeast corner, traveling only south (•) or east (!) between intersections. The attractions shown are: Carnegie Hall (1), Tiffany & Co. (2), the Sony Building (3), the Museum of Modern Art (4), the Four Seasons Hotel (5), St. Patrick’s Cathedral (6), the General Electric Building (7), Radio City Music Hall (8), Rockefeller Center (9), the Paramount Building (10), the New York Times Building (11), Times Square (12), the General Society of Mechanics and Tradesmen (13), Grand Central Terminal (14), and the Chrysler Building (15). (Right) The directed graph \( \text{ManhattanGraph} \) in which every edge is weighted by the number of attractions along that city block (edge weights equal to 0 are not shown).

Weight of edge: number of attractions along the edge.
Manhattan Tourist as a Network

CHAPTER 5

FIGURE 5.2 (Left) A simplification of Midtown Manhattan. You start at the intersection of 59th Street and 8th Avenue in the northwest corner and end at the intersection of 42nd Street and 3rd Avenue in the southeast corner, traveling only south (\#) or east (!) between intersections. The attractions shown are: Carnegie Hall (1), Tiffany & Co. (2), the Sony Building (3), the Museum of Modern Art (4), the Four Seasons Hotel (5), St. Patrick’s Cathedral (6), the General Electric Building (7), Radio City Music Hall (8), Rockefeller Center (9), the Paramount Building (10), the New York Times Building (11), Times Square (12), the General Society of Mechanics and Tradesmen (13), Grand Central Terminal (14), and the Chrysler Building (15). (Right) The directed graph \text{ManhattanGraph} in which every edge is weighted by the number of attractions along that city block (edge weights equal to 0 are not shown).

Weight of edge: number of attractions along the edge.

Goal: Find a longest path from source (top left) to sink (bottom right).
Manhattan Tourist Problem:

- **Input:** A weighted $n \times m$ rectangular grid ($n + 1$ rows and $m + 1$ columns).
- **Output:** A longest path from source $(0, 0)$ to sink $(n, m)$ in the grid.
Exercise: What is the longest path in this city? What algorithm did you use?
A “Greedy” Manhattan Algorithm Taking the Best Choice in Each Node

STOP: Does the greedy algorithm solve the problem?
A “Greedy” Manhattan Algorithm Taking the Best Choice in Each Node

Answer: No! Much like with genome assembly, we need a more clever approach.
Manhattan Tourist as a Network Problem

Longest Path in a Directed Graph:
• **Input:** An edge-weighted directed graph with source and sink nodes.
• **Output:** A longest path from source to sink in the graph.
Manhattan Tourist as a Network Problem

STOP: What is the longest path in this graph?
Manhattan Tourist as a Network Problem

Answer: Cycles in graphs cause infinite paths ...
Directed acyclic graph (DAG): A directed graph that contains no cycles.
Directed acyclic graph (DAG): A directed graph that contains no cycles.

Longest Path in a DAG Problem:
- **Input:** An edge-weighted DAG with source and sink nodes.
- **Output:** A longest path from source to sink in the DAG.
<table>
<thead>
<tr>
<th>Directed acyclic graph (DAG): A directed graph that contains no cycles.</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th><strong>Longest Path in a DAG Problem:</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>• <strong>Input:</strong> An edge-weighted DAG with source and sink nodes.</td>
</tr>
<tr>
<td>• <strong>Output:</strong> A longest path from source to sink in the DAG.</td>
</tr>
</tbody>
</table>

... but what does finding a longest path in a DAG have to do with sequence comparison?
SEQUENCE ALIGNMENT AS A PATH IN A NETWORK
Returning to Sequence Alignment ...

\[
\begin{array}{ccccccccc}
A & T & - & G & T & T & T & A & T & A \\
A & T & C & G & T & - & C & - & C & C
\end{array}
\]
Returning to Sequence Alignment ...

FIGURE 5.5 An alignment of \textit{ATGTTATA} and \textit{ATCGTCC}.

The array in the first row counts the number of symbols of \textit{ATGTTATA} used up to a given position. The array in the fourth row counts the number of symbols of \textit{ATCGTCC} used up to a given position. And the array in the last row records whether each column of the alignment represents a match/mismatch (\&/\&), insertion (!), or deletion (#).

Note that in addition to horizontal and vertical edges, we have added diagonal edges connecting \((i, j)\) to \((i+1, j+1)\) in Figure 5.6.

FIGURE 5.6 Every alignment corresponds to a path in the alignment graph from source to sink, and vice-versa. (Left) The path \((0, 0)\) \& \((1, 1)\) \& \((2, 2)\) \& \((2, 3)\) \& \((3, 4)\) \& \((4, 5)\) \& \((5, 5)\) \& \((6, 6)\) \& \((7, 6)\) \& \((8, 7)\) is highlighted above and corresponds to the alignment of \textit{ATGTTATA} and \textit{ATCGTCC} in Figure 5.5. (Right) Another path in the alignment graph.

We call the DAG in Figure 5.6 the \textbf{alignment graph} \textit{ALIGNMENT GRAPH} \((v, w)\), and we call a path from source to sink in this DAG an \textbf{alignment path}. Every alignment of \textit{v} and \textit{w} can be viewed as a set of instructions to construct a unique alignment path in \textit{ALIGNMENT GRAPH} \((v, w)\), where each match/mismatch, insertion, and deletion corresponds to an edge \&/\&, !, and #, respectively. For an introduction to the alignment graph, see Compeau and Skiena (2011).
Returning to Sequence Alignment ...

**Figure 5.5** An alignment of `ATGTTATA` and `ATCGTCC`. The array in the first row counts the number of symbols of `ATGTTATA` used up to a given position. The array in the fourth row counts the number of symbols of `ATCGTCC` used up to a given position. And the array in the last row records whether each column of the alignment represents a match/mismatch (&/&), insertion (!), or deletion (#).

Note that in addition to horizontal and vertical edges, we have added diagonal edges connecting `(i, j)` to `(i+1, j+1)` in Figure 5.6.

**Figure 5.6** Every alignment corresponds to a path in the alignment graph from source to sink, and vice-versa. (Left) The path `(0, 0) ↓ (1, 1) ↓ (2, 2) → (2, 3) ↓ (3, 4) ↓ (4, 5) ↓ (5, 5) ↓ (6, 6) ↓ (7, 6) ↓ (8, 7)` is highlighted above and corresponds to the alignment of `ATGTTATA` and `ATCGTCC` in Figure 5.5. (Right) Another path in the alignment graph.

We call the DAG in Figure 5.6 the **alignment graph** of strings `v` and `w`, denoted `ALIGNMENT GRAPH(v, w)`, and we call a path from source to sink in this DAG an **alignment path**. Every alignment of `v` and `w` can be viewed as a set of instructions to construct a unique alignment path in `ALIGNMENT GRAPH(v, w)`, where each match/mismatch, insertion, and deletion corresponds to an edge &/&, !, and #, respectively.
Returning to Sequence Alignment ... 

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>T</td>
<td>-</td>
<td>G</td>
<td>T</td>
<td>T</td>
<td>A</td>
<td>T</td>
<td>A</td>
<td></td>
<td></td>
</tr>
<tr>
<td>A</td>
<td>T</td>
<td>C</td>
<td>G</td>
<td>T</td>
<td>-</td>
<td>C</td>
<td>-</td>
<td>C</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

0 1 2 3 4 5 5 6 6 7

(0, 0) ↘ (1, 1) ↘ (2, 2) → (2, 3) ↘ (3, 4)
↘ (4, 5) ↓ (5, 5) ↘ (6, 6) ↓ (7, 6) ↘ (8, 7)

This is a path in a 2-D network!
Representing an Alignment as a Path in a Manhattan-like DAG

This network is called the alignment network of the strings ATGTTATA and ATCGTCC.
We can also construct an alignment from a path

Exercise: What alignment does this path correspond to?
We can also construct an alignment from a path

Exercise: What alignment does this path correspond to?

Answer:

A T G T T A - T - - A
- - A T - C G T C C -
Solving the Symbol Matching Problem

Symbol Matching Problem:
- **Input:** Two strings.
- **Output:** The greatest number of matched symbols in any alignment of the two strings.

STOP: How can we use the alignment network to solve this problem?
Counting Matches Only

**Answer:** If we weight the red edges as 1 and the other edges as 0, then a maximum-weight path from source to sink solves the Symbol Matching Problem!
Counting Matches Only

How do we compare DNA sequences?

Moreover, this process is reversible, as we can convert each alignment path into a unique alignment.

Exercise Break:

Construct the alignment of ATGTTATA and ATCGTCC corresponding to the alignment path in Figure 5.6 (right).

Stop and Think:

Can you use the alignment graph to find a longest common subsequence of two strings?

Recall that finding a longest common subsequence of two strings is equivalent to finding an alignment of these strings maximizing the number of matches. In Figure 5.7, we highlight all diagonal edges of ALIGNMENT GRAPH (ATGTTATA, ATCGTCC) corresponding to matches. If we assign a weight of 1 to all these edges and 0 to all other edges, then the Longest Common Subsequence Problem is equivalent to finding a longest path in this weighted DAG!

Answer: If we weight the red edges as 1 and the other edges as 0, then a maximum-weight path from source to sink solves the Symbol Matching Problem!

But we haven’t said how to find the maximum length of a path.

© 2022 by Phillip Compeau
THE CHANGE PROBLEM: AN INTRO TO DYNAMIC PROGRAMMING
Different coin values are called denominations.

**USA Denominations**
(100, 50, 25, 10, 5, 1)

**Roman Republic Denominations**
(120, 40, 30, 24, 20, 10, 5, 4, 1)

**Minimum Change Problem:**
- **Input:** An integer money and a collection of denominations coins.
- **Output:** A minimum number of coins with these denominations that add to money.
Making Change

Different coin values are called **denominations**.

**USA Denominations**
(100, 50, 25, 10, 5, 1)

**Roman Republic Denominations**
(120, 40, 30, 24, 20, 10, 5, 4, 1)

**STOP:** What algorithm would you propose to provide as few coins as possible when making change for a given set of denominations?
Making Change Greedily

**GreedyChange**(money, coins)
1. Choose the largest coin denomination \(coin\) in \(coins\) that is less than or equal to \(money\).
2. Subtract \(coin\) from \(money\).
3. If \(money\) is equal to zero, STOP. Otherwise, return to step 1.
Making Change Greedily

**GreedyChange**(money, coins)
1. Choose the largest coin denomination coin in coins that is less than or equal to money.
2. Subtract coin from money.
3. If money is equal to zero, STOP. Otherwise, return to step 1.

STOP: Does **GreedyChange** always make change with the minimum number of coins?
Greedy Change is Suboptimal!

Roman Republic Denominations
(120, 40, 30, 24, 20, 10, 5, 4, 1)

When changing 48 Roman denarii, GreedyChange would suggest five coins (48 = 40 + 5 + 1 + 1 + 1).
Greedy Change is Suboptimal!

Roman Republic Denominations
(120, 40, 30, 24, 20, 10, 5, 4, 1)

When changing 48 Roman denarii, GreedyChange would suggest five coins (48 = 40 + 5 + 1 + 1 + 1).

But we can make change with just two coins (48 = 24 + 24)!
Let MinNumCoins($money$) denote the minimum number of coins needed to change an amount $money$ for a collection of denominations $Coins$.

**Minimum Change Problem:**

- **Input:** An integer $money$ and a collection of denominations $coins$.
- **Output:** A minimum number of coins with these denominations that add to $money$. 
Say that you need to change 76 denarii, and you only have three denominations: Coins = (5, 4, 1).

STOP: Is there anything that you can do to simplify the computation of MinNumCoins(76)?
Key Insight: split problem into simpler cases

Say that you need to change 76 denarii, and you only have three denominations: Coins = (5, 4, 1).

**Answer:** A minimum collection of coins totaling 76 denarii must be one of the following:
- a minimal collection of coins totaling 75 denarii, plus a 1-denarius coin;
- a minimal collection of coins totaling 72 denarii, plus a 4-denarius coin;
- a minimal collection of coins totaling 71 denarii, plus a 5-denarius coin;

© 2022 Phillip Compeau
For an arbitrary set of denominations \( \text{Coins} \), we obtain the following recurrence relation:

\[
\text{MinNumCoins}(\text{money}) = \min \left\{ \begin{array}{l}
\text{MinNumCoins}(\text{money} - \text{coin}_1) + 1 \\
\vdots \\
\text{MinNumCoins}(\text{money} - \text{coin}_d) + 1
\end{array} \right. 
\]
For an arbitrary set of denominations $\textit{Coins}$, we obtain the following recurrence relation.

$$\text{MIN\textsc{Num\textit{Coins}}}(\textit{money}) = \min \left\{ \begin{array}{l}
\text{MIN\textsc{Num\textit{Coins}}}(\textit{money} - \textit{coin}_1) + 1 \\
\vdots \\
\text{MIN\textsc{Num\textit{Coins}}}(\textit{money} - \textit{coin}_d) + 1
\end{array} \right\}$$

**Recurrence relation:** An expression for a function $f(x)$ in terms of values of $f(y)$ where $y < x$. 

© 2022 Phillip Compeau
RecursiveChange\((money, coins)\)

if \(money = 0\)
    return 0

\(minNumCoins \leftarrow \infty\)

for \(i \leftarrow 1\) to \(|coins|\)
    if \(money \geq coins[i]\)
        \(numCoins \leftarrow \text{RecursiveChange}(money - coins[i], coins)\)
        if \(numCoins + 1 < minNumCoins\)
            \(minNumCoins \leftarrow numCoins + 1\)

return \(minNumCoins\)

**Recursive algorithm:** An algorithm that calls itself.
Recurrence Relations Imply Recursive Algorithms

**RecursiveChange**(*money, coins*)

if *money* = 0
  return 0

*minNumCoins* ← ∞

for *i* ← 1 to |*coins*|
  if *money* ≥ *coins*[*i*]
    *numCoins* ← **RecursiveChange**(*money* − *coins*[*i*], *coins*)
    if *numCoins* + 1 < *minNumCoins*
      *minNumCoins* ← *numCoins* + 1

return *minNumCoins*

**Recursive algorithm:** An algorithm that calls itself.

**STOP:** Is **RecursiveChange** a good algorithm? Why or why not?
Recursive Calls Can Grow Exponentially

When money = 76 and Coins = (5,4,1), RecursiveChange(70, Coins) is computed six times.
Recursive Calls Can Grow Exponentially

STOP: How many times do you think $\text{RecursiveChange}(30, \text{Coins})$ will be computed?
Recursive Calls Can Grow Exponentially

Answer: Billions! 😞
Computing Values “Bottom-Up” Avoids Many Recursive Calls

**Broke: RecursiveChange** causes exponential branching because of repeated recursive calls.
Computing Values “Bottom-Up” Avoids Many Recursive Calls

**Broke:** `RecursiveChange` causes exponential branching because of repeated recursive calls.

**Woke:** First compute `MinNumCoins(money)` for small values of `money` and work our way upward.

$$\begin{array}{cccccccccccc}
0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 & 11 & 12 \\
0 & 1 & 2 & 3 & 1 & 1 & & & & & & & \\
\end{array}$$

$\textit{money}$

$\textit{MinNumCoins}$

$$coins = (1, 4, 5)$$
Computing Values “Bottom-Up” Avoids Many Recursive Calls

\[ \text{MinNumCoins}(\text{money}) = \min \left\{ \begin{align*}
\text{MinNumCoins}(\text{money} - \text{coin}_1) + 1 \\
\vdots \\
\text{MinNumCoins}(\text{money} - \text{coin}_d) + 1
\end{align*} \right\} \]

**Woke:** First compute MinNumCoins(money) for small values of money and work our way upward.

\[ \text{coins} = (1, 4, 5) \]
Computing Values “Bottom-Up” Avoids Many Recursive Calls

\[ \text{MinNumCoins}(\text{money}) = \min \left\{ \text{MinNumCoins}(\text{money} - \text{coin}_1) + 1, \ldots, \text{MinNumCoins}(\text{money} - \text{coin}_d) + 1 \right\} \]

Woke: First compute MinNumCoins(money) for small values of money and work our way upward.

<table>
<thead>
<tr>
<th>coins = (1, 4, 5)</th>
<th>money</th>
</tr>
</thead>
<tbody>
<tr>
<td>MinNumCoins</td>
<td></td>
</tr>
<tr>
<td>0 1 2 3 4 5 6 7 8 9 10 11 12</td>
<td>0 1 2</td>
</tr>
</tbody>
</table>

© 2022 Phillip Compeau
Computing Values “Bottom-Up” Avoids Many Recursive Calls

\[ \text{MinNumCoins}(\text{money}) = \min \begin{cases} \text{MinNumCoins}(\text{money} - \text{coin}_1) + 1 \\ \vdots \\ \text{MinNumCoins}(\text{money} - \text{coin}_d) + 1 \end{cases} \]

**Woke:** First compute \( \text{MinNumCoins}(\text{money}) \) for small values of \( \text{money} \) and work our way upward.

\[ \begin{array}{cccccccccccc} \hline \text{money} & 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 & 11 & 12 \\ \hline \text{MinNumCoins} & 0 & 1 & 2 & 3 & 1 & 1 & 2 & 3 & \text{ } & \text{ } & \text{ } & \text{ } & \text{ } \\ \hline \end{array} \]

\( \text{coins} = (1, 4, 5) \)
Computing Values “Bottom-Up” Avoids Many Recursive Calls

\[
\text{MinNumCoins}(\text{money}) = \min \left\{ \begin{array}{l}
\text{MinNumCoins}(\text{money} - \text{coin}_1) + 1 \\
\vdots \\
\text{MinNumCoins}(\text{money} - \text{coin}_d) + 1
\end{array} \right. 
\]

**Woke:** First compute MinNumCoins(money) for small values of money and work our way upward.

\[\text{coins} = (1, 4, 5)\]
Computing Values “Bottom-Up” Avoids Many Recursive Calls

\[
\text{MinNumCoins}(money) = \min \left\{ \begin{array}{c}
\text{MinNumCoins}(money - coin_1) + 1 \\
\vdots \\
\text{MinNumCoins}(money - coin_d) + 1
\end{array} \right. 
\]

Woke: First compute MinNumCoins(money) for small values of money and work our way upward.

\begin{tabular}{|c|c|c|c|c|c|c|c|c|c|c|c|c|}
\hline
0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 & 11 & 12 \\
\hline
0 & 1 & 2 & 3 & 1 & 1 & 2 & 3 & 2 & 2 & & & \\
\hline
\end{tabular}

coins = (1, 4, 5)

\copyright 2022 Phillip Compeau
Computing Values “Bottom-Up” Avoids Many Recursive Calls

\[ \text{MinNumCoins}(\text{money}) = \min \left\{ \text{MinNumCoins}(\text{money} - \text{coin}_1) + 1, \ldots, \text{MinNumCoins}(\text{money} - \text{coin}_d) + 1 \right\} \]

**Woke:** First compute MinNumCoins(money) for small values of money and work our way upward.

<table>
<thead>
<tr>
<th>money</th>
<th>MinNumCoins</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
</tr>
<tr>
<td>4</td>
<td>2</td>
</tr>
<tr>
<td>5</td>
<td>2</td>
</tr>
<tr>
<td>6</td>
<td>2</td>
</tr>
<tr>
<td>7</td>
<td>3</td>
</tr>
<tr>
<td>8</td>
<td>2</td>
</tr>
<tr>
<td>9</td>
<td>2</td>
</tr>
<tr>
<td>10</td>
<td>2</td>
</tr>
<tr>
<td>11</td>
<td>2</td>
</tr>
<tr>
<td>12</td>
<td>2</td>
</tr>
</tbody>
</table>

money = (1, 4, 5)
Computing Values “Bottom-Up” Avoids Many Recursive Calls

\[
\text{MinNumCoins}(money) = \min \begin{cases} 
\text{MinNumCoins}(money - coin_1) + 1 \\
\vspace{10pt}
\vdots \\
\text{MinNumCoins}(money - coin_d) + 1 
\end{cases}
\]

**Woke:** First compute \( \text{MinNumCoins}(money) \) for small values of \( money \) and work our way upward.

<table>
<thead>
<tr>
<th>money</th>
<th>MinNumCoins</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>3</td>
<td>2</td>
</tr>
<tr>
<td>4</td>
<td>3</td>
</tr>
<tr>
<td>5</td>
<td>2</td>
</tr>
<tr>
<td>6</td>
<td>3</td>
</tr>
<tr>
<td>7</td>
<td>2</td>
</tr>
<tr>
<td>8</td>
<td>2</td>
</tr>
<tr>
<td>9</td>
<td>2</td>
</tr>
<tr>
<td>10</td>
<td>2</td>
</tr>
<tr>
<td>11</td>
<td>2</td>
</tr>
<tr>
<td>12</td>
<td>2</td>
</tr>
</tbody>
</table>

\( \text{coins} = (1, 4, 5) \)
Computing Values “Bottom-Up” Avoids Many Recursive Calls

\[
\text{MinNumCoins}(\text{money}) = \min \left\{ \begin{align*}
\text{MinNumCoins}(\text{money} - \text{coin}_1) + 1 \\
\vdots \\
\text{MinNumCoins}(\text{money} - \text{coin}_d) + 1
\end{align*} \right\}
\]

**Woke:** First compute MinNumCoins(money) for small values of money and work our way upward.

<table>
<thead>
<tr>
<th>money</th>
<th>MinNumCoins</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>4</td>
<td>1</td>
</tr>
<tr>
<td>5</td>
<td>1</td>
</tr>
<tr>
<td>6</td>
<td>2</td>
</tr>
<tr>
<td>7</td>
<td>3</td>
</tr>
<tr>
<td>8</td>
<td>2</td>
</tr>
<tr>
<td>9</td>
<td>2</td>
</tr>
<tr>
<td>10</td>
<td>3</td>
</tr>
<tr>
<td>11</td>
<td>3</td>
</tr>
<tr>
<td>12</td>
<td></td>
</tr>
</tbody>
</table>

\(\text{coins} = (1, 4, 5)\)
Computing Values “Bottom-Up” Avoids Many Recursive Calls

\[
\text{MinNumCoins}(\text{money}) = \min \left\{ \begin{array}{l}
\text{MinNumCoins}(\text{money} - \text{coin}_1) + 1 \\
\vdots \\
\text{MinNumCoins}(\text{money} - \text{coin}_d) + 1
\end{array} \right. 
\]

Dynamic Programming: Using a recurrence relation to fill in and “remember” values in a table.

<table>
<thead>
<tr>
<th>money</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>11</th>
<th>12</th>
</tr>
</thead>
<tbody>
<tr>
<td>MinNumCoins</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>3</td>
<td>3</td>
</tr>
</tbody>
</table>

\[\text{coins} = (1, 4, 5)\]
Origins of Dynamic Programming?

STOP: Wait … why would such a simple idea be called “dynamic programming”?

Dynamic Programming: Using a recurrence relation to fill in and “remember” values in a table.
Richard Bellman, a Wise Man

“We had a very interesting gentleman in Washington named Wilson. He was Secretary of Defense, and he actually had a pathological fear and hatred of the word "research". I’m not using the term lightly; I’m using it precisely. His face would suffuse, he would turn red, and he would get violent if people used the term research in his presence. You can imagine how he felt, then, about the term mathematical. The RAND Corporation was employed by the Air Force, and the Air Force had Wilson as its boss, essentially. Hence, I felt I had to do something to shield Wilson and the Air Force from the fact that I was really doing mathematics inside the RAND Corporation. What title, what name, could I choose? In the first place I was interested in planning, in decision making, in thinking. But planning, is not a good word for various reasons. I decided therefore to use the word "programming". I wanted to get across the idea that this was dynamic, this was multistage, this was time-varying. I thought, let's kill two birds with one stone. Let's take a word that has an absolutely precise meaning, namely dynamic, in the classical physical sense. It also has a very interesting property as an adjective, and that is it's impossible to use the word dynamic in a pejorative sense. Try thinking of some combination that will possibly give it a pejorative meaning. It's impossible. Thus, I thought dynamic programming was a good name. It was something not even a Congressman could object to. So I used it as an umbrella for my activities.”
FINDING THE LENGTH OF A LONGEST PATH IN A DAG
Returning to Manhattan

**Manhattan Tourist Problem:**

- **Input:** A weighted $n \times m$ rectangular grid ($n + 1$ rows and $m + 1$ columns).
- **Output:** A longest path from source $(0, 0)$ to sink $(n, m)$ in the grid.

**Exercise:** Find a recurrence relation for the length of a longest path from $(0, 0)$ to node $(i, j)$, which we will call $\text{length}(i,j)$. 
Returning to Manhattan

Column $j – 1$  Column $j$

Row $i – 1$

Row $i$

(weight of vertical edge into $i$,$j$)

(weight of horizontal edge)

**Answer:** $length(i,j) = \max\{$

$length(i-1,j) + \text{weight(vertical edge into } i,j\text{)},$

$length(i, j-1) + \text{weight(horizontal edge into } i,j\text{)}\}$. 

© 2022 Phillip Compeau
Returning to Manhattan

**Manhattan Tourist Problem:**

- **Input:** A weighted $n \times m$ rectangular grid ($n + 1$ rows and $m + 1$ columns).
- **Output:** A longest path from source (0, 0) to sink $(n, m)$ in the grid.

**STOP:** Will a recursive algorithm for Manhattan Tourist have the same problem that the recursive change-making function encountered?
Returning to Manhattan

Manhattan Tourist Problem:
- **Input:** A weighted $n \times m$ rectangular grid ($n + 1$ rows and $m + 1$ columns).
- **Output:** A longest path from source (0, 0) to sink ($n$, $m$) in the grid.

**Answer:** Yes! Because the same $\text{length}(i, j)$ can get re-computed many times…
Let’s Use Dynamic Programming Instead

Recurrence relation

\[
\text{length}(i, j) = \max\{\text{length}(i-1, j) + \text{down}(i, j), \\
\text{length}(i, j-1) + \text{right}(i, j)\}
\]
Let's Use Dynamic Programming Instead

STOP: Which element of the table should we fill in next and what should its value be?

Recurrence relation

\[ \text{length}(i, j) = \max\{\text{length}(i-1, j) + \text{down}(i, j), \text{length}(i, j-1) + \text{right}(i, j)\} \]
Let’s Use Dynamic Programming Instead

**Answer:** We only know the values of MaxWeight for the two nodes adjacent to the node \((1, 1)\); it gets the value \(\max(3+0, 1+3) = 4\).

Recurrence relation:

\[
\text{length}(i, j) = \max\{\text{length}(i-1, j) + \text{down}(i, j), \text{length}(i, j-1) + \text{right}(i, j)\}
\]
Let’s Use Dynamic Programming Instead

STOP: Which elements should we fill in next and what should their values be?

Recurrence relation

\[ \text{length}(i, j) = \max\{\text{length}(i-1, j) + \text{down}(i, j), \text{length}(i, j-1) + \text{right}(i, j)\} \]
Let’s Use Dynamic Programming Instead

**Answer:** We can fill in all of row 1 or all of column 1 (it doesn’t matter which).

Recurrence relation

\[
\text{MaxWeight}(i, j) = \max\{\text{MaxWeight}(i-1, j) + \text{down}(i, j), \text{MaxWeight}(i, j-1) + \text{right}(i, j)\}
\]
Let’s Use Dynamic Programming Instead

Exercise: Fill in the remaining values of length for this network.

Recurrence relation

\[ length(i, j) = \max\{length(i-1, j) + down(i, j), length(i, j-1) + right(i, j)\} \]
Let’s Use Dynamic Programming Instead

STOP: Now do you see a longest path in this grid? How might we find one in general?

Recurrence relation

\[ \text{length}(i, j) = \max\{\text{length}(i-1, j) + \text{down}(i, j), \text{length}(i, j-1) + \text{right}(i, j)\} \]
Finding an LCS

Exercise: What is the recurrence relation for finding a longest common subsequence?
Our Recurrence Has Two Cases

**Case 1**

\[
\begin{align*}
\text{Row } i - 1 & \quad \text{Column } j - 1 \quad \text{Column } j \\
\text{Row } i & \quad \quad \quad 0 \\
\end{align*}
\]

\[\text{length}(i, j) = \text{maximum of:} \]
\[\quad \text{length}(i - 1, j) + 0 \]
\[\quad \text{length}(i, j - 1) + 0 \]
\[\quad \text{length}(i - 1, j - 1) + 0 \]

**Case 2**

\[
\begin{align*}
\text{Row } i - 1 & \quad \text{Column } j - 1 \quad \text{Column } j \\
\text{Row } i & \quad \quad \quad 1 \\
\end{align*}
\]

\[\text{length}(i, j) = \text{maximum of:} \]
\[\quad \text{length}(i - 1, j) + 0 \]
\[\quad \text{length}(i, j - 1) + 0 \]
\[\quad \text{length}(i - 1, j - 1) + 1 \]

© 2022 Phillip Compeau
Our Recurrence Has Two Cases

**Case 1**

<table>
<thead>
<tr>
<th>Row $i - 1$</th>
<th>Column $j - 1$</th>
<th>Column $j$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

**Case 2**

<table>
<thead>
<tr>
<th>Row $i - 1$</th>
<th>Column $j - 1$</th>
<th>Column $j$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

**STOP:** when will the diagonal edge weight be equal to 1?
Counting Matches Only

Answer: A diagonal edge connecting \((i - 1, j - 1)\) to \((i, j)\) is 1 when the corresponding symbols \(v[i - 1]\) and \(w[i - 1]\) of the two strings match.
A Recurrence for an Arbitrary DAG?

Longest Path in a DAG

Problem:

• **Input:** An edge-weighted DAG with source and sink nodes.
• **Output:** A longest path from source to sink in the DAG.

How do we compare DNA sequences?

A similar approach can be developed to find the longest path in any DAG, as suggested by the next exercise.

Exercise Break:

What is the length of a longest path between the blue and red nodes in the DAG shown in Figure 5.12?

**Figure 5.12**
A weighted DAG without the obvious order inherent in the graphs previously encountered in this chapter.

Topological orderings

Do not worry if you struggled to solve the last exercise. The hitch to using dynamic programming in order to find the length of a longest path in a DAG is that we must decide on the order in which to visit nodes when computing the values according to the recurrence:

$$s_b = \max_{\text{all predecessors } a \text{ of node } b} \{s_a + \text{weight of edge from } a \text{ to } b\}.$$  

This ordering of nodes is important, since by the time we reach node $b$, the values $s_a$ for all its predecessors must have already been computed. We have managed to hide this issue for rectangular grids because the order in which we have computed the $s_i, j$ ensured that we would never consider a node before visiting all of its predecessors.

To illustrate the importance of visiting nodes in the correct order, consider the DAG in Figure 5.13, which corresponds to a “Dressing Challenge Problem”. How would you order the nodes of this graph so that you don’t put on your boots before your tights?
A Recurrence for an Arbitrary DAG?

Exercise: Try finding a longest path from source to sink in this DAG. Can you find a recurrence relation for an arbitrary DAG?
A Recurrence for an Arbitrary DAG?

Let $s(b)$ be the length of a longest path from source to sink.

If there is an edge connecting $a$ to $b$, we call $a$ a predecessor of $b$. 

Exercise Break: What is the length of a longest path between the blue and red nodes in the DAG shown in Figure 5.12?
Let $s(b)$ be the length of a longest path from source to sink.

If there is an edge connecting $a$ to $b$, we call $a$ a predecessor of $b$.

$$s_b = \max_{\text{all predecessors } a \text{ of node } b} \{ s_a + \text{weight of edge from } a \text{ to } b \}$$
STOP: What makes computing this recurrence difficult?

\[ s_b = \max_{\text{all predecessors } a \text{ of node } b} \{ s_a + \text{weight of edge from } a \text{ to } b \} \]
A Recurrence for an Arbitrary DAG?

STOP: What makes computing this recurrence difficult?

**Answer:** We need to know the order to consider the nodes.

\[
s_b = \max_{\text{all predecessors } a \text{ of node } b} \{ s_a + \text{weight of edge from } a \text{ to } b \}
\]
“Dressing Challenge”: Ordering Nodes in a DAG
Topological Orderings

The critical part of computing \( s(b) \) is ensuring that \( s(a) \) has already been computed for all predecessors.

That is, we need to have an ordering of the nodes in a DAG so that no node is considered before its predecessor.
The critical part of computing $s(b)$ is ensuring that $s(a)$ has already been computed for all predecessors.

That is, we need to have an ordering of the nodes in a DAG so that no node is considered before its predecessor.

An ordering of nodes $(a_1, ..., a_k)$ of nodes in a DAG is a topological ordering if every edge $a_i \rightarrow a_j$ is such that $i < j$. 
The critical part of computing $s(b)$ is ensuring that $s(a)$ has already been computed for all predecessors.

**Theorem:** Every DAG must have at least one topological ordering (and there is an algorithm for finding it).

An ordering of nodes $(a_1, \ldots, a_k)$ of nodes in a DAG is a **topological ordering** if every edge $a_i \rightarrow a_j$ is such that $i < j$. 
Two Topological Orderings for Dressing DAG
Stop: What topological order(s) do you see for the alignment graph?
Three Topological Orderings for the Alignment Graph

STOP and Think:

Another topological ordering of the rectangular grid from Figure 5.15.

FIGURE 5.16 (see DETOUR: Constructing a Topological Ordering)

Problem to explain how to construct a longest path in a DAG.

Since every edge participates in only a single recurrence, the runtime of the only node with indegree 0 in achieving the nodes of the DAG in the order dictated by the topological ordering, which is an ordering, we can compute the length of the longest path from source to sink by visiting

It can be proven that any DAG has a topological ordering, and that this topological ordering can be constructed in time proportional to the number of edges in the graph.

We can now efficiently compute the length of a longest path in an arbitrary DAG, but

© 2022 Phillip Compeau
Pseudocode for Finding Length of Longest Path

STOP: What is the approximate ("big O" for the initiated) runtime of **LongestPath**?
Pseudocode for Finding Length of Longest Path

```plaintext
LongestPath(Graph, source, sink)
    for each node b in Graph
        \( s_b \leftarrow -\infty \)
    \( s_{source} \leftarrow 0 \)
    topologically order Graph
    for each node b in Graph (following the topological order)
        \( s_b \leftarrow \max \{ s_a + \text{weight of edge } a \rightarrow b \} \)
    return \( s_{sink} \)
```

**Answer:** We consider each edge exactly once, so (if we know a topological order) the runtime is proportional to the number of edges.
Topological Orderings for the Alignment Graph

STOP: How many edges does the alignment graph of strings $v$ and $w$ have?
STOP: How many edges does the alignment graph of strings \(v\) and \(w\) have?

Answer: Each node has 0, 1, or 3 predecessors. So, the number of edges is proportional to \(|v| \cdot |w|\).
From Finding the Maximum Length to Finding a Path

**LongestPath**(*Graph, source, sink*)

```plaintext
for each node b in Graph
    \( s_b \leftarrow -\infty \)

\( s_{source} \leftarrow 0 \)

topologically order Graph

for each node b in Graph (following the topological order)
    \( s_b \leftarrow \max_{a\text{ all predecessors of } b} \{s_a + \text{weight of edge } a \rightarrow b\} \)

return \( s_{sink} \)
```

**Note:** We can find the length of a longest path, but we still don’t know how to *construct* a longest path.
Finding a Longest Path

STOP: Take a moment to look at our solution from before when we found the maximum weight of a path. How might we have reconstructed the longest path?
BACKTRACKING IN THE ALIGNMENT GRAPH
Note: we highlighted the edge used at each node when computing length of longest path.
From a Recurrence to a Longest Path

Note: we highlighted the edge used at each node when computing length of longest path.

We remember one predecessor at each node, so following predecessors backward from sink yields longest path!
Recall that these Problems are the Same

**Longest Common Subsequence Length Problem:**
- **Input:** Two strings.
- **Output:** The length of a longest common subsequence of these strings.

**Symbol Matching Problem:**
- **Input:** Two strings.
- **Output:** The greatest number of matched symbols in any “alignment” of the two strings.
Putting it All Together

Longest Common Subsequence Length Problem:
• **Input:** Two strings.
• **Output:** The length of a longest common subsequence of these strings.

**STOP:** How can we find an LCS of two strings?
Putting it All Together

Longest Common Subsequence Length Problem:

- **Input:** Two strings.
- **Output:** The length of a longest common subsequence of these strings.

**Answer:**

1. Build the alignment graph, with "match" edges weighted 1.
2. Find the length of an LCS using recurrence relation.
3. Backtrack to find longest path.
We call node $b$.

To compute scores for any node $b$, you can arrive at the following recurrence for computing the length of an LCS:

$$s_b = \max_{\text{all predecessors } a \text{ of node } b} \left\{ s_a + \text{weight of edge from } a \text{ to } b \right\}$$

When computing the recurrence, we store a “pointer” to the predecessor node $a$ that achieved the maximum.
GLOBAL ALIGNMENT
Strengthening Alignment Scoring

Alignment score: Divided into three components:
• **match** reward (+1)
• **mismatch** penalty (-\(\mu\))
• **insertion/deletion** penalty (-\(\sigma\))

**STOP:** What were \(\mu\) and \(\sigma\) when finding a longest common subsequence?
Strengthening Alignment Scoring

Alignment score: Divided into three components:
- **match** reward (+1)
- **mismatch** penalty (-\(\mu\))
- **insertion/deletion** penalty (-\(\sigma\))

Answer: They were both equal to zero…
Global Alignment Problem: Find a highest-scoring alignment of two strings.

- **Input:** Two strings.
- **Output:** An alignment of the strings with maximum alignment score.
Strengthening Alignment Scoring

**Global Alignment Problem:** *Find a highest-scoring alignment of two strings.*

- **Input:** Two strings.
- **Output:** An alignment of the strings with maximum alignment score.

**STOP:** How can we modify the alignment network to solve this problem?
Strengthening Alignment Scoring

Answer: Slight modification to alignment network ... a longest path will yield an alignment of maximum score!
Strengthening Alignment Scoring

Answer: Slight modification to alignment network ... a longest path will yield an alignment of maximum score!

Exercise: What is the recurrence relation?
Two Cases: Mismatch vs. Match

Case 1

- Row $i - 1$ to Column $j - 1$
- Column $j$

- Row $i - 1$
- Row $i$

Case 2

- Row $i - 1$ to Column $j - 1$
- Column $j$

- Row $i - 1$
- Row $i$

$\text{length}(i, j) = \text{maximum of:}$

- $\text{length}(i - 1, j) - \sigma$
- $\text{length}(i, j - 1) - \sigma$
- $\text{length}(i - 1, j - 1) - \mu$

- $\text{length}(i - 1, j) - \sigma$
- $\text{length}(i, j - 1) - \sigma$
- $\text{length}(i - 1, j - 1) + 1$

© 2022 by Phillip Compeau
Further Strengthening Scoring with a Scoring Matrix

Scoring matrix:
Penalizes indels and matches/mismatches differently depending on individual symbols.

STOP: How do you think this matrix was computed?

PAM250 matrix
A Quick Aside About the BLOSUM Scoring Matrices

BLOSUM62 miscalculations improve search performance

Mark P Styczynski, Kyle L Jensen, Isidore Rigoutsos & Gregory Stephanopoulos

Nature Biotechnology 26, 274–275 (2008) | Cite this article
1144 Accesses | 53 Citations | 77 Altmetric | Metrics

To the editor:

The BLOSUM1 family of substitution matrices, and particularly BLOSUM62, is the de facto standard in protein database searches and sequence alignments. In the course of analyzing the evolution of the Blocks database2, we noticed errors in the software source code used to create the initial BLOSUM family of matrices (available online at ftp://ftp.ncbi.nih.gov/repository/blocks/...). The result of these errors is that the BLOSUM matrices—BLOSUM62, BLOSUM50, etc.—are quite different from the matrices that should have been calculated using the algorithm described by Henikoff and Henikoff3. Obviously, minor errors in research, and particularly in software source code, are quite common. This case is noteworthy for three reasons: first, the BLOSUM matrices are ubiquitous in computational biology; second, these errors have gone unnoticed for 15 years; and third, the ‘incorrect’ matrices perform better than the ‘intended’ matrices.
Global Alignment Problem: Find a highest-scoring alignment of two strings.

- **Input:** Two strings and a scoring matrix.
- **Output:** An alignment of the strings with maximum alignment score according to the scoring matrix.

**STOP:** How does this change the alignment graph?
Global Alignment Problem: Find a highest-scoring alignment of two strings.

- **Input:** Two strings and a scoring matrix.
- **Output:** An alignment of the strings with maximum alignment score according to the scoring matrix.

**Answer:** Every edge simply gets weighted with the cost of the corresponding scoring matrix value.
Summarizing our Global Alignment Algorithm

1. Form a 2-D array using the recurrence relation for dynamic programming.
2. Create array containing “backtracking pointers”.
3. After reaching the sink, backtrack to source to produce a maximum-weight path.
4. Infer the alignment corresponding to this path.
Summarizing our Global Alignment Algorithm

**STOP (biologists):** Would you rather align two genes as DNA strings (nucleotides) or as proteins (amino acids)?

A general method applicable to the search for similarities in ... by SB Needleman · 1970 · Cited by 14225 · Related articles

A computer adaptable method for finding similarities in the amino acid sequences of two proteins has been developed. From these findings it is possible to determine whether significant homology exists between the proteins. This information is used to trace their possible evolutionary development.
Answer: *If* we know that the genes wind up as protein, then a protein-level function will be more informative since there is a larger alphabet and the amino acids determine function of the protein.

**A general method applicable to the search for similarities in ...**

by SB Needleman · 1970 · Cited by 14225 · Related articles

A computer adaptable method for finding similarities in the amino acid sequences of two **proteins** has been developed. From these findings it is possible to determine whether significant homology exists between the **proteins**. This information is used to trace their possible evolutionary development.
Applying to Real Data

STOP: Let’s apply this to the same protein (say, hemoglobin subunit alpha) in a few different species. What do you think we will see?

- *Homo sapiens* vs. *Gorilla gorilla gorilla*
- *Homo sapiens* vs. *Bos Taurus* (cow)
- *Homo sapiens* vs. *Danio rerio* (zebrafish)

*Homo sapiens*: https://www.uniprot.org/uniprot/P69905
*Gorilla gorilla gorilla*: https://www.uniprot.org/uniprot/P01923
*Bos taurus*: https://www.uniprot.org/uniprot/P01966
*Danio rerio*: https://www.uniprot.org/uniprot/Q90487

EMBOSS “Needle” server: https://www.ebi.ac.uk/Tools/psa/emboss_needle/
Results of Hemoglobin Alignments

<table>
<thead>
<tr>
<th></th>
<th>Sequence 1</th>
<th>Sequence 2</th>
<th>Alignment Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>HBA_HUMAN</td>
<td>MVLSAPDKTNVKAAGKVGAHGAGEYGAAILFSLFPTKTYYFPFDDL</td>
<td>MVLSAPDKTNVKAAGKVGAHGAGEYGAAILFSLFPTKTYYFPFDDL</td>
<td>50</td>
</tr>
<tr>
<td>HBA_GORGO</td>
<td>VSAPDKTNVKAAGKVGAHGAGEYGAAILFSLFPTKTYYFPFDDL</td>
<td>VSAPDKTNVKAAGKVGAHGAGEYGAAILFSLFPTKTYYFPFDDL</td>
<td>49</td>
</tr>
<tr>
<td>HBA_HUMAN</td>
<td>HGSAQKVGHGKVDALTNAHVDDMPNLALSDDLHAKLRLVPDNFK</td>
<td>HGSAQKVGHGKVDALTNAHVDDMPNLALSDDLHAKLRLVPDNFK</td>
<td>100</td>
</tr>
<tr>
<td>HBA_GORGO</td>
<td>HGSAQKVGHGKVDALTNAHVDDMPNLALSDDLHAKLRLVPDNFK</td>
<td>HGSAQKVGHGKVDALTNAHVDDMPNLALSDDLHAKLRLVPDNFK</td>
<td>99</td>
</tr>
<tr>
<td>HBA_HUMAN</td>
<td>LLSHCLLVTLAHLPAEFTPAVHASLDKFLASVSTVLTSDKYR</td>
<td>LLSHCLLVTLAHLPAEFTPAVHASLDKFLASVSTVLTSDKYR</td>
<td>142</td>
</tr>
<tr>
<td>HBA_GORGO</td>
<td>LLSHCLLVTLAHLPAEFTPAVHASLDKFLASVSTVLTSDKYR</td>
<td>LLSHCLLVTLAHLPAEFTPAVHASLDKFLASVSTVLTSDKYR</td>
<td>141</td>
</tr>
</tbody>
</table>

Note: “|” means exact similarity, “:” means strong similarity, and “:" means weak similarity.

STOP: What is our hypothesis?
From the point of view of hemoglobin structure, it appears that gorilla is just an abnormal human.

Émile Zuckerkandl
From the point of view of hemoglobin structure, it appears that gorilla is just an abnormal human.

...that is of course nonsense. What the comparison really indicates is that hemoglobin is a bad choice and has nothing to tell us about attributes.
FROM GLOBAL TO LOCAL ALIGNMENT
From Global to “Local” Similarities

**Homeobox genes:** regulate embryonic development and are present in a large variety of species.
Finding “Local” Similarities

Homeobox genes: regulate embryonic development and are present in a large variety of species.

Mouse

...ARRSRTHFTKFOQTDILIEAFEKNRFPGIVTREKLAQQTGIPESRIHIWFOQRARRHPDPG...

Human

...ARQKQTFITWTQKNRLVQAFERNPPFDATATRKKLAETGQLQESRIQMWFQKQRSLYLKKS...

Homeodomain: area within homeobox genes of shared “local” similarity.
Finding “Local” Similarities

**Homeobox genes:** regulate embryonic development and are present in a large variety of species.

Mouse

...ARRSRTHFTKFQTDILIEAFEKNRFPGIVTREKLAQQTGIPESRIHIWFQNRARHPDPG...

...ARKQQTFTITWTKNRVLQAFERNPPFDTATRKLQAEQTGLQESRIQMWFOQKQRSLYLLKK...

Human

**Homeodomain:** area within homeobox genes of shared “local” similarity.

**STOP:** Will our dynamic programming algorithm find regions of local similarity?
Finding “Local” Similarities

**Exercise:** Score these alignments ($\sigma = \mu = 1$). Does our scoring function make sense?
Local alignments may be well away from “main diagonal” because they have a lot of indels on ends of the alignment.
Global Alignment Problem:

- **Input**: Two strings and a scoring matrix.
- **Output**: An alignment of the strings with maximum alignment score according to the scoring matrix.
Revisiting Global Alignment

Global Alignment Problem:
• **Input:** Two strings and a scoring matrix.
• **Output:** An alignment of the strings with maximum alignment score according to the scoring matrix.

**STOP:** How can we reformulate the problem statement to find areas of “local” similarity?
Local Alignment Problem:

- **Input:** Two strings \( v \) and \( w \) and a scoring matrix.
- **Output:** Substrings of \( v \) and \( w \) whose best global alignment score is maximized over all substrings.
Revisiting Global Alignment

Local Alignment Problem:
- **Input:** Two strings $v$ and $w$ and a scoring matrix.
- **Output:** Substrings of $v$ and $w$ whose best global alignment score is maximized over all substrings.

**STOP:** One idea for solving this is to solve the Global Alignment Problem for every pair of substrings of $v$ and $w$. Why is this an issue?
Revisiting Global Alignment

Local Alignment Problem:

• **Input:** Two strings $v$ and $w$ and a scoring matrix.
• **Output:** Substrings of $v$ and $w$ whose best global alignment score is maximized over all substrings.

**Answer:** There are $C(|v|, 2)$ substrings of $v$ and $C(|w|, 2)$ substrings of $w$. As a result we have about $|v|^2|w|^2$ alignments to construct!

This was understood in 1970, and yet the problem remained open …
A general method applicable to the search for similarities in the amino ...
by SB Needleman - 1970 - Cited by 12553 - Related articles
A computer adaptable method for finding similarities in the amino acid sequences of two proteins has been developed. From these findings it is possible to determine whether significant homology exists between the proteins. ... The maximum match is a number dependent upon the similarity of the sequences.

Identification of common molecular subsequences. - NCBI
by TF Smith - 1981 - Cited by 11181 - Related articles
Identification of common molecular subsequences. Smith TF, Waterman MS. PMID: 7265238;
[Indexed for ... MeSH terms. Base Sequence*; Models, Chemical *
“Free Rides” for Local Alignment

Add a zero-weight edge from the source to every node and the sink to every node.

This will allow a local alignment to start and end anywhere with no penalty.
“Free Rides” for Local Alignment

**Exercise:** What is the recurrence relation for the local alignment problem?
"Free Rides" for Local Alignment

Answer: It is given by

\[ s_{i,j} = \max \begin{cases} 
0 \\
 s_{i-1,j} + \text{Score}(v_i, -) \\
 s_{i,j-1} + \text{Score}(-, w_j) \\
 s_{i-1,j-1} + \text{Score}(v_i, w_j) 
\end{cases} \]

where the scores here are \(-\sigma, -\sigma\), and either +1 or \(-\mu\) (depending on a match vs. a mismatch).
“Free Rides” for Local Alignment

**Exercise:** After we apply the recurrence, where should we start backtracking? (That is, where does the best local alignment end?)
“Free Rides” for Local Alignment

Exercise: After we apply the recurrence, where should we start backtracking? (That is, where does the best local alignment end?)

Answer: Wherever the maximum value of the scoring table is.
STOP: Recall that the dynamic programming algorithm has runtime proportional to the number of edges in the network. How many zero-weight edges did we add?
STOP: Recall that the dynamic programming algorithm has runtime proportional to the number of edges in the network. How many zero-weight edges did we add?

Answer: Just ~2nm. 😊
The Solution to a Problem Unsolved for Ten Years Nearly Fits on One Slide


Identification of Common Molecular Subsequences

The identification of maximally homologous subsequences among sets of long sequences is an important problem in molecular sequence analysis. The problem is straightforward only if one restricts consideration to contiguous subsequences (segments) containing no internal deletions or insertions. The more general problem has its solution in an extension of sequence metrics (Sellers, 1974; Waterman et al., 1976) developed to measure the minimum number of "events" required to convert one sequence into another.

These developments in the modern sequence analysis began with the heuristic homology algorithm of Needleman & Wunsch (1970) which first introduced an iterative matrix method of calculation. Numerous other heuristic algorithms have been suggested including those of Fitch (1968) and Dayhoff (1969). More mathematically rigorous algorithms were suggested by Sankoff (1972), Reicbert et al. (1973) and Beyer et al. (1979), but these were generally not biologically satisfying or interpretable. Success came with Sellers (1974) development of a true metric measure of the distance between sequences. This metric was later generalized by Waterman et al. (1976) to include deletions/insertions of arbitrary length. This metric represents the minimum number of "mutational events" required to convert one sequence into another. It is of interest to note that Smith et al. (1980) have recently shown that under some conditions the generalized Sellers metric is equivalent to the original homology algorithm of Needleman & Wunsch (1970).

In this letter we extend the above ideas to find a pair of segments, one from each of two long sequences, such that there is no other pair of segments with greater similarity (homology). The similarity measure used here allows for arbitrary length deletions and insertions.

Algorithm

The two molecular sequences will be \( A = a_1a_2 \ldots a_n \) and \( B = b_1b_2 \ldots b_m \). A similarity \( s(a,b) \) is given between sequence elements \( a \) and \( b \). Deletions of length \( k \) are given weight \( W_k \). To find pairs of segments with high degrees of similarity, we set up a matrix \( H \). First set

\[
H_0 = H_{w} = 0 \text{ for } 0 \leq k \leq n \text{ and } 0 \leq l \leq m.
\]

Preliminary values of \( H \) have the interpretation that \( H_{i,j} \) is the maximum similarity of two segments ending in \( a_i \) and \( b_j \), respectively. These values are obtained from the relationship

\[
H_{i,j} = \max (H_{i-1,j-1} + s(a_ia_j), \max_{k=1}^{l} (H_{i-k,j} - W_k), \max_{l=1}^{m} (H_{i,j-1} - W_l), 0).
\]

1 \leq i \leq n \text{ and } 1 \leq j \leq m.

The formula for \( H_{i,j} \) follows by considering the possibilities for ending the segments at any \( a_i \) and \( b_j \).

1. If \( a_i \) and \( b_j \) are associated, the similarity is

\[
H_{i,j} = H_{i-1,j-1} + s(a_ia_j).
\]

2. If \( a_i \) is at the end of a deletion of length \( k \), the similarity is

\[
H_{i-k,j} = W_k.
\]

3. If \( b_j \) is at the end of a deletion of length \( l \), the similarity is

\[
H_{i,j-l} = W_l.
\]

4. Finally, a zero is included to prevent calculated negative similarity, indicating no similarity up to \( a_i \) and \( b_j \).

The pair of segments with maximum similarity is found by first locating the maximum element of \( H \). The other matrix elements leading to this maximum value are then sequentially determined with a traceback procedure ending with an element of \( H \) equal to zero. This procedure identifies the segments as well as the corresponding alignment. The pair of segments with the next best similarity is found by applying the traceback procedure to the second largest element of \( H \) not associated with the first traceback.

A simple example is given in Figure 1. In this example the parameters \( s(a,b) \) and \( W_k \) required were chosen on an a priori statistical basis. A match, \( a_i = b_j \), produced an \( s(a,b) \) value of unity while a mismatch produced a minus one-third. These values have an average for long, random sequences over an equally probable four letter set of zero. The deletion weight must be chosen to be at least equal to the difference between a match and a mismatch. The value used here was \( W_k = 1 + 1/3k \).

\[
\begin{array}{cccccccccc}
A & C & G & U & A & C & G & U & A & G \\
\hline
A & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
C & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
G & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
U & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
\end{array}
\]

\[
\begin{array}{cccccccccc}
A & C & G & U & A & C & G & U & A & G \\
\hline
A & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
C & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
G & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
U & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
\end{array}
\]

\[
\begin{array}{cccccccccc}
A & C & G & U & A & C & G & U & A & G \\
\hline
A & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
C & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
G & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
U & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
\end{array}
\]


* Zero need not be included unless there are negative values of \( s(a,b) \).
LETTERS TO THE EDITOR

Note, in this simple example, that the alignment obtained:

-G-C-A-U-U-G-
-G-C-C-U-C-G-

contains both a mismatch and an internal deletion. It is the identification of the latter which has not been previously possible in any rigorous manner.

This algorithm not only puts the search for pairs of maximally similar segments on a mathematically rigorous basis but it can be efficiently and simply programmed on a computer.

Northern Michigan University
T. F. SMITH

Los Alamos Scientific Laboratory
P.O. Box 1663, Los Alamos
N. Mex. 87545, U.S.A.

Received 14 July 1980

REFERENCES


Note added in proof: A weighting similar to that given above was independently developed by Walter Goad of Los Alamos Scientific Laboratory.
Smith and Waterman's Scoring Table

![Scoring Table](image)

<table>
<thead>
<tr>
<th>Δ</th>
<th>A</th>
<th>C</th>
<th>A</th>
<th>G</th>
<th>C</th>
<th>C</th>
<th>U</th>
<th>C</th>
<th>G</th>
<th>C</th>
<th>U</th>
<th>U</th>
<th>A</th>
<th>G</th>
</tr>
</thead>
<tbody>
<tr>
<td>Δ</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td>A</td>
<td>0.0</td>
<td>0.0</td>
<td>1.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>1.0</td>
<td>0.0</td>
</tr>
<tr>
<td>A</td>
<td>0.0</td>
<td>0.0</td>
<td>1.0</td>
<td>0.7</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>1.0</td>
<td>0.7</td>
</tr>
<tr>
<td>U</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.7</td>
<td>0.3</td>
<td>0.0</td>
<td>1.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>1.0</td>
<td>1.0</td>
<td>0.7</td>
</tr>
<tr>
<td>G</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>1.0</td>
<td>0.3</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.7</td>
<td>1.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.7</td>
<td>1.0</td>
</tr>
<tr>
<td>C</td>
<td>0.0</td>
<td>1.0</td>
<td>0.0</td>
<td>0.0</td>
<td>2.0</td>
<td>1.3</td>
<td>0.3</td>
<td>1.0</td>
<td>0.3</td>
<td>2.0</td>
<td>0.7</td>
<td>0.3</td>
<td>0.3</td>
<td>0.3</td>
</tr>
<tr>
<td>C</td>
<td>0.0</td>
<td>1.0</td>
<td>0.7</td>
<td>0.0</td>
<td>1.0</td>
<td>3.0</td>
<td>1.7</td>
<td>1.3</td>
<td>1.0</td>
<td>1.7</td>
<td>0.3</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td>A</td>
<td>0.0</td>
<td>0.0</td>
<td>2.0</td>
<td>0.7</td>
<td>0.3</td>
<td>1.7</td>
<td>2.7</td>
<td>1.3</td>
<td>1.0</td>
<td>0.7</td>
<td>1.0</td>
<td>1.3</td>
<td>1.3</td>
<td>0.0</td>
</tr>
<tr>
<td>U</td>
<td>0.0</td>
<td>0.0</td>
<td>0.7</td>
<td>1.7</td>
<td>0.3</td>
<td>1.3</td>
<td>2.7</td>
<td>2.3</td>
<td>1.0</td>
<td>0.7</td>
<td>1.7</td>
<td>2.0</td>
<td>1.0</td>
<td>1.0</td>
</tr>
<tr>
<td>U</td>
<td>0.0</td>
<td>0.0</td>
<td>0.3</td>
<td>0.3</td>
<td>1.3</td>
<td>1.0</td>
<td>2.3</td>
<td>2.3</td>
<td>2.0</td>
<td>0.7</td>
<td>1.7</td>
<td>2.7</td>
<td>1.7</td>
<td>1.0</td>
</tr>
<tr>
<td>U</td>
<td>0.0</td>
<td>0.0</td>
<td>0.3</td>
<td>0.3</td>
<td>1.3</td>
<td>1.0</td>
<td>2.3</td>
<td>2.3</td>
<td>2.0</td>
<td>0.7</td>
<td>1.7</td>
<td>2.7</td>
<td>1.7</td>
<td>1.0</td>
</tr>
<tr>
<td>G</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>1.3</td>
<td>0.0</td>
<td>1.0</td>
<td>1.0</td>
<td>2.0</td>
<td>3.3</td>
<td>2.0</td>
<td>1.7</td>
<td>1.3</td>
<td>2.3</td>
<td>2.7</td>
</tr>
<tr>
<td>A</td>
<td>0.0</td>
<td>0.0</td>
<td>1.0</td>
<td>0.0</td>
<td>1.0</td>
<td>0.3</td>
<td>0.7</td>
<td>0.7</td>
<td>2.0</td>
<td>3.0</td>
<td>1.7</td>
<td>1.3</td>
<td>2.3</td>
<td>2.0</td>
</tr>
<tr>
<td>C</td>
<td>0.0</td>
<td>1.0</td>
<td>0.0</td>
<td>0.7</td>
<td>1.0</td>
<td>2.0</td>
<td>0.7</td>
<td>1.7</td>
<td>1.7</td>
<td>3.0</td>
<td>2.7</td>
<td>1.3</td>
<td>1.0</td>
<td>2.0</td>
</tr>
<tr>
<td>G</td>
<td>0.0</td>
<td>0.0</td>
<td>0.7</td>
<td>1.0</td>
<td>0.3</td>
<td>0.7</td>
<td>1.7</td>
<td>0.3</td>
<td>2.7</td>
<td>2.7</td>
<td>2.3</td>
<td>1.0</td>
<td>2.0</td>
<td>2.0</td>
</tr>
<tr>
<td>G</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>1.7</td>
<td>0.7</td>
<td>0.3</td>
<td>0.3</td>
<td>1.3</td>
<td>1.3</td>
<td>2.3</td>
<td>1.3</td>
<td>2.3</td>
<td>2.0</td>
<td>2.0</td>
</tr>
</tbody>
</table>

Fig. 1. $H_{ij}$ matrix generated from the application of eqn (1) to the sequences A-A-U-G-C-C-A-U-U-G-A-C-G-G and C-A-G-C-C-U-C-G-C-U-U-A-G. The underlined elements indicate the trackback path from the maximal element 3.30.
ONE MORE INNOVATION: AFFINE ALIGNMENT
Comparing Same-Score Alignments

STOP: Which of these two alignments (which have the same score) is “better”? Why?
Comparing Same-Score Alignments

**Affine penalty:** a way of scoring contiguous gaps higher than discontiguous gaps.

- **gap opening penalty** (**σ**): given to first symbol.
- **gap extension penalty** (**ε**): given to extra symbols.
Comparing Same-Score Alignments

Affine penalty: a way of scoring contiguous gaps higher than discontiguous gaps.
- gap opening penalty ($\sigma$): given to first symbol.
- gap extension penalty ($\epsilon$): given to extra symbols.

If $\sigma = 5$ and $\epsilon = 1$, then the alignment on the left is penalized by $2\sigma = 10$, whereas the alignment on the right is only penalized by $\sigma + \epsilon = 6$. 

© 2022 Phillip Compeau
Adding Affine Gap Penalties

**Alignment with Affine Gap Penalties Problem:**

Construct a highest-scoring global alignment of two strings (with affine gap penalties).

- **Input:** Two strings along with numbers $\sigma$ and $\varepsilon$ and a scoring matrix.
- **Output:** A highest scoring global alignment between these strings, as defined by the gap opening and extension penalties $\sigma$ and $\varepsilon$.

**STOP:** How can we modify the alignment graph to solve this problem?
Adding “Long” Edges to Graph

One solution: Add (huge number of) new edges to alignment graph to facilitate longer gaps.

Figure 5.23 illustrates how affine gap penalties can be modeled in the alignment graph by introducing a new “long” edge for each gap. Since we do not know in advance where gaps should be located, we need to add edges accounting for every possible gap. Thus, affine gap penalties can be accommodated by adding all possible vertical and horizontal edges in the alignment graph to represent all possible gaps. Specifically, we add edges connecting \((i, j)\) to both \((i+k, j)\) and \((i, j+k)\) with weights \(s + e \cdot k\) for all possible gap sizes \(k\), as illustrated in Figure 5.24. For two sequences of length \(n\), the number of edges in the resulting alignment graph modeling affine gap penalties increases from \(O(n^2)\) to \(O(n^3)\).
Adding “Long” Edges to Graph

One solution: Add (huge number of) new edges to alignment graph to facilitate longer gaps.
Adding “Long” Edges to Graph

The runtime of our algorithm is proportional to the number of edges, so maybe we can use fewer edges.
Three-Level Manhattan for Affine Alignment

This is the same path in a “three-level” Manhattan.

CHAPTER 5

FIGURE 5.25

Building a three-level graph for alignment with a \( \text{aff} \) gap penalties. The lower level corresponds to gap extensions in \( v \), the middle level corresponds to matches and mismatches, and the upper level corresponds to gap extensions in \( w \).

FIGURE 5.26

Every path from source to sink in the standard alignment graph shown in Figure 5.23 corresponds to a path from source to sink in the three-level graph of the same length (and vice-versa). Every node in the middle level has one outgoing (blue) edge to the upper level and one outgoing (green) edge to the lower level, both represented by dashed edges and having weight equal to the gap opening penalty. Every node in the middle level also has one incoming blue edge from the upper level and one incoming green edge from the lower level, both represented by dashed edges and having zero weight (these edges close a gap).

Exercise Break:

Prove that the number of edges in the graph described in Figure 5.26 is at most \( 7 \cdot n \cdot m \) for sequences of length \( n \) and \( m \).

© 2022 Phillip Compeau
We Still are Finding a Longest Path and Have a Recurrence Relation

$lower_{i,j} = \max \left\{ lower_{i-1,j} - \epsilon, \middle_{i-1,j} - \sigma \right\}$

$middle_{i,j} = \max \left\{ lower_{i,j}, \middle_{i-1,j-1} + \text{Score}(v_i, w_j), upper_{i,j} \right\}$

$upper_{i,j} = \max \left\{ upper_{i,j-1} - \epsilon, \middle_{i,j-1} - \sigma \right\}$
The Number of Edges is Still Manageable

**Exercise:** What is the approximate number of edges in this graph?
The Number of Edges is Still Manageable

**Answer:** Approximately (in fact, at most) $7 \cdot |v| \cdot |w|$.
ALIGNING MULTIPLE STRINGS
Moving to Multiple Sequences

Multiple Alignment Problem: Find the highest-scoring alignment between multiple strings.

• **Input:** A collection of $t$ strings (and some way of scoring columns of a multiple alignment).
• **Output:** A multiple alignment of these strings having maximum score.
Multiple Alignment Problem: *Find the highest-scoring alignment between multiple strings.*

- **Input:** A collection of $t$ strings (and some way of scoring columns of a multiple alignment).
- **Output:** A multiple alignment of these strings having maximum score.

STOP: What algorithm would you propose to solve this problem?
The multiple alignment matrix is a generalization of the pairwise alignment matrix to more than two sequences. The three arrays shown below this alignment record the respective number of symbols in \texttt{ATGTTATA}, \texttt{AGCGATCA}, and \texttt{ATCGTCTC} encountered up to a given position. Together, these three arrays correspond to a path in a three-dimensional grid:

\[(0, 0, 0) \rightarrow (1, 1, 1) \rightarrow (2, 2, 2) \rightarrow (2, 3, 3) \rightarrow (3, 4, 4) \rightarrow (4, 5, 5) \rightarrow (5, 6, 5) \rightarrow (6, 7, 6) \rightarrow (7, 7, 7) \rightarrow (8, 8, 8)\]

As the alignment graph for two sequences is a grid of squares, the alignment graph for three sequences is a grid of cubes. Every node in the 3-way alignment graph has up to seven incoming edges, as shown in Figure 5.31.

The score of a multiple alignment is defined as the sum of scores of the alignment columns (or, equivalently, weights of edges in the alignment path), with an optimal alignment being one that maximizes this score. In the case of an amino acid alphabet, the score of a multiple alignment is defined as the sum of scores of the alignment columns (or, equivalently, weights of edges in the alignment path), with an optimal alignment being one that maximizes this score. In the case of an amino acid alphabet, the score of a multiple alignment is defined as the sum of scores of the alignment columns (or, equivalently, weights of edges in the alignment path), with an optimal alignment being one that maximizes this score.
The multiple alignment matrix is a generalization of the pairwise alignment matrix to more than two sequences. The three arrays shown below this alignment record the respective number of symbols in ATGTTATA, AGCGATCA, and ATCGTCTC encountered up to a given position. Together, these three arrays correspond to a path in a three-dimensional grid:

\[(i-1, j-1, k-1), (i-1, j, k-1), (i-1, j-1, k), (i-1, j, k), (i, j-1, k), (i, j, k-1), (i, j, k)\]

As the alignment graph for two sequences is a grid of squares, the alignment graph for three sequences is a grid of cubes. Every node in the 3-way alignment graph has up to seven incoming edges, as shown in Figure 5.31.

The score of a multiple alignment is defined as the sum of scores of the alignment columns (or, equivalently, weights of edges in the alignment path), with an optimal alignment being one that maximizes this score. In the case of an amino acid alphabet, 278

STOP: What is the issue with the dynamic programming approach in multiple dimensions?
Answer: The number of edges in a single block grows like $2^t - 1$...
Moving to Multiple Dimensions

**STOP:** What heuristic might you propose to align multiple sequences?
Greedy Heuristic for Multiple Alignment

1. Find an optimal pairwise alignment of each pair of strings.
2. Combine the set of optimal pairwise alignments into a multiple alignment.
Greedy Heuristic for Multiple Alignment

1. Find an optimal pairwise alignment of each pair of strings.
2. Combine the set of optimal pairwise alignments into a multiple alignment.

STOP: Try this approach on the strings CCCCTTTT, TTTTGGGG, and GGGGC CCC.
Greedy Heuristic for Multiple Alignment

1. Find an optimal pairwise alignment of each pair of strings.
2. Combine the set of optimal pairwise alignments into a multiple alignment.

There is no way to combine these optimal pairwise alignment into a meaningful multiple alignment!

© 2022 Phillip Compeau
Fortunately, strings that we are aligning will often be so similar that even simple heuristics will find correct alignments. But not always...
INTERLUDE: WHY DON’T WE HAVE AN HIV VACCINE?
Waiting for an HIV Vaccine …

Yet another terrible disease is about to yield to patience, persistence and outright genius.

Margaret Heckler
1984

© 2022 Phillip Compeau
Waiting for an HIV Vaccine …

Yet another terrible disease is about to yield to patience, persistence and outright genius.

It is no longer a question of whether we can develop an AIDS vaccine, it is simply a question of when.

© 2022 Phillip Compeau
Waiting for an HIV Vaccine …

The failed HIV Merck vaccine study: a step back or a launching point for future vaccine development?

Rafick-Pierre Sekaly

See the article "An HIV-1 clade C DNA prime, NYVAC boost vaccine regimen induces reliable, polyfunctional, and long-lasting T cell responses" on page 63.

This article has been cited by other articles in PMC.

Abstract

The world of human immunodeficiency virus (HIV) vaccines has suffered a baffling setback. The first trial of a vaccine designed to elicit strong cellular immunity has shown no protection against infection. More alarmingly, the vaccine appeared to increase the rate of HIV infection in individuals with prior immunity against the adenovirus vector used in the vaccine. A new study in this issue suggests that a different vaccine approach—using a DNA prime/poxvirus boost strategy—induces polyfunctional immune responses to an HIV immunogen. The disappointing results of the recent vaccine trial suggest that a more thorough assessment of vaccine-induced immune responses is urgently needed, and that more emphasis should be placed on primate models before efficacy trials are undertaken.
... and yet we have a coronavirus vaccine in under a year???
Many Vaccines Target Viral Surface Proteins

How coronavirus vaccine will work

Scientists have taken genes for the spike protein on the surface of coronavirus, and put them into a harmless virus to make a vaccine.

This is injected into the patient.

The vaccine enters cells, which then start to produce the coronavirus spike protein.

This prompts the immune system to produce antibodies and activate killer T-cells to destroy infected cells.

If the patient encounters coronavirus again, the antibodies and T cells are triggered to fight the virus.

Citation: https://www.bbc.com/news/health-52394485
Many Vaccines Target Viral Surface Proteins

Vaccines training the immune system to recognize HIV’s surface proteins fail because HIV strains are so variable.

https://www.frontiersin.org/articles/10.3389/fimmu.2015.00336/full
HIV Drug “Cocktails” Have to Deal with Variability

The HIV population in a single infected individual rapidly evolves to evade the immune system.

envelope glycoprotein gp120

© 2022 Phillip Compeau
HIV Drug “Cocktails” Have to Deal with Variability

The HIV population in a single infected individual rapidly evolves to evade the immune system.

envelope glycoprotein gp120

HIV strains from different patients are diverged phenotypes requiring different drug cocktails.
Multiple Alignment Problem: Find the highest-scoring alignment between multiple strings.

- **Input:** A collection of $t$ strings (and some way of scoring columns of a multiple alignment).
- **Output:** A multiple alignment of these strings having maximum score.

A single misalignment could lead to an error, so we have to be accurate. And so we need a problem formulation that scores different columns differently.
Once we have a collection of known protein alignments ("families"), we need to be able to identify which family a new protein belongs to. That is, add a new string into an existing alignment.

```
VKKLGEQR-NKTIIFNQPSGGDLEIVMSFNCGGEFFYCNTTQLFN-----------NSTES-------DTITL
VKKLGEQR-NKTIIFNQPSGGDLEIVMSFNCGGEFFYCNTTQLFN-----------NSTDNG-------DTITL
VKKLGEQR-NKTIIFNQPSGGDLEIVMSFNCGGEFFYCNTTQLFD-----------NSTESNN------DTITL
VDKLREQFGKNKTIIIFNQPSGGDLEIVMTFNCGGEFFYCNTTQLFNSTWNS----TGNGTESYNGQENGITTNL
VDKLREQFGKNKTIIIFNQPSGGDLEIVMTFNCGGEFFYCNTTQLFNSTWNG----TNTT---GLDG--NDTITL
VDKLREQFGKNKTIIIFNQSSGGGLEIVMTFNCGGEFFYCNTTQLFNSWTE-----NSTE--GLHG--DDTITL
VKKLGEQFG-NKTIIFNQSSGGGLEIVMTFNCGGEFFYCNTTQLFNNS--TR-----NSTESNNGQGNDTTTL
VKKLREQFGKNTIIFKQSSGGGLEIVTHTFNCAGEFFYCNTTQLFNSNWTE------NSITGLDG--NDTITL
VGKREQFGRKTIIFNQPSGGGLEIVMHSFNQCGEFFYCNTTTLFNSTWDNSTWNSTGKDEENGN--NDTITL
```
GAMBLING WITH YAKUZA
Chō-Han and "Heads or Tails"

**Chō-Han**: A game played in 18th Century Japanese casinos in which players wager that the sum will be even ("chō") or odd ("han").

We will think about an equivalent game called "Heads or Tails" in which we bet on a coin toss.
Chō-Han and "Heads or Tails"

**Chō-Han:** A game played in 18\textsuperscript{th} Century Japanese casinos in which players wager that the sum will be even ("chō") or odd ("han").

We will think about an equivalent game called "Heads or Tails" in which we bet on a coin toss.
A crooked dealer may use one of two coins:

- The **fair** coin \((F)\) gives heads with probability \(\frac{1}{2}\):
  \[
  \Pr_F(\text{“Head”}) = \frac{1}{2} \quad \Pr_F(\text{“Tail”}) = \frac{1}{2}
  \]

- The **biased** coin \((B)\) gives heads with probability \(\frac{3}{4}\):
  \[
  \Pr_B(\text{“Head”}) = \frac{3}{4} \quad \Pr_B(\text{“Tail”}) = \frac{1}{4}
  \]
A crooked dealer may use one of two coins:

• The **fair** coin \((F)\) gives heads with probability \(\frac{1}{2}\):
  \[
  \Pr_{F}(\text{“Head”}) = \frac{1}{2} \quad \Pr_{F}(\text{“Tail”}) = \frac{1}{2}
  \]

• The **biased** coin \((B)\) gives heads with probability \(\frac{3}{4}\):
  \[
  \Pr_{B}(\text{“Head”}) = \frac{3}{4} \quad \Pr_{B}(\text{“Tail”}) = \frac{1}{4}
  \]

**STOP:** Say that you play Heads or Tails 100 times, and the coin produces heads 63 times. Is the dealer cheating? Was the coin fair or biased?
Identifying a Biased Coin

A crooked dealer may use one of two coins:
- The fair coin (F) gives heads with probability \( \frac{1}{2} \):
  \[
  \Pr_F(\text{“Head”}) = \frac{1}{2} \quad \quad \Pr_F(\text{“Tail”}) = \frac{1}{2}
  \]
- The biased coin (B) gives heads with probability \( \frac{3}{4} \):
  \[
  \Pr_B(\text{“Head”}) = \frac{3}{4} \quad \quad \Pr_B(\text{“Tail”}) = \frac{1}{4}
  \]

**STOP:** A better question would be, “Which coin is more likely to have been used if we see heads 63 times?”
A crooked dealer may use one of two coins:

- The **fair** coin ($F$) gives heads with probability $1/2$:
  $$\Pr_F(“\text{Head”}) = 1/2 \quad \Pr_F(“\text{Tail”}) = 1/2$$

- The **biased** coin ($B$) gives heads with probability $3/4$:
  $$\Pr_B(“\text{Head”}) = 3/4 \quad \Pr_B(“\text{Tail”}) = 1/4$$

**Answer:** 63 is closer to 75 than 100, but there must be a more quantitative answer …
Identifying a Biased Coin

Given a sequence of $n$ flips with $k$ “Heads”:

$$x = x_1 x_2 \ldots x_n$$
Identifying a Biased Coin

Given a sequence of $n$ flips with $k$ “Heads”:

$$x = x_1 x_2 \ldots x_n$$

The probability this sequence was generated by the fair coin:

$$\Pr(x|F) = \Pr_F(x_1) \cdot \ldots \cdot \Pr_F(x_n) = (1/2)^n$$
Identifying a Biased Coin

Given a sequence of $n$ flips with $k$ “Heads”:

$$x = x_1 x_2 \ldots x_n$$

The probability this sequence was generated by the fair coin:

$$\Pr(x|F) = \Pr_F(x_1) \cdot \ldots \cdot \Pr_F(x_n) = (1/2)^n$$

The probability that it was generated by the biased coin:

$$\Pr(x|B) = \Pr_B(x_1) \cdot \ldots \cdot \Pr_B(x_n) = (3/4)^k \cdot (1/4)^{n-k}$$
Identifying a Biased Coin

\[
\Pr(x|F) > \Pr(x|B) \rightarrow \text{fair is more likely}
\]

\[
\Pr(x|F) < \Pr(x|B) \rightarrow \text{biased is more likely}
\]

The probability this sequence was generated by the fair coin:

\[
\Pr(x|F) = \Pr_F(x_1) \cdot \ldots \cdot \Pr_F(x_n) = (1/2)^n
\]

The probability that it was generated by the biased coin:

\[
\Pr(x|B) = \Pr_B(x_1) \cdot \ldots \cdot \Pr_B(x_n) = (3/4)^k \cdot (1/4)^{n-k}
\]
**Exercise:** For a sequence of 100 flips with 63 heads, which coin is more likely?

The probability this sequence was generated by the fair coin:

\[
\Pr(x|F) = \Pr_F(x_1) \cdot \ldots \cdot \Pr_F(x_n) = (1/2)^n
\]

The probability that it was generated by the biased coin:

\[
\Pr(x|B) = \Pr_B(x_1) \cdot \ldots \cdot \Pr_B(x_n) = (3/4)^k \cdot (1/4)^{n-k}
\]
Both \((1/2)^{100}\) and \((3/4)^{63} \cdot (1/4)^{37}\) are so close to zero that this question is harder than it seems!

The probability this sequence was generated by the **fair** coin:

\[
\Pr(x|F) = \Pr_F(x_1) \cdot \ldots \cdot \Pr_F(x_n) = (1/2)^n
\]

The probability that it was generated by the **biased** coin:

\[
\Pr(x|B) = \Pr_B(x_1) \cdot \ldots \cdot \Pr_B(x_n) = (3/4)^k \cdot (1/4)^{n-k}
\]
Both \((1/2)^{100}\) and \((3/4)^{63} \cdot (1/4)^{37}\) are so close to zero that this question is harder than it seems!

Equilibrium occurs when

\[ \Pr(x|F) = \Pr(x|B) \]
Both $(1/2)^{100}$ and $(3/4)^{63} \cdot (1/4)^{37}$ are so close to zero that this question is harder than it seems!

Equilibrium occurs when

$$\Pr(x|F) = \Pr(x|B)$$

$$(1/2)^n = (3/4)^k \cdot (1/4)^{n-k}$$
Both \((1/2)^{100}\) and \((3/4)^{63} \cdot (1/4)^{37}\) are so close to zero that this question is harder than it seems!

Equilibrium occurs when

\[
\Pr(x|F) = \Pr(x|B)
\]

\[
(1/2)^n = (3/4)^k \cdot (1/4)^{n-k}
\]

\[
(1/2)^n = 3^k/4^n
\]
Both \((1/2)^{100}\) and \((3/4)^{63} \cdot (1/4)^{37}\) are so close to zero that this question is harder than it seems!

Equilibrium occurs when

\[
\Pr(x|F) = \Pr(x|B) \\
(1/2)^n = (3/4)^k \cdot (1/4)^{n-k} \\
(1/2)^n = 3^k/4^n \\
2^n = 3^k
\]
Identifying a Biased Coin

Both $(1/2)^{100}$ and $(3/4)^{63} \cdot (1/4)^{37}$ are so close to zero that this question is harder than it seems!

Equilibrium occurs when

$$\Pr(x|F) = \Pr(x|B)$$

$$(1/2)^n = (3/4)^k \cdot (1/4)^{n-k}$$

$$(1/2)^n = 3^k/4^n$$

$$2^n = 3^k$$

$$n = k \cdot \log_2(3)$$
Both $(1/2)^{100}$ and $(3/4)^{63} \cdot (1/4)^{37}$ are so close to zero that this question is harder than it seems!

Equilibrium occurs when

\[ \Pr(x|F) = \Pr(x|B) \]
\[ (1/2)^n = (3/4)^k \cdot (1/4)^{n-k} \]
\[ (1/2)^n = 3^k/4^n \]
\[ 2^n = 3^k \]
\[ n = k \cdot \log_2(3) \]
\[ k = n / \log_2(3) \approx 0.632 \cdot n \]
Identifying a Biased Coin

STOP: So … which coin was more likely?

Equilibrium occurs when

\[
\Pr(x|F) = \Pr(x|B)
\]

\[
(1/2)^n = (3/4)^k \cdot (1/4)^{n-k}
\]

\[
(1/2)^n = 3^k/4^n
\]

\[
2^n = 3^k
\]

\[
n = k \cdot \log_2(3)
\]

\[
k = n / \log_2(3) \approx 0.632 \ n
\]
Identifying a Biased Coin

**Answer:** The fair coin (!) because \( k < 0.632 \ n. \)

Equilibrium occurs when

\[
\Pr(x|F) = \Pr(x|B) \\
(1/2)^n = (3/4)^k \cdot (1/4)^{n-k} \\
(1/2)^n = 3^k/4^n \\
2^n = 3^k \\
n = k \cdot \log_2(3) \\
k = n / \log_2(3) \approx 0.632 \ n
\]
Log-odds ratio: The logarithm of the ratio of \( \Pr(x|F) \) and \( \Pr(x|B) \):

\[
\log_2(\Pr(x|F) / \Pr(x|B)) = \log_2 \left( \frac{2^n}{3^k} \right) = n - k \cdot \log_2(3)
\]

The log-odds ratio is positive when \( \Pr(x|F) > \Pr(x|B) \) and negative when \( \Pr(x|B) > \Pr(x|F) \).
Bakuto Dealers Were Shirtless for a Reason…

Now let’s assume that the dealer has both a fair and biased coin and can switch back and forth.
Bakuto Dealers Were Shirtless for a Reason…

Now let’s assume that the dealer has both a fair and biased coin and can switch back and forth.

**Casino Problem:** Given a sequence of coin flips, determine when the dealer used a fair coin and a biased coin.

- **Input:** A sequence $x = x_1 x_2 \dotsc x_n$ of flips made by coins $F$ and $B$.
- **Output:** A sequence $\pi = \pi_1 \pi_2 \dotsc \pi_n$, with each $\pi_i$ being equal to either $F$ or $B$. 
Bakuto Dealers Were Shirtless for a Reason…

This is not a computational problem! Any of the $2^n$ sequences $\pi$ can generate any $x$.

**Casino Problem:** Given a sequence of coin flips, determine when the dealer used a fair coin and a biased coin.

- **Input:** A sequence $x = x_1 \ x_2 \ldots \ x_n$ of flips made by coins $F$ and $B$.
- **Output:** A sequence $\pi = \pi_1 \ \pi_2 \ldots \ \pi_n$, with each $\pi_i$ being equal to either $F$ or $B$. 
Grading $\pi$ Using Log-Odds Ratio

$\text{HHHTHTHHHT}$  $\text{BBBBBB}$  $\Pr(x|F) < \Pr(x|B)$
Grading $\pi$ Using Log-Odds Ratio

$\textbf{HHHTHTHHHT}$

$\textbf{BBBBBB}$

$\textbf{FFFFF}$

$\Pr(x|F) < \Pr(x|B)$

$\Pr(x|F) > \Pr(x|B)$
Grading π Using Log-Odds Ratio

\[ \frac{\Pr(x|F)}{\Pr(x|B)} < 1 \]

\[ \frac{\Pr(x|F)}{\Pr(x|B)} > 1 \]
Grading $\pi$ Using Log-Odds Ratio

If $n = \# \text{ tosses and } k = \# \text{ heads}$, use log-odds ratio:
$$\log_2(\Pr(x|F)/\Pr(x|B)) = \log_2(2^n/3^k) = n - k \cdot \log_2(3) .$$

- $\text{HHHTHTHHHT}$
- $\text{BBBBBB}$
- $\text{FFFFF}$

$\Pr(x|F)/\Pr(x|B) < 1$
$\Pr(x|F)/\Pr(x|B) > 1$
Grading \( \pi \) Using Log-Odds Ratio

If \( n = \# \) tosses and \( k = \# \) heads, use log-odds ratio:

\[
\log_2\left( \frac{\Pr(x|F)}{\Pr(x|B)} \right) = \log_2(2^n/3^k) = n - k \cdot \log_2(3) .
\]

- Log-odds ratio \( < 0 \): Biased coin more likely
- Log-odds ratio \( > 0 \): Fair coin more likely
Grading $\pi$ Using Log-Odds Ratio
Grading $\pi$ Using Log-Odds Ratio

HHHTHHTHHHT
BBBBBB
FFFFF
FFFFF
FFFFF
Grading $\pi$ Using Log-Odds Ratio
Grading $\pi$ Using Log-Odds Ratio

STOP: What are the disadvantages of this approach?
Grading $\pi$ Using Log-Odds Ratio

**Answer:** Overlapping windows may make different prediction for the same flip.
Grading $\pi$ Using Log-Odds Ratio

(Also, there is no clear choice for window length.)
HIDDEN MARKOV MODELS
Think of the dealer as a machine with $k$ hidden states ($F$ and $B$) that proceeds in a sequence of steps.
Turning the Dealer into a Machine

Think of the dealer as a machine with $k$ hidden states ($F$ and $B$) that proceeds in a sequence of steps.

In each step, it emits a symbol (H or T) with certain probability based on its current state.
Think of the dealer as a machine with $k$ hidden states ($F$ and $B$) that proceeds in a sequence of steps.

In each step, it emits a symbol (H or T) with certain probability based on its current state.

While in a certain state, the machine makes two decisions:

1. Which *symbol* will I emit?
2. Which *hidden state* will I move to next?
Why are the States “Hidden”?

An observer can see the emitted symbols of an HMM but does not know which state the HMM is currently in.
Why are the States “Hidden”? 

An observer can see the emitted symbols of an HMM but *does not* know which state the HMM is currently in.

**Goal:** infer the most likely sequence of hidden states of an HMM based on the sequence of emitted symbols.
Why are the States “Hidden”?

An observer can see the emitted symbols of an HMM but does not know which state the HMM is currently in.

**Goal:** infer the most likely sequence of hidden states of an HMM based on the sequence of emitted symbols.

If we also have a collection of probabilities for the likelihood of changing states, we have a hidden Markov model (HMM).
An HMM Consists of Four Objects

$\Sigma$: an alphabet of emitted symbols

H and T
An HMM Consists of Four Objects

Σ: an alphabet of emitted symbols

States: a set of hidden states

© 2022 Phillip Compeau
An HMM Consists of Four Objects

Σ: an alphabet of emitted symbols

States: a set of hidden states

Transition = (transition\textsubscript{\textit{i},\textit{k}}): a |States| × |States| matrix of transition probabilities

(of changing from state \textit{i} to state \textit{k})
An HMM Consists of Four Objects

**Σ:** an *alphabet* of emitted symbols

**States:** a set of *hidden states*

*Transition* = \((transition_{l,k}): \text{a } |\text{States}| \times |\text{States}|\)

matrix of *transition probabilities*

(of changing from state \(l\) to state \(k\))

<table>
<thead>
<tr>
<th></th>
<th>(F)</th>
<th>(B)</th>
</tr>
</thead>
<tbody>
<tr>
<td>(F)</td>
<td>0.9</td>
<td>0.1</td>
</tr>
<tr>
<td>(B)</td>
<td>0.1</td>
<td>0.9</td>
</tr>
</tbody>
</table>

*Emission* = \((emission_k(b)): \text{a } |\text{States}| \times |\Sigma|\)

matrix of *emission probabilities*

(emitting symbol \(b\) when HMM is in state \(k\))

<table>
<thead>
<tr>
<th></th>
<th>(H)</th>
<th>(T)</th>
</tr>
</thead>
<tbody>
<tr>
<td>(F)</td>
<td>0.50</td>
<td>0.50</td>
</tr>
<tr>
<td>(B)</td>
<td>0.75</td>
<td>0.25</td>
</tr>
</tbody>
</table>
The HMM Diagram Visualizes an HMM

HMM Diagram:
• solid nodes are hidden states
• dashed nodes are emitted symbols
• solid directed edges: connect states and are labeled by transition probabilities
• dashed directed edges: connect state to symbol and labeled by emission probabilities.
**Hidden path:** a sequence $\pi = \pi_1 \ldots \pi_n$ of states that an HMM passes through.

$\Pr(x, \pi)$: the probability that an HMM follows the hidden path $\pi$ and emits the string $x = x_1 x_2 \ldots x_n$.

$x$: T H T H H H T H T T H

$\pi$: F F F B B B B B F F F

© 2022 Phillip Compeau
Representing $\Pr(x, \pi)$ as a Product

HMM follows $\pi$ and emits $x$ when two events occur.

1. The HMM follows the path $\pi$. The probability of this event is $\Pr(\pi)$.

2. Given that HMM follows path $\pi$, it emits $x$. This is the conditional probability $\Pr(x|\pi)$. 
Representing \( \Pr(x, \pi) \) as a Product

HMM follows \( \pi \) and emits \( x \) when two events occur.
1. The HMM follows the path \( \pi \). The probability of this event is \( \Pr(\pi) \).
2. Given that HMM follows path \( \pi \), it emits \( x \). This is the conditional probability \( \Pr(x|\pi) \).

This is a more general result in probability:

\[
\Pr(x, \pi) = \Pr(\pi) \cdot \Pr(x|\pi).
\]

Let’s compute each of the terms on the right.
First: Computing \( \Pr(\pi) \)

\( \Pr(\pi) \) is just the \textit{product} of the probabilities \( \Pr(\pi_i \rightarrow \pi_{i+1}) \), where each \( \Pr(\pi_i \rightarrow \pi_{i+1}) \) is the probability of transitioning from state \( \pi_i \) to state \( \pi_{i+1} \).

\[

table
\begin{array}{cccccccccccc}
  i & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 & 11 \\
  \pi & F & F & F & B & B & B & B & B & F & F & F \\
  x & T & H & T & H & H & H & T & H & T & T & H \\
  \Pr(\pi_i \rightarrow \pi_{i+1}) & \frac{1}{2} & \frac{9}{10} & \frac{9}{10} & \frac{1}{10} & \frac{9}{10} & \frac{9}{10} & \frac{9}{10} & \frac{9}{10} & \frac{1}{10} & \frac{9}{10} & \frac{9}{10} \\
\end{array}
\]
First: Computing $\Pr(\pi)$

$\Pr(\pi)$ is just the *product* of the probabilities $\Pr(\pi_i \rightarrow \pi_{i+1})$, where each $\Pr(\pi_i \rightarrow \pi_{i+1})$ is the probability of transitioning from state $\pi_i$ to state $\pi_{i+1}$.

Below: $\Pr(\pi_0 \rightarrow \pi_1)$ is $\frac{1}{2}$ since we assume there is a 50-50 chance of starting in state $\pi_1$.

<table>
<thead>
<tr>
<th>$i$</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>11</th>
</tr>
</thead>
<tbody>
<tr>
<td>$x$</td>
<td>T</td>
<td>H</td>
<td>T</td>
<td>H</td>
<td>H</td>
<td>H</td>
<td>T</td>
<td>H</td>
<td>T</td>
<td>T</td>
<td>H</td>
</tr>
<tr>
<td>$\pi$</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>B</td>
<td>B</td>
<td>B</td>
<td>B</td>
<td>B</td>
<td>F</td>
<td>F</td>
<td>F</td>
</tr>
<tr>
<td>$\Pr(\pi_i \rightarrow \pi_{i+1})$</td>
<td>$\frac{1}{2}$</td>
<td>$\frac{9}{10}$</td>
<td>$\frac{9}{10}$</td>
<td>$\frac{9}{10}$</td>
<td>$\frac{9}{10}$</td>
<td>$\frac{9}{10}$</td>
<td>$\frac{9}{10}$</td>
<td>$\frac{9}{10}$</td>
<td>$\frac{9}{10}$</td>
<td>$\frac{9}{10}$</td>
<td>$\frac{9}{10}$</td>
</tr>
</tbody>
</table>
Next: Computing $\Pr(x | \pi)$

If we know the hidden path, then the probability of emitting a string $x = x_1 \ldots x_n$ is just the product of the emission probabilities of each symbol $x_i$.

$$
\Pr(x | \pi) = \prod_{i=1}^{n} \Pr(x_i | \pi_i) = \prod_{i=1}^{n} \text{emission}_{\pi_i}(x_i).
$$

<table>
<thead>
<tr>
<th>$i$</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>11</th>
</tr>
</thead>
<tbody>
<tr>
<td>$x$</td>
<td>T</td>
<td>H</td>
<td>T</td>
<td>H</td>
<td>H</td>
<td>H</td>
<td>T</td>
<td>H</td>
<td>T</td>
<td>T</td>
<td>H</td>
</tr>
<tr>
<td>$\pi$</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>B</td>
<td>B</td>
<td>B</td>
<td>B</td>
<td>B</td>
<td>F</td>
<td>F</td>
<td>F</td>
</tr>
</tbody>
</table>

$\Pr(\pi_i \rightarrow \pi_{i+1})$:  

<table>
<thead>
<tr>
<th></th>
<th>1/2</th>
<th>9/10</th>
<th>9/10</th>
<th>1/10</th>
<th>9/10</th>
<th>9/10</th>
<th>9/10</th>
<th>9/10</th>
<th>1/10</th>
<th>9/10</th>
<th>9/10</th>
</tr>
</thead>
</table>

$\Pr(x_i | \pi_i)$:  

<table>
<thead>
<tr>
<th></th>
<th>1/2</th>
<th>1/2</th>
<th>1/2</th>
<th>3/4</th>
<th>3/4</th>
<th>3/4</th>
<th>1/4</th>
<th>3/4</th>
<th>1/2</th>
<th>1/2</th>
<th>1/2</th>
</tr>
</thead>
</table>

© 2022 Phillip Compeau
Putting it All Together

Exercise: Compute \( \Pr(x, \pi) = \Pr(\pi) \cdot \Pr(x|\pi) \) for the \( x \) and \( \pi \) below. Can you find a better explanation for \( x = \text{“THTHHHHTHTTH”} \) than \( \pi = FFFBBBBBFFFF \)?

\[
\Pr(x|\pi) = \prod_{i=1}^{n} \Pr(x_i|\pi_i) = \prod_{i=1}^{n} \text{emission}_{\pi_i}(x_i).
\]

<table>
<thead>
<tr>
<th>( i )</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>11</th>
</tr>
</thead>
<tbody>
<tr>
<td>( x )</td>
<td>T</td>
<td>H</td>
<td>T</td>
<td>H</td>
<td>H</td>
<td>H</td>
<td>T</td>
<td>H</td>
<td>T</td>
<td>T</td>
<td>H</td>
</tr>
<tr>
<td>( \pi )</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>B</td>
<td>B</td>
<td>B</td>
<td>B</td>
<td>B</td>
<td>F</td>
<td>F</td>
<td>F</td>
</tr>
<tr>
<td>( \Pr(\pi_i \rightarrow \pi_{i+1}) )</td>
<td>( \frac{1}{2} )</td>
<td>( \frac{9}{10} )</td>
<td>( \frac{9}{10} )</td>
<td>( \frac{1}{10} )</td>
<td>( \frac{9}{10} )</td>
<td>( \frac{9}{10} )</td>
<td>( \frac{9}{10} )</td>
<td>( \frac{9}{10} )</td>
<td>( \frac{1}{10} )</td>
<td>( \frac{9}{10} )</td>
<td>( \frac{9}{10} )</td>
</tr>
<tr>
<td>( \Pr(x_i</td>
<td>\pi_i) )</td>
<td>( \frac{1}{2} )</td>
<td>( \frac{1}{2} )</td>
<td>( \frac{1}{2} )</td>
<td>( \frac{3}{4} )</td>
<td>( \frac{3}{4} )</td>
<td>( \frac{3}{4} )</td>
<td>( \frac{1}{4} )</td>
<td>( \frac{3}{4} )</td>
<td>( \frac{1}{2} )</td>
<td>( \frac{1}{2} )</td>
</tr>
</tbody>
</table>
THE DECODING PROBLEM
Decoding Problem: Find an optimal hidden path in an HMM given its emitted string.

- **Input:** A string $x = x_1 \ldots x_n$ emitted by an HMM ($\sum, States, Transition, Emission$).

- **Output:** A path $\pi$ that maximizes the probability $Pr(x, \pi)$ over all possible paths through this HMM.
Building a DAG for the Crooked Casino

HMM diagram

© 2022 Phillip Compeau
Building a DAG for the Crooked Casino

HMM diagram

The source and sink are "silent states" (don’t emit a symbol).
Building a DAG for the Crooked Casino

HMM diagram

© 2022 Phillip Compeau
Building a DAG for the Crooked Casino

HMM diagram

source

sink
Building a DAG for the Crooked Casino

HMM diagram

source

sink
Building a DAG for the Crooked Casino

HMM diagram

© 2022 Phillip Compeau
Building a DAG for the Crooked Casino

HMM diagram

© 2022 Phillip Compeau
Building a DAG for the Crooked Casino

HMM diagram
Building a DAG for the Crooked Casino

This is the Viterbi graph of this HMM.
A DAG for an Arbitrary HMM

Exercise: What is the Viterbi graph of this HMM diagram?
A DAG for an Arbitrary HMM

HMM diagram
A DAG for an Arbitrary HMM

Exercise: What about this HMM diagram? It has “forbidden transitions” between states.
A DAG for an Arbitrary HMM

FIGURE 10.8 (Left) An HMM diagram for an HMM that has four states with some forbidden transitions, such as from A to D and from C to itself. Edges corresponding to forbidden transitions between states are not included in the HMM diagram. (Right) The Viterbi graph for this HMM emitting a string of length 6.

Finding the Most Likely Outcome of an HMM

Dynamic programming allows us to answer questions about HMMs extending beyond the most likely hidden path. For example, we have already computed the probability \( \Pr(p) \) of a hidden path \( p \). But what about computing \( \Pr(x) \), the probability that the HMM will emit a string \( x \)?

Exercise Break: Which outcome is more likely in the crooked casino: “HHTT” or “HTHT”? How would you find the most likely sequence of four coin flips?

Outcome Likelihood Problem:

Find the probability that an HMM emits a given string.

Input: A string \( x = x_1 \ldots x_n \) emitted by an HMM \( (S, \text{States}, \text{Transition}, \text{Emission}) \).

Output: The probability \( \Pr(x) \) that the HMM emits \( x \).

STOP and Think: To solve the Outcome Likelihood Problem, you can make a slight change to the Viterbi recurrence

\[
\text{sk},i = \max_{\text{all states } l} \{ \text{sk},i-1 \cdot \text{WEIGHT}_{i}(l,k) \}
\]

What is the change?

We have already observed that \( \Pr(x) \) is equal to the sum of \( \Pr(x,p) \) over all hidden paths \( p \). However, the number of paths through the Viterbi graph is exponential in the
Alignment Manhattan vs. Decoding Manhattan

**Alignment**
*three valid directions*

**Decoding**
*many valid directions*
The edge from \((l, i-1)\) to \((k, i)\) corresponds to:

- transitioning from state \(l\) to state \(k\) (with probability \(transition_{l,k}\))
- emitting symbol \(x_i\) (with probability \(emission_k(x_i)\))
We weight this edge with $\text{transition}_{l, k} \cdot \text{emission}_k(x_i)$.

The product weight of a path $\pi$ through the Viterbi graph is the product of its edge weights:

$$\prod_{i=1}^{n} \text{transition}_{\pi_{i-1}, \pi_i} \cdot \text{emission}_{\pi_i}(x_i)$$
STOP: How does the product weight differ from $\text{Pr}(x, \pi)$?

$$\prod_{i=1}^{n} \text{transition}_{\pi_{i-1}, \pi_i} \cdot \text{emission}_{\pi_i}(x_i)$$
Answer: It is the same ... so to maximize $\Pr(x, \pi)$, we are looking for a path of maximum product-weight!

$$\prod_{i=1}^{n} \text{transition}_{\pi_{i-1}, \pi_i} \cdot \text{emission}_{\pi_i}(x_i)$$
Finding a “Longest” Path

Maximum Product-Weight Path in a DAG Problem:
Find a path in a DAG of maximum product weight.

• **Input:** A DAG with positive edge weights, along with source and sink nodes.
• **Output:** A path from source to sink of maximum product weight.
Finding a “Longest” Path

Maximum Product-Weight Path in a DAG Problem:
Find a path in a DAG of maximum product weight.

• **Input:** A DAG with positive edge weights, along with source and sink nodes.

• **Output:** A path from source to sink of maximum product weight.

**STOP:** How do we use what we have learned to solve this problem?
Define $s_{k,i}$ as the weight of an optimal path from source to the node $(k, i)$.
Define $s_{k,i}$ as the weight of an optimal path from source to the node $(k, i)$. We have “optimal substructure” because an optimal path from source to $(k, i)$ must be an optimal path from source to $(l, i-1)$ for some node in column $i-1$. 
Answer 1: Dynamic Programming with a Recurrence Relation

Define \( s_{k,i} \) as the weight of an optimal path from \( source \) to the node \((k, i)\).

We have “optimal substructure” because an optimal path from \( source \) to \((k, i)\) must be an optimal path from \( source \) to \((l, i-1)\) for some node in column \( i-1 \).

\[
\begin{align*}
   s_{k,i} &= \max_{\text{all states } l} \left\{ s_{l,i-1} \cdot \text{(weight of edge between nodes} (l, i-1) \text{ and } (k, i)) \right\} \\
   &= \max_{\text{all states } l} \left\{ s_{l,i-1} \cdot \text{WEIGHT}_i(l,k) \right\} \\
   &= \max_{\text{all states } l} \left\{ s_{l,i-1} \cdot \text{transition}_{\pi_{i-1}, \pi_i} \cdot \text{emission}_{\pi_i}(x_i) \right\}
\end{align*}
\]
Two logarithm properties:
1. $\log(x_1 \cdot x_2) = \log(x_1) + \log(x_2)$.
2. It’s increasing; that is, if $x_1 < x_2$, then $\log(x_1) < \log(x_2)$.
Two logarithm properties:
1. \( \log(x_1 \cdot x_2) = \log(x_1) + \log(x_2) \).
2. It’s increasing; that is, if \( x_1 < x_2 \), then \( \log(x_1) < \log(x_2) \).

STOP: How are these properties useful for our purposes?
Two logarithm properties:

1. \( \log(x_1 \cdot x_2) = \log(x_1) + \log(x_2) \).
2. It’s increasing; that is, if \( x_1 < x_2 \), then \( \log(x_1) < \log(x_2) \).

If we take the logarithm of a product of edge weights \( w_1 \ldots w_n \), then by property 1, we obtain a sum of edge weights \( \log(w_1) + \ldots + \log(w_n) \).
Two logarithm properties:
1. \( \log(x_1 \cdot x_2) = \log(x_1) + \log(x_2) \).
2. It’s increasing; that is, if \( x_1 < x_2 \), then \( \log(x_1) < \log(x_2) \).

If we take the logarithm of a product of edge weights \( w_1 \ldots w_n \), then by property 1, we obtain a sum of edge weights \( \log(w_1) + \ldots + \log(w_n) \).

And if the weights correspond to a maximum weight path, this optimality will be preserved by property 2.
Our Problem is “Longest Path in a DAG” in Disguise!

Maximum Product-Weight Path in a DAG Problem: 
Find a path in a DAG of maximum product weight.

• **Input:** A DAG with positive edge weights, along with source and sink nodes.

• **Output:** A path from source to sink of maximum product weight.
PROFILE HMMS FOR SEQUENCE ALIGNMENT
Once we have a collection of known protein alignments ("families"), we need to be able to identify which family a new protein belongs to. That is, add a new string into an existing alignment.
Once we have a collection of known protein alignments ("families"), we need to be able to identify which family a new protein belongs to. That is, add a new string into an existing alignment.

This sets up as an HMM problem, since when adding a new string to an alignment, we have:

- a decision to make at each step (align? Gap symbol?)
- We’re looking for a "path" (decisions) of sorts that "makes the most sense".
From an Alignment to a Profile

Seed alignment: remove columns if the fraction of space symbols ("-") exceeds a threshold $\theta$. 

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>C</td>
<td>D</td>
<td>E</td>
<td>F</td>
<td>A</td>
<td>C</td>
<td>A</td>
<td>D</td>
</tr>
<tr>
<td>A</td>
<td>F</td>
<td>D</td>
<td>A</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>C</td>
<td>C</td>
</tr>
<tr>
<td>A</td>
<td>-</td>
<td>-</td>
<td>E</td>
<td>F</td>
<td>D</td>
<td>-</td>
<td>F</td>
<td>D</td>
</tr>
<tr>
<td>A</td>
<td>C</td>
<td>A</td>
<td>E</td>
<td>F</td>
<td>-</td>
<td>-</td>
<td>A</td>
<td>-</td>
</tr>
<tr>
<td>A</td>
<td>D</td>
<td>D</td>
<td>E</td>
<td>F</td>
<td>A</td>
<td>A</td>
<td>A</td>
<td>D</td>
</tr>
</tbody>
</table>

FIGURE 10.9: A $5 \times 10$ multiple alignment (first panel), its $5 \times 8$ seed alignment (second panel), the profile matrix (third panel), and the diagram of a simple HMM that models this profile (fourth panel). The seed alignment is obtained from the original alignment by ignoring poorly conserved columns (shaded gray); in this case, we ignore columns for which the fraction of space symbols is greater than or equal to the column removal threshold $\theta = 0.35$. To better illustrate the relationship between the alignment and its seed alignment, we have separated the first five columns in the seed alignment from its last three columns and numbered these columns above the original alignment. The match states $\text{MATCH}(i)$ are abbreviated as $M_i$. The HMM only has one possible path; it is initially in state $\text{MATCH}(1)$, the transition probability from state $\text{MATCH}(i)$ to state $\text{MATCH}(i+1)$ is equal to 1 for all $i$, and all other transitions are forbidden. Emission probabilities are equal to frequencies in the profile, e.g., emission probabilities for $M_2$ are 0 for $A$, $2/4$ for $C$, $1/4$ for $D$, 0 for $E$, and $1/4$ for $F$. When the HMM enters state $\text{MATCH}(i)$, it emits symbol $x_i$ with probability equal to the frequency of this symbol in the $i$-th column of the profile. The HMM then moves into state $\text{MATCH}(i+1)$ with transition probability equal to 1.
### From an Alignment to a Profile

#### Alignment

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>C</td>
<td>D</td>
<td>E</td>
<td>F</td>
<td>A</td>
<td>C</td>
<td>A</td>
<td>D</td>
</tr>
<tr>
<td>A</td>
<td>F</td>
<td>D</td>
<td>A</td>
<td>–</td>
<td>–</td>
<td>C</td>
<td>C</td>
<td>F</td>
</tr>
<tr>
<td>A</td>
<td>–</td>
<td>–</td>
<td>E</td>
<td>F</td>
<td>D</td>
<td>–</td>
<td>F</td>
<td>D</td>
</tr>
<tr>
<td>A</td>
<td>C</td>
<td>A</td>
<td>E</td>
<td>F</td>
<td>–</td>
<td>A</td>
<td>–</td>
<td>C</td>
</tr>
<tr>
<td>A</td>
<td>D</td>
<td>D</td>
<td>E</td>
<td>F</td>
<td>A</td>
<td>A</td>
<td>A</td>
<td>D</td>
</tr>
</tbody>
</table>

#### Alignment*

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>C</td>
<td>D</td>
<td>E</td>
<td>F</td>
<td>A</td>
<td>D</td>
<td>F</td>
<td></td>
</tr>
<tr>
<td>A</td>
<td>F</td>
<td>D</td>
<td>A</td>
<td>–</td>
<td>C</td>
<td>C</td>
<td>F</td>
<td></td>
</tr>
<tr>
<td>A</td>
<td>–</td>
<td>–</td>
<td>E</td>
<td>F</td>
<td>F</td>
<td>D</td>
<td>C</td>
<td></td>
</tr>
<tr>
<td>A</td>
<td>C</td>
<td>A</td>
<td>E</td>
<td>F</td>
<td>A</td>
<td>–</td>
<td>C</td>
<td></td>
</tr>
<tr>
<td>A</td>
<td>D</td>
<td>D</td>
<td>E</td>
<td>F</td>
<td>A</td>
<td>D</td>
<td>F</td>
<td></td>
</tr>
</tbody>
</table>
From an Alignment to a Profile

<table>
<thead>
<tr>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>C</td>
<td>D</td>
<td>E</td>
<td>F</td>
<td>A</td>
<td>C</td>
<td>A</td>
</tr>
<tr>
<td>A</td>
<td>F</td>
<td>D</td>
<td>A</td>
<td>-</td>
<td>–</td>
<td>C</td>
<td>C</td>
</tr>
<tr>
<td>A</td>
<td>–</td>
<td>–</td>
<td>E</td>
<td>F</td>
<td>D</td>
<td>–</td>
<td>F</td>
</tr>
<tr>
<td>A</td>
<td>C</td>
<td>A</td>
<td>E</td>
<td>F</td>
<td>–</td>
<td>–</td>
<td>A</td>
</tr>
<tr>
<td>A</td>
<td>D</td>
<td>D</td>
<td>E</td>
<td>F</td>
<td>A</td>
<td>A</td>
<td>A</td>
</tr>
</tbody>
</table>

**Alignment**

<table>
<thead>
<tr>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>C</td>
<td>D</td>
<td>E</td>
<td>F</td>
<td>A</td>
<td>D</td>
<td>F</td>
</tr>
<tr>
<td>A</td>
<td>F</td>
<td>D</td>
<td>A</td>
<td>-</td>
<td>C</td>
<td>C</td>
<td>F</td>
</tr>
<tr>
<td>A</td>
<td>–</td>
<td>–</td>
<td>E</td>
<td>F</td>
<td>F</td>
<td>D</td>
<td>C</td>
</tr>
<tr>
<td>A</td>
<td>C</td>
<td>A</td>
<td>E</td>
<td>F</td>
<td>A</td>
<td>–</td>
<td>C</td>
</tr>
<tr>
<td>A</td>
<td>D</td>
<td>D</td>
<td>E</td>
<td>F</td>
<td>A</td>
<td>D</td>
<td>F</td>
</tr>
</tbody>
</table>

**Alignment**

<table>
<thead>
<tr>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>1/4</td>
<td>1/5</td>
<td>0</td>
<td>3/5</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>2/4</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1/5</td>
<td>1/4</td>
<td>2/5</td>
</tr>
<tr>
<td>0</td>
<td>1/4</td>
<td>3/4</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>3/4</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>4/5</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1/4</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1/5</td>
<td>0</td>
<td>3/5</td>
</tr>
</tbody>
</table>

**Profile (Alignment)**

© 2022 Phillip Compeau
### From an Alignment to a Profile

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>C</td>
<td>D</td>
<td>E</td>
<td>F</td>
<td>A</td>
<td>C</td>
<td>A</td>
<td>D</td>
</tr>
<tr>
<td>A</td>
<td>F</td>
<td>D</td>
<td>A</td>
<td>--</td>
<td>--</td>
<td>C</td>
<td>C</td>
<td>F</td>
</tr>
<tr>
<td>A</td>
<td>--</td>
<td>--</td>
<td>E</td>
<td>F</td>
<td>D</td>
<td>--</td>
<td>F</td>
<td>D</td>
</tr>
<tr>
<td>A</td>
<td>C</td>
<td>A</td>
<td>E</td>
<td>F</td>
<td>--</td>
<td>A</td>
<td>--</td>
<td>C</td>
</tr>
<tr>
<td>A</td>
<td>D</td>
<td>D</td>
<td>E</td>
<td>F</td>
<td>A</td>
<td>A</td>
<td>A</td>
<td>D</td>
</tr>
<tr>
<td></td>
<td>A</td>
<td>C</td>
<td>D</td>
<td>E</td>
<td>F</td>
<td>A</td>
<td>D</td>
<td>F</td>
</tr>
<tr>
<td>Alignment*</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>A</td>
<td>F</td>
<td>D</td>
<td>A</td>
<td>--</td>
<td>--</td>
<td>C</td>
<td>C</td>
<td>F</td>
</tr>
<tr>
<td>A</td>
<td>--</td>
<td>--</td>
<td>E</td>
<td>F</td>
<td>F</td>
<td>D</td>
<td>C</td>
<td></td>
</tr>
<tr>
<td>A</td>
<td>C</td>
<td>A</td>
<td>E</td>
<td>F</td>
<td>A</td>
<td>--</td>
<td>C</td>
<td></td>
</tr>
<tr>
<td>A</td>
<td>D</td>
<td>D</td>
<td>E</td>
<td>F</td>
<td>A</td>
<td>D</td>
<td>F</td>
<td></td>
</tr>
<tr>
<td></td>
<td>A</td>
<td>C</td>
<td>D</td>
<td>E</td>
<td>F</td>
<td>A</td>
<td>D</td>
<td>F</td>
</tr>
<tr>
<td>Profile(Alignment*)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>A</td>
<td>1</td>
<td>0</td>
<td>1/4</td>
<td>1/5</td>
<td>0</td>
<td>3/5</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>C</td>
<td>0</td>
<td>2/4</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1/5</td>
<td>1/4</td>
<td>2/5</td>
</tr>
<tr>
<td>D</td>
<td>0</td>
<td>1/4</td>
<td>3/4</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>3/4</td>
<td>0</td>
</tr>
<tr>
<td>E</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>4/5</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>F</td>
<td>0</td>
<td>1/4</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1/5</td>
<td>0</td>
<td>3/5</td>
</tr>
<tr>
<td>HMM diagram</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

© 2022 Phillip Compeau
STOP: How do we model insertions?
Toward a Profile HMM: Insertions

A   F   D   D   A   F   F   F   D   F
Toward a Profile HMM: Insertions

A F D D A F F F D F

© 2022 Phillip Compeau
Toward a Profile HMM: Insertions
Toward a Profile HMM: Insertions
Toward a Profile HMM: Insertions

STOP: How do we model deletions?
Toward a Profile HMM: Deletions
Toward a Profile HMM: Deletions

© 2022 Phillip Compeau
Toward a Profile HMM: Deletions

STOP: What issues do you see with this approach?
Answer: Just like with affine alignment, we can have fewer edges if we create separate “deletion states”.

© 2022 Phillip Compeau
Toward a Profile HMM: Deletions

© 2022 Phillip Compeau
STOP: Are any edges still missing in this HMM diagram?
Adding Edges Between Insertion/Deletion States
The Profile HMM is Ready to Use!

This is the HMM diagram of the profile HMM of a seed alignment.
Summarizing a Profile HMM

Σ: an alphabet of emitted symbols

States: a set of hidden states

Transition = (transition_{l,k}): a |States| \times |States| matrix of transition probabilities
(of changing from state \(l\) to state \(k\))

Emission = (emission_{k}(b)): a |States| \times |Σ| matrix of emission probabilities
(emitting symbol \(b\) when HMM is in state \(k\))

Amino acids
Start, end, match, insertion, and deletion states

It is not yet clear what the transition and emission probabilities should be!
Hidden Paths Through Profile HMM

```
<table>
<thead>
<tr>
<th>A</th>
<th>C</th>
<th>D</th>
<th>E</th>
<th>F</th>
<th>A</th>
<th>C</th>
<th>A</th>
<th>D</th>
<th>F</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>F</td>
<td>D</td>
<td>A</td>
<td></td>
<td></td>
<td>C</td>
<td>C</td>
<td>F</td>
<td></td>
</tr>
<tr>
<td>A</td>
<td></td>
<td>E</td>
<td>F</td>
<td>D</td>
<td>F</td>
<td>D</td>
<td>C</td>
<td></td>
<td></td>
</tr>
<tr>
<td>A</td>
<td>C</td>
<td>A</td>
<td>E</td>
<td>F</td>
<td></td>
<td>A</td>
<td></td>
<td>C</td>
<td></td>
</tr>
<tr>
<td>A</td>
<td>D</td>
<td>D</td>
<td>E</td>
<td>F</td>
<td>A</td>
<td>A</td>
<td>A</td>
<td>D</td>
<td>F</td>
</tr>
</tbody>
</table>
```

Diagram showing states and transitions.
Hidden Paths Through Profile HMM

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>C</th>
<th>D</th>
<th>E</th>
<th>F</th>
<th>A</th>
<th>C</th>
<th>A</th>
<th>D</th>
<th>F</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>F</td>
<td>D</td>
<td>A</td>
<td>–</td>
<td>–</td>
<td>C</td>
<td>C</td>
<td>F</td>
<td></td>
<td></td>
</tr>
<tr>
<td>A</td>
<td>–</td>
<td>–</td>
<td>E</td>
<td>F</td>
<td>D</td>
<td>–</td>
<td>F</td>
<td>D</td>
<td>C</td>
<td></td>
</tr>
<tr>
<td>A</td>
<td>C</td>
<td>A</td>
<td>E</td>
<td>F</td>
<td>–</td>
<td>–</td>
<td>A</td>
<td>–</td>
<td>C</td>
<td></td>
</tr>
<tr>
<td>A</td>
<td>D</td>
<td>D</td>
<td>E</td>
<td>F</td>
<td>A</td>
<td>A</td>
<td>A</td>
<td>D</td>
<td>F</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>l₀</th>
<th>l₁</th>
<th>l₂</th>
<th>l₃</th>
<th>l₄</th>
<th>l₅</th>
<th>l₆</th>
<th>l₇</th>
<th>l₈</th>
<th>E</th>
</tr>
</thead>
<tbody>
<tr>
<td>M₁</td>
<td>I₀</td>
<td>M₂</td>
<td>M₃</td>
<td>M₄</td>
<td>M₅</td>
<td>M₆</td>
<td>M₇</td>
<td>M₈</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>A</td>
<td>C</td>
<td>D</td>
<td>E</td>
<td>F</td>
<td>A</td>
<td>C</td>
<td>A</td>
<td>D</td>
<td>F</td>
</tr>
<tr>
<td>M₂</td>
<td>I₁</td>
<td>M₃</td>
<td>M₄</td>
<td>M₅</td>
<td>M₆</td>
<td>M₇</td>
<td>M₈</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>A</td>
<td>C</td>
<td>D</td>
<td>E</td>
<td>F</td>
<td>A</td>
<td>C</td>
<td>A</td>
<td>D</td>
<td>F</td>
</tr>
<tr>
<td>M₃</td>
<td>I₂</td>
<td>M₄</td>
<td>M₅</td>
<td>M₆</td>
<td>M₇</td>
<td>M₈</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>A</td>
<td>F</td>
<td>D</td>
<td>A</td>
<td>(–)</td>
<td>C</td>
<td>C</td>
<td>F</td>
<td></td>
<td></td>
</tr>
<tr>
<td>M₄</td>
<td>I₃</td>
<td>M₅</td>
<td>M₆</td>
<td>M₇</td>
<td>M₈</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>A</td>
<td>F</td>
<td>D</td>
<td>A</td>
<td>(–)</td>
<td>C</td>
<td>C</td>
<td>F</td>
<td></td>
<td></td>
</tr>
<tr>
<td>M₅</td>
<td>I₄</td>
<td>M₆</td>
<td>M₇</td>
<td>M₈</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>A</td>
<td>(–)</td>
<td>(–)</td>
<td>(–)</td>
<td>E</td>
<td>F</td>
<td>D</td>
<td>F</td>
<td>D</td>
<td>C</td>
</tr>
</tbody>
</table>

Diagram showing transitions between states and emissions.
Hidden Paths Through Profile HMM

Note: this is a hidden path in an HMM diagram (not in a Viterbi graph).
4 transitions from $M_5$:

1 + 1 + 1 = 3 into $I_5$

1 into $M_6$

0 into $D_6$
Transition Probabilities of Profile HMM

4 transitions from $M_5$:

1 + 1 + 1 = 3 into $I_5$
1 into $M_6$
0 into $D_6$

$transition_{\text{Match}(5),\text{Insertion}(5)} = \frac{3}{4}$
$transition_{\text{Match}(5),\text{Match}(6)} = \frac{1}{4}$
$transition_{\text{Match}(5),\text{Deletion}(6)} = 0$
Transition Probabilities of Profile HMM

Symbols emitted from $M_2$: $C, F, C, D$

- $\text{emission}_{\text{Match}(2)}(A) = 0$
- $\text{emission}_{\text{Match}(2)}(C) = 2/4$
- $\text{emission}_{\text{Match}(2)}(D) = 1/4$
- $\text{emission}_{\text{Match}(2)}(E) = 0$
- $\text{emission}_{\text{Match}(2)}(F) = 1/4$
Forbidden Transitions

|   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
|   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
| S | I₀ | M₁ | D₁ | I₁ | M₂ | D₂ | I₂ | M₃ | D₃ | I₃ | M₄ | D₄ | I₄ | M₅ | D₅ | I₅ | M₆ | D₆ | I₆ | M₇ | D₇ | I₇ | M₈ | D₈ | I₈ | E |
| S | 1 |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
| I₀ |   | .8 | .2 |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
| M₁ |   |   | 1 |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
| D₁ |   |   |   | 1 |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
| I₁ |   |   |   |   | .8 | .2 |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
| M₂ |   |   |   |   |   | 1 |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
| D₂ |   |   |   |   |   |   | 1 |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
| I₂ |   |   |   |   |   |   |   | .25 | .75 |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
| M₃ |   |   |   |   |   |   |   |   | .33 | .67 |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
| D₃ |   |   |   |   |   |   |   |   |   | 1 |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
| I₃ |   |   |   |   |   |   |   |   |   |   | 1 |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
| M₄ |   |   |   |   |   |   |   |   |   |   |   | 1 |   |   |   |   |   |   |   |   |   |   |   |   |   |
| D₄ |   |   |   |   |   |   |   |   |   |   |   |   | 1 |   |   |   |   |   |   |   |   |   |   |   |   |
| I₄ |   |   |   |   |   |   |   |   |   |   |   |   |   | 1 |   |   |   |   |   |   |   |   |   |   |   |
| M₅ |   |   |   |   |   |   |   |   |   |   |   |   |   |   | .25 | .75 |   |   |   |   |   |   |   |   |   |
| D₅ |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   | .33 | .67 |   |   |   |   |   |   |   |   |
| I₅ |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   | 1 |   |   |   |   |   |   |   |   |   |
| M₆ |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   | 1 |   |   |   |   |   |   |   |   |
| D₆ |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   | .8 | .2 |   |   |   |   |   |   |
| I₆ |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   | 1 |   |   |   |   |   |   |
| M₇ |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   | 1 |   |   |   |   |   |
| D₇ |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   | 1 |   |   |   |   |
| I₇ |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   | 1 |   |   |   |
| M₈ |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   | 1 |   |   |
| D₈ |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   | 1 |   |
| I₈ |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   | 1 |
| E  |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |

Gray cells: edges in the HMM diagram.

Clear cells: forbidden transitions.

Empty gray cells: equal to zero.
Having zero weights will cause issues for two reasons:
1. log(0) is undefined.
2. One weight being zero shouldn’t disqualify a path.
Forbidden Transitions

STOP: What should we do?
## Forbidden Transitions

**STOP:** What should we do?

**Answer:** Add pseudocounts (!) to the zero values and normalize.

<table>
<thead>
<tr>
<th></th>
<th>S</th>
<th>l₀</th>
<th>M₁</th>
<th>D₁</th>
<th>l₁</th>
<th>M₂</th>
<th>D₂</th>
<th>l₂</th>
<th>M₃</th>
<th>D₃</th>
<th>l₃</th>
<th>M₄</th>
<th>D₄</th>
<th>l₄</th>
<th>M₅</th>
<th>D₅</th>
<th>l₅</th>
<th>M₆</th>
<th>D₆</th>
<th>l₆</th>
<th>M₇</th>
<th>D₇</th>
<th>l₇</th>
<th>M₈</th>
<th>D₈</th>
<th>l₈</th>
<th>E</th>
</tr>
</thead>
<tbody>
<tr>
<td>S</td>
<td></td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>STOP: What should we do?</td>
</tr>
<tr>
<td>l₀</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Answer: Add pseudocounts (!) to the zero values and normalize.</td>
</tr>
</tbody>
</table>
CLASSIFYING PROTEINS WITH PROFILE HMMS
Aligning a Protein Against a Profile HMM

**Alignment**

```
S → D → D → D → D → D → D → D → D → E
```

**Protein**

```
ACAFDEAEF
```

**Exercise Break:**

Construct a profile HMM for the HIV sequences shown in Figure 10.1 with $q = 0.35$.

Classifying proteins with profile HMMs

Given a protein family, represented by **Alignment**, we can now return to the problem of deciding whether a newly sequenced protein, represented by **Text**, belongs to the family.

We first form HMM (**Alignment**, $q$) for some parameter $q$. As shown in Figure 10.17, a hidden path through HMM (**Alignment**, $q$) corresponds to a sequence of match, insertion, and deletion states for aligning **Text** against **Alignment**.

A C -- D E F AC A D F
A F -- D A -- -- C C F
A -- -- -- E F D-- F D C
A C -- A E F -- A -- C
A D -- D E F AA A D F

**FIGURE 10.17** (Top) A path through HMM (**Alignment**, 0.35) for the multiple alignment from Figure 10.9 and the emitted string **Text** = ACAFDEAF. (Bottom) The emitted symbols correspond to aligning **Text** against **Alignment**. Specifically, the first two symbols are emitted from two match states and belong in the first two positions of the alignment. The next two symbols are emitted from an insertion state and belong in columns of their own (shown in pink). The space symbols in the seventh and eleventh columns above correspond to deletion states; these symbols are not emitted by the HMM. The space symbols in the gray columns do not correspond to any states and are passed over. The non-shaded columns form an augmented 6 x 8 seed alignment for comparison against newly sequenced proteins.
Aligning a Protein Against a Profile

HMM

Apply Viterbi algorithm to find optimal hidden path.

© 2022 Phillip Compeau
Profile HMM diagram
STOP: How many rows and columns does the Viterbi graph of this profile HMM have?
Profile HMM diagram

Viterbi graph of profile HMM:
#columns=
#visited states
Profile HMM diagram

Viterbi graph of profile HMM:
#columns = #visited states

STOP: What is wrong with this Viterbi graph?
Profile HMM diagram

Viterbi graph of profile HMM:
#columns = #visited states

By definition,
#columns = #emitted symbols
Profile HMM diagram

Nearly correct Viterbi graph of profile HMM:

Vertical edges enter “silent” deletion states
Profile HMM diagram

Correct Viterbi graph of profile HMM:

Adding 0-th column that contains only silent states
Sequence Alignment with Profile HMM Problem: Align a new sequence to a family of aligned sequences using a profile HMM.

- **Input:** A multiple alignment Alignment, a string Text, a threshold $\theta$ (maximum fraction of insertions per column), and a pseudocount $\sigma$.
- **Output:** An optimal hidden path emitting Text in the profile HMM $HMM(\text{Alignment}, \theta, \sigma)$. 
Have I Wasted Your Time?
Have I Wasted Your Time?

\[
\begin{align*}
\text{s}_{M(j),i} &= \max \left\{ \text{s}_{I(j-1),i-1} \cdot \text{weight}(I(j-1),M(j),i-1), \right. \\
& \quad \left. \text{s}_{D(j-1),i-1} \cdot \text{weight}(D(j-1),M(j),i-1), \right. \\
& \quad \left. \text{s}_{M(j-1),i-1} \cdot \text{weight}(M(j-1),M(j),i-1) \right\} \\
\text{s}_{i,j} &= \max \left\{ \text{s}_{i-1,j} + \text{score}(v_i,-), \right. \\
& \quad \left. \text{s}_{i,j-1} + \text{score}(-,w_j), \right. \\
& \quad \left. \text{s}_{i-1,j-1} + \text{score}(v_i,w_j) \right\}
\end{align*}
\]
The choice of alignment path is now based on varying transition and emission probabilities.
I Hope Not! 😊

Individual scoring parameters for each edge in the alignment graph capture subtle similarities that evade traditional alignments.
Citations

Biological sequence analysis

HMMER web server: interactive sequence similarity searching
RD Finn, J Clements, SR Eddy - Nucleic acids research, 2011 - academic.oup.com
… HMMER is a software suite for protein sequence similarity searches using probabilistic methods. Previously, HMMER has mainly been … A HMMER web server ( http://hmmr.janelia.org ) has been designed and implemented such that most protein database searches return …
☆ Save  Cite  Cited by 3707  Related articles  All 17 versions