Exercise 7: Motifs and alignment

Let \( P \) be a PWM of length \( l \) with consensus sequence \( c \).
Let \( s \) be a DNA sequence of length \( n > l \).

1. Create an algorithm that finds the best ungapped match of \( c \) in \( s \).
2. Generalise your algorithm to find the best ungapped match of \( P \) in \( s \).

### Naïve string search

**PWM**

\[
\begin{array}{cccccccc}
A & 1 & 0 & 6 & 0 & 0 & 1 & 0 \\
C & 2 & 7 & 0 & 0 & 1 & 0 & 0 \\
G & 4 & 0 & 1 & 1 & 7 & 0 & 7 \\
T & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
\end{array}
\]

**Consensus**

\( GCACGGG \)

Input string:

\[ s = acgcggccaatcagaggccacG\text{CACG}A\text{GG}\text{gtgaggcaagtccgcctctcgggagg} \]

Input string:

\[ s = acgcggccaatcagaggccacG\text{CACG}A\text{GG}\text{gtgaggcaagtccgcctctcgggagg} \]

### Extending NaïveStringSearch to find position with most matches

**StringSearchMostMatch**

\[
\text{bestIndex} = 0 \\
\text{bestCount} = 0
\]

for \( i \) in range(len(s)):

\[
\text{count} = 0 \\
\text{for} \ j \ 	ext{in range(len(p))}: \\
\quad \text{if} \ s[i + j] == p[j]: \\
\quad \quad \text{count} = \text{count} + 1
\]

if count > bestCount:

\[
\text{bestIndex} = i \\
\text{bestCount} = \text{count}
\]

return bestIndex

### Scanning with PWM is straightforward extension

**PWMscan**

\[
\begin{array}{cccccccc}
A & 1 & 0 & 6 & 0 & 0 & 1 & 0 \\
C & 2 & 7 & 0 & 0 & 1 & 0 & 0 \\
G & 4 & 0 & 1 & 1 & 7 & 0 & 7 \\
T & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
\end{array}
\]

**PWM**

\[
\begin{array}{cccccccc}
A & 1 & 0 & 6 & 0 & 0 & 1 & 0 \\
C & 2 & 7 & 0 & 0 & 1 & 0 & 0 \\
G & 4 & 0 & 1 & 1 & 7 & 0 & 7 \\
T & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
\end{array}
\]

### Introducing gaps

3. Create an algorithm that finds the best gapped match (alignment) of \( c \) in \( s \).
4. Generalise your algorithm to find the best ungapped match of \( P \) in \( s \).
What if gaps are allowed?

**PWM:**

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>C</th>
<th>G</th>
<th>T</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>6</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>6</td>
</tr>
<tr>
<td>3</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

Gap penalty = 1

Input string:
acgppcacaatcaagggcnaaGCACGTGGgtggpcaagtctcctgppggggg

Scan:
- GCACGTGG = 44
- GCACGTGG = 45

Consider first alignment with consensus

String 1: GCACGTGG
String 2: ..acGCACGAGGtg..

```
Gap penalty = 1
PWM:
A 1 0 6 0 0 1 0 0
C 2 7 0 6 0 1 0 0
G 4 0 1 1 7 0 7 6
T 0 0 0 0 0 5 0 1
```

What kind of alignment is this? Local? Global?

Recap: global alignment

- **Initialization**
  - **Insertion cost:** -1
- **Recurrence**
  - $PA(S_i,R_j) = \max_{i,j} \{ PA(S_{i-1},R_{j-1}) + \delta(s_i,r_j), PA(S_{i-1},R_j) - 1, PA(S_i,R_{j-1}) - 1 \}$

Recap: local alignment

- **Initialization**
  - **Insertion cost:** -1
- **Recurrence**
  - $PA(S_i,C_j) = \max_{i,j} \{ PA(S_{i-1},C_{j-1}) + \delta(s_i,c_j), PA(S_{i-1},C_j) - 1, PA(S_i,C_{j-1}) - 1 \}$

This case: map the whole of c locally into s

- **Initialization**
  - **Insertion cost:** -1
- **Recurrence**
  - $PA(S_i,C_j) = \max_{i,j} \{ PA(S_{i-1},C_{j-1}) + \delta(s_i,c_j), PA(S_{i-1},C_j) - 1, PA(S_i,C_{j-1}) - 1 \}$

Variant: gaps allowed only in c

- **Initialization**
  - **Insertion cost:** -1
- **Recurrence**
  - $PA(S_i,C_j) = \max_{i,j} \{ PA(S_{i-1},C_{j-1}) + \delta(s_i,c_j), PA(S_{i-1},C_j) - 1, PA(S_i,C_{j-1}) - 1 \}$
Generalisation to PWM

- **Initialization**
  - Insertion cost: -1
- **Recurrence**

\[
P(A(S_i, C_j)) = \max \begin{cases} 
P(A(S_{i-1}, C_{j-1})) + \text{PWM}[j][s], 
P(A(S_{i-1}, C_j)) - 1, 
P(A(S_{i}, C_{j-1})) - 1 \end{cases}
\]

Comparing two PWMs

5. Create an algorithm that calculates the similarity (or "edit distance") of two PWMs.

Similarity of two PWMs?

- **Similarity of two PWM columns**

<table>
<thead>
<tr>
<th>C1</th>
<th>C2</th>
<th>Diff</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>C</td>
<td>2</td>
<td>4</td>
</tr>
<tr>
<td>G</td>
<td>4</td>
<td>6</td>
</tr>
<tr>
<td>T</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Total:</td>
<td>8</td>
<td></td>
</tr>
</tbody>
</table>

\[
d(P_{1,i}, P_{2,j}) = \sum_{a \in \text{DNA}} |P_{1,i,a} - P_{2,j,a}| 
\]

Aligning two PWMs

- **Initialization**
  - Insertion cost: $k$
- **Recurrence**

\[
d(P_{1,i}, P_{2,j}) = \sum_{a \in \text{DNA}} |P_{1,i,a} - P_{2,j,a}| 
\]