Введение

Задача выравнивания биологических последовательностей является одной из наиболее часто решаемых в биоинформатике. Методы выравнивания последовательностей основаны на размещении похожих участков двух или более последовательностей ДНК, РНК или белков друг под другом.

Если в последовательностях существуют совпадающие участки, то, например, можно предположить, что две последовательности ДНК относятся к одному и тому же виду гена, несмотря на то, что подпоследовательности нуклеотидов не полностью совпадают.

Обычно, когда мы выравниваем последовательности, то стремимся определить наилучшее из большего числа возможных сравнений. Выбор наилучшего происходит по количественной оценки выравнивания.

Расстояние Хэмминга

Одним из примеров количественной оценки выравнивания является расстояние Хэмминга - число позиций, в которых различаются соответствующие символы двух строк одинаковой длины.

Реализуйте функцию **hamming**, которая принимает две строки и вычисляет расстояние Хэмминга. Гарантируется, что на вход поступают строки одинаковой длины.

```
>>> seq1 = "TGCTCGGACTACACGCATTATTGCAG"
>>> seq2 = "TATTATGTATAGAGTTACACGGGCAT"
>>> hamming(seq1, seq2)
16
```

Алгоритм Нидлмана-Вунша

Инициализация

Реализуйте функцию **init**, которая по m, n и ошибке σ строит матрицу с $m+1$ строкой и $n+1$ столбцом:

$$A_{m,n} = \begin{pmatrix} 0 & -\sigma & \cdots & -n\sigma \\ -\sigma & 0 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ -m\sigma & 0 & \cdots & 0 \end{pmatrix}$$

Вычисление элемента

Пусть a и b - две последовательности, которые хотим выравнять. Теперь имея пустую матрицу, нужно научиться её заполнять. Вычисление очередного элемента матрицы:

$$A_{i\ j} = max \begin{cases} A_{i-1\ j-1} + s(a_i, b_j) & \text{Match / Mismatch} \\ A_{i\ j-1} - \sigma & \text{Insertion} \\ A_{i-1\ j} - \sigma & \text{Deletion} \end{cases}$$

$$s(a_i, b_j) = \begin{cases} \mu & \text{if } a_i = b_j \text{ Match} \\ \epsilon & \text{if } a_i \neq b_j \text{ Mismatch} \end{cases}$$

Реализуйте функцию **calc_score**, которая принимает на вход 7 параметров - up ($A_{i-1\ j}$), left ($A_{i\ j-1}$), middle ($A_{i-1\ j-1}$), matched ($a_i = b_j$), gap_penalty (σ), match (μ), mismatch (ϵ) и вычисляет $A_{i\ j}$.

```
>>> calc_score(93, 83, 77, 1, 3, 3, -1)
90.0
>>> calc_score(84, 50, 50, 0, 3, 3, -2)
81.0
```

Заполнение матрицы

Мы научились инициализировать матрицу, считать отдельный элемент в матрице выравнивания. Теперь нужно посчитать все элементы матрицы выравнивания и вывести элемент $A_{m,n}$, где m, n - длины входных последовательностей.

Реализуйте функцию **align**, которая на вход принимает две последовательности ДНК, штраф за пропуск (σ) , баллы за совпадение (μ) и несовпадение (ϵ) и возвращает $A_{m,n}$

```
>>> seq1 = "TGTTACCCATTACATTG"
>>> seq2 = "TTTCCAAGGCATCTT"
>>> align(seq1, seq2, 4, 4, -1)
20.0
```

Построение выравнивания

Теперь имея матрицу выравнивания построим самое выравнивание.

Реализуйте функцию **get_alignment**, которая по двум последовательностям, матрице выравнивания, штрафа за пропуски, бонусам за совпадение/несовпадение нуклеотидов строит выравнивание. Функцию **get_alignment** нужно добавить в конце функции **align**.

```
>>> seq1 = "AGTGTCGGCT"
>>> seq2 = "ACTTCTACCCCAGC"
>>> align(seq1, seq2, 1.399, 2.2168, -4.4499)
AC-T-TCTACCCCAG-C-
A-GTGTC-------GGCT
-3.4872
```