

Algoritmos para Análise de Sequências Biológicas

Ficha 5

Objetivo

- Scores de alinhamentos globais e locais
- Reconstrução de alinhamentos

ScoreMatriz

- 1 Implemente a classe `ScoreMatriz` que implementa uma matriz de scoring
- 2 O construtor deve receber
 - ▶ `match_pairs` e opcionalmente os scores para dois caracteres iguais e diferentes, do alfabeto e do custo do espaçamento
 - ▶ `blosum62`
 - ▶ `file` seguido do nome de um ficheiro
- 3 Implemente o método `getScore` que recebe dois caracteres e devolve o score correspondente

Esqueleto que pode utilizar

```
class ScoreMatrix:
    def __init__(self, **kwargs):
        def defaulting(kwarg, key, default_value):
            if key not in kwarg:
                print(f"Using default value for {key}: {default_value}")
                return default_value
            else:
                return kwarg[key]
        if 'file' in kwargs:
            print("Reading from", kwargs["file"])
        elif 'blosum62' in kwargs:
            print("Using the blosum62 matrix")
        elif 'match_pairs' in kwargs:
            eqs = defaulting(kwarg, "eqs", 1)
            diffs = defaulting(kwarg, "diffs", 0)
            alphabet = defaulting(kwarg, "alphabet", "ACGT")
        g = defaulting(kwarg, "g", -8)
```

AlignSeq

- 1 Implemente a classe `AlignSeq` que implementa o alinhamento entre duas sequências
- 2 O construtor deve receber duas sequências, a matriz de scoring, por omissão BLOSUM62 e o keyword `global` ou `local`, sendo por omissão utilizado o alinhamento global
- 3 Implemente o método `score` que devolve o score do alinhamento
- 4 Implemente o método `alignment` que devolve a lista de tuplos correspondendo aos melhores alinhamentos
- 5 Implemente corretamente o método `__str__` que é invocado ao imprimir o alinhamento
- 6 Implemente corretamente o método `__repr__` que é invocado no REPL e que usa o método anterior

Sugestões

- Crie atributos para armazenar as matrizes de score e trace para além de armazenar as sequências e a matriz de scoring
- Crie métodos que imprimam de forma legível as matrizes para ajudar no debug do código
- Inicialmente crie uma versão que não se preocupe com empates

Exemplos de utilização

```
>>> sm = ScoreMatrix(file="blosum80.txt")
Reading from blosum80.txt
>>> sm = ScoreMatrix(blosum62 = True)
Using the blosum62 matrix
>>> sm = ScoreMatrix(match_pairs = True)
Using default value for eqs: 1
Using default value for diffs: 0
Using default value for alphabet: ACGT
Using default value for g: -8
>>> sm = ScoreMatrix(match_pairs = True, diffs = -1, eqs = 2)
Using default value for alphabet: ACGT
Using default value for g: -8
>>> sm.getScore('A','G')
-1
```

Exemplos de utilização

```
>>> AlignSeq("ATGAAGGT", "AGAGAGGC",  
             ScoreMatrix(match_pairs = True), eqs = 2, diffs = 0,  
             g = -1, alphabet = "ACGT")
```

Melhor score do alinhamento otimo global: 10

ATGA-AGGT

A-GAGAGGC

```
>>> AlignSeq("LGPSGCASGIWTKSA", "TGPSGGSRIWTKSG",  
             ScoreMatrix(blosum62 = True), g = -8)
```

Melhor score do alinhamento otimo global: 45

LGPSGCASGIWTKSA

TGPSG-GSRIWTKSG

Exemplos de utilização

```
>>> AlignSeq("HGWAG", "PHSWG",  
             ScoreMatrix(blosum62 = True), g = -8)  
Melhor score do alinhamento otimo local: 19  
HGW  
HSW  
>>> AlignSeq("GKYESVI", "KYVSSWI",  
             ScoreMatrix(blosum62 = True), g = -1)  
Melhor score do alinhamento otimo global: 16  
Alinhamento 1  
GKY-ES-VI  
-KYVSSW-I  
Alinhamento 2  
GKY-ESV-I  
-KYVSS-WI
```

Exemplos de utilização

```
>>> AlignSeq("GKYESVI", "KYVSSWI",  
              ScoreMatrix(blosum62 = True),  
              local = True, g = -1)
```

Melhor score do alinhamento otimo local: 14

Alinhamento 1

KYES

KYVS

Alinhamento 2

KY-ESVI

KYVSSWI