Laborator 7 – Bioinformatică

Alinierea de secvenţe - continuare

A. Biblioteca NumPy

NumPy este o bibliotecă dezvoltată pentru manipularea de tablouri multidimensionale (arrays).

Tablourile sunt obiecte mutabile. Tipurile de date conţinute în tablouri pot fi: *int*, *float*, *complex*, *double*, *byte*, *long*, *string*, etc.

Să se testeze următoarele exemple. Ce afişează?

Observaţie. Ca orice bibliotecă, *NumPy* trebuie inclusă în scriptul Python care o utilizează:

```
from numpy import *
```

1.1.1 Funcţii de bază pentru manipularea de tablouri

Exemplul 1

```
# crearea unui vector dintr-o lista

a = array([1,2,3,4,5])

print a

# crearea unui vector de lungime 4 cu elementele 0.0

b = zeros(4)

print b

# crearea unui vector de lungime 4 cu elementele 1.0

c = ones(4)

print c

# crearea unei matrici

d = array([[1,2],[3,4]])
```

```
print d

# matrice de 4x3 cu toate elementele 0.0

f = zeros((4,3))

print f

# matrice de 4x3 cu toate elementele nr intregi 0

g = zeros((4,3),int)

print g

# matrice de 4x3 cu toate elementele nr complexe

h = zeros((4,3), complex)

print h
```

1.1.2 Funcţii matematice de bază pe tablouri

Exemplul 2

```
a = array([1,5,4.3])

b = array([-0.9,3,4])

print a + b

print a-b

print a*b

print a/b
```

Exemplul 3

```
a = array([[1,2,3,4],[5,6,7,8],[9,10,11,12]])

print a
```

```
print a[0]

print a[-1]

print a[0][0]

print a[1][2]

print a[:2]

print a[:,1]

print a.shape #dimensiunea unei matrici

c = array([[1,2,3,4],[5,6,7,8],[3]) #eroare, de ce?
```

Exemplul 4

```
a = array([1,2,3,4])

b = array([5,6,7,8])

# inmultire pe componente

print a*b

# produs scalar - inmultire vector linie a cu vector colana b

print dot(a,b)

# inmultire un vector coloana a cu un vector linie b

print outer (a,b)
```

Exemplul 5

```
from numpy.random import *

a = ranf((4,3))

print a
```

```
b = random_integers(0,5,(4,3))

print b
```

Exemplul 6

```
b = random_integers(0,5,(4,3))

print sqrt(b)

print b**2

print exp(b)

print log(b)

print log2(b)
```

1.1.3 Copieri, comparări şi *slicing*

Exemplul 7

Variabilele conţin referinţe către obiectele din memorie. Tablourile sunt tot obiecte. Ce afişează codul de mai jos şi de ce?

```
b = array ([5.6, 4.2, 3.8])

c = b

c[0] = 0

print b
```

Exemplul 8

Pentru a copia locaţia de memorie se utilizează funcţia *copy*.

```
from copy import *

b = array([5.6, 4.2, 3.8])

c=copy(b)

c[0]=0
```

```
print b
```

Copia se mai poate realiza prin adunarea lui 0 la vectorul *b* care creează alt obiect, la altă locaţie de memorie.

```
b = array([5.6, 4.2, 3.8])

c=b+0

c[0]=0

print b
```

Exemplul 9

```
a = ranf(5)

b = ranf(5)

print a

print b

print greater(a,b)

print greater(a,b).astype(int)
```

Alte funcţii de comparare sunt: *less*, *less_equal*, *greater_equal*, *equal*.

Exemplul 10

```
a = ranf(5) - 0.5

print a

print sign(a)
```

Exemplul 11

```
a = [True, False, False,True]

b = [True, False, True,False]

print logical_and(a,b)
```

```
print logical_or(a,b)
```

Exemplul 12

- Conversii:

```
c = array([[1,2,3],[4,5,6],[7,8,9]])

print c

c[0,2] = 10.5

c[0][1] = 0.5

print c

c = c + 0.0

c[0,2] = 10.5

print c

c = c.astype(int)

print c
```

Exemplul 13

```
a = arange(15) #intoarce un array de la 0 la 14

print a

a = a/4

print a

a = arange(15)

a = a/4.

print a

b = [3,5,14,9,2]
```

```
print a[b]
```

Exemplul 14

- Sortare:

```
a = ranf(5)

print a

a.sort()

print a

b = a[::-1]

print b

a = ranf(5)

print a

print a.argsort() #indicii sevcventei sortate
```

Exemplul 15

```
a = ranf((3,2))

print a

print ravel(a)

print

print a.reshape((2,3))

print

print a

print

b = ranf((2,2))
```

```
print b

print

c = ranf((1,2))

print c

print

print concatenate((a,b,c))
```

Exemplul 16

- Permutarea random a unui tablou:

```
a = ranf(5)

print a

shuffle(a)

print a
```

Exemplul 17

- Funcţii statistice:

```
a = ranf((4,3))

print a

# maximul din tot tabloul

print a.max()

# intoarce un vector cu maximele de pe fiecare coloana

print a.max(0)

# intoarce un vector cu maximele de pe fiecare linie

print a.max(1)
```

```
print a.min()

print a.min(0)

print a.min(1)

print a.sum()

print a.sum(0)

print a.sum(1)

print a.mean()

print a.mean(0)

print a.mean(1)
```

1.1.4 Conversii de la tablouri la alte tipuri

Exemplul 18

```
a = ranf(5)

print a.tolist()

a = ranf((4,3))

print a.tolist()
```

Exemplul 19

```
a = ranf(5)

print str(a)+'este string'

print map(str,a)
```

B. Altă modalitate de calculare a scorului în alinierea de secvenţe:

Pentru calcularea scorului alinierilor secvenţelor se iau în calcul şi alţi factori biologici precum tipurile de aminoacizi (polari, nepolari, ionizaţi). Astfel, când doi aminoacizi diferă, contează în calculul scorului final şi dacă ambii sunt încărcaţi pozitiv, de exemplu, sau dacă unul este încărcat, iar celălalt nu. Astfel, scorul se poate calcula ţinând cont şi de probabilitatea înlocuirii unui aminoacid cu altul în cadrul aceleiaşi familii de proteine.

Să se implementeze o funcţie, *blosum_score(seq1, seq2, gap = -8)*, care calculează scorul a doua secvenţe aliniate astfel:

- când se întâlneşte "–", se adună la scor valoarea *gap = -8*;
- se adună ponderile perechilor de aminoacizi aliniaţi;
- în matricea *BLOSUM*, elementul *(i,j)* reprezintă valoarea care se adună la scor dacă se întalneşte aliniată perechea *PBET[i] – PBET[j]*, unde *PBET* este un vector care conţine codurile aminoacizilor în urmatoarea ordine:
  PBET ='ARNDCQEGHILKMFPSTWYV'
- se observă că pe diagonala principală a matricii *BLOSUM* se găsesc valori pozitive deoarece corespund alinierii aceluiaşi aminoacid în ambele secvenţe.
- se pot utiliza funcţia *sir.index('caracter')* care întoarce poziţia caracterului în şirul *sir*.
- să se utilizeze următoarea matrice *BLOSUM* de scoruri:

```
BLOSUM = array([
[ 5,-2,-1,-2,-1,-1,-1, 0,-2,-1,-2,-1,-1,-3,-1, 1, 0,-3,-2, 0],
[-2, 7,-1,-2,-1, 1, 0,-3, 0,-4,-3, 3,-2,-3,-3,-1,-1,-3,-1,-3],
[-1,-1, 7, 2,-2, 0, 0, 0, 1,-3,-4,-0,-2,-4,-2,-1, 0,-4,-2,-3],
[-2,-2, 2, 8,-4, 0, 2,-1,-1,-4,-4,-1,-4,-5,-1, 0,-1,-5,-3,-4],
[-1,-4,-2,-4,13,-3,-3,-3,-3,-2,-2,-3,-2,-2,-4,-1,-1,-5,-3,-1],
[-1,-1, 0, 0,-3, 7, 2,-2, 1,-3,-2, 2, 0,-4,-1,-0,-1,-1,-1,-3],
[-1, 0, 0, 2,-3, 2, 6,-3, 0,-4,-3, 1,-2,-3,-1,-1,-1,-3,-2,-3],
[ 0,-3, 0,-1,-3,-2,-3, 8,-2,-4,-4,-2,-3,-4,-2, 0,-2,-3,-3,-4],
[-2, 0, 1,-1,-3, 1, 0,-2,10,-4,-3, 0,-1,-1,-2,-1,-2,-3,-1, 4],
[-1,-4,-3,-4,-2,-3,-4,-4,-4, 5, 2,-3, 2, 0,-3,-3,-1,-3,-1, 4],
[-2,-3,-4,-4,-2,-2,-3,-4,-3, 2, 5,-3, 3, 1,-4,-3,-1,-2,-1, 1],
[-1, 3, 0,-1,-3, 2, 1,-2, 0,-3,-3, 6,-2,-4,-1, 0,-1,-3,-2,-3],
[-1,-2,-2,-4,-2, 0,-2,-3,-1, 2, 3,-2, 7, 0,-3,-2,-1,-1, 0, 1],
[-3,-3,-4,-5,-2,-4,-3,-4,-1, 0, 1,-4, 0, 8,-4,-3,-2, 1, 4,-1],
[-1,-3,-2,-1,-4,-1,-1,-2,-2,-3,-4,-1,-3,-4,10,-1,-1,-4,-3,-3],
[ 1,-1, 1, 0,-1, 0,-1, 0,-1,-3,-3, 0,-2,-3,-1, 5, 2,-4,-2,-2],
[ 0,-1, 0,-1,-1,-1,-1,-2,-2,-1,-1,-1,-1,-2,-1, 2, 5,-3,-2, 0],
[-3,-3,-4,-5,-5,-1,-3,-3,-3,-3,-2,-3,-1, 1,-4,-4,-3,15, 2,-3],
[-2,-1,-2,-3,-3,-1,-2,-3, 2,-1,-1,-2, 0, 4,-3,-2,-2, 2, 8,-1],
[0,-3,-3,-4,-1,-3,-3,-4,-4, 4, 1,-3, 1,-1,-3,-2, 0,-3,-1, 5]])
```

C. Algoritm de programare dinamică pentru alinierea globală de secvenţe. Algoritmul Needleman-Wunsch

Pentru proiectarea algoritmului de aliniere prin metoda programării dinamice este nevoie de o analiză mai atentă a problemei pentru a identifica modul de descompunere a acesteia.

Problema dată este următoarea: Se dau două secvenţe, *x* şi *y*, nu neapărat de aceeaşi dimensiune, peste alfabetul Σ şi o funcţie de scor *s*, $s : \Sigma \times \Sigma \rightarrow Z$. Se doreşte determinarea celei mai bune alinieri globale, folosind modelul liniar de penalizare a golurilor (*gaps*). Modelul liniar presupune o penalizare constantă pentru fiecare gol.

Observaţii:

- Nu se acceptă în aliniere corespondenţa gap – gap, deoarece aceasta poate fi ignorată.
- Funcţia scor, *s*, se reprezintă ca o matrice.

Tehnica de programare dinamică se aplică pentru probleme care pot fi descompuse în subprobleme ale căror soluţii se suprapun. Optimul global este obţinut din soluţiile optime ale subproblemelor în care este descompusă problema iniţială. Pentru a nu recalcula soluţiile subproblemelor, acestea sunt reţinute într-o structură de date, cel mai adesea o matrice sau un vector.

Pentru a descompune problema alinierii globale, se va rezolva o problemă mai generală decât cea iniţială şi anume:

Se dau secvenţele *x*, *y*, peste alfabetul Σ şi o funcţie de scor, *s*. Se doreşte obţinerea celei mai bune alinieri globale a subsecvenţei formate din primele *i* caractere din secvenţa *x* cu primele *j* caractere din secvenţa *y*, pentru $\forall i \in \{1,\ldots,|x|\}$ şi $\forall j \in \{1,\ldots,|y|\}$.

Se notează cu *D(i,j)* scorul celei mai bune alinieri dintre subsecvenţa primelor *i* caractere din *x* şi subsecvenţa primelor *j* caractere din *y*.

Se notează cu *g* penalizarea unui *gap*, $g<0$, pentru a descuraja introducerea gap-urilor.

Este evident ca $D(0,0)=0$.

De asemenea, $D(i,0) = i \cdot g$ şi $D(0,j) = j \cdot g$.

D(i,0) reprezintă scorul alinierii primelor *i* caractere din *x* cu o subsecvenţă de 0 caractere din *y*. Este clar că secvenţa din *x* este aliniată cu o secvenţă de gap-uri.

In acelaşi mod se obţine şi *D(0, j)* pentru alinierea subsecvenţei primelor *j* caractere din *y* cu gap-uri.

Formula recurentă pentru *D(i, j)* este următoarea:

$$D(i,j) = \max \begin{cases} D(i-1,j-1) + s(x_i, y_j) \\ D(i-1,j) + g \\ D(i,j-1) + g \end{cases}, i \in \{1,\ldots,|x|\}, j \in \{1,\ldots,|y|\} \quad (1)$$

D(0,0), *D(i,0)*, *D(0, j)* reprezintă condiţiile iniţiale pentru formula recurentă.

x_i reprezintă al *i*-lea element din secvenţa *x* şi y_j reprezintă al *j*-lea element din secvenţa *y*.

Formula recurentă (1) are următoarea semnificaţie:

- Prima ramura a recurenţei corespunde cazului în care s-a realizat corespondenţa dintre caracterul x_i şi caracterul y_j. Este nevoie să se cunoască scorul alinierii lui x_i cu y_j, adică $s(x_i, y_j)$ şi scorul celei mai bune alinieri dintre primele *i-1* caractere din secvenţa *x* şi primele *j-1* caractere din secvenţa *y*, adică *D(i-1, j-1)*.
- A doua ramură a recurenţei corespunde cazului în care s-a realizat corespondenţa dintre caracterul x_i şi un *gap*. Este nevoie să se cunoască scorul celei mai bune alinieri dintre primele *i-1* caractere din secvenţa *x* şi primele *j* caractere din secvenţa *y*, adică *D(i-1, j)*, la care se adaugă penalizarea de gap, *g*.
- A treia ramură a recurenţei corespunde cazului în care s-a realizat corespondenţa

dintre caracterul y_j şi un gap. Este nevoie să se cunoască scorul celei mai bune alinieri dintre primele *i* caractere din secvenţa *x* şi primele *j-1* caractere din secvenţa *y*, adică *D(i, j-1)*, la care se adaugă penalizarea de gap, *g*.

$$\begin{array}{c|c} x_1 \dots x_{i-1} & x_i \\ y_1 \dots y_{j-1} & y_j \end{array} \qquad \begin{array}{c|c} x_1 \dots x_{i-1} & x_i \\ y_1 \dots y_j & - \end{array} \qquad \begin{array}{c|c} x_1 \dots x_i & - \\ y_1 \dots y_{j-1} & y_j \end{array}$$

Scorul alinierii globale optime a secvenţelor *x* şi *y* se va găsi în $D(|x|,|y|)$. Structura de date utilizată pentru implementarea acestei recurenţe este o matrice de dimensiune $(|x|+1)\times(|y|+1)$. Matricea va fi initializată cu condiţiile iniţiale şi apoi va fi completată pe linii de la elementul *(1,1)* până la elementul $(|x|,|y|)$.

<u>*Observaţie*</u>:

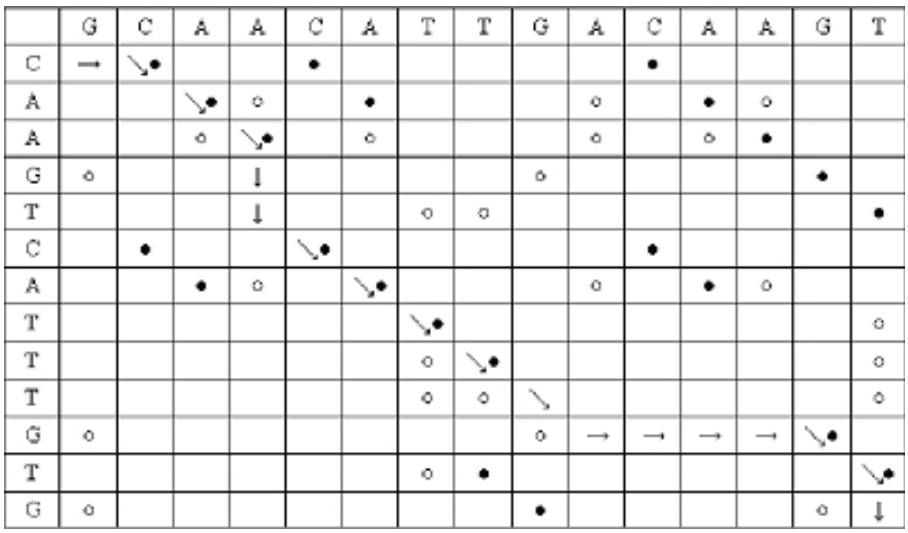
Anumite alinieri pot avea scoruri negative, datorită penalizarilor de gap sau a nepotrivirilor dintre caracterele aliniate.

Până acum s-a discutat despre obţinerea scorului alinierii globale optime. Pentru obţinerea alinierii propriu-zise, se porneşte din elementul $(|x|,|y|)$ din matricea *D*, şi se reconstruieşte alinierea dinspre ultimele caractere înspre primele, astfel:

- poziţia iniţială în matrice este $(|x|,|y|)$;
- dacă valoarea elementului de la poziţia curentă, *(i, j)*, din matrice este egală cu rezultatul obţinut din prima ramură a recurenţei (1), atunci elementul curent va deveni *(i-1, j-1)*, iar la aliniere se va adăuga la stânga corespondenţa dintre x_i şi y_j.
- dacă valoarea elementului de la poziţia curentă, *(i, j)*, din matrice este egală cu rezultatul obţinut din a doua ramură a recurenţei (1), atunci elementul curent va deveni *(i-1, j)*, iar la aliniere se va adăuga la stânga corespondenţa dintre x_i şi un *gap* (" – ").
- dacă valoarea elementului de la poziţia curentă, *(i, j)*, din matrice este egală cu rezultatul obţinut din a treia ramură a recurenţei (1), atunci elementul curent va deveni *(i, j-1)*, iar la aliniere se va adăuga la stânga corespondenţa dintre un *gap* (" – ") şi y_j.
- se repetă algoritmul până se ajunge în elementul *(0, 0)*.
- când se ajunge în elementul *(0,0)*, s-a obţinut alinierea corespunzătoare celui mai bun scor pentru alinierea globală a şirului *x* cu şirul *y*.

<u>*Observaţii*</u>:

- Dacă se ajunge într-o pozitie *(i, 0)*, se adaugă, la stânga în aliniere, subsecvenţa primelor *i* caractere din *x* aliniate cu *gap*-uri.
- Dacă se ajunge într-o poziţie *(0, j)*, se adaugă, la stânga în aliniere, o secvenţă de *gap*-uri alinată cu subsecvenţa primelor *j* caractere din *y*.
- alinierea globală optimă a două secvenţe se poate determina grafic astfel:



Aliniere globală optimă

D. Aplicaţii. Algoritmul Needleman-Wunsch

1. Să se completeze scheletul de cod din fisieru *alin_nw.py* si sa se testeze implementarea algoritmului Needleman-Wunsch.

2. În implementarea prezentată s-a utilizat o funcţie de scor simplă, *simple(c_1, c_2)*, care întoarce valoarea 1 în caz de potrivire a două caractere şi 0 în caz de nepotrivire. Să se scrie o funcţie de scor, *BLOSUM(c_1, c_2)*, care să întoarcă valoarea corespunzatoare caracterelor c_1 şi c_2 din matricea *BLOSUM*. Să se testeze implementarea algoritmului, utilizând această funcţie de scor.

3. Sa se testeze algoritmul pe mai multe secvente ale proteinei ribonucleaza pancreatica si sa se compare rezultatele cu cele obtinute prin alinierea cu programul.