UNIVERSIDAD DE TALCA
FACULTAD DE INGENIERÍA
ESCUELA DE INGENIERÍA EN BIOINFORMÁTICA

Algoritmos y Estructuras de Datos
Proyecto Unidad III
Implementación del Algoritmo Needleman-Wunsch.

Introducción

Existen muchos algoritmos de comparación que permiten ser usados para evaluar similitudes en secuencias de proteínas y ADN, uno de ellos es el algoritmo de Needleman-Wunsch, propuesto en 1970 por Saul Needelman y Christian Wunsh.

El algoritmo Needleman-Wunsch realiza un alineamiento total entre dos secuencias para calcular un puntaje que indica qué tantas similitudes existen en ambas secuencias (ya sean cadenas de proteínas o ácidos nucleicos). El algoritmo siempre termina y asegura una solución óptima, siendo apropiado cuando las similitudes en los símbolos del alfabeto son establecidas previamente.

Needleman-Wunsch fue de las primeras aplicaciones de programación dinámica para la comparación de secuencias biológicas.

Elementos de entrada necesarios para el alineamiento global de dos secuencias con el algoritmo Needleman-Wunsch:

- Un alfabeto **A** que tiene los símbolos que forman las secuencias a alinear. Por ejemplo, el alfabeto podría ser **A={'A', 'G', 'C', 'T'}** si se trabaja con cadenas de ADN, o podría ser **A={'A', 'R', 'N', 'D', 'C', 'Q', 'E', 'G', 'H', 'I', 'L', 'K', 'M', 'F', 'P', 'S', 'T', 'W', 'Y', 'V'}** si se trabaja con aminoácidos.
- Dos cadenas de texto **S** y **T** de tamaño **n** y **m** respectivamente, que representan las dos secuencias a alinear. En detalle:
  - Sea **S[1..n]** la primera cadena a alinear, que tiene **n** caracteres (S[1] sería el primer caracter de la cadena S y S[n] sería el último caracter de la cadena S).
  - Sea **T[1..m]** la segunda cadena a alinear, que tiene **m** caracteres (T[1] sería el primer caracter de la cadena T y T[m] sería el último caracter de la cadena T).
- Una función **U(c,d)** que dadas dos letras **c** y **d** del alfabeto **A** da como resultado el puntaje de similitud de la letra **c** con la letra **d**, o sea, qué tanto se parecen las letras **c** y **d**.
- Un valor **V** que indica el puntaje de penalidad. Este puntaje se asigna cuando un par de letras no empareja (apertura de *gap*).

Respuesta dada por el algoritmo Needleman-Wunsch dados los parámetros de entrada:

- El puntaje máximo que se logra al alinear las secuencias **A** y **B**.

- El emparejamiento que da el puntaje máximo.

Para entender los parámetros de entrada y la respuesta del algoritmo, observe los siguientes ejemplos:

Considere el alfabeto **A={'A','B','C'}** y la siguiente función **U** de similaridad de las letras del alfabeto:

	'A'	'B'	'C'
'A'	3	-8	-7
'B'	-8	5	-9
'C'	-7	-9	4

Sea **V=-2** el puntaje de penalidad. Sean **S='BCAB'** y **T='ABAB'** las dos secuencias a emparejar. Bajo estas circunstancias, el siguiente emparejamiento da el puntaje máximo:

```
-BCAB
AB-AB
```

Cada uno de los **'-'** coincide con una letra no emparejada y da un puntaje de **V**. El puntaje total de este emparejamiento es:

V+U('B','B')+V+U('A','A')+U('B','B')=(-2)+(5)+(-2)+(3)+(5)=9.

Obviamente hay muchos más emparejamientos, pero ninguno con mayor puntaje. Por ejemplo, el emparejamiento

```
BCAB
ABAB
```

Da un puntaje de:

U('B','A')+U('C','B')+U('A','A')+U('B','B')=(-8)+(-9)+(3)+(5)=-9.

Descripción del Algoritmo

- Sea **f(i,j)** el máximo puntaje que se logra alineando los primeros **i** caracteres de la cadena **S** contra los primeros **j** caracteres de la cadena **T**, para todo **i** entre 0 y **n** y para todo **j** entre 0 y **m**.
- Se define **f** recursivamente con programación dinámica así:
  - Si i=0 y j=0, entonces f(i,j)=0.
  - Si i>0 y j=0, entonces f(i,j)=f(i-1,j)+V=...=V*i porque como j=0 entonces se tiene que la segunda cadena es vacía y que la única opción es no emparejar ninguna letra de S[1..j].
  - Si i=0 y j>0, entonces f(i,j)=f(i,j-1)+V=...=V*j porque como i=0 entonces se tiene que la primera cadena es vacía y que la única opción es no emparejar ninguna letra de T[1..i].
  - Si i>0 y j>0, entonces f(i,j)=max{f(i-1,j)+V,f(i,j-1)+V,f(i-1,j-1)+U[S[i]][T[j]]} porque se busca lo mejor entre las siguientes tres opciones:
    - f(i-1,j)+V que resulta de no emparejar S[i] (lo que da puntaje V) y de mirar el puntaje de emparejar el resto de cadena S[1..i-1] contra T[i..j] (lo que da puntaje f(i-1,j).
    - f(i,j-1)+V que resulta de no emparejar T[j] (lo que da puntaje V) y de mirar el puntaje de emparejar S[1..i] contra el resto de cadena T[1..j-1] (lo que da puntaje f(i,j-1).

- f(i-1,j-1)+U[S[i]][T[j]] que resulta de emparejar S[i] con T[j] (lo que da puntaje [S[i]][T[j]]) y de mirar el puntaje de emparejar el resto de cadena S[1..i-1] contra el resto de cadena T[1..j-1] (lo que da puntaje f(i-1,j-1).

- El puntaje máximo se encuentra calculando **f(n,m)**, que revisa todos los **n** caracteres de la cadena **S** contra todos los **m** caracteres de la cadena **T**.

Calculando todos los posibles valores de la función **f** usando una matriz de tamaño n+1 por m+1 podemos reconstruir el alineamiento que da el puntaje máximo.

Pseudocódigo para hallar el máximo puntaje que se logra alineando las cadenas **S** y **T**:

- Declare f[0..n][0..m] como una matriz de enteros con n filas y m columnas.
- Para cada i desde 0 hasta n:
  - Para cada j desde 0 hasta m:
    - Si i=0 y j=0: asigne a f[i][j] el valor 0.
    - De lo contrario, si i>0 y j=0: asigne a f[i][j] el valor de f[i-1][j]+V.
    - De lo contrario, si i=0 y j>0: asigne a f[i][j] el valor de f[i][j-1]+V.
    - De lo contrario, si i>0 y j>0: asigne a f[i][j] el máximo entre f[i-1][j]+V, f[i][j-1]+V, y f[i-1][j-1]+U[S[i]][T[j]].
- El máximo puntaje está en f[n][m].

Pseudocódigo para reconstruir el alineamiento que da el puntaje máximo:

- Declare alineamientoS como una cadena vacía.
- Declare alineamientoT como una cadena vacía.
- Inicialice i en n y j en m.
- Mientras i sea mayor que 0 ó j sea mayor que 0:
  - Si i>0 y j=0: concatene el caracter S[i] al principio de alineamientoS, concatene el caracter - al principio de alineamientoT, y decrezca i en uno.
  - De lo contrario, si i=0 y j>0: concatene el caracter - al principio de alineamientoS, concatene el caracter T[j] al principio de alineamientoT, y decrezca j en uno.
  - De lo contrario, si i>0 y j>0 y f[i][j]=f[i-1][j]+V: concatene el caracter S[i] al principio de alineamientoS, concatene el caracter - al principio de alineamientoT, y decrezca i en uno.
  - De lo contrario, si i>0 y j>0 y f[i][j]==f[i][j-1]+V: concatene el caracter - al principio de alineamientoS, concatene el caracter T[j] al principio de alineamientoT, y decrezca j en uno.
  - De lo contrario: concatene el caracter S[i] al principio de alineamientoS, concatene el caracter T[j] al principio de alineamientoT, decrezca i en uno, y decrezca j en uno.

Descripción del trabajo

Tomando como referencia la información entregada en este documento más otras referencias que investigue, implemente el algoritmos de Needleman-Wunsch para el alineamiento de secuencias de ADN (con albafeto **A={'A', 'G', 'C', 'T'}**). Su programa debe:

- Implementarse en el lenguaje de programación C++.
- Leer como parámetro de entrada:
  - Las dos cadenas, cada una de ellas en archivos de textos independientes.
  - Un arhivo con la matriz de emparejamiento **U**. El formato es libre.
  - El valor **V** correspondiente al puntaje de no emparejar.
  - Por ejemplo:

```
:~$ ./programa -C1 cad1.tex -C2 cad2.tex -U funU.tex -V val
```

- Entregar la reconstrucción del alineamiento que da el puntaje más alto. Muestre este resultado utilizando Graphviz (http://www.graphviz.org/).
- Puede utilizar tanto estructuras de datos estáticas como dinámicas en su implementación.

Referencias

- https://pmc.ncbi.nlm.nih.gov/articles/PMC7123042/
- https://www.ebi.ac.uk/jdispatcher/psa
- https://open.oregonstate.education/appliedbioinformatics/chapter/chapter-3/