TEC | Tecnológico de Costa Rica

Proyecto 1

Comparación de cadenas

IC-8022

Introducción a la Biología Molecular Computacional

Instituto Tecnológico de Costa Rica, Sede Central Cartago

Escuela de Computación, Ingeniería en Computación

II Semestre 2017

Prof. Esteban Arias Méndez

Estudiantes:

Julian Salinas Rojas 2015114132

Johanna Elizondo Elizondo 2013049616

Fecha de entrega:

30/Oct/2017

Abstract

This document contains relevant information about the development and execution of the final implementation for the sequence aligner as defined by the first project assignment as part of the Introduction to Computational Molecular Biology course.

Tabla de Contenidos

Introducción

El alineamiento de secuencias resulta de gran utilidad dentro del campo de la biología molecular, ya que, se usa para analizar distintos aspectos sobre secuencias de ADN, por ejemplo, la existencia de mutaciones, entre ellas deleciones, sustituciones e inserciones.

Por lo anterior, el objetivo es la creación de una herramienta que permita, mediante distintos algoritmos comparar cadenas de una manera más eficiente. El presente documento contiene información relevante acerca del desarrollo de la misma.

Como herramienta principal para la implementación de los algoritmos se utiliza el lenguaje Python 3 y el ambiente de desarrollo Pycharm, además se hace uso de la biblioteca Numpy la cual permite un manejo más simple y eficiente de matrices. Se implementa una línea de comandos para que el usuario pueda ejecutar cada uno de los alineamientos, especificando para ello una secuencia o un archivo de texto.

Los procedimientos implementados se basan en diferentes algoritmos conocidos extensamente para el alineamiento de secuencias, entre ellos se encuentran el algoritmo Needleman–Wunsch para alineamiento global. el algoritmo Smith-Waterman para el alineamiento local, y finalmente el algoritmo de Hirschberg para los alineamientos con espacio linear.

Marco teórico

Algoritmo Needleman-Wunsch

Es un algoritmo de programación dinámica, es decir, busca dividir el problema en otros más pequeños y a su vez aprovechando resultados obtenidos con anterioridad. Sirve para realizar alineamientos globales entre dos secuencias y es comúnmente usado en bioinformática para alinear secuencias de nucleótidos o proteínas.

Fue propuesto en 1970 por Saul Needleman y Christian Wunsch en su paper *A general method applicable to the search for similarities in the amino acid sequence of two proteins, J Mol Biol. 48(3):443-53.*

De forma general, el algoritmo define una matriz donde se colocan las dos cadenas por alinear, se define valores para los resultados de la comparación de elementos de las cadenas, luego celda por celda se llena cada una con el mejor resultado de las celdas anteriores más el valor de comparación actual. Como paso final se reconstruye el alineamiento siguiendo el origen del valor de las celdas empezando por la última de la matriz, hasta llegar a la primera de la misma.

Pseudocódigo

```
for i=0 to length(A)
 F(i,0) ← d*i
for j=0 to length(B)
 F(0,j) ← d*j
for i=1 to length(A)
 for j=1 to length(B)
 {
   Match ← F(i-1,j-1) + S(A_i, B_j)
   Delete ← F(i-1, j) + d
   Insert ← F(i, j-1) + d
   F(i,j) ← max(Match, Insert, Delete)
 }
```

Algoritmo Smith-Waterman

Es una variación del algoritmo Needleman-Wunsch, sin embargo, este algoritmo está enfocado en realizar alineamientos locales de secuencias; esto es, determinar regiones similares entre dos secuencias de nucleótidos o proteínas.

Fue propuesto por Temple Smith y Michael Waterman en 1981. La principal diferencia con el algoritmo Needleman-Wunsch es que las celdas no pueden ser negativas, es decir, el minino que se puede asignar a una celda es cero, lo cual representa que los alineamientos locales sean visibles.

Otra diferencia con el algoritmo de Needleman-Wunsch es que el retroceso comienza en la celda de la matriz con el puntaje más alto y continua hasta que una celda con puntaje cero es encontrada, proporcionando el puntaje más alto para el alineamiento local.

Según González, una motivación para alineamientos locales es la dificultad para obtener alineamientos correctos en regiones de baja similitud entre secuencias biológicas lejanamente emparentadas, porque las mutaciones agregaron mucho "ruido" con la evolución para permitir una comparación significativa de estas regiones. Los alineamientos locales evitan estas regiones completamente y se concentran en aquellas con un puntaje positivo, por ejemplo, aquellas con señales de similitud conservadas por la evolución.

Como se menciona anteriormente, este algoritmo es derivado del Needleman-Wunsch, por tanto el pseudocódigo también es similar. Primero se inicializa la primera fila y la primera columna con ceros y luego, al escoger los nuevos valores, se debe tomar en cuenta el cero y descartar los negativos.

Se hace uso de un algoritmo derivado para obtener alineamientos semiglobales. La única modificación con respecto al algoritmo Needleman-Wunsch es que tanto la primera fila como la primera columna inician en ceros, es decir, no se penalizar por los espacios entre una secuencia y otra.

Algoritmo de Hirschberg

Al igual que el anterior, es una variación que se puede aplicar a los algoritmos derivados de Needleman-Wunsch. Su objetivo es buscar el alineamiento óptimo entre dos secuencias mientras utiliza una cantidad de almacenamiento mucho menor al algoritmo original. Para

lograrlo, en vez de tener una matriz completa en la memoria, se tienen solo dos vectores, los cuales son los únicos necesarios para calcular el vector siguiente. La desventaja de este algoritmo es que su reconstrucción resulta más compleja, por tanto, ahorra espacio pero aumenta el tiempo de cómputo.

Desarrollo

Se desarrollaron seis algoritmos en total, cada uno de ellos muy similares entre sí, por tanto se hace uso de herencia, para reciclar el código que sea común en cada uno de los algoritmos, por ejemplo, el cálculo de la matriz resulta igual para todos, sin embargo, el algoritmo de alineamiento global inicializa tanto la primera fila como la primera columna con el valor gap multiplicado por el número de fila o columna según corresponda.

La jerarquía utilizada no inicia con el algoritmo de Needleman-Wunsch o de alineamiento global, esta inicia con el algoritmo de alineamiento semiglobal. La razón de esto es que dicho algoritmo contiene la implementación que resulta más en común con el resto de algoritmos, por ejemplo, al hacer que la clase que implementa el algoritmo de alineamiento global herede del semiglobal implica sólo tener que cambiar el cómo se inicializa la matriz para proceder a realizar los cálculos sobre esta. De igual forma la clase que implementa el algoritmo local solo requiere sobrescribir el modo en que cada una de las celdas de la matriz debe ser calculada.

Para realizar la implementación de cada uno de los algoritmos se hace uso de los ejemplos propuestos por la escuela de computación de Carnegie Mellon University. Estos ejemplos muestran de forma visual cómo se comporta cada uno de los algoritmos, además permite la verificación de los resultados obtenidos.

Es necesario mostrar al usuario el procedimiento que se realizó sobre la matriz, por tanto, para lograr este objetivo se hace uso de una estructura adicional en la ejecución del algoritmo, esta estructura es una matriz que posee el mismo tamaño que la matriz original donde se realizan los cálculos, sin embargo, la finalidad de esta es guardar las flechas, es decir el procedimiento que se siguió para encontrar el valor óptimo. Así, cada vez que se calcula una celda de la matriz, se coloca una flecha en la matriz adicional o alternativa justo en el mismo número de fila y columna que tiene la celda que acaba de ser calculada.

Con el objetivo de facilitar el trabajo al usuario final se implementa una interfaz textual que permite realizar las operaciones usando palabras u oraciones sencilla e intuitivas. Por ejemplo,para realizar un alineamiento el usuario puede ingresar el siguiente texto:

Ingrese un comando: alineamiento global ACGTCACGT ACGTGTGTT

Está interfaz textual es implementada como un módulo que interpreta las entradas del usuario con la finalidad de ejecutar las funcionalidades correspondientes por medio de introspección ya que dichas funcionalidades tienen internamente el mismo nombre con que el usuario las invoca. Esto resulta en dos grandes ventajas, es más intuitivo para el usuario al momento de hacer uso de una funcionalidad y facilita tanto el desarrollo como el mantenimiento por parte de los desarrolladores.

Otro requerimiento es que cada uno de los alineamientos debe mostrar cuánto tiempo duró procesando y cuánta memoria necesito durante su ejecución. Para cumplir con lo anterior, se hace uso de el patrón decorador, esto con el fin de no afectar la lógica de los algoritmos, es decir no tener que alterar el código ya implementado. Cabe destacar que la memoria es medida en bytes mientras que el tiempo es medido en segundos. Otra nota importante es que al realizar la medición de memoria, se toma en cuenta el peso actual del objeto que realizó el algoritmo, rescatando que dicho objeto contiene la matriz original, la matriz que contiene las flechas, así como otras propiedades, lo cual, hace que la memoria necesitada sea un poco más de la esperada, sin embargo, la diferencia del uso de memoria es fácilmente notable al comparar un algoritmo en su versión estándar con su versión de espacio lineal mediante el algoritmo de Hirschberg.

Análisis de resultados

En esta sección se va a mostrar un análisis de resultados donde se evidencie la funcionalidad de cada algoritmo de alineamiento, además de la funcionalidad de los diferentes comandos implementados.

Al iniciar el programa se muestra los siguiente:

```
Opciones disponibles:
alineamiento, tablas, optimo, listar, val
match, mismatch, gap, recursos, ayuda, salir

Para mostrar información detallada sobre cada opción anteponga la palabra 'ayuda'
Ejemplo: >> ayuda alineamiento

Ingrese un comando:
```

En la parte de ingresar comando es donde vamos a escribir todo lo que necesitamos y así mostrar el análisis de resultados. En todas las pruebas vamos a seleccionar que si queremos ver la tabla generada y mostrar procedimiento en la tabla, esto para tener una mejor evidencia del análisis.

Resultado esperado: El resultado esperado para todos los algoritmos es que el sistema muestre el alineamiento correcto entre las secuencias que se ingresen, además del valor óptimo.

Alineamiento Global

Se va a ingresar el siguiente comando: **alineamiento global sala salon**

Resultado obtenido:

```
Ingrese un comando: alineamiento global sala salon

Valor óptimo:
valor: 0

Alineamiento obtenido:
sal_a
salon

¿Desea ver la tabla generada? [S/N]: s
¿Mostrar procedimiento en la tabla [S/N]: s

Tabla obtenida
[          _          s          a          l          o          n ]
[    _     0    ←     -1    ←    -2    ←    -3    ←    -4    ←    -5 ]
[          ↑    ↖                                                    ]
[    s    -1          1    ←     0    ←    -1    ←    -2    ←    -3 ]
[          ↑          ↑    ↖                                         ]
[    a    -2          0          2    ←     1    ←     0    ←    -1 ]
[          ↑          ↑          ↑    ↖                              ]
[    l    -3         -1          1          3    ←     2    ←     1 ]
[          ↑          ↑                     ↑    ↖     ↑    ↖     ↑ ]
[    a    -4         -2          0          2    ←     1    ←     0 ]
```

Alineamiento semiglobal

Se va a ingresar el siguiente comando: **alineamiento global files/test1 files/test2**

Resultado obtenido:

```
Valor óptimo:
valor: 0

Alineamiento obtenido:
CORR_EA
CORRALES

¿Desea ver la tabla generada? [S/N]: s
¿Mostrar procedimiento en la tabla [S/N]: s

Tabla obtenida
[          _          C          O          R          R          A          L          E          S ]
[    _     0          0          0          0          0          0          0          0          0 ]
[                  ↖          ↖    ↑    ↖          ↖          ↖          ↖          ↖          ↖    ]
[    C     0          1         -1         -1         -1         -1         -1         -1         -1 ]
[                        ↖                    ↖    ↑    ↖          ↖          ↖          ↖    ]
[    O     0         -1          2    ←     0         -2         -2         -2         -2         -2 ]
[                  ↖               ↑    ↖          ↖    ↑                    ↖    ↑    ↖          ↖    ]
[    R     0         -1          0          3          1    ←    -1         -3         -3         -3 ]
[                  ↖                              ↖                                        ↖    ↑ ]
[    R     0         -1         -2          1          4    ←     2    ←     0    ←    -2         -4 ]
[                  ↖          ↖               ↑          ↑    ↖          ↖    ↑    ↖               ]
[    E     0         -1         -2         -1          2          3          1          1    ←    -1 ]
[                  ↖          ↖                         ↑    ↖          ↖          ↖    ↑    ↖    ]
[    A     0         -1         -2         -3          0          3          2          0          0 ]
```

Alineamiento local

Se va a ingresar el siguiente comando: **alineamiento local files/test1 files/test2**

Resultado obtenido:

```
Ingrese un comando: alineamiento local files/test1 files/test2

Valor óptimo:
valor: 1

Alineamiento obtenido:
A
A

¿Desea ver la tabla generada? [S/N]: s
¿Mostrar procedimiento en la tabla [S/N]: s

Tabla obtenida
[          _          G          G          A          T ]
[    _     0          0          0          0          0 ]
[                                   ↖                    ]
[    A     0          0          0          1          0 ]
[                                   ↖          ↖         ]
[    A     0          0          0          1          0 ]
[                  ↖          ↖                    ↖    ]
[    G     0          1          1          0          0 ]
[                        ↖          ↖          ↖         ]
[    T     0          0          0          0          1 ]
```

La alineación se hizo con los siguiente archivos:

Alineamiento global con K-Band

Se va a ingresar el siguiente comando: **alineamiento kband files/test1 GATO**, con un valor de k = 5. Se utiliza el mismo archivo 'test1', que se muestra en la imagen anterior.

Resultado obtenido:

```
Ingrese un comando: alineamiento kband files/test1 GATO
Ingrese un valor para k: 5

Valor óptimo:
valor: -2

Alineamiento obtenido:
AAGT
GATO

¿Desea ver la tabla generada? [S/N]: s
¿Mostrar procedimiento en la tabla [S/N]: s

Tabla obtenida
[          G       A       T       0 ]
[  _   0       -2      -4      -6      -8 ]
[        ↖       ↖                      ]
[  A   -2      -1      -1   ←  -3   ←  -5 ]
[        ↖       ↖                      ]
[  A   -4      -3       0   ←  -2   ←  -4 ]
[        ↖       ↖       ↖              ]
[  G   -6      -3      -4      -1   ←  -3 ]
[        ↖       ↖       ↖       ↖      ]
[  T   -8      -7      -4      -3      -2 ]
```

Seguidamente se muestra los alineamientos con espacio lineal, en este si va ir mostrando la matriz como se va eliminando cada fila y columna para ver la buena funcionalidad de cada algoritmo.

Alineamiento global con espacio lineal

Se va a ingresar el siguiente comando: **alineamiento globallineal CASA COSAS**.

Resultado obtenido:

```
Ingrese un comando: alineamiento globallineal CASA COSAS

Valor óptimo:
valor: 0

Alineamiento obtenido:
[[  0  -2  -4  -6  -8 -10]
 [ -2   1  -1  -3  -5  -7]
 [ -4  -1   0  -2  -2  -4]
 [ -6  -3  -2   1  -1  -1]
 [ -8  -5  -4  -1   2   0]]

 Delete Column
[[ 0 -2 -4 -6 -8]
 [-2  1 -1 -3 -5]
 [-4 -1  0 -2 -2]
 [-6 -3 -2  1 -1]
 [-8 -5 -4 -1  2]]

 Delete Row
[[ 0 -2 -4 -6 -8]
 [-2  1 -1 -3 -5]
 [-4 -1  0 -2 -2]
 [-6 -3 -2  1 -1]]

 Delete Column
[[ 0 -2 -4 -6]
 [-2  1 -1 -3]
 [-4 -1  0 -2]
 [-6 -3 -2  1]]
```

```
 Delete Column
[[ 0 -2 -4 -6]
 [-2  1 -1 -3]
 [-4 -1  0 -2]
 [-6 -3 -2  1]]

 Delete Row
[[ 0 -2 -4 -6]
 [-2  1 -1 -3]
 [-4 -1  0 -2]]

 Delete Column
[[ 0 -2 -4]
 [-2  1 -1]
 [-4 -1  0]]

 Delete Row
[[ 0 -2 -4]
 [-2  1 -1]]

 Delete Column
[[ 0 -2]
 [-2  1]]
CASA_
COSAS
```

```
¿Desea ver la tabla generada? [S/N]: s
¿Mostrar procedimiento en la tabla [S/N]: s

Tabla obtenida
[                  C         O         S         A         S ]
[    _     0  ←   -2   ←   -4   ←   -6   ←   -8   ←  -10 ]
[          ↑  ↖                                          ]
[    C    -2        1   ←   -1   ←   -3   ←   -5   ←   -7 ]
[          ↑        ↑   ↖        ↖   ↑   ↖              ]
[    A    -4       -1        0        -2        -2   ←   -4 ]
[          ↑        ↑                  ↖             ↖    ]
[    S    -6       -3        -2        1   ←    -1        -1 ]
[          ↑        ↑                  ↑   ↖              ]
[    A    -8       -5        -4        -1        2   ←    0 ]
```

Alineamiento semiglobal con espacio lineal

Se va a ingresar el siguiente comando: **alineamiento semigloballineal SACO SAPOS**

Resultado obtenido:

```
Ingrese un comando: alineamiento semigloballineal SACO SAPOS

Valor óptimo:
valor: 0

Alineamiento obtenido:
[[ 0  0  0  0  0  0]
 [ 0  1 -1 -1 -1  1]
 [ 0 -1  2  0 -2 -1]
 [ 0 -1  0  1 -1 -3]
 [ 0 -1 -2 -1  2  0]]

 Delete Column
[[ 0  0  0  0  0]
 [ 0  1 -1 -1 -1]
 [ 0 -1  2  0 -2]
 [ 0 -1  0  1 -1]
 [ 0 -1 -2 -1  2]]

 Delete Row
[[ 0  0  0  0  0]
 [ 0  1 -1 -1 -1]
 [ 0 -1  2  0 -2]
 [ 0 -1  0  1 -1]]

 Delete Column
[[ 0  0  0  0]
 [ 0  1 -1 -1]
 [ 0 -1  2  0]
 [ 0 -1  0  1]]
```

```
 Delete Row
[[ 0  0  0  0]
 [ 0  1 -1 -1]
 [ 0 -1  2  0]]

 Delete Column
[[ 0  0  0]
 [ 0  1 -1]
 [ 0 -1  2]]

 Delete Row
[[ 0  0  0]
 [ 0  1 -1]]

 Delete Column
[[0 0]
 [0 1]]
SACO_
SAPOS
```

```
¿Desea ver la tabla generada? [S/N]: s
¿Mostrar procedimiento en la tabla [S/N]: s

Tabla obtenida
[          _          S          A          P          O          S ]
[    _     0          0          0          0          0          0 ]
[                ↖          ↖     ↑     ↖          ↖          ↖     ]
[    S     0          1         -1         -1         -1          1 ]
[                     ↖                     ↖     ↑          ↑     ]
[    A     0         -1          2     ←    0         -2         -1 ]
[                ↖               ↑     ↖          ↖     ↑     ↖     ]
[    C     0         -1          0          1         -1     ←   -3 ]
[                ↖                               ↖               ]
[    O     0         -1         -2         -1          2     ←    0 ]
```

Alineamiento local con espacio lineal

Se va a ingresar el siguiente comando: **alineamiento locallineal AGTTA AGUUA**

Resultado obtenido:

```
Alineamiento obtenido:
Delete Row
[[0 0 0 0 0 0]
 [0 1 0 0 0 1]
 [0 0 2 0 0 0]
 [0 0 0 1 0 0]
 [0 0 0 0 0 0]
 [0 1 0 0 0 1]]

Delete Row
[[0 1 0 0 0 1]
 [0 0 2 0 0 0]
 [0 0 0 1 0 0]
 [0 0 0 0 0 0]
 [0 1 0 0 0 1]]

Delete Row
[[0 0 2 0 0 0]
 [0 0 0 1 0 0]
 [0 0 0 0 0 0]
 [0 1 0 0 0 1]]
```

```
Delete Row
[[0 0 2 0 0 0]
 [0 0 0 1 0 0]
 [0 0 0 0 0 0]
 [0 1 0 0 0 1]]

Delete Row
[[0 0 0 1 0 0]
 [0 0 0 0 0 0]
 [0 1 0 0 0 1]]

Delete Row
[[0 0 0 0 0 0]
 [0 1 0 0 0 1]]

Delete Row
[[0 1 0 0 0 1]]

A_AG
A_AG
```

```
¿Desea ver la tabla generada? [S/N]: s
¿Mostrar procedimiento en la tabla [S/N]: s

Tabla obtenida
[          _          A          G          U          U          A ]
[    _     0          0          0          0          0          0 ]
[                ↖                                               ↖    ]
[    A     0          1          0          0          0          1 ]
[                     ↖                                              ]
[    G     0          0          2     ←    0          0          0 ]
[                               ↑     ↖                              ]
[    T     0          0          0          1          0          0 ]
[                                              ↖                    ]
[    T     0          0          0          0          0          0 ]
[                ↖                                               ↖    ]
[    A     0          1          0          0          0          1 ]
```

Comandos especiales

Ayuda:

```
Ingrese un comando: ayuda

        Realiza un alineamiento entre dos hileras con el algoritmo indicado.
        Las hileras pueden ser escritas manualmente o puede especificarse por medio de archivos.
        Sintaxis: alineamiento [global|semiglobal|local|kband] [texto_1|arhivo_1] [texto_2|arhivo_2] ?lineal
        Ejemplos:
            1. alineamiento global AAGCTGGT ACCCTTCGG
            2. alineamiento local AGTCGTCC archivo.txt
            3. alineamiento global archivo.txt otro_archivo.txt lineal
```

Tablas:

```
Ingrese un comando: alineamiento global casa cosa

Valor óptimo:
valor: 2

Alineamiento obtenido:
casa
cosa

¿Desea ver la tabla generada? [S/N]: n

Ingrese un comando: tablas

Tabla obtenida
[          _    c    o    s    a ]
[    _     0   -2   -4   -6   -8 ]
[    c    -2    1   -1   -3   -5 ]
[    a    -4   -1    0   -2   -2 ]
[    s    -6   -3   -2    1   -1 ]
[    a    -8   -5   -4   -1    2 ]
```

Listar:

```
Ingrese un comando: listar

Algoritmos implementados
0. locallineal
1. global
2. globallineal
3. local
4. semigloballineal
5. kband
6. semiglobal
```

Val:

```
Ingrese un comando: val

Valor actual de los pesos
match: 1
mismatch: -1
gap: -2
```

Match, mismach y gap:

```
Ingrese un comando: match

Valor actual
match: 1

Ingrese un comando: mismatch

Valor actual
mismatch: -1

Ingrese un comando: gap

Valor actual
gap: -2
```

Config:

```
Ingrese un comando: config match

Valor actual
match: 1

Ingrese un comando: config match 2

Valor actual
match: 2
```

Recursos:

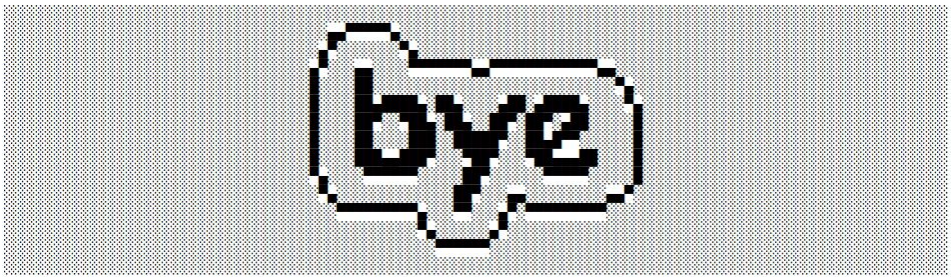
```
Ingrese un comando: recursos

Recursos utilizados
tiempo: 0.0017056465 segundos
memoria: 2759 bytes
```

Salir:

```
Ingrese un comando: salir

Recursos utilizados
tiempo: 0.0017056465 segundos
memoria: 2759 bytes
```




```
--------------------------------- Instituto Tecnológico de Costa Rica ---------------------------------
                                Ingeniería en Computación, Sede Cartago
                          Introducción a la Biología Molecular Computacional
                                          Segundo semestre 2017
                                        Profesor: Esteban Arias

-------------------------------------------- Integrantes --------------------------------------------
                                             Julian Salinas
                                            Johanna Elizondo
```

Conclusion analisis final: en general se ha obtenido el resultado esperado.

Ejemplos documentados

En esta sección se van a mostrar funciones importante que se han creado para el sistema, dando una breve explicación del mismo.

En la siguiente función de la clase 'BaseAligment' se inicializan las variables importantes para cada uno de los algoritmos, se está empleando herencia por lo que las diferentes clases de cada algoritmo pueden hacer uso de ello.

```
class BaseAlignment(object):

    # --------------------------------------------------------------------

    def __init__(self, v_text, h_text, match=1, mismatch=-1, gap=-2):

        self.v_text = "_" + v_text
        self.h_text = "_" + h_text

        self.match = match
        self.mismatch = mismatch
        self.gap = gap
        self.show_arrows = True

        self.shape = len(self.v_text), len(self.h_text)
        self.matrix = np.zeros(self.shape, dtype="int32")
        self.arrows = np.copy(self.matrix)
        self.arrows[:, :] = Row.NONE
        self.aligned_strings = None
```

La siguiente función se encuentra en la clase "SemiglobalAlignment", es aquí donde se construye la tabla del algoritmo global y semiglobal, adicionalmente se calcula la fila y columna inicial que es lo que diferencia a cada algoritmo.

```
@timeit
def calc_matrix(self):

    for i, j in it.product(range(1, len(self.v_text)), range(1, len(self.h_text))):
        neighbors = self.calc_neighbors(i, j)
        best_val = neighbors[np.argmax(neighbors)]
        self.matrix[i, j] = best_val
        self.arrows[i, j] = self.calc_arrow(best_val, neighbors)

    memit(self)
    return self.matrix
```

La siguiente imagen muestra el algoritmo diseñado para recorrer la tabla y encontrar la alineación de las cadenas, luego de dicha tabla ha sido creada. Esta funcion se encuentra en la clase “SemiglobalAlignment” y tanto el alineamiento global como semiglobal hacen uso de ella.

```
def traceback(self):
    v_align = ""
    h_align = ""
    matrix = self.matrix[:, :]
    h_text = self.h_text[0::]
    v_text = self.v_text[0::]
    i, j = self.calc_score_index()
    while i > 0 or j > 0:
        neighbors = self.calc_neighbors(i, j)
        if i > 0 and j > 0 and matrix[i, j] == neighbors[2]:
            v_align = v_text[i] + v_align
            h_align = h_text[j] + h_align
            i -= 1
            j -= 1
        elif i > 0 and matrix[i, j] == neighbors[0]:
            v_align = v_text[i] + v_align
            h_align = "_" + h_align
            i -= 1
        else:
            v_align = "_" + v_align
            h_align = h_text[j] + h_align
            j -= 1
    return v_align, h_align
```

Esta es la función de recorrido para el alineamiento global, se encuentra la clase ‘LocalAlignment’, ya que solo esta clase hace uso de dicha función.

```
def traceback(self):
    v_align = ""
    h_align = ""
    h_text = self.h_text[0::]
    v_text = self.v_text[0::]
    i, j = self.calc_score_index()

    neighbors = self.calc_neighbors(i, j)
    argmax = np.argmax(neighbors)
    maxval = neighbors[argmax]

    while maxval > 0:
        if maxval == neighbors[2]:
            v_align = v_text[i] + v_align
            h_align = h_text[j] + h_align
            i -= 1
            j -= 1
        elif maxval == neighbors[1]:
            v_align = "_" + v_align
            h_align = h_text[j] + h_align
            j -= 1
        elif maxval == neighbors[0]:
            v_align = v_text[i] + v_align
            h_align = "_" + h_align
            i -= 1

        neighbors = self.calc_neighbors(i, j)
        argmax = np.argmax(neighbors)
        maxval = neighbors[argmax]

    return v_align, h_align
```

La siguiente función es la encargada del recorrido de la alineación global y semiglobal con espacio lineal. Se puede visualizar como las filas y columnas se van eliminando conforme se van utilizando. Esta función se puede encontrar en la clase 'SemiglobaLinealAlignment'.

```
def traceback(self):

    v_align = ""
    h_align = ""
    self.matrixLineal = self.matrix
    h_text = self.h_text[0::]
    v_text = self.v_text[0::]
    i, j = self.calc_score_index(False)
    banderaR = False
    banderaC = False
    print(self.matrixLineal)
    while i > 0 or j > 0:

        neighbors = self.calc_neighbors(i, j)
        if banderaR:
            self.matrixLineal = np.delete(self.matrixLineal, i+1, 0)
            print('\n Delete Row')
            print(self.matrixLineal)
            banderaR = False

        if banderaC:
            self.matrixLineal = np.delete(self.matrixLineal, j+1, 1)
            print('\n Delete Column')
            print(self.matrixLineal)
            banderaC = False

        if i > 0 and j > 0 and  self.matrixLineal[i, j] == neighbors[2]:
            v_align = v_text[i] + v_align
            h_align = h_text[j] + h_align
            i -= 1
            j -= 1
            banderaR = True
            banderaC = True


        elif i > 0 and  self.matrixLineal[i, j] == neighbors[0]:
            v_align = v_text[i] + v_align
            h_align = "_" + h_align
            i -= 1
            banderaR = True

        else:
            v_align = "_" + v_align
            h_align = h_text[j] + h_align
            j -= 1
            banderaC = True

    return v_align, h_align
```

Esta es la implementación de uno de los comandos especiales del sistema, el de 'recursos'. Esta función se encarga de mostrar el tiempo invertido y memoria consumida por el sistema. Se encuentra en la clase 'GUI'.

```
def recursos(self, args):

    """
    Muestra el consumo actual de memoria y de tiempo. Si se especifíca
    de recursos utilizados durante la ejecución de lo contrario se mos
    """

    print("\n" + Font.BOLD + Font.PURPLE +
          "Recursos utilizados" +
          Font.END + Font.END)

    print("tiempo: %.10f segundos" % Timem.last_time)
    print("memoria: %s bytes\n" % str(Timem.last_memory_usage))

    if args.__contains__("totales"):

        print(Font.BOLD + Font.PURPLE +
              "Sumatoria de recursos utilizados durante la ejecución" +
              Font.END + Font.END)

        print("tiempo: %.10f segundos" % (time() - Timem.start_time))
        print("memoria: %s bytes \n" % str(Timem.total_memory_usage))
```

Problemas presentados

Unos de los mayores problemas presentados fue la dificultad de entender por completo el cada uno de los algoritmos, sobre todo los que se debían resolver usando espacio lineal y los que implementan costo por gap.

Uno de los requerimientos solicita que se debe mostrar la cantidad de memoria usada y la cantidad de tiempo de ejecución para cada uno de los algoritmos, esto debido a falta de experiencia provocó algunos problemas, sin embargo, el requerimiento fue implementado

Otros problemas encontrados se relacionan a detalles de implementación particulares, relacionados ya sea a errores de interpretación de los algoritmos en el momento, o a errores ocurridos por la falta de experiencia y dominio del lenguaje de programación específico utilizado.

Conclusiones y observaciones

1. Las herramientas que brinda la computación son de inmensa importancia para el área de la biología molecular, dicha situación ya es sabida ampliamente, pero el poner en práctica dichos conceptos ayuda a comprender mejor dicha importancia y su potencial a explotar en un futuro con el avance de la tecnología.
2. El uso del lenguaje Python facilita en gran medida al desarrollador el concentrarse aún más en el objetivo final de su trabajo, y despreocuparse un poco por luchar con las particularidades del medio de programación. Esto pues Python es un lenguaje muy flexible y fácil de manipular en sus niveles más básicos.
3. La mayoría de algoritmos de alineamiento se basan en el algoritmo de Needleman-Wunsch, al que luego agregan variaciones para obtener diferentes resultados. Con lo cual es importante dominar dicho algoritmo para facilitar el entendimiento de los demás.
4. La biblioteca Numpy para Python resulta en una de las herramientas más productivas, esto gracias a su simplicidad, eficiencia y facilidad de instalación.

Referencias bibliográficas

[1] Joao Meidanis, João Carlos Setubal (1997). “Introduction to Computational Molecular Biolog”.

[2] Germán González, “Algoritmo Needleman-Wunsch”, Recuperado de: http://www.bioinformaticos.com.ar/algoritmo-needleman-wunsch/

[3] Germán González, “Algoritmo Smith-Waterman”, Recuperado de: http://www.bioinformaticos.com.ar/algoritmo-smith-waterman/

[4] Virtual Amrita Laboratories Univerzaling Education, “Global alignment of two sequences - Needleman-Wunsch Algorithm”. Recuperado de: http://vlab.amrita.edu/?sub=3&brch=274&sim=1431&cnt=1

[5] Carnegie Mellon University, School of Computer Science. “String comparison”. Recuperado de: https://www.cs.cmu.edu/~ckingsf/bioinfo-lectures/globalalign.pdf

[6] Carnegie Mellon University, School of Computer Science. “Local Alignment”. Recuperado de: https://www.cs.cmu.edu/~ckingsf/bioinfo-lectures/local.pdf

[7] Carnegie Mellon University, School of Computer Science. “Space-Efficient Alignment”. Recuperado de: https://www.cs.cmu.edu/~ckingsf/bioinfo-lectures/linspace.pdf

[8] Stackoverflow – Programmer Community. “Medición de memoria de objetos”. Recuperado de: http://stackoverflow.com/questions/11301295/