TP final Heurística GRASP para MSA

Multiple Sequence Alignment (MSA)

El objetivo de la comparación de secuencias biológicas es descubrir similitudes funcionales entre ellas. Ácidos nucléicos y proteínas biológicamente similares pueden no tener una fuerte similitud de secuencia, pero aún nos gustaría reconocer la semejanza incluso cuando las secuencias apenas comparten lejanas similitudes.
Si la similitud es débil, la alineación de a pares puede fallar en identificar secuencias relacionadas porque las similitudes débiles por pares pueden fallar las pruebas estadísticas de
significancia. Si bien no hemos visto estas pruebas de significancia estadística, basta con saber que son el siguiente paso a un alineamiento y son las que nos informan si efectivamente las secuencias alineadas son similares.

Por ej., la β-galactosidasa es una enzima fundamental en la digestión de productos lácteos. La tienen los humanos y todo tipo de organismos, hasta bacterias. Y si bien es la misma enzima, con la misma función, no es exactamente idéntica. La secuencia de aminoácidos (de "letras", en lo que a nosotros respecta), varia no sólo entre especies, sino que hasta entre individuos de la misma especie. Y esto sucede con todas nuestras enzimas.

Entonces, si tuvieramos la β-galactosidasa de 2 organismos muy dispares y las alineásemos, es posible que a primera vista parezcan proteínas (las enzimas son una clase de proteínas), totalmente distintas. Nos estaríamos perdiendo nuestra respuesta.

Entonces, la comparación de 2 secuencias lejanamente similares puede no dar frutos. Sin embargo, la comparación simultánea de muchas secuencias a menudo permite encontrar similitudes que son invisibles en un alineamiento de a pares.

Complejidad del MSA

Recordarán que la complejidad de nuestro algoritmo de Needleman-Wunsch tenía una complejidad temporal y espacial de $O(mn)$, o $O(n^2)$, para mayor simplicidad. Esta

complejidad era consecuencia del doble *loop* anidado en el que a cada paso se resuelve un subproblema, tomando la decisión entre alguna de las siguientes 3 posibilidades:

1. Insertar un *gap* en la secuencia **A**
2. Insertar un *gap* en la secuencia **B**
3. Alinear 2 caracteres, uno de la secuencia **A** y el otro de la **B**

Esto era necesario para obtener un alineamiento global óptimo. Qué deberíamos hacer para obtener el alineamiento entre 3 secuencias, como el siguiente?

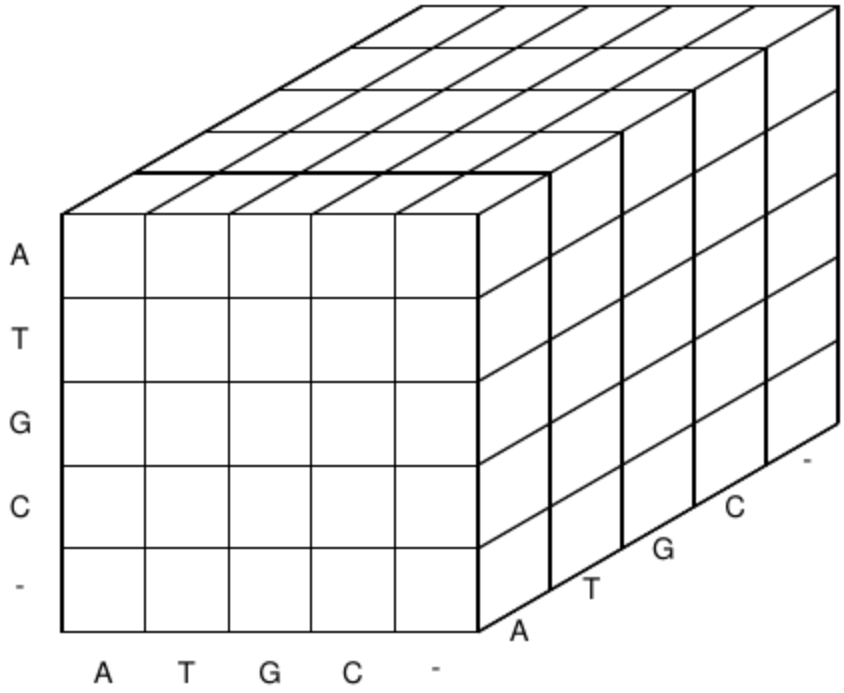
```
--T--CC-C-AGT--TATGT-CAGGGGACACG--A-GCATGCAGA-GAC
  |  || |  ||  | | | |||    || |  | |  | ||||   |
AATTGCCGCC-GTCGT-T-TTCAG----CA-GTTATG--T-CAGAT--C
 ||||| |   X|||| |          || XXX|||  | ||| |  |
-ATTGC-G--ATTCGTAT------GGGACA-TGGATGCATGCAG-TGAC
```

Si añadiéramos una tercera secuencia, naturalmente las posibilidades se expandrían:

1. Insertar 2 *gaps,* uno en la secuencia **A**, otro en **B**, alineados con un caracter de **C**
2. Insertar 2 *gaps,* uno en la secuencia **A**, otro en C, alineados con un caracter de **B**
3. Insertar 2 *gaps,* uno en la secuencia **B**, otro en **C** alineados con un caracter de **A**
4. Insertar un *gap* en la secuencia **A**, alineado con 2 caracteres, uno de **B**, y otro de **C**
5. Insertar un *gap* en la secuencia **B**, alineado con 2 caracteres, uno de **A**, y otro de **C**
6. Insertar un *gap* en la secuencia **C**, alineado con 2 caracteres, uno de **A**, y otro de **B**
7. Alinear 3 caracteres, de secuencias **A**, **B**, **C**

Notar que al igual que en el alineamiento de 2 secuencias, no ponderamos agregar un gap en cada secuencia, ya que eso no avanza el alineamiento y sólo implica pagar *gap_score* sin beneficio alguno.

Entonces, ahora cada subproblema depende de 7 subproblemas más pequeños y además tendremos un *loop* adicional por la secuencia extra. Así, nuestra tabla de memonización tendrá una dimensión adicional, sería un cubo. Así como nuestra matriz de scoring se convertiría en un cubo:



Y la complejidad de nuestro algoritmo pasaría a ser de $O(n^3)$ o, en términos más generales$O(n^k)$, donde k es el número de secuencias a alinear. Esta complejidad no resulta aceptable para una operación tan rutinaria y fundamental como la de hacer un alineamiento múltiple. Por eso todas las herramientas que realizan alineamientos múltiples, son heurísticas para este problema y no algoritmos.

Heurística

La heurística base de los algoritmos de MSA es la de alinear siempre 2 objetos. Y decimos objetos porque en nuestro caso, uno de ellos será una secuencia y otro será lo que se llama *Profile*.

Nuestra heurística empezará alineando 2 secuencias, obtendrá un alineamiento inicial y a ese alineamiento le irá agregando el resto de las secuencias, una por una. Ahora, sabemos alinear 2 secuencias, cómo se alinea 1 secuencia y 1 alineamiento? Bueno, en realidad no se alínea la nueva secuencia con el alineamiento, sino que se alinea contra el ya mencionado *profile*.

Profile

El p*rofile* es una abstracción sobre la noción de secuencia, puede representar la misma información que 1 secuencia, más información adicional. Podríamos decir que la secuencia es un *subset* de un *profile.* Mientras la secuencia sólo informa el caracter que corresponde a cada posición (en ácidos nucléicos, A, C, G o T), el *profile* está pensado para contener información de varias secuencias a la vez, por lo que, a cada posición, informa la **fracción** o el **conteo** de A, C, G o T. Veamos un ejemplo.

Un pequeño MSA. Aquí está toda la información respecto a un alineamiento, pero

T	C	G	G	G	G	g	T	T	T	t	t
c	C	G	G	t	G	A	c	T	T	a	C
a	C	G	G	G	G	A	T	T	T	t	C
T	t	G	G	G	G	A	c	T	T	t	t
a	a	G	G	G	G	A	c	T	T	C	C
T	t	G	G	G	G	A	c	T	T	C	C
T	C	G	G	G	G	A	T	T	c	a	t
T	C	G	G	G	G	A	T	T	c	C	t
T	a	G	G	G	G	A	a	c	T	a	C
T	C	G	G	G	t	A	T	a	a	C	C

El *profile* de conteo correspondiente a este alineamiento se vería así:

A:	2	2	0	0	0	0	9	1	1	1	3	0
C:	1	6	0	0	0	0	0	4	1	2	4	6
G:	0	0	10	10	9	9	1	0	0	0	0	0
T:	7	2	0	0	1	1	0	5	8	7	3	4

y uno fraccional sería así:

A:	.2	.2	0	0	0	0	.9	.1	.1	.1	.3	0
C:	.1	.6	0	0	0	0	0	.4	.1	.2	.4	.6
G:	0	0	1	1	.9	.9	.1	0	0	0	0	0
T:	.7	.2	0	0	.1	.1	0	.5	.8	.7	.3	.4

Nótese que es fácil obtener cualquiera de los 2 *profiles* a partir del alineamiento, pero no se puede recuperar un alineamiento a partir de un *profile.* El *profile* fraccional debe ser acompañado por el número de secuencias que está representado. Aún así, no se sabría a que secuencia corresponde cada nucleótido.

Entonces, si por ejemplo se nos pidiera el MSA de 10 secuencias, comenzaremos alineando 2 secuencias y luego alinearemos el profile resultante contra 1 de las 8 secuencias restantes, generando así un nuevo profile, que alinearemos con 1 de las, ahora, 7 secuencias restantes y así iremos consumiendo nuestras secuencias hasta terminar con 1 *profile*, y más importante aún, 1 alineamiento que contenga a las 10 secuencias.

Ya sabemos alinear 2 secuencias, pero cómo alineamos 1 secuencia y 1 *profile*?

Needleman-Wunsch para 1 secuencia y 1 *profile*

La buena noticia es que hay pocas diferencias con el algoritmo original.
En primer lugar, las estructuras de datos que manejaremos serán distintas y ahora deberemos retornar 1 alineamiento y 1 profile, eso no es mayor problema.

El único verdadero cambio estará en la forma de obtener el *score* de una operación. Pero antes, vamos a calcular el alineamiento de una columna de un MSA ya resuelto.

score de un MSA

Veamos, primero, la forma de obtener el *score* del alineamiento, mirando el ejemplo de la posición **5** del alineamiento anterior. Podemos hacer esto ya que al igual que en el alineamiento de pares de secuencias, el *score* global del alineamiento será igual a la suma de los *scores* de cada columna.

Cuál sería el *score* final de esta columna? Si sólo hubiera 2 secuencias, entonces el *score* sería el de match entre **G**s, digamos `score["GG"]`, si tuvieramos almacenada nuestra *scoring matrix* en un diccionario. O si una de esas 2 secuencias contuviera la `T`, entonces sería el *score* del *mismatch*, `score["GT"]`. Pero ambos apareamientos existen y se repiten numerosas veces. Así que el score de esta columna lo obtendremos por **suma de pares**: utilizando nuestra matriz de *scoring*, obtendremos los *scores* de todos los pares posibles que se puedan formar entre el caracter en la posición **5** de cada secuencia. Por ejemplo, la suma de los 9 pares que incluyen a la secuencia 1:

```
score["GT"] + score["GG"] + score["GG"] + score["GG"] + score["GG"] + score["GG"] + score["GG"] + score["GG"] + score["GG"] +...
```

Luego vendrán otros 8 pares en los que aparecerá la `T`:

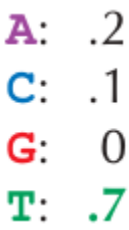
```
score["TG"] + score["TG"] + score["TG"] + score["TG"] + score["TG"] + score["TG"] + score["TG"] + score["TG"] +...
```

y luego otros 7 pares en los que aparecerá la 3ra secuencia, y así hasta completar los 45 posibles pares.

Como vemos, no hay algo particularmente nuevo y tampoco lo habrá en la forma de calcular el *score* entre una posición de *profile* y una de una secuencia.

score entre una posición de *profile* y una de secuencia

Veamos la columna 1 del *profile* anterior:



Supongamos que a este *profile* le queremos alinear una secuencia que empieza con **A**. Existe la posibilidad de poner un gap y en ese caso corresponde utilizar el *score* del *gap* (*gap penalty*), pero si quisieramos alinear la **A** de esta secuencia con esta posición del *profile*, cuál sería el *score*? Esta posición no es ni una **A**, ni una **C**, ni una T y definitivamente no es una **G.** Qué *score corresponde?*

La respuesta es, todos. Esta posición es **A** en un 20%, **C** en un 10% y **T** en un 70%. Entonces, el score de alinear esta posición con la **A** proveniente de la secuencia sería:

$$score_{posición\ 1} = 0.2\ score[AA] + 0.1\ score[AC] + 0.7\ score[AT] + 0.0\ score[AG]$$

Claro, el último término es irrelevante, pero lo dejaremos para completar la idea. Y si existiera un *gap* en alguna proporción, también incluíremos la *gap penalty* ponderada por su fracción correspondiente.

Y así es como calcularemos el *score* de cada operación posible al momento de resolver un subproblema. Ahora sí, estaremos en condiciones de escribir nuestro Needleman-Wunsch modificado.

TP final - primera parte - deadline: 30-11

1. Implemente su propia clase *Profile*
2. Modifique su implementación de Needleman-Wunsch para que sea capaz de alinear 1 secuencia y 1 *Profile.* Recuerde que el usuario debe ser capaz de ajustar la *gap penalty* y de ingresar su propia matriz de *scoring.* Acá va una de ejemplo:

https://s3-us-west-2.amazonaws.com/secure.notion-static.com/b3234696-b4ad-4651-b367-10d8c0ac3dd3/NUC.4.2

Ya tienen secuencias del TP anterior y acá también les dejamos más secuencias para que prueben su implementación:

https://s3-us-west-2.amazonaws.com/secure.notion-static.com/a7829aa3-bbac-4360-8bd8-4e19085c2478/10.fas

Los links están disponibles la versión online de este documento.

GRASP

1) Proponer un algoritmo goloso para el problema de alineamiento múltiple de secuencias. ***// hecho***

2) Aleatorizar el algoritmo anterior.

3) Proponer un algoritmo de búsqueda local para el problema de alineamiento múltiple de secuencias.

4) Variar parámetros y la estrategia del algoritmo de búsqueda local que optimicen el funcionamiento del mismo.

5) Construir un algoritmo GRASP para el problema de alineamiento múltiple de secuencias.
La entrada de su algirtmo será un `.fas` con secuencias a alinear (ej: `10.fas`), y una matriz de *scoring*.
La salida deberá ser un archivo de texto plano con el alineamiento y gráfico de *scoring* Vs # de iteraciones.

En la notebook de μbenchmark de funciones de orden tienen ejemplos de matplotlib para hacer el gráfico.
Esta vez no usaremos *notebooks*, el formato de entrega serán scripts de Python, en un repositorio de github y con un *Minimum Working Example* (MWE), para poder reproducir sus resultados.