

Programación, Máster de Bioinformática 2013-2014

Trabajo individual

El **peso en la nota final** de las distintas partes de la asignatura es: 2 puntos las cuestiones y programas de la parte de introducción a la programación, 2 puntos las cuestiones y programas de la parte de iniciación a las estructuras de datos y algoritmos, y **6 puntos** del trabajo individual (este).

El trabajo tratará sobre el problema de alineación múltiple de cadenas (información en: sección 6.10 del libro [An Introduction to Bioinformatics Algorithms](#), <http://bix.ucsd.edu/bioalgorithms/>; y en la página <http://www.biostat.wisc.edu/bmi576/lectures/multiple-alignment.pdf>). Todos los alumnos trabajarán sobre el mismo problema pero la función a optimizar será distinta y harán programas distintos.

Los alumnos se agruparán en grupos de tres, y cada grupo se asignará a un profesor distinto, que se encargará de indicar a cada alumno la función objetivo* a utilizar, de asesorarlo en la realización de la práctica, y de evaluarla. Una vez los alumnos han formado grupos de tres deben mandar un correo a domingo@um.es indicando los componentes del grupo, y se les asignará profesor.

En clase se verá un algoritmo de Programación Dinámica para alineación de dos cadenas y su implementación en Perl, y también la técnica de avance rápido, en la que se pueden utilizar heurísticas para intentar mejorar soluciones iniciales. Los alumnos trabajarán con estas dos técnicas para aplicarlas al problema que se les plantea.

La resolución del problema tendrá dos partes (que pueden repetirse o entremezclarse a criterio, justificado, del alumno):

- En una parte inicial se generarán soluciones, de forma aleatoria, haciendo combinaciones por pares de cadenas, o con otros métodos que el alumno decida y de los que justifique su selección.
- Una vez generada una o varias soluciones iniciales se programarán métodos de mejora (avance rápido) de las soluciones teniendo en cuenta la función objetivo que se quiere optimizar. Algunos ejemplos de técnicas heurísticas se pueden encontrar en <http://www.diva-portal.org/smash/get/diva2:3318/FULLTEXT02.pdf>.

La entrada de datos estará en un fichero con el siguiente formato:

la primera línea contendrá el valor de N y la segunda el de M, con N el número de cadenas y M la longitud de las cadenas (consideramos todas las cadenas de la misma longitud por simplicidad)

a continuación habrá N líneas, cada línea almacenando una cadena de ADN de longitud M.

La salida de datos será un fichero que tendrá en la primera línea la función objetivo calculada (un número real), y a continuación N líneas, cada una con las mismas cadenas de entrada en el mismo orden, e incluyendo en cada cadena los huecos (carácter "-") incluidos para obtener la alineación con que se ha obtenido esa función objetivo.

El trabajo consistirá en:

- 1) Una introducción breve de los problemas biológicos relacionados con el tipo de algoritmos que se implementan.
- 2) Desarrollar y explicar el algoritmo, incluyendo su pseudocódigo. Habrá que incluir
 - consideraciones sobre los requerimientos de memoria y tiempo de ejecución,
 - justificación de la forma de generación de los alineamientos iniciales y de las heurísticas utilizadas para intentar mejorar las soluciones iniciales. Dependerán de la función objetivo a optimizar
- 3) Implementar el algoritmo en un programa Perl.
- 4) Realización de experimentos para la validación y evaluación del programa. Se trata de comprobar que los resultados que da son "correctos" y estudiar experimentalmente el tiempo de ejecución. Para esto se utilizarán ficheros de entrada proporcionados por los profesores. Además de algunos ficheros de entrada se proporciona un programa de generación de entradas, que el alumno puede usar para realizar experimentos adicionales.

Cada alumno tendrá que **entregar** al profesor encargado de su trabajo una documentación que contenga los 4 puntos anteriores. El plazo de entrega de la documentación es el 17 de enero, y el profesor la revisará y puede citar al

alumno para la revisión del trabajo. Cada alumno también preparará una presentación de aproximadamente 20 minutos, y la expondrá en una sesión de presentaciones que se planificará para la semana a partir del 28 de enero.

Para la **evaluación** del trabajo se tendrá en cuenta:

- A) Iniciativa y autonomía del alumno en su desarrollo.
- B) Cuestiones formales de la documentación y la presentación del trabajo.
- C) El contenido de cada uno de los cuatro puntos del trabajo:
 - El punto 1) no tiene que desarrollarse en mucha profundidad ni se le asignará una valoración muy alta, pues la asignatura es de programación, pero debe incluirse para relacionar los temas de programación con problemas de biología.
 - En el punto 2) se trata de identificar las estructuras de datos con los que se trabaja y las operaciones sobre ellas para resolver el problema. Como paso previo a desarrollar el programa hay que identificar cómo resolver el problema computacional. Se trata de realizar un pseudocódigo, explicándolo y teniendo en cuenta si puede haber problemas de memoria o de tiempo de ejecución cuando se vaya a ejecutar, lo que nos ayudará a tomar decisiones sobre el algoritmo y su posterior implementación antes de empezar a desarrollar código. Dado que es un problema de optimización hay que justificar que las decisiones heurísticas que se tomen van encaminadas a mejorar la función objetivo.
 - La implementación del algoritmo en un programa (punto 3) es la parte más importante del trabajo (la asignatura es de programación) y será la que más se valore. Se tendrán en cuenta especialmente cuestiones de programación: facilidad de entendimiento del código, con inclusión de comentarios y nombres de variables significativos, la estructura del código, la correcta manipulación de las estructuras de datos... El programa se realizará en Perl, pero se valorará positivamente su implementación también en otro lenguaje visto en prácticas.
 - Una vez realizado un programa es necesario validarlo y evaluarlo (punto 4). En el tipo de problema con el que estamos tratando, al no conocerse la solución óptima ni poder encontrarse si el tamaño del problema no es muy pequeño, la validación consistirá en idear alguna forma de justificación de que los resultados son satisfactorios. Hay que justificar las pruebas que se han realizado y dar los resultados al menos con los ficheros de datos proporcionados. Los resultados serán el tiempo de ejecución y de valor de la función objetivo obtenidos. Habrá que sacar conclusiones en cuanto a evolución del tiempo de ejecución con el tamaño del problema, la idoneidad de las heurísticas utilizadas, y otros aspectos que el alumno considere relevantes.

* Algunos ejemplos de funciones de distancia y en algunos casos su significado biológico se pueden consultar en http://bioinformatics.psb.ugent.be/downloads/psb/Userman/treecon_distance.html. Y algunos ejemplos que se podrían utilizar son:

1. Sumatorio de las distancias entre todos los pares de secuencias, donde la distancia:

- a) se calcula según el número de caracteres en que no coinciden, sin contar los gaps. Sería un problema de minimización.
- b) se calcula según el número de caracteres en que no coinciden, contando los gaps. Sería un problema de minimización.
- c) se calcula según una matriz de distancias $d(x,y)$. Si consideramos distancias cero cuando coinciden y 1 como el caso más negativo, sería un problema de minimización.
- d) se calcula sumando las longitudes de los huecos en los que no coinciden los caracteres en las dos secuencias, sin incluir los gaps. Es un problema de minimización.
- e) se calcula sumando las longitudes de los huecos en los que no coinciden los caracteres en las dos secuencias, incluyendo los gaps. Es un problema de minimización.
- f) se calcula sumando las longitudes de las subcadenas coincidentes elevadas al cuadrado: en **AGTATG** y **AGCATG** tenemos cadenas coincidentes AG y ATG, con lo que el peso es $2*3+3*3$. Es un problema de maximización.
- g) Se calcula sumando las coincidencias pero multiplicando cada coincidencia por la distancia al final de la cadena (por la izquierda o derecha). Por ejemplo, en **ACGTA-G-** y **A-G-ACGT**, la primera coincidencia de A está a distancia 1 en cada cadena, por lo que se suma 2; la primera coincidencia de G está a distancia 3 en la primera cadena y 2 (no se cuentan los gaps) en la segunda, por lo que se suma 5; la segunda de A está a

distancia 2 en la primera cadena (por la derecha) y 3 en la segunda (por la izquierda) y se suma 5; y la segunda de G está a distancia 1 y 2, por lo que se suma 3. El peso total es 20. Es un problema de maximización.

2. Alternativamente calcular el peso utilizando todas las cadenas:

- h) se quiere maximizar el número de posiciones en que coinciden todas las cadenas.
- i) si tenemos N cadenas, el número de posiciones en que coinciden N cadenas se multiplica por N-1, en el que coinciden N-1 cadenas se multiplica por N-2,... , y se suman todos estos valores. Es un problema de maximización.
- j) se quiere maximizar el número de posiciones en que coinciden al menos X caracteres, con X menor que N.
- k) se asigna un valor de 1 a posiciones donde coinciden los caracteres de todas las cadenas, y cuando hay un único carácter distinto se asigna un valor dependiendo de una matriz de distancia entre caracteres. Por ejemplo si $d(A,T)=0.5$ y $d(A,G)=0.2$, en una posición donde todo sean A y aparezca una T se suma 0.5, y si lo que aparece es una G se suma 0.2.
- l) como en f) pero considerando coincidencias en todas las cadenas.
- m) contando coincidencias en todas las cadenas, pero con pesos distintos según el carácter. Es de maximización.