Pontificia Universidad Javeriana
Departamento de Ingeniería de Sistemas
Estructuras de Datos
Proyecto del curso, 2015-10
Tercera Parte

1. Descripción del problema

En general, la información genética de un organismo se encuentra contenida en su genoma. Para manipular la información del genoma de un organismo cualquiera, se usan secuencias de información conocidas como códigos genéticos. Estos códigos genéticos están compuestos de letras que representan las cuatro bases nitrogenadas del ácido desoxirribonucleico (ADN) o del ácido ribonucleico (ARN). Mientras en el caso del ADN las bases son adenina (A), timina (T), guanina (G) y citosina (C); para el ARN las bases son adenina (A), uracilo (U), guanina (G) y citosina (C).

Aunque hay muchos avances relacionados con el descubrimiento de los códigos genéticos de los organismos del planeta, mucho trabajo aún queda por hacer. Para guardar la información que se conoce (y señalar la que se desconoce) se usa un formato de códigos genéticos conocido como FASTA.

2. Descripción del proyecto

El objetivo del presente proyecto es construir un sistema que permita algunas manipulaciones sencillas sobre archivos que contienen códigos genéticos

2.1. Formato FASTA

Los archivos en formato FASTA (extensión .fa) son basados en texto. Pueden contener uno o varios códigos genéticos que componen un genoma. Cada código genético empieza con una línea de texto que tiene el formato:

```
>[descripcion de la secuencia]
```

Debe notarse que siempre empieza con el caracter '>' y justo después se presenta una cadena no vacía que describe la secuencia. Las siguientes líneas (hasta la siguiente secuencia o hasta el final del archivo) contienen los datos que describen el código genético. Estas líneas están justificadas y todas tienen un ancho constante, con la posible excepción de la última línea. Un ejemplo de un archivo FASTA puede ser:

```
>Full_SEQUENCE
CTCCGGTGAGAAATTTTGGGATGTATCAAATCACGGTCCTACTAC
TTACCGCCAACACGAGCCGGAACCCCCTAGATCAATTCATGCTTT
TCCCTTCACGCGAAGGAGTCGGAAGTGATCTGTATGAAGCTATTA
CCCTAGGTGGCCACACCTAC
>Incomplete_sequence
URDNSHVNKSCUANADDDVMMDVHRSRGNUNTSBTTMCGNBUUBTUMNAYKSTRYMVBWVNSC
SUUDDYVBCGNDRUURUBDHVGTYAVAVVSAKHUTSWCUBUUMSMDVDKBWRHHBHDBDWUDC
CAUWBRNYSTVTBBCKGDCMSNTKBNTRTNYRNWNWNVCSCNDDWHUHWRRTUMWNKHUBDWA
DUNYUVKHNKSDCYAVARTUWDVSTDVVMMVUHVBCDGVBSKKBHKSVHHGKSAADDVBRKSS
UKURTKTNAWYBKVRRBGKGBMGKUCGSMDTDTKCKVUYNVDWRWNBRCGKDWWUNBYMTGBN
YTAHCUTAGNBKWWGNDRKMNCDVDTKNRGVBRVMKGBKUBGBRHVUSTBCGDV
```

El signifcado de cada letra se explica en la Tabla 1.

2.2. Tercera entrega

A continuación se describen los componentes individuales que conforman la tercera entrega del presente proyecto. El sistema se implementará como una aplicación que recibe comandos textuales.

Código	Significado
A	Adenina
C	Citosina
G	Guanina
T	Timina
U	Uracilo
R	A o G
Y	C, T o U
K	G, T o U
M	A o C
S	C o G
W	A, T o U
B	C, G, T o U
D	A, G, T o U
H	A, C, T o U
V	A, C o G
N	A, C, G, T o U
X	Máscara
-	Espacio de longitud indeterminada

Cuadro 1: Relación de cada código del formato FASTA con su significado biológico (tomada de http://www.ncbi.nlm.nih.gov/blast/fasta.shtml).

2.2.1. Componente 1: Resumen de la información de un genoma

La primera entrega de este proyecto debe estar completamente terminada y funcional (ver documento del enunciado de la primera entrega).

2.2.2. Componente 2: Compresión y decompresión de archivos FASTA

La segunda entrega de este proyecto debe estar completamente terminada y funcional (ver documento del enunciado de la segunda entrega).

2.2.3. Componente 3: Relaciones entre bases de las secuencias

Objetivo: A partir de las secuencias de un genoma, utilizar representaciones en grafos de las secuencias para obtener información adicional sobre la distribución de las bases.

En el archivo FASTA las secuencias se presentan en bloques de bases con una indentación específica. De esta forma, cada secuencia puede interpretarse como una matriz de bases, organizadas en filas y columnas. Cada fila corresponde a una línea del texto de la secuencia, y cada columna almacena una única base, de forma que hay tantas columnas como el número de indentación. Para el archivo FASTA de ejemplo presentado anteriormente, la secuencia `Full_SEQUENCE` puede verse como una matriz de 4 filas y 45 columnas, mientras que la secuencia `Incomplete_sequence` puede corresponder a una matriz de 6 filas y 63 columnas. Cada base se ubica entonces en una fila `i` y una columna `j` de la secuencia.

Esta configuración facilita la construcción de un grafo sobre cada secuencia. Para esto, cada una de las diferentes bases corresponde a un vértice del grafo, y estos vértices se interconectan teniendo en cuenta los 4 vecinos (superior, inferior, izquierdo, derecho) más cercanos a cada base. Para una base ubicada en la posición `[i,j]`, estos vecinos estarían ubicados así:

- superior: en la misma posición de la fila anterior, es decir `[i-1,j]`
- inferior: en la misma posición de la fila siguiente, es decir `[i+1,j]`
- izquierdo: sobre la misma fila, en la siguiente posición, es decir `[i,j+1]`
- derecho: sobre la misma fila, en la posición anterior, es decir `[i,j-1]`

La siguiente figura ilustra las conexiones en el grafo para una base dada, de acuerdo a la descripción anterior.

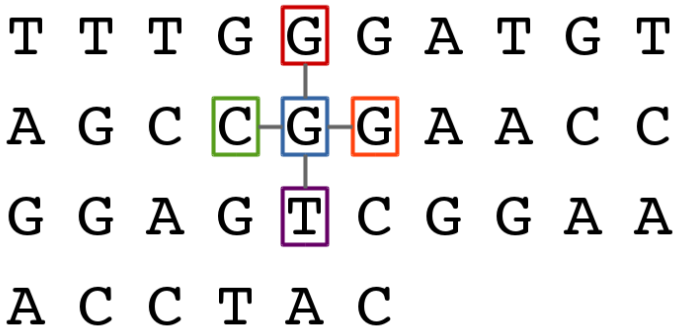

Figura 1: Ubicación de los vecinos para una base dada (azul): sobre la misma fila, a la derecha (naranja) y a la izquierda (verde); en la misma posición de la fila anterior (rojo), y en la misma posición de la fila siguiente (morado).

Interconectando todas las bases en la secuencia, se construye entonces el grafo que representa la secuencia, el cual se ilustra en la siguiente figura.

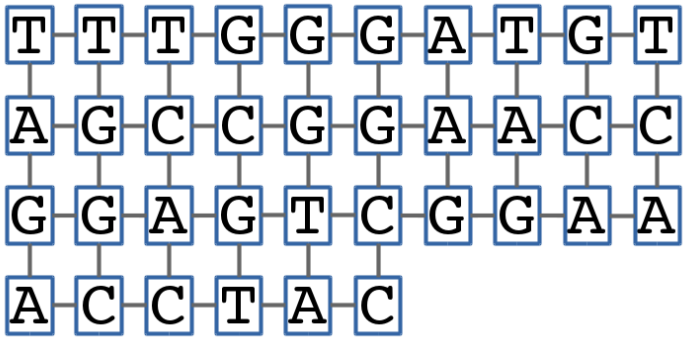

Figura 2: Construcción del grafo que representa una secuencia de bases.

Cada conexión entre bases vecinas tiene asignado un peso que se calcula teniendo en cuenta el código ASCII de la letra que representa cada base. Así, el peso de la arista que conecta el vértice en la posición `[i,j]` con el vértice en la posición `[x,y]` se calcula de acuerdo a la ecuación

$$C_{ij-xy} = \frac{1}{1 + |ASCII_{ij} - ASCII_{xy}|}$$

donde $ASCII_{ij}$ y $ASCII_{xy}$ corresponden al valor del código ASCII para la base en `[i,j]` y en `[x,y]`, respectivamente.

Teniendo en cuenta el grafo construído sobre cada secuencia del genoma, los comandos que se deben desarrollar son:

- **comando:** `shortest_path <sequence_description> <i> <j> <x> <y>`
  salida en pantalla:
  (La secuencia no existe) `Invalid sequence.`
  (Posición de base origen inválida) `The base in position [<i>,<j>] does not exists.`
  (Posición de base destino inválida) `The base in position [<x>,<y>] does not exists.`
  (La secuencia existe) `The shortest path between base in [<i>,<j>] and base in [<x>,<y>] is: .... The total cost of the path is: ...`

descripción: El comando debe imprimir en pantalla la secuencia de vértices del grafo que describen la ruta más corta entre la base ubicada en la posición `[i,j]` de la matriz de la secuencia `<sequence_description>` y la base ubicada en la posición `[x,y]` de la misma matriz. Así mismo, debe imprimir el costo total de la ruta, teniendo en cuenta el peso que tiene cada conexión entre bases.

- **comando:** `remote_base <sequence_description> <i> <j>`
  salida en pantalla:
  (La secuencia no existe) `Invalid sequence.`
  (Posición de base inválida) `The base in position [<i>,<j>] does not exists.`
  (La secuencia existe) `The remote base is located in [<a>,<b>], and the path between base in [<i>,<j>] and base in [<a>,<b>] is: .... The total cost of the path is: ...`
  descripción: Para la base ubicada en la posición `[i,j]` de la matriz de la secuencia `<sequence_description>`, el comando busca la ubicación de la misma base (misma letra) más lejana dentro de la matriz. Para esta base remota, el comando debe imprimir en pantalla su ubicación, la secuencia de vértices que describen la ruta entre la base origen y la base remota, y el costo total de la ruta.

2.3. Interacción con el sistema

La interfaz de la aplicación a construir debe ser una consola interactiva donde los comandos correspondientes a los componentes serán tecleados por el usuario. El indicador de línea de comando debe ser el caracter '$'. Para cada comando, se debe incluir una ayuda de uso que indique la forma correcta de hacer el llamado (i.e. el comando '`help <command>`' debe existir, además de los descritos anteriormente). El comando debe presentar en pantalla los mensajes de resultado especificados antes. Los comandos de los componentes deben ensamblarse en un único sistema (es decir, funcionan todos dentro de un mismo programa, no como programas independientes por componentes).

3. Evaluación

Las entregas se harán en la correspondiente actividad de Uvirtual, hasta la media noche del día indicado. Se debe entregar un archivo comprimido (únicos formatos aceptados: .zip, .tar, .tar.gz, .tar.bz2, .tgz) que contenga documentos (único formato aceptado: .pdf) y código fuente (.h, .hxx, .hpp, .c, .cxx, .cpp). Si la entrega contiene archivos en cualquier otro formato, será descartada y no será evaluada, es decir, la nota definitiva de la entrega será de 0 (cero) sobre 5 (cinco).

3.1. Entrega (viernes 5 de junio de 2015)

Componente 3 completo y funcional. Además, entregar lo que no haya sido entregado en las fechas anteriores. Esta entrega tendrá una sustentación (del proyecto completo) durante el viernes 5 de junio de 2015 entre las 9am y las 6pm. Esta entrega se compone de:

- (25 %) Completar la funcionalidad que aún no haya sido desarrollada o completada de la primera y segunda entregas.
- (30 %) Documento de diseño. El documento de diseño debe seguir las pautas de ingeniería que usted ya conoce: Diagrama de clases (si ya vio ADOO o POO) o diseño con precondiciones, poscondiciones e invariantes (si no ha visto ADOO). Además, se exigirá un(os) esquemático(s) que resuma(n) el(los) problema(s) y su(s) solución(ones) gráficamente.
- (45 %) Código fuente compilable en el compilador gnu-g++ (versión 4.0.0, como mínimo). Este porcentaje de la entrega será un promedio de la evaluación de cada comando.