Estructuras de Datos (Gestión), curso 2006/07

Práctica única – Secuenciado de Genes (Shotgun análisis)

Introducción

Como todos sabéis, la información genética se encuentra codificada en las hebras de ADN en la forma de una secuencia de bases que son de cuatro tipos distintos (A,C,G y T). Por lo tanto esa información consiste, en principio, en una gigantesca cadena de caracteres (con sólo cuatro caracteres posibles).

No es necesario resaltar la importancia que ha cobrado en la actualidad la secuenciación del genoma tanto de personas como de todo tipo de organismos, y los grandes avances que se espera que esa información proporcione al campo de la medicina. Lo que seguramente sea menos conocido es que en esta actividad la informática, en particular el diseño de algoritmos adecuados, juega un papel esencial.

El obtener la secuencia de bases de una hebra de ADN es un proceso extremadamente difícil y laborioso de realizar por métodos bioquímicos (los que estén interesados pueden obtener información consultando la dirección http://en.wikipedia.org/wiki/DNA_Sequencing y los enlaces asociados). En el momento actual sólo se puede realizar de forma semiautomática para hebras cortas, que tengan menos de unas 500 bases (además cuanto más larga sea más tiempo se tarda y menos fiables son los resultados). Sin embargo los genomas relevantes constan de millones de bases (el genoma humano consta de unos 3000 millones de bases).

Por lo tanto los métodos actuales se basan en obtener la secuencia de muchos fragmentos cortos del genoma y a partir de esa información intentar reconstruir la secuencia de todo el genoma: Aquí es donde entra en juego la informática.

El análisis Shotgun (una traducción posible sería análisis *por perdigonada*) se basa en la siguiente idea:

- Se hacen múltiples copias de la hebra que queremos secuenciar (esto es sencillo gracias al método PCR).
- Se rompen esas copias en múltiples fragmentos al azar.
- Se secuencian unos cuantos de esos fragmentos (los que tengan un tamaño adecuado, no mayor de 500 bases).
- Se intenta reconstruir la secuencia de esa hebra a partir de la secuencia de los fragmentos. Aquí es vital que en las secuencias de esos fragmentos existan muchos **solapamientos** (que cada base esté representada en varios fragmentos).

Esta última etapa supone resolver lo que se denomina el **problema de la supercadena minimal** (*shortest superstring*): Encontrar la secuencia de longitud mínima que contiene a todos las secuencias proporcionadas.

El análisis shotgun no garantiza que se vaya a obtener siempre la secuencia correcta (existen problemas cuando hay muchas repeticiones de una subsecuencia en el genoma), pero a cambio permite hacer un análisis muy rápido, y cuando se la combina con otras técnicas más laboriosas se obtienen resultados muy aceptables.

Descripción formal

El problema **de la supercadena minimal** se puede definir de la siguiente forma: Dadas *p* cadenas de caracteres $\mathbf{C}_i$, con $i \in [1\ldots p]$**,** de longitudes $n_1 \ldots n_p$, que llamaremos **fragmentos** (es decir, el fragmento *i*-ésimo se representa por el vector $C_i[1..n_i]$), el objetivo es obtener una cadena **S**[1..*m*] que cumpla que:

- Todo fragmento es una subcadena de S: Para todo fragmento *i* existe por lo menos una parte de S que es igual a él:

  $$\forall i : \exists j : \mathbf{C}_i[1..n_i] = \mathbf{S}[j..j+n_i-1]$$

- De entre todas las cadenas **S** que cumplan lo anterior queremos la de menor longitud posible (*m* mínimo), o si hay varias con la misma longitud mínima *m*, cualquiera de ellas.

Por ejemplo, si la entrada consistiera en los siguientes fragmentos: “CT”, “EST”, “URAS”, “STRU”, “UCTU”, “CTUR” y “TRUC”, la supercadena minimal sería “ESTRUCTURAS”:

```
E S T
  S T R U
    T R U C
        C T
        C T U R
            U R A S
-----------------------
E S T R U C T U R A S
```

Este problema es NP-Completo, y por lo tanto su solución exacta tiene un orden no polinómico. Sin embargo existen varios algoritmos voraces que, aunque no garantizan el obtener una supercadena minimal, si permiten obtener una supercadena en tiempo polinómico y, en la mayoría de los casos, minimal.

Nuestro objetivo será implementar uno de éstos algoritmos voraces, que se detalla a continuación:

> Sea **C** el conjunto original de fragmentos.
> **repetir**
> Quitar de **C** los dos fragmentos, **x** e **y**, cuyo *solapamiento* sea máximo.
> Sea **z** el resultado de *fusionar* **x** e **y**
> Insertar **z** en **C**
> Quitar de **C** todo fragmento *incluido* en **z** (salvo el propio **z**)
> **hasta_que** sólo quede un fragmento en **C**
> El fragmento que queda en **C** es la solución

Como se puede observar el algoritmo consiste en un bucle que en cada iteración sustituye dos fragmentos (o más) por otro, y por lo tanto se itera como máximo *p* veces (*p* es el número original de fragmentos). Las operaciones que marcan en cursiva tienen el significado siguiente:

- El **solapamiento** de dos cadenas **x**[1..n_x] e **y**[1..n_y] se define como la longitud de la mayor subcadena *final* de **x** que es igual a la subcadena *inicial* de **y**: Es decir el mayor valor *l* para el que $\mathbf{x}[n_x - l + 1 \ldots n_x] = \mathbf{y}[1\ldots l]$

Por ejemplo, las cadenas "CTUR" y "URAS" tienen un solapamiento de 2, ya que los dos últimos caracteres de "CTUR" son iguales a los dos primeros de "URAS". Esta función no es simétrica: El solapamiento de "URAS" y "CTUR" es 0.

- La **fusión** de dos cadenas consiste en concatenarlas pero omitiendo en una de ellas la parte que se solapa. Por ejemplo, al fusionar "CTUR" y "URAS" se obtiene la cadena "CTURAS", y al fusionar "URAS" y "CTUR" se obtiene "URASCTUR".
- Una cadena, **x**[1..n_x], está **incluida** en otra, **y**[1..n_y], si se cumple que:

$$\exists i : \mathbf{y}[i \ldots i + n_x - 1] = \mathbf{x}[1 \ldots n_x]$$

Si existen varios pares de fragmentos con el mismo valor máximo de solapamiento, se escoge cualquiera de ellos.

Requisitos de la Práctica

La práctica consistirá en crear **DOS** programas que resuelvan el problema de la obtención de la supercadena "minimal" mediante el algoritmo voraz detallado en la sección anterior.

Ambos programas se comportarán de la misma manera: Pedirán el nombre de un fichero de texto donde se encuentran los fragmentos, aplicarán el algoritmo y escribirán la supercadena resultante.

Los ficheros de los fragmentos tienen el siguiente formato: La primera línea constará de un entero que indica el número de fragmentos, *p*, y a continuación seguirán *p* líneas cada una de ellas indicando un fragmento. Para el ejemplo propuesto anteriormente, el contenido del fichero sería:

```
7
CT
EST
URAS
STRU
UCTU
CTUR
TRUC
```

Los dos programas se diferencian únicamente en la forma en que se implementa el algoritmo:

Primer programa

En el primer programa los fragmentos se representan mediante vectores de caracteres o bien mediante el tipo o clase STRING del lenguaje elegido. El conjunto de fragmentos se representa mediante un vector de fragmentos sin imponer ninguna estructura u ordenación especial. Es decir, en este primer programa NO se debe utilizar ninguna estructura de datos avanzada ni hacer esfuerzos para optimizar la eficiencia de las operaciones *solapamiento, fusión, inclusión,* o el encontrar los fragmentos de máximo solapamiento.

Segundo programa

En el segundo programa el objetivo es optimizar la eficiencia (respecto al tiempo) al máximo, mediante el uso de las estructuras de datos más adecuadas para las operaciones de *solapamiento*, *fusión*, o encontrar los fragmentos de máximo solapamiento. Se puede usar cualquier estructura, aunque no se haya contemplado en la asignatura, y se anima a los alumnos a buscar en la red información sobre el tema. Lo que no está permitido es cambiar de algoritmo o modificar su estructura: Tiene que corresponder al esquema mostrado.

Documentación que se debe entregar

Se deberá entregar el código fuente de los programas que resuelven la práctica (se recomienda el uso del lenguaje Eiffel, aunque alternativamente se permite realizarlo en Java, Free-Pascal o Delphi/Kylix) y un documento (en formato Word, Latex (DVI) o Acrobat PDF) donde se indique lo siguiente:

- Nombre de los alumnos que han realizado la práctica.
- Una evaluación de la eficiencia del primer programa en función de los parámetros *m* y *n* para el siguiente caso: Todos los fragmentos tienen el mismo tamaño, *n*, y el número de ellos es $p = 10\ m/n$ donde *m* es el tamaño de la supercadena.
- Una breve descripción (1 o 2 páginas) del tipo de estructuras de datos que se han usado en el segundo programa, donde se han obtenido y la razón para usarlas.
- Una breve evaluación de la mejora de eficiencia obtenida en el segundo programa.

Las prácticas se enviarán por **correo electrónico** (cvaca@infor.uva.es) al profesor de la asignatura, y la fecha límite de presentación es el **26 de enero de 2007**.

Comprobación de los programas

En la página de la asignatura se publicarán casos de prueba para que podáis evaluar vuestros programas.