Universidad Simón Bolívar
Departamento de Computación y Tecnología de la Información

CI-2692 – Laboratorio de Algoritmos y Estructuras II
Sep-Dic 2013

Proyecto 4 (25%)

Entrega: 30 de enero de 2013 (jueves S.12) antes de la 1:30 PM en el espacio de su grupo en el Aula Virtual.
Corrección y revisión (presencial): mismo día desde las 2:30PM
No se aceptarán retrasos.

Contexto

Su grupo acaba de integrarse a la unidad de investigación a cargo del *Explorador Submarino FUKAI*, un robot semi-autónomo utilizado para explorar las áreas más profundas del océano.

La construcción del FUKAI requirió de materiales y técnicas muy particulares para que éste pudiera resistir la intensa presión del fondo del mar. Por esta razón, es comparativamente limitado en memoria y poder de procesamiento. Además, los recursos del procesador del FUKAI deben utilizarse eficientemente ya que en el mismo se ejecutan continuamente procesos relacionados a múltiples proyectos de investigación y a la navegación del robot.

El FUKAI recientemente realizó un descubrimiento de gran importancia: fragmentos de secuencias de bases nitrogenadas, las cuales pudieran estar codificando información genética y por lo tanto ser evidencia de una nueva forma de vida.

Se le ha asignado a la unidad de investigación del FUKAI como misión prioritaria la recolección y análisis de la mayor cantidad posible de fragmentos de este tipo. Por tanto se planea enviar una actualización de software al robot con programas orientados a cumplir este objetivo.

Su grupo tiene como misión la elaboración del módulo de almacenamiento de las secuencias encontradas. Su módulo deberá comunicarse con otros, como el módulo de observación, módulo de corrección de errores y módulo de análisis.

Estructura de datos

El objetivo básico del módulo a desarrollar es el registro de las secuencias de bases observadas por el FUKAI, incluyendo el número de veces que haya sido observada cada una.

Si bien el ADN presente en todos los seres vivos hasta ahora conocidos en el planeta codifica la información genética utilizando 4 bases nitrogenadas distintas, las secuencias encontradas por el FUKAI tienen la peculiaridad de consistir únicamente en **2** bases, denotadas con las letras '**A'** y '**T'**. La información de una secuencia puede entonces visualizarse como un *string* compuesto por estos dos caracteres, p. ej: ATTAATTTA.

Esto, combinado con el hecho de que muchos de los fragmentos poseen los mismos inicios de secuencia, ha llevado a determinar que el uso más eficiente de la memoria se obtendrá almacenando las secuencias en un **árbol binario**. Bajo este esquema, cada nodo del árbol registra la cantidad de veces que ha sido observada una secuencia en particular. Además, la

secuencia resultante de anexar una ‘A’ al final de dicha secuencia queda representada por el **hijo izquierdo** del nodo, mientras que la anexión de una ‘T’ es representada análogamente por el **hijo derecho**. Cada uno de los hijos puede ser nulo para significar la no-observación de secuencias más largas.

Es decir, la observación de la siguiente serie de secuencias:

```
ATA
ATT
ATT
T
TT
```

sería representada por el árbol:

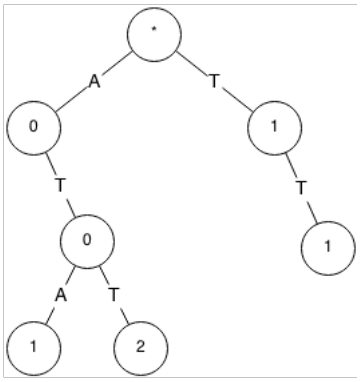


Figura 1
Nota: el nodo raíz representa la secuencia vacía. Por conveniencia su contador tiene siempre valor nulo.

Formalmente, se puede definir el tipo de datos a usar como:

> **freetype** *arbbinp*(int) ::= *avac* | *nodo*(int, *arbbinp*(int), *arbbinp*(int))

La forma abstracta de este tipo de datos *arbbinp*(*T*) es la misma que es definida como *arbbin”*(*T*) en [1].

Operaciones

Su módulo se comunicará con los otros vía archivos de texto. Debe ejecutarse mediante el comando:

```
> python3 fukai_gen.py <arch_entrada> <arch_salida>
```

<arch_entrada> será un archivo de texto que contendrá en cada línea una de las instrucciones que siguen a continuación. Algunas de las instrucciones deben producir salida escrita, la cual debe agregarse secuencialmente al archivo de texto *<arch_salida>*.

Al iniciarse la ejecución, el programa debe inicializar inmediatamente la estructura de datos. Las operaciones a ejecutar sobre dicha estructura siguen a continuación.

a) Lectura
Entrada: `GET <secuencia>`
Salida: *<secuencia> <cantidad>*
Imprime la cantidad de veces que ha sido observada la secuencia hasta el momento. En caso de que la secuencia no haya sido observada la cantidad es 0.

b) Adición
Entrada: `ADD <secuencia>`
Salida: <ninguna>
Indica una nueva observación de una secuencia. Debe aumentarse el registro del número de veces que ha sido observada la secuencia en 1.

c) Listado
Entrada: `GETALL`
Salida:
```
<secuencia0> <cantidad0>
<secuencia1> <cantidad1>
...
<secuenciaK> <cantidadK>
```
Imprime en cada línea la cantidad de veces que ha sido observadas cada una de las secuencias registradas hasta el momento. Las secuencias deben ser impresas en orden lexicográfico. Note que las cantidades siempre deben ser mayores a 0.

d) Longitud Máxima
Entrada: `MAXLENGTH`
Salida: `maxlength == <longitud>`
Imprime la longitud de la secuencia más larga registrada hasta el momento.

e) Eliminación
Entrada: `DELETE <secuencia>`
Salida: <ninguna>
Indica que el módulo de corrección de errores ha detectado que las observaciones realizadas hasta ahora de la secuencia especificada fueron falsos positivos, por lo que se debe registrar que no ha sido observada.

Para preservar el uso de la memoria, después de la eliminación el estado del árbol debe ser idéntico al que tendría si la secuencia nunca hubiera sido agregada. Es decir, deben borrarse los nodos que se hayan vuelto innecesarios, lo cual también se conoce como "podar el árbol". **Sugerencia:** Existe una expresión que permite determinar si un nodo es necesario o no, descúbrala.

Si la secuencia especificada no existe, se debe escribir como salida el string:
```
ERROR: Cannot DELETE.
```

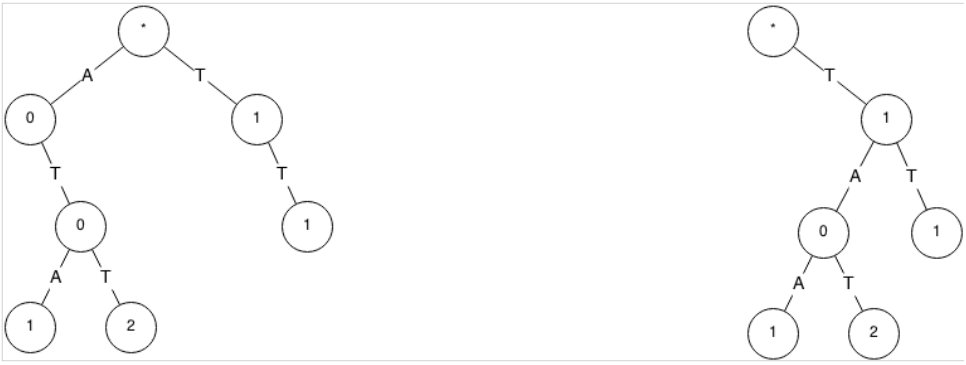
f) Reemplazo
Entrada: `SET <secuencia> <cantidad>`
Salida: <ninguna>
Indica que el módulo de corrección de errores ha detectado fallas de lectura, tras lo cual ha determinado el valor correcto. Debe fijarse la cantidad de veces que ha sido observada la secuencia a `<cantidad>`, independientemente del valor previamente registrado. En caso de que la cantidad sea cambiada a 0, debe podarse el árbol (si es necesario).

g) Migración
Entrada: `CHANGE <secuenciaOrigen> <secuenciaDestino>`
Salida: <ninguna>
Indica que el módulo de corrección de errores ha detectado un error recurrente en la interpretación de fragmentos que inician con `<secuenciaOrigen>`, y determinado que la interpretación correcta del inicio de la secuencia es `<secuenciaDestino>`.

Es decir, siendo *f* un fragmento que inicia con s_o, y siendo *f'* el resultado de sustituir s_o por s_D en *f*, entonces tras la instrucción `CHANGE` s_o s_D, el estado del árbol debe ser el que hubiese resultado si por cada adición de todo fragmento *f* en su lugar se hubiese añadido su correspondiente fragmento *f'*.

Ejemplo:

El árbol resultante de observar las secuencias	tras la operación	debe resultar idéntico al árbol resultante de observar las secuencias
ATA ATT ATT T TT	CHANGE AT TA	TAA TAT TAT T TT



La implementación de esta operación se pudiera realizar iterando sobre las secuencias afectadas y para cada una realizar las adiciones, eliminaciones o reemplazos necesarios. Sin embargo, dicha implementación tendría un orden proporcional a la cantidad de nodos afectados, por lo que sería prohibitivamente costosa para árboles de gran tamaño.

En cambio, mediante la manipulación adecuada de las referencias del árbol, es posible lograr una implementación proporcional a la longitud de las secuencias involucradas en la operación. Implemente la operación utilizando esta segunda estrategia.

Precondición: Esta operación tiene como precondición que en el árbol no existan fragmentos que inicien con `<secuenciaDestino>`. Si esta condición no se cumple, en vez de ejecutarse la operación se debe escribir como salida el string:

```
ERROR: Cannot CHANGE. Use CHANGEMERGE instead.
```

h) Fusión

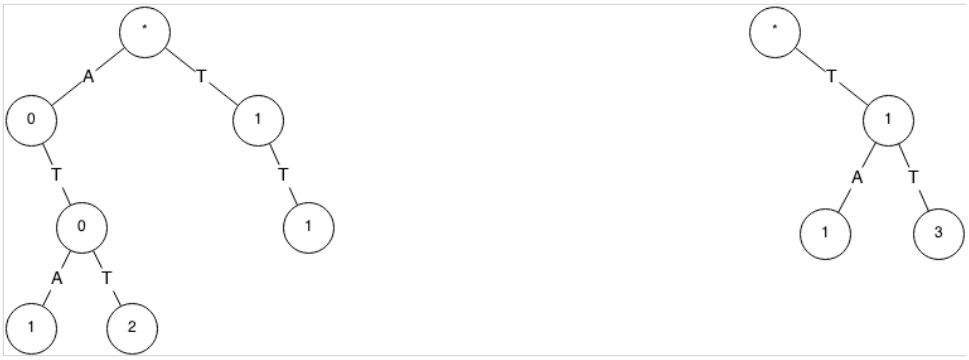
Entrada: `CHANGEMERGE <secuenciaOrigen> <secuenciaDestino>`

Salida: <ninguna>

Tiene la misma semántica que la operación de Migración, pero sin la restricción de no-existencia de fragmentos que inicien con `<secuenciaDestino>`.

Ejemplo:

El árbol resultante de observar las secuencias	tras la operación	debe resultar idéntico al árbol resultante de observar las secuencias
ATA ATT ATT T TT	CHANGEMERGE AT T	TA TT TT T TT



Implemente esta operación basándose en su implementación de la operación de Migración. Notará que el costo de la operación es proporcional a la similitud entre el sub-árbol definido por *<secuenciaOrigen>* y el sub-árbol definido por *<secuenciaDestino>*, siendo el peor caso cuando ambos sub-árboles son idénticos.

Sugerencia: Por otra parte, preste especial atención al caso especial que se produce cuando estos dos sub-árboles no son disjuntos.

i) Impresión
Entrada: `PRINT` *<string>*
Salida: *<string>*
Replica *<string>* en el archivo de salida. Puede utilizarse para organizar o dividir el archivo de salida.

Formato de los parámetros (tanto en la entrada como en la salida):

<secuencia> consiste exclusivamente en uno o más caracteres 'T' ó 'A' (en mayúsculas) seguidos, sin delimitadores al inicio o final de la secuencia. Ejm: `TTAAAATTTTATA`.
<cantidad> y *<longitud>* son `int`.
<string> es un `string` adecuadamente delimitado con comillas.

Requerimientos de la entrega

1) Cada integrante del grupo deberá imprimir, firmar y entregar a su profesor la *"Declaración de Autenticidad para Entregas"*, cuya plantilla se encuentra en el Aula Virtual del curso.

2) La entrega del proyecto será directamente en el espacio de su grupo en el Aula Virtual. Deposite un archivo comprimido que contenga los archivos de código fuente, así como otros que sean necesarios para la ejecución de la aplicación. El nombre del archivo debe ser `Proyecto4-XX.tar.gz`, donde XX es el número de su grupo.

3) Todos sus archivos de código deben tener un encabezado con los nombres y números de carnet de ***ambos*** miembros del equipo.

4) La entrega del proyecto debe realizarse **ANTES** de la 1:30 PM del 30 de enero de 2013 (jueves S.12). Se recomienda realizar la entrega con antelación y considerar imprevistos como fallas de conectividad.

El no cumplimento de los requerimientos podrá resultar en el rechazo de su entrega.

Preguntas y consultas

Se recomienda hacer preguntas con la mayor antelación posible. A fin de realizar las consultas de manera pública y persistente, coloque sus preguntas en el foro del Aula Virtual.

Referencias

[1] Ravelo, J. y Fernández, K. Tipos Algebraico-Libres. http://ldc.usb.ve/~crodrig/cursos/ci2612/MaterialTALibres.pdf

---

Saúl González D. (esaulgd) 2013.01.09