Universidad Simón Bolívar
Depto. de Computación y T.I.
Taller de Algoritmos y Estructuras III (CI-2693)
Septiembre – Diciembre 2.012

Proyecto 4

Composición de Funciones

Por Eduardo Blanco

Un grupo de investigadores de la USB ha estado trabajando en la definción de mecanismos que permitan el desarrollo de aplicaciones complejas en base a un conjunto de funciones simples contenidas en algún repositorio. En otras palabras, estos investigadores están diseñando estrategias que permitan seleccionar y coordinar un conjunto apropiadado de funciones de manera de llevar a cabo actividades más complejas. Estas actividades serán descritas en función de las entradas y salidas de la misma forma en como se define el dominio y el rango de una función (**A: Dom → Ran**).

A tal fin una actividad se define como un par de tuplas (E, S), en donde E representa los tipos de los datos que serán las entradas de la actividad (dominio) y S los tipos de los datos que se deben generar (rango). Sin embargo, para poder seleccionar y coordinar funciones apropiadas, éstas también deben ser descritas de la misma manera, es decir, la actividad realizada por una función será descrita en función de su dominio y su rango.

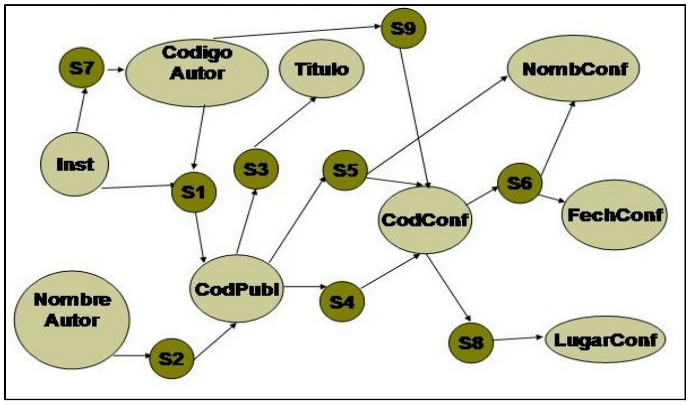
Como primer paso, los investigadores definieron cómo representar las dependencias existentes entre funciones con el fin de definir un mecanismo de planificación para la ejecución de un conjunto de funciones en forma coordinada. Las dependencias entre estas funciones se da a nivel de los tipos de datos que éstas necesitan para invocarse y los tipos de datos que éstas retornan. Todas las funciones están implementadas en un lenguaje que les permite retornar uno o más valores por cada invocación, es decir, el rango puede ser el producto cartesiano de dos conjuntos. Por ejemplo:

f: A x B x C → D x E,

f es una función que toma tres parámetros de tipos A, B y C respectivamente. Adicionalmente, si **(a, b, c) Є A x B x C**, entonces la invocación de **f(a, b, c)** retorna una tupla, **(d, e) Є D x E**, con dos valores, el primero de tipo

D y el segundo de tipo **E**.

Los investigadores acaban de establecer que una manera bastante flexible de representar este problema es usando grafos bipartitos: grafos cuyo conjunto de vértices **V** se puede particionar en dos subconjuntos disjuntos **U y W**, tal que E ⊆ UxW ∪ WxU, es decir, todas las aristas tienen origen en uno de los conjuntos y destino en el otro conjunto. En este caso particular los vértices en **W** denotan funciones y los vértices en **U** denotan tipos de datos. La siguiente figura muestra cómo se representan estas dependencias usando este tipo de grafos.



En la figura, las funciones están denotadas por el color verde oscuro y los tipos de datos por el color verde claro. De esta manera, note que la función **S2** recibe un sólo parámetro de tipo *NombreAutor* y genera un dato de tipo *CodPubl*. Asímismo, la función **S5** recibe un parámetro de tipo CodPub y genera un par de valores de retorno de tipos *CodConf* y *NombConf*.

Este tipo de representación establece una dependencia entre funciones de la siguiente manera: la función **S1** sólo puede ser invocada cuando existe al menos un dato de tipo *Inst* y otro de tipo *CodigoAutor*; y después de ser invocada generará al menos un dato de tipo *CodPubl*.

El problema

Dado un grafo **G=(V, E)**, donde V = W ∪ U y E ⊆ UxW ∪ WxU, este grupo de investigadores le pide a Ud que diseñe una aplicación que identifique composiciones de funciones que sean capaces de dar respuesta a las actividades especificadas. Adicionalmente, Ud debe identificar, para cada actividad, la composición de funciones que requiera el menor tiempo en ejecutarse.

Su aplicación debe poderse correr de la siguiente forma:

java Main <archivo.in> <archivo.out>

donde <archivo.in> es el nombre del archivo de entrada y <archivo.out> es el nombre del archivo de salida

Ud debe generar los casos de pruebas que permitan probar la correctitud de su proyecto.

La corrida del proyecto se realizará en base a los casos de prueba desarrollado por el grupo profesoral. Con el fin de que este programa funcione en forma correcta con las clases que Ud implemente, es importante que Ud. respete las especificaciones del proyecto.

Entrada

La primera línea del archivo estará un número F que indicará el número de funciones. Luego vendrán F líneas con las especificaciones de las funciones. Cada una de estas líneas tendrá una cadena con el siguiente formato:

f: (A, B, C), (D, E) t

en donde f es el nombre de la función, **(A, B, C)** y **(D, E)** representan el dominio y rango de la función respectivamente; y t es un entero representa el tiempo de ejecución medido en milisegundos. Note que cada función puede tomar varios parámetros y retornar varios valores al ser invocada. En ambos casos Ud puede asumir al menos un parámetro, es decir, al menos uno de entrada y al menos uno de salida.

En la siguiente línea, vendrá un entero T, que indica el número de actividades para las que se debe identificar una secuencia (composición) de funciones que permita llevar a cabo la actividad correspondiente. Luego vendrá un conjunto de T líneas donde se especifican las actividades que se deben resolver. Cada línea tendrá el siguiente formato:

(A, B, C, H), (D, E)

donde A, B, C y H son los parámetros de entrada de la entrada de la actividad y D y E representan los tipos de los valores que se deben generar en la ejecución de la actividad.

Salida

La salida constará de T líneas, cada una de las cuales contendrá la composición de las funciones de menor costo para la actividad correspondiente, el tiempo que tomá en ejecutarse la composición identificada[1] y el número de órdenes totales que corresponden a la composición reportada. Las líneas deben corresponder en orden con la lista de actividades. En el caso de no existir una secuencia, Ud debe imprimir la secuencia vacía (). Las funciones de cada composición deben imprimirse respetando en orden alfabético.

Ejemplo Entrada

```
3
f3: (A), (C) 2
f1: (B, C), (D) 3
f2: (B), (E) 1
2
(A, B), (D, E)
(A), (B)
```

Ejemplo Salida

```
(f1, f2, f3), 6, 3
()
```

Entrega

Para el martes 21 de Noviembre (Semana 10) a la 3:30 pm usted deberá entregar a su profesor (en el salón de clases), un sobre sellado y debidamente identificado con su nombre, carnet y profesor de laboratorio. Éste debe contener:

- **Uds debe incluir un pequeño informe de máximo 3 páginas en donde explique y justifique su solución en forma detallada.**
- Códigos fuente **debidamente documentados** de los tipos implementados. **Incluya sólo los fuentes**

1 El tiempo de ejecución de una composición se calcula como la suma de los tiempos de ejecución de las funciones que la integran.

nuevos. Si requiere hacer alguna modificación a clases ya existentes, agregue reporte de una página explicando la naturaleza del cambio y la justificación.

- **La declaración de autenticidad debidamente llenada por los integrantes del grupo.**
- El lunes 20 de Noviembre hasta las 11:59pm, Ud deberá colocar el archivo con su proyecto en Aula Virtual, para lo cual deberá crear el directorio PROY4 dentro de la carpeta documentos de su grupo. Note que debe estar suscrito a algún grupo en aula Virtual para poder optar a esta opción. En este directorio colocará los archivos con los fuentes de su aplicación. El archivo debe tener como nombre **P4G<#grupo>.zip** que contenga un directorio **P4G<#grupo>** que a su vez contenga todos los archivos .java de su proyecto.

Observaciones

En la evaluación del proyecto se tomará en cuenta el estilo de programación, desempeño, uso de herencia, correcto uso y manejo de excepciones, etc. Proyectos que no compilen serán calificados con cero.

Recuerde que no puede utilizar

- ninguna de las clases en java.util.*
- ninguno de los algoritmos de ordenamiento provistos por java