Universidad de los Andes
Facultad de Ingeniería

Facultad de Ciencias

Maestría en Biología Computacional
BCOM4006 – Algoritmos en Biología Computacional
Semestre: 2020-20

Tarea 1 – Estructuras de datos

Objetivos

1. Implementar diferentes estructuras para guardar un conjunto de datos
2. Comparar cada implementación en tiempos de respuesta
3. Recordar conceptos básicos de programación

Contexto

Como actividad práctica para entender las diferencias entre varias estructuras de datos para resolver un problema, en esta tarea se explorará el uso de listas, arboles y tablas de hashing como estructuras para guardar datos de genes y anotaciones funcionales. La información de base que se procesará en esta tarea proviene de los siguientes dos archivos:

- sacCer_Annotations.gff: Contiene la anotacion estructural y funcional de diferentes características en el genoma de referencia de levadura (Saccharomyces cerevisiae). De este archivo vamos a cargar los ids de los genes y los ids de sus correspondientes ontologías (ids tipo GO:XXXXXXX) . El archivo está en formato gff3 (https://github.com/The-Sequence-Ontology/Specifications/blob/master/gff3.md)
- go.obo: Descripción general de ontologías de la base de datos “Gene ontology”. Cada ontología tiene un id, un nombre, una categoría principal y una descripción en texto.

En una primera versión, el objetivo es implementar las funcionalidades de buscar un gen dado su id y de buscar el conjunto de genes asociados a una ontología dada.

Para poder comparar los tiempos de respuesta de estas operaciones, se desarrollaron simulaciones en las que se ejecuta cada una de las operaciones de búsqueda con un grupo de 100000 ids generados de manera aleatoria. El programa adjunto permite ejecutar estas simulaciones e informa el tiempo de respuesta del proceso.

Instrucciones

1. Cargar en Eclipse el proyecto java adjunto. Explorar las clases utilizando el diagrama de clases como guía

2. [25%] Implementar los métodos de la clase “ArrayListGeneCatalog”

3. Ejecutar el programa desde la linea de comandos. Para esto, el método main se encuentra ubicado en la clase “uniandes.algobc.structures.ExampleGeneCatalog”. Los parámetros del programa son los siguientes:

args[0]: Archivo de ontologías
args[1]: Archivo de anotación de genes en formato GFF3
args[2]: Tipo de estructura a cargar. Puede ser ArrayList, TreeMap o HashMap. Ejecutar por ahora con ArrayList
args[3]: Tipo de proceso a correr. Si escribe 1, corre una simulación en la que busca 100,000 ids aleatorios como ids de genes. Si escribe 2, corre una simulación en la que busca 100,000 ids aleatorios como ids de ontologías. Si escribe 3, permite buscar un gen específico dado un id
args[4]: Para el proceso 3, id del gen a buscar

Si el proceso se ejecuta bien, en la pantalla deben salir algunos mensajes indicando que algunos ids de ontologías no fueron encontrados. Luego debe indicar la cantidad de genes que fueron cargados (6,603). Finalmente, para el proceso 1 y 2, indica el tiempo en el que se realizó la simulación y para el proceso 3 indica los datos del gen buscado.

4. [25%] Implementar la clase “HashMapGeneCatalog” utilizando como atributo para guardar los genes un objeto de la clase “HashMap” indexado por el id del gen, en lugar de utilizar un “ArrayList”. Para esto, hacer clic derecho en el paquete, y seleccionar “New → class”. En “Name:” escribir “HashMapGeneCatalog” y en “Interfaces:”, utilizar el botón “Add...” para buscar la interfaz “GeneCatalog”. Una vez presione “Finish” notará que las signaturas de los métodos se agregan automáticamente a la clase.
Para completar la implementación de la clase, debe crear un atributo de tipo “HashMap” para el catálogo de genes. Se puede guiar por el HashMap para el catalogo general de ontologías implementado en la clase “ExampleGeneCatalog”. Consultar en el javadoc (https://docs.oracle.com/javase/8/docs/api/java/util/HashMap.html) el manejo de los métodos “get”, “put” y “values” de la clase “HashMap” para implementar los métodos. Una vez terminada la implementación, ejecutar el programa utilizando “HashMap” como estructura a cargar. Verificar si el tiempo de las simulaciones mejoró respecto al tiempo empleado utilizando “ArrayList”

5. [25%] Implementar la clase “TreeMapGeneCatalog” siguiendo los pasos explicados en el punto anterior y utilizando como atributo para guardar los genes un objeto de la clase “TreeMap” indexado por el id del gen. Esta clase tiene métodos con los mismos nombres que la clase “HashMap”. Ejecutar el programa utilizando “TreeMap” como estructura a cargar y comparar los tiempos de respuesta de las simulaciones.

6. [15%] Para aprovechar mejor las nuevas estructuras de datos aprendidas en esta tarea, en la clase Gene, reemplazar el ArrayList que guarda las ontologías por un mapa (“TreeMap” o “HashMap”). Modificar los métodos necesarios para que la clase quede bien implementada (Advertencia: No puede cambiar ningún contrato de método). Ejecutar las simulaciones para verificar si mejoraron los tiempos de respuesta.

7. [10%] Para intentar mejorar el tiempo de respuesta de la simulación de búsqueda por ontologías, agregar otro atributo de tipo HashMap a la clase “HashMapGeneCatalog” indexado por los ids de ontología en lugar de los ids de los genes. Razonar de qué tipo de datos deberían ser los elementos guardados en este nuevo mapa. Hacer los cambios necesarios en los métodos addGene y getGenes para llenar correctamente este nuevo mapa y para poder utilizarlo en la consulta por id de ontología. Verificar si el tiempo de respuesta para la simulación por id de ontología mejora después de realizar los cambios.

Bono. [10%] Al revisar el código de la clase ExampleGeneCatalog, notará que en ninguna parte se hace referencia directa a las clases ArrayListGeneCatalog, TreeMapGeneCatalog o HashMapGeneCatalog. De hecho, el programa funcionaba correctamente desde antes de implementar las dos últimas clases. Investigar y describir el mecanismo que permite que esto sea posible.