Universidad de los Andes

Ingeniería de Sistemas y Computación
ISIS1206 – Estructura de Datos
Taller 5

Taller 5: Colas de Prioridad

Objetivos

- Estudiar, implementar, probar y documentar una estructura de cola de prioridad
- Utilizar adecuadamente herramientas para el desarrollo de software en equipos

Lectura Previa

Estudiar la teoría de las colas de prioridad. Consultar la sección 2.4 del libro guía "*Algorithms*" de Sedgewick y Wayne.

Lo que su grupo debe hacer (parejas)

Parte 1 – Trabajo en casa

1. Cree en bitbucket un repositorio llamado T5_201910. Al momento de crearlo recuerde la URL que se muestra en la parte superior derecha de la página de bitbucket: Por ejemplo

Repository_url = https://login-usuario@bitbucket.org/login-usuario/T5_201910.git

donde login-usuario corresponde a su login en Bitbucket.

2. Cree el README del repositorio donde aparezcan los nombres y códigos de los miembros del grupo de trabajo.

3. Realice el procedimiento para crear el directorio en su computador de trabajo para que relacione este directorio con el repositorio remoto T5_201910.

El trabajo que sigue debe repartirse entre ambos estudiantes del grupo dado que se deben implementar dos versiones de una cola de prioridad: (i) basado en la estructura Queue (cola), (ii) basado en la estructura Heap (montículo)

El trabajo debe balancearse entre ambos estudiantes del grupo.

4. Descargue la información del archivo de infracciones para el mes de Enero, Febrero, Marzo, y Abril de 2018 (formato CSV) que se encuentra en los siguientes enlaces:

   - http://opendata.dc.gov/datasets/moving-violations-issued-in-january-2018/data
   - http://opendata.dc.gov/datasets/moving-violations-issued-in-february-2018/data
   - http://opendata.dc.gov/datasets/moving-violations-issued-in-march-2018/data
   - http://opendata.dc.gov/datasets/moving-violations-issued-in-april-2018/data

   Copie las versiones de estos archivos (CSV) en la carpeta data del repositorio.

5. Definir e implementar la(s) Estructura de Datos (su interface y su implementación) genérica de tipo `T`. En esta estructura se van a almacenar las infracciones. Definirla en el paquete `model.data_structures`.

6. Implementar el método `loadMovingViolations ()` en la clase `Controller` que carga las infracciones registradas en los archivos CSV para los meses: Enero, Febrero, Marzo, y Abril. Las infracciones deben almacenarse en la estructura de datos definida.

7. Implemente las clases `MaxColaPrioridad <T extends Comparable<T>>` y `MaxHeapCP <T extends Comparable<T>>` con los siguientes métodos que siguen el comportamiento de una cola de prioridad orientada a mayor:

Operación	Descripción
`MetodoConstructor()`	Crea un objeto de la clase (sin elementos)
`int darNumElementos()`	Retorna número de elementos presentes en la cola de prioridad
`void agregar(T elemento)`	Agrega un elemento a la cola. Si el elemento ya existe y tiene una prioridad diferente, el elemento debe actualizarse en la cola de prioridad.
`T delMax ()`	Saca/atiende el elemento máximo en la cola y lo retorna; `null` en caso de cola vacía
`T max()`	Obtener el elemento máximo (sin sacarlo de la Cola); `null` en caso de cola vacía
`boolean esVacia ()`	Retorna si la cola está vacía o no

   Incluir estas clases en el paquete `model.data_structures`.

8. Defina una clase `LocationVO` comparable con los atributos `int addressId`, `String location`, `int numberOfRegisters`; el atributo `numberOfRegisters` corresponde al número de veces que aparece el `addressId` en los registros cargados. Implemente el método `compareTo` asumiendo que el criterio de ordenamiento es `numberOfRegisters`; si este valor es igual, la comparación se resuelve por su `location`.

9. Implementar el método `generarMuestra(…)` en la clase `Controller` que permite generar una muestra aleatoria de tamaño N de infracciones. El valor N es un dato de entrada del requerimiento.

10. Haga una lista de escenarios de prueba para los métodos `agregar` y `delMax` de las clases `MaxColaPrioridad<T>` y `MaxHeapCP<T>`, asumiendo que son colas de prioridad de objetos de tipo `LocationVO`. Debe incluir la lista de escenarios en su reporte.

11. Implemente una clase de pruebas **PruebaColaPrioridad** que implemente los escenarios definidos anteriormente.

12. Realizar una gráfica mostrando los tiempos promedios de ejecución para los métodos **agregar** y **delMax** usando las dos estructuras con diferentes cargas; es decir con diferentes volúmenes de datos [0, 350000] con un delta de 50000.

13. Preparar un documento con:
    - Análisis de eficiencia para cada estructura implementada; el documento debe explicar cuando se presenta su peor caso y su mejor caso en ambas operaciones. Para estos casos dar la complejidad teórica en tiempo usando la notación O( ).
    - Reportar los tiempos promedios de ejecución en una tabla y realizar una gráfica comparativa de los tiempos promedios para cada tamaño de muestras.
    - Incluir el documento preparado en la carpeta docs de su proyecto Eclipse.

Parte 2 – Trabajo en clase

1. El Departamento de Transporte de Washington D.C. quiere conocer cuáles son las N vías que tienen la mayor cantidad de infracciones registradas en un periodo de tiempo definido (Fecha Inicial – Fecha Final).
2. La clase que da respuesta a este requerimiento debe tener dos métodos de creación de la cola de prioridad y comparar los tiempos de respuesta de las mismas:
    - `MaxColaPrioridad <LocationVO> crearMaxColaP (LocalDateTime fInicial, LocalDateTime fFinal)`,
    - `MaxHeapCP <LocationVO> crearMaxHeapCP (LocalDateTime fInicial, LocalDateTime fFinal)`.

Entrega

1. Para hacer la entrega del taller usted debe agregar a su repositorio los usuarios de los monitores y su profesor, siguiendo las instrucciones del documento “Guía Creación de Repositorios para Talleres y Proyectos.docx”.
2. Entregue su taller por medio de BitBucket. Recuerde, dar acceso a su profesor y monitores (modo Read) para que pueda ser calificado.