Tarea 4: TDA para Mediana

Profesores: Nelson Baloian
Patricio Poblete
Auxiliares: Manuel Cáceres
Sebastián Ferrada
Sergio Peñafiel

Fecha de Entrega: 15 de Mayo 23:59hrs

1 Introducción

En muchos problemas de la vida diaria el uso de estadísticos es útil al momento de trabajar o tomar decisiones para conjuntos de datos grandes. El estadístico más típico para representar un conjunto es la media, sin embargo muchas veces este puede ser muy sensible a errores en los datos (por ejemplo que sólo un valor sea muy alto con respecto al resto). Frente a esto, un mejor estadístico es seleccionar un representante medio del conjunto, esto es la mediana.

En esta tarea se estudiará e implementará un tipo de dato abstracto (TDA) que permita insertar dinámicamente datos numéricos y permita obtener la mediana en tiempo constante.

Para implementarlo se le pide que utilice 2 colas de prioridad. Una de las colas de prioridad tendrá todos los elementos menores o iguales a la mediana en un MAX-Heap, y la otra tendrá todos los elementos mayores o iguales a la mediana en un MIN-Heap.

Note que si ambas colas están balanceadas según el número de elementos y la cantidad total de elementos es impar, entonces necesariamente una cola de prioridad tiene 1 elemento más que la otra, y justamente el elemento que está en el tope de esa cola es la mediana. Si la cantidad de elementos es par, entonces los elementos en los topes de ambas pilas son los centrales, por lo tanto la mediana se calcula como el promedio de estos.

Para que lo anterior funcione al momento de insertar un nuevo elemento debe decidir a que cola de prioridad insertarlo. Eventualmente podría ocurrir que al insertar un elemento los heaps queden desbalanceados, por lo que luego de insertar debe asegurar que la diferencia de los largos heaps sea a lo más 1, si esto no se cumple debe intercambiar un elemento de un heap al otro.

2 Implementación

2.1 El TDA ColaPrioridadMediana

Debe implementar un TDA `ColaPrioridadMediana` con las siguientes operaciones:

- Un constructor, `ColaPrioridadMediana([int cap])`:

  Construye una representación para el TDA, debe inicializar todo lo necesario para que el resto de las operaciones funcionen correctamente, opcionalmente se le puede pasar como parámetro una capacidad máxima para el almacenamiento de números.

- Insertar, `void insertar(double x)`:

  Inserta el elemento x al TDA según las reglas descritas anteriormente, este método debe tomar tiempo $O(\log n)$, siendo n la cantidad de elementos que tiene el TDA.

- Obtener la mediana, `double getMediana()`:

  Devuelve el valor asociado a la mediana de los elementos del TDA, este método debe tomar tiempo $O(1)$.

2.2 Ejemplo de uso del TDA

Suponga que se aprueba un subsidio a la vivienda, sin embargo este sólo puede ser otorgado a la mitad de los postulantes, el criterio de asignación es que aquellos que están en la mitad con menores ingresos familiares lo obtienen. Usando el TDA anterior, cree un programa que pida al usuario números para el ingreso familiar por la entrada estándar (uno por linea), y los inserte en el TDA, si el usuario ingresa vacío, es decir no ingresa un número, entonces debe mostrar la mediana de todos los números anteriores como el máximo ingreso familiar que puede acceder al beneficio. Observe el siguiente ejemplo (`>>` indica entrada de texto del usuario).

```
>> 480000
>> 920000
>> 650000
>>
El subsidio aplica hasta $650000 de ingreso familiar
```

2.3 Consideraciones y Guías

- Para el trabajo con heaps sólo se permite el uso de arreglos de java para representarlos, cualquier implementación distinta de heaps **no** puede ser utilizada en la solución.
- Para la inicialización de los heaps si no se tiene una capacidad por parámetro asígnele una que estime conveniente. Si se intenta insertar cuando se sobrepasa la capacidad muestre un mensaje de error y termine el programa.
- Puede asumir que los elementos que entrega el usuario por la entrada estándar son siempre números.
- Es obligatorio responder las preguntas que se detallan en la sección discusión del informe.

3 Condiciones de Entrega

- Esta tarea debe ser resuelta en Java.
- Es obligatorio la entrega de un informe en formato pdf junto con su tarea (Ver siguiente sección).
- Esta tarea es de carácter individual, cualquier caso de copia se evaluará con la nota mínima.
- No olvide subir a U-cursos todos los archivos necesarios para que su tarea funcione correctamente.
- Debe subir los archivos de código fuente (*.java). Los archivos compilados (*.class) no serán evaluados.
- Cualquier duda respecto a la tarea puede ser consultada usando el foro del curso.
- **NO** se aceptarán atrasos.

4 Informe

El informe debe describir el trabajo realizado, la solución implementada, los resultados obtenidos y las conclusiones o interpretaciones de estos. Principalmente debe ser breve, describiendo cada uno de los puntos que a continuación se indican:

- **Portada:** Indicando número de la tarea, fecha, autor, email, código del curso, etc.
- **Introducción:** Descripción breve del problema y su solución.
- **Análisis del problema:** Exponga en detalle el problema, los supuestos que pretende ocupar, casos de borde y brevemente la metodología usada para resolverlo.
- **Solución del problema:** Indique claramente los pasos que siguió para llegar a la solución del problema. Muestre mediante figuras y ejemplos qué es lo que realiza su código. Evite copiar todo el código fuente en el informe, sin embargo, puede mostrar las partes relevantes de éste.
- **Modo de uso:** Explicando brevemente cualquier dato necesario para la compilación y ejecución de su programa.
- **Resultados:** Muestre ejemplos de entradas y salidas de su programa.
- **Discusión:**
  - Discuta sobre la complejidad de los métodos del TDA, en especial argumente porque sus implementaciones cumplen con las restricciones de tiempo.
  - Para el caso de uso del TDA (problema del subsidio), considere un algoritmo que inserta todos los elementos en un arreglo en el orden de llegada, luego los ordena con algoritmo eficiente y finalmente extrae la mediana como el elemento en la posición $n/2$. Compare este algoritmo con su solución, ¿Es mejor, peor o igual en términos de complejidad? argumente.