EDA (E.T.S. de Ingeniería Informática)
Curso 2015-2016
Práctica 5. Montículo Binario

Departamento de Sistemas Informáticos y Computación
Universitat Politècnica de València

1. Objetivos

- Comprender el código de la clase `MonticuloBinario`, vista en teoría, y ser capaz de hacer modificaciones sobre dicho código.
- Utilizar la estructura de datos *Cola de Prioridad* para implementar un servidor de impresión eficiente.

2. Contexto y trabajo previo

Para que aproveches al máximo la sesión de prácticas, antes debes realizar una lectura comprensiva de este boletín y familiarizarte con el código de las clases que se te proporcionan a través de PoliformaT; en concreto observa qué atributos tienen las clases y la funcionalidad de cada una de ellas.

Los montículos binarios (*binary heaps* en inglés) son árboles binarios completos, los cuáles pueden representarse fácilmente mediante un array. En un montículo binario minimal (*MinHeap*), el dato de un nodo es siempre menor o igual que el de sus hijos. Esta es la implementación más habitual y la que se ha estudiado en teoría con la clase `MonticuloBinario`. En un montículo binario maximal (*MaxHeap*), la propiedad de orden es justo la contraria (el dato de un nodo es siempre mayor o igual que el de sus hijos).

Otra característica de la clase `MonticuloBinario` es que deja libre la posición 0 del array. De esta forma, se simplifica el cálculo de la posición del padre y los hijos de un nodo. Concretamente, para el nodo i-ésimo:

- Su hijo izquierdo está en la posición $2 * i$, si $2 * i \leq talla$.
- Su hijo derecho está en la posición $2 * i + 1$, si $2 * i + 1 \leq talla$.
- Su padre está en la posición $i/2$, si $i \neq 1$.

3. Actividades en el laboratorio

En esta práctica se va a realizar una simulación que nos permita comparar el tiempo medio de espera de dos servidores de impresión distintos:

- *ImpresoraCola*: esta impresora usa internamente una *Cola* para gestionar los documentos que recibe. De esta forma, los documentos se van imprimiendo en el mismo orden en el que se van enviando a la impresora.
- *ImpresoraCP*: esta impresora usa internamente una *Cola de Prioridad*, implementada mediante un montículo binario maximal, de forma que se imprimen primero los documentos más cortos (con un menor número de páginas).

Esta simulación nos permitirá comprobar que el tiempo medio de espera para la impresión de trabajos es menor utilizando una *ImpresoraCP*.

3.1. Implementación de un Monticulo Binario Maximal

La clase `MonticuloBinario`, disponible en PoliformaT, muestra el código de un Monticulo Binario Minimal en el que la posicion 0 del array se deja libre. En esta práctica, sin embargo, vamos a requerir la implementación de un Monticulo Binario Maximal en el que los elementos comiencen en la posición 0 del array. Aparte del cambio en la propiedad de orden, debe tenerse en cuenta también que el cálculo de la posición del padre y de los hijos de un nodo también varía. En concreto, para el nodo i-ésimo:

- Su hijo izquierdo está en la posición $2 * i + 1$, si $2 * i + 1 < talla$.
- Su hijo derecho está en la posición $2 * i + 2$, si $2 * i + 2 < talla$.
- Su padre está en la posición $(i - 1)/2$, si $i \neq 0$.

Para ello el alumno deberá implementar en el paquete *librerias/estructurasDeDatos/jerarquicos* una nueva clase, `MonticuloBinarioMaxR0`, que modifique el código de `MonticuloBinario` para convertirlo en un Montículo Binario Maximal con el nodo raíz situado en la posición 0 del array. El alumno puede copiar el programa `TestMonticuloBinarioMaxR0.class`, disponible en PoliformaT, dentro de este mismo paquete y ejecutarlo para comprobar si el código implementado funciona correctamente.

3.2. La aplicación Impresora

Crear el paquete *aplicaciones/impresora*, que se encarga de realizar la simulación para comparar los dos tipos de impresora. A este paquete se deberan añadir las siguientes clases:

- `Documento`: representa un documento a imprimir. Esta clase tiene tres atributos: el título del documento (*String*), el número de páginas que tiene (*int*) y el instante de tiempo, en segundos, en el que se envía a imprimir (*int*).

- `Impresora`: interfaz que contiene los métodos que han de implementar las impresoras. Estos métodos son los siguientes:

  `void guardarTrabajo(Documento doc)`: permite almacenar un documento en la impresora. Este método solo añade documentos a la cola de impresión, no los imprime.

  `boolean hayTrabajos()`: permite consultar si hay documentos pendientes de imprimir.

  `Documento siguienteTrabajo()`: permite consultar el siguiente documento que se imprimirá.

  `int imprimirSiguiente()`: imprime (elimina de la cola de impresión) el siguiente documento. Esta función devuelve el tiempo requerido para imprimir dicho documento (en segundos), que depende del número de páginas del documento y de la velocidad de impresión (declarada mediante la constante `PAGINAS_POR_MINUTO`).

- `ImpresoraCola`: impresora que almacena internamente los documentos usando una *Cola*. Esta clase se proporciona completamente implementada.

- `ImpresoraCP`: impresora que almacena internamente los documentos usando una *Cola de Prioridad*.

- `SimuladorImpresion`: esta clase realiza la comparación de ambas impresoras mediante una simulación.

3.3. La clase Documento

Añade el código necesario a la clase `Documento` para que sea posible almacenar objetos de este tipo en una *Cola de Prioridad*. Se ha de tener en cuenta lo siguiente:

- La *Cola de Prioridad* que se va a utilizar está implementada mediante un Monticulo Binario **Maximal**.

- Un documento tendrá mayor prioridad que otro si tiene un menor número de páginas. Es decir, se priorizan los documentos más cortos.

3.4. La clase ImpresoraCP

Completa el código de la clase `ImpresoraCP`, teniendo en cuenta que usa internamente una *Cola de Prioridad* para almacenar los documentos. La implementación de dicha *Cola de Prioridad* ha de ser la que ha diseñado el alumno en la clase `MonticuloBinarioMaxR0`.

Se aconseja consultar el código de la clase `ImpresoraCola`, ya que el código va a ser muy similar.

3.5. Simulación de impresión

La práctica se puede validar ejecutando el programa `SimuladorImpresion`. Si el código de la práctica está correcto, se mostrará una pequeña simulación donde se compara el tiempo medio de espera resultante de imprimir un conjunto de documentos mediante la `ImpresoraCola` y la

`ImpresoraCP`. En esta simulación se muestra una línea para cada uno de los documentos, en el orden en el que se van imprimiendo. El formato de estas líneas es el siguiente:

```
[156] El hobbit - J.R.R. Tolkien (61 pag.) Envio: 34 (122,00 seg. de espera)
```

- `[156]` ⟶ Instante de tiempo (en segundos) en el que finaliza la impresión del documento.
- `El hobbit - J.R.R. Tolkien` ⟶ Título del documento.
- `(61 pag.)` ⟶ Número de páginas del documento.
- `Envio: 34` ⟶ Instante (en segundos) en el que se envió el documento a la impresora.
- `(122,00 seg. de espera)` ⟶ Tiempo de espera para la impresión del documento. Este valor es la diferencia entre el tiempo en el que acaba la impresión y el instante de envío.

Nótese que la simulación se realiza con valores aleatorios, por lo que será diferente cada vez que se ejecute el programa. Al final de la simulación de cada impresora se muestra el tiempo de espera promedio. Por ejemplo, el siguiente resultado se corresponde con una posible ejecución del programa:

```
SIMULACION CON COLA: Tiempo medio de espera = 690,58 seg.
SIMULACION CON COLA DE PRIORIDAD: Tiempo medio de espera = 483,75 seg.
```

En este ejemplo se puede observar que la impresora que utiliza una *Cola de Prioridad* para gestionar los documentos reduce el tiempo medio de espera respecto a la impresora que usa internamente una *Cola*.