EDA (ETS de Ingeniería Informática). Curso Académico 2020-2021

Práctica 3 - Parte 2. Elementos para la implementación eficiente y el análisis del rendimiento de una Tabla Hash Enlazada

Departamento de Sistemas Informáticos y Computación. Universitat Politècnica de València

1. Objetivo

El objetivo de esta segunda parte de la práctica 3 es incorporar a la clase `TablaHash` que se utilizó en su primera parte algunas de las operaciones que requiere la implementación eficiente y el análisis del rendimiento de una Tabla Hash Enlazada: respectivamente, la operación de *Rehashing* y las del cálculo de la desviación típica de las longitudes de las cubetas de la tabla y del coste promedio de localizar una de sus claves.

2. Descripción del problema

Como ya se ha comentado, la clase `TablaHash` que implementa el `Map index` de una Biblioteca Digital (BD) representa una Tabla *Hash* Enlazada **SIN** *Rehashing*. Para ver cómo ello puede acarrear una importante degradación de su eficiencia cuando, por cualquier motivo, aumenta la longitud media de sus cubetas, basta comparar los análisis de eficiencia que se muestran en la siguiente figura y que corresponden a dos `TablaHash` que únicamente difieren en la talla máxima estimada que se usa para construirlas: 22310 para la tabla correspondiente a la Figura 1(a) y 112 para la correspondiente a la Figura 1(b).

```
TablaHash para la BD "lista10.txt": talla 22310, hashCode() con
                                    base 31 y sin Rehashing

   - Talla Máxima Estimada = 22310

   - Factor de Carga = 0,750

   - Desviación Típica = 0,868

   - Tras 0 rehashings, han habido 30737 colisiones
     Coste promedio de localizar una clave = 1,378

   - Histograma de ocupación

     0       14057
     1       10576
     2       3900
     3       984
     4       205
     5       24
     6       7
     7       0
     8       0
     9       0
```

(a)

```
TablaHash para la BD "lista10.txt": talla 22310, hashCode() con
                                    base 31 y sin Rehashing

   - Talla Máxima Estimada = 112

   - Factor de Carga = 149,732

   - Desviación Típica = 12,496

   - Tras 0 rehashings, han habido 1693044 colisiones
     Coste promedio de localizar una clave = 75,887

   - Histograma de ocupación

     0       0
     1       0
     2       0
     3       0
     4       0
     5       0
     6       0
     7       0
     8       0
     9       149
```

(b)

Figura 1: Cómo se puede degradar la eficiencia de una Tabla *Hash* Enlazada SIN *Rehashing*

Así pues, para dotar a una Tabla *Hash* Enlazada de un "botón del pánico" que pulsar cuando su eficiencia se vea comprometida, en esta segunda parte de la práctica el alumno deberá implementar la operación de *Rehashing* en la clase `TablaHash` que implementa el `Map index` del Buscador de una BD. Además, deberá añadir a dicha clase las implementación de las operaciones que obtienen la desviación típica de las longitudes de las cubetas de la tabla y el coste promedio de localizar una de sus claves, pues sus resultados son dos de los valores que obtiene el programa `TestEficiencia` para analizar el rendimiento de dicho `Map`.

3. Actividades a realizar

Antes de llevar a cabo las actividades que se proponen en este apartado, es necesario que el alumno ubique en el directorio *eda/librerias/estructurasDeDatos/deDispersion* de su carpeta personal los ficheros `TablaHash.java` y `TestTablaHash.class` disponibles en *PoliformaT*.

3.1. Completar el código de la clase `TablaHash` y validarlo

Tras abrir su proyecto *BlueJ eda* y acceder a su paquete *librerias.estructurasDeDatos.deDispersion*, el alumno debe completar el cuerpo de los siguientes métodos de la clase `TablaHash`:

- Método `desviacionTipica`, que devuelve la desviación típica de las longitudes de las cubetas de `this TablaHash`.

  `NOTA:` si la longitud de la i-ésima cubeta de la tabla se denota con l_i y la longitud media de sus N cubetas con $\bar{l}$, la desviación típica σ se calcula como sigue:

$$\sigma = \sqrt{\frac{\sum (l_i - \bar{l})^2}{N}}$$

- Método `costeMLocalizar`, que devuelve el coste promedio de localizar una clave de `this TablaHash`, calculado a partir del número de colisiones que se producen al localizar sus `this.talla` claves.

- Método `rehashing`, que incrementa la capacidad de `elArray` que tiene `this TablaHash` SII tras insertar una nueva Entrada en ella su factor de carga es mayor que 0.75, su factor de carga por defecto; la nueva capacidad de `elArray` debe ser igual al siguiente número primo del doble de su capacidad actual (`elArray.length`).

Finalmente, una vez compilada la clase `TablaHash`, el alumno debe validar su corrección ejecutando el programa `TestTablaHash`.

3.2. Analizar la eficiencia del `Map index` de `lista10.txt` usando la (nueva) clase `TablaHash`

Para realizar esta actividad, el alumno debe:

- Acceder al paquete *biblioteca* de su proyecto *eda* y recompilar sus clases.

- Repetir las ejecuciones de `TestEficiencia` que realizó en la última actividad de la primera parte de esta práctica PERO, esta vez, pasando a su `main` como segundo argumento el `String "CON"`.

  NOTA: los nombres de los ficheros con los histogramas de ocupación de cada tabla `"CON"` *Rehashing* incluyen el número de *Rehahings* realizados para que su factor de carga no supere 0.75 seguido del `String "RH"`.

- Comparar la eficiencia de las tablas resultantes con la de las obtenidas en la última actividad de la primera parte de esta práctica para comprobar que, en efecto, la operación de *Rehashing* es el "botón del pánico" de una Tabla *Hash* Enlazada eficiente.