Universidad de los Andes

Ingeniería de Sistemas y Computación
ISIS1206 – Estructura de Datos
Taller 6

Taller 6: Tablas de Hash

Objetivos

- Estudiar, implementar, probar y documentar la Estructura de Datos *Tabla de Hash*
- Utilizar adecuadamente herramientas para el desarrollo de software en equipos

Lectura Previa

Estudiar la teoría de las tablas de hash. Consultar la sección 3.4 del libro guía "*Algorithms*" de Sedgewick y Wayne.

Las Fuentes de Datos

Se van a usar los datos en formato JSON de las infracciones en el distrito de Columbia. En la página de DC.gov se pueden descargar los JSON en el menú API y luego usando la URL que aparece bajo "GeoJSON". Sin embargo por facilidad se usara unos archivos preprocesador con el siguiente formato:


```
[
  {
    "OBJECTID": 14469881,
    "ROW_": null,
    "LOCATION": "100 BLK MICHIGAN AVE NW E/B",
    "ADDRESS_ID": 815694,
    "STREETSEGID": 1405,
    "XCOORD": 398728.3,
    "YCOORD": 139835.9,
    "TICKETTYPE": "Moving",
    "FINEAMT": 100,
    "TOTALPAID": 100,
    "PENALTY1": 0,
    "PENALTY2": null,
    "ACCIDENTINDICATOR": "No",
    "TICKETISSUEDATE": "2018-03-08T18:49:00.000Z",
    "VIOLATIONCODE": "T119",
    "VIOLATIONDESC": "SPEED 11-15 MPH OVER THE SPEED LIMIT",
```

```
    "ROW_ID": null
  },
  {
    "OBJECTID": 14469882,
    "ROW_": null,
    "LOCATION": "14TH ST N/B @ K ST NW",
    "ADDRESS_ID": 900719,
    "STREETSEGID": 3438,
    "XCOORD": 397228.19001138,
    "YCOORD": 137185.83334074,
    "TICKETTYPE": "Moving",
    "FINEAMT": 50,
    "TOTALPAID": 50,
    "PENALTY1": 0,
    "PENALTY2": null,
    "ACCIDENTINDICATOR": "No",
    "TICKETISSUEDATE": "2018-03-08T10:40:00.000Z",
    "VIOLATIONCODE": "T334",
    "VIOLATIONDESC": "TURN RIGHT ON RED WITHOUT COMPLETE STOP",
    "ROW_ID": null
  },
 ...
]
```


Para la lectura de archivos JSON se recomienda usar el API (librería) GSON. Consultar https://github.com/google/gson
Para descargas del API (librería extensión .jar) consulte: https://maven-badges.herokuapp.com/maven-central/com.google.code.gson/gson
Para documentación del API consulte: http://www.javadoc.io/doc/com.google.code.gson/gson

Lo que su grupo debe hacer (parejas)

Parte 1 – Trabajo en casa

1. Cree en bitbucket/GitHub un repositorio llamado T6_201910.

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 que acaba de crear.
   El directorio debe conservar la estructura aprendida en los talleres anteriores.

El trabajo debe balancearse entre ambos estudiantes del grupo.

4. Implementar la estructura de datos **Tabla Hash genérica** con los tipos K (para la clase Llave) y V (para la clase Valor asociado a cada llave). El tipo K debe ser Comparable<K>. Las operaciones básicas deben ser:

Operación	Descripción
`MetodoConstructor( int m )`	Crea una tabla de hash con capacidad inicial de m posiciones
`void put(K, V)`	Agregar una dupla (K, V) a la tabla. Si la llave K existe, se reemplaza su valor V asociado. V no puede ser *null*.
`V get(K)`	Obtener el valor V asociado a la llave K. V no puede ser *null*.
`V delete(K)`	Borrar la dupla asociada a la llave K. Se obtiene el valor V asociado a la llave K. Se obtiene *null* si la llave K no existe.
`Iterator<K> keys()`	Conjunto de llaves K presentes en la tabla.

a. Implementar una **Tabla Hash** usando ***Linear Probing*** para resolución de conflictos. En esta versión se define un factor de carga (N/M) máximo de 0.75; es decir, para factores de carga mayores debe aplicarse el *rehash* de la tabla. Adicionalmente se deben definir las pruebas unitarias para cada una de las operaciones (incluyendo *rehash*).

b. Implementar una **Tabla Hash** usando ***Separate Chaining*** para resolución de conflictos. En esta versión se define un factor de carga (N/M) máximo de 5.0; es decir, para factores de carga mayores debe aplicarse el *rehash* de la tabla. Al igual que para el punto anterior se debe definir las pruebas unitarias para cada una de las operaciones (incluyendo *rehash*).

Recomendación: Recurrir a un número primo de entradas en la Tabla de Hash para mejorar la distribución uniforme de las llaves en la tabla. Tener en cuenta también al hacer el *rehash*.

5. Descargue la información del archivo de infracciones para el mes de Enero, Febrero, Marzo, Abril, Mayo y Junio de 2018 (formato JSON) que se encuentra en el archivo data.zip. data.zip tiene la misma información que la contenida en:

- 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
- http://opendata.dc.gov/datasets/moving-violations-issued-in-may-2018/data
- http://opendata.dc.gov/datasets/moving-violations-issued-in-june-2018/data

Parte 2 – Trabajo en clase

A partir de los archivos de las infracciones, resuelva los siguientes requerimientos:

1. **Infracciones con accidente por ADDRESS_ID (Tabla de Hash Linear Probing)**
   Crear una tabla de hash donde se guarden las infracciones por la dirección (ADDRESS_ID). Dado una dirección mostrar todas las infracciones que terminaron en un accidente en esa dirección. El resultado debe estar ordenado cronológicamente.

   De cada infracción se debe mostrar: OBJECTID, LOCATION, TICKETISSUEDATE, VIOLATIONCODE y FINEAMT.

2. **Infracciones con accidente por ADDRESS_ID (Tabla de Hash Separate Chaining)**
   Implementar la misma funcionalidad del punto anterior, pero con una tabla de hash implementadad con separate chaining.

3. **Pruebas.**
   Usando las dos tablas creadas en los puntos anteriores, hacer una prueba JUnit Test donde se hacen 10.000 consultas get(…). Luego llenar la siguiente tabla:

	Tabla de Hash Linear Probing	Tabla de Hash Separate Chaining
Número de duplas (K, V) en la tabla		
Tamaño inicial del arreglo de la tabla		
Tamaño final del arreglo de la tabla		
Factor de carga final (N/M)		
Número de rehashes que tuvo la tabla (desde que se creó)		
Tiempo promedio de consultas get(…)		

4. **Comparación Implementaciones**
   Preparar un documento con un análisis comparativo de las dos implementaciones. Incluir el documento en la carpeta docs de su proyecto Eclipse.

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/GitHub. Recuerde, si su repositorio no tiene el taller o está vacío, su taller no será calificado por más de que lo haya desarrollado.