LABORATORIO NO. 8: TABLAS DE SÍMBOLOS ORDENADAS y BALANCEADAS

Jennifer Alejandra Montoya Peñuela 202012739 j.montoyap@uniandes.edu.co

Santiago Pardo Bravo 202013024 s.pardob@uniandes.edu.co

Cristhian David Perdomo Gómez 201316701 cd.perdomo10@uniandes.edu.co

1 OBJETIVOS

Incluir en el diseño de soluciones el uso de *Maps* o Tablas de símbolos ordenadas y utilizar los árboles binarios de búsqueda balanceados *(RBTs)* para su implementación

a) Analizar los órdenes de crecimiento y el desempeño de los *RBTs* como índices ordenados para la consulta de datos
b) Integrar los *RBTs* con las otras estructuras de datos vistas en el curso

2 DESARROLLO

En los grupos previamente definidos sigan los siguientes pasos para el laboratorio de hoy.

2.1 CARGAR EL EJEMPLO ISIS1225-SAMPLETREE

Asegúrense que todos los miembros del equipo tienen acceso al ejemplo, en caso de que no lo hayan hecho para el laboratorio 7. Para ello utilicen el URL https://github.com/ISIS1225DEVS/ISIS1225-SampleTree y sigan el proceso acostumbrado: Fork del proyecto a su espacio en GitHub y clone a su máquina local. Si ya tienen el ejemplo, actualícenlo a la nueva versión del ejemplo.

2.2 EJECUTAR EL EJEMPLO

Cuando tenga los datos de prueba, ejecute el programa, desde el archivo view.py tal como lo hicieron en el laboratorio 7. Ejecute primero la opción 1 que inicializa las estructuras de datos, luego la opción 2, para cargar la información.

Al cargar la información, verá el número de registros cargados, la menor llave encontrada, la mayor llave encontrada, la altura del árbol *(BST)* y el número de nodos en el árbol *(BST)*.

Tome nota del total de elementos en el árbol y la altura del BST reportada. Recuerde a que conclusión llegaron en el laboratorio anterior sobre estos valores.

Número de elementos (BST):

Altura del árbol (BST):

2.3 Ejecutar el Ejemplo con un RBT

Salgan del programa y ahora editen el archivo model.py, específicamente donde se crea el BST.

```
analyzer['dateIndex'] = om.newMap(omaptype='BST',
                                  comparefunction=compareDates)
```

Cambien el valor del parámetro ***omaptype*** y ahora usen un árbol rojo-negro (RBT), como se muestra en el ejemplo.

```
analyzer['dateIndex'] = om.newMap(omaptype='RBT',
                                  comparefunction=compareDates)
```

Ahora, ejecuten nuevamente el programa desde el archivo *view.py*. Seleccionen la opción 1 y luego la opción 2.

Nuevamente verán el número de elementos en el árbol y la altura del árbol.

Número de elementos (RBT):

Altura del árbol (RBT):

Pregunta 1: ¿Qué diferencia existe entre las alturas de los dos árboles (BST y RBT) ?, ¿por qué pasa esto?

BST:

```
Cargando información de crimenes ....
Crimenes cargados: 319073
Altura del arbol: 29
Elementos en el arbol: 1177
Menor Llave: 2015-06-15
Mayor Llave: 2018-09-03
```

RBT:

```
Cargando información de crimenes ....
Crimenes cargados: 319073
Altura del arbol: 13
Elementos en el arbol: 1177
Menor Llave: 2015-06-15
Mayor Llave: 2018-09-03
```

La Altura del árbol es mayor cuando se implementa como un BST, esto sucede porque con esta implementación los datos se ordenan de manera relativa, no estricta (como sucede cuando se implementa un RBT). Por lo tanto, el árbol no quedará balanceado correctamente y por lo tanto esto se verá reflejado en la altura del mismo. En un RBT se ordenan los datos mientras van ingresando, por lo que se balancea correctamente el árbol y la altura tendrá un valor optimo

2.4 Modificar el requerimiento 1 del Reto 3

Utilicen la implementación del requerimiento 1 del reto 3 del laboratorio anterior (Lab 7) en el que utilizaron un BST, para averiguar la altura del árbol. Hagan la prueba con los datos del año 2016.

Pregunta 2: ¿Cuántos elementos tiene el árbol (size)? ¿Qué altura tiene el árbol (height)?

BST, año 2016:

```
Acidentes cargados: 131254
Altura del arbol: 14
Elementos en el arbol: 344
Menor Llave: 2016-02-08
Mayor Llave: 2017-01-26
```

Para el 2016 se cargaron en total 344 elementos y el árbol BST tiene una altura de 14

2.5 Modificar el requerimiento 1 del Reto 3

Cambien el requerimiento 1 del reto 3, desarrollado en el laboratorio anterior para que ahora utilice un RBT.

Pregunta 3: Qué tan difícil fue hacer el cambio de una estructura de datos por otra? ¿Cuántas líneas de código tuvieron que modificar para hacer el cambio?

-> Fue necesario solo modificar una línea de código para hacer el cambio de BST a RBT, esto se hizo en el archivo model, con la función implementada al momento de crear el árbol

Ahora prueben con el mismo archivo de datos (año 2016) la nueva estructura de datos (RBT)

Pregunta 4: Cuántos elementos tiene el árbol? ¿Qué altura tiene el árbol? ¿Qué puede concluir sobre las alturas de los árboles cuando se usa un BST o un RBT?

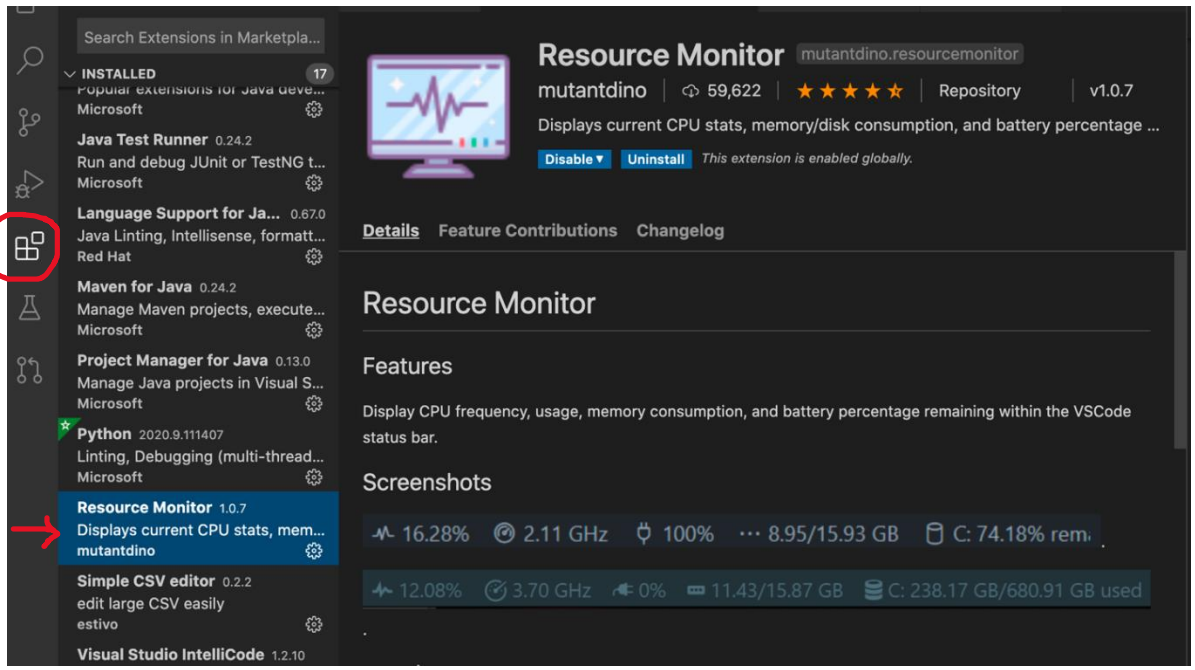
RBT:

```
Acidentes cargados: 131254
Altura del arbol: 11
Elementos en el arbol: 344
Menor Llave: 2016-02-08
Mayor Llave: 2017-01-26
```

-> Cuando se implementa un RBT se ve una clara disminución en la altura del árbol, esto se debe gracias a que cuando se construye un árbol rojo-negro este va a ordenar los elementos conforme van entrando, por lo que su distribución es mucho más acertada en comparación a los BST, por lo tanto, la altura del árbol será menor y al momento de realizar las búsquedas estas se harán mucho más rápido

2.6 Utilizar el conjunto completo de datos con el requerimiento 1 del Reto 3

Ahora, instalen en su VS Code cualquier plugin de ayuda para la verificación de uso de los recursos disponibles de su computador (procesador, memoria, etc.). Se sugiere la utilización de “Resource Monitor”.



En la barra inferior de VSCode verán la información de los recursos de su computador:

Tomen nota de la memoria utilizada que reporta VSCode

Un arhivo

Terminen el programa y ejecuten esta misma medición varias veces (por ejemplo 3) y calculen un promedio.

Mem Inicial:

	Memoria inicial
Intento #1	3.93
Intento #2	3.94
Intento #3	3.96

Ejecuten el reto 1 del requerimiento 3 (solo usando un RBT), cargando los datos de un solo año (2016) e inspeccionen la cantidad de memoria utilizada (con el programa aun ejecutándose)

Mem Final:

	Memoria final
Intento #1	5.07
Intento #2	5.02
Intento #3	4.96

La diferencia es la memoria utilizada durante la ejecución de su programa

Mem Utilizada:

	Memoria utilizada
Intento #1	1.14
Intento #2	1.08
Intento #3	1

Promedio 1.07

4 archivos

Ahora corran de nuevo el programa, pero esta vez utilicen todos los datos del proyecto (archivo completo con los 4 años) y ejecuten el mismo requerimiento 1 del reto 3. Inspeccionen nuevamente el uso de la memoria reportada en VSCode.

Mem Inicial:

	Memoria inicial
Intento #1	7.21
Intento #2	3.46
Intento #3	4.03

Ejecuten el reto 1 del requerimiento 3 (solo usando un RBT), cargando los datos de un solo año (2016) e inspeccionen la cantidad de memoria utilizada (con el programa aun ejecutándose)

Mem Final:

	Memoria final
Intento #1	11.30
Intento #2	10.32
Intento #3	11.04

La diferencia es la memoria utilizada durante la ejecución de su programa

Mem Utilizada:

	Memoria utilizada
Intento #1	4.09
Intento #2	6.86
Intento #3	7.01

Promedio 5.98

Pregunta 5: Existe diferencia en el consumo de memoria? ¿Pueden proponer una relación entre el total de datos cargados y la memoria utilizada?

Existe una clara diferencia, ya que el promedio al usar los 4 archivos es casi 5 veces mayor, por otro lado, también se observó que esta relación puede ser proporcional, ya que por cada archivo se utilizaba 1.5 de memoria más

2.7 COMPARTIR EL PRODUCTO DE LA PRACTICA CON LOS EVALUADORES

El resultado de este laboratorio es la implementación del requerimiento 1 del reto 3. Para entregar exitosamente sus resultados de este laboratorio, por favor recuerde las siguientes indicaciones:

a) Crear un Commit en el depósito de su reto 3 con el formato ***EDA-2020-20-Lab-08-SEC-<<Número de la sección>>-GRUPO-<<Número del grupo>>.***
b) Invitar a los monitores del laboratorio asignados.
c) Incluir en el **README** del repositorio los datos completos de los integrantes del grupo (nombre completo, correo Uniandes y código de estudiante).
d) Incluir en la carpeta *Docs* un documento en formato PDF que indique lo siguiente:
   i. Datos completos de los integrantes del grupo (nombre completo, correo Uniandes y código de estudiante).
   ii. Las preguntas del laboratorio con sus respectivas respuestas, en un documento PDF llamado *"respuestas-lab8.pdf"*, marcado con el nombre de los integrantes del grupo. Dentro del documento la respuesta a las preguntas:

Pregunta 1: Qué diferencia existe entre las alturas de los dos árboles (BST y RBT) por qué pasa esto?

BST:

```
Cargando información de crimenes ....
Crimenes cargados: 319073
Altura del arbol: 29
Elementos en el arbol: 1177
Menor Llave: 2015-06-15
Mayor Llave: 2018-09-03
```

RBT:

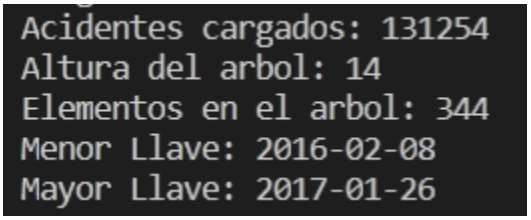
```
Cargando información de crimenes ....
Crimenes cargados: 319073
Altura del arbol: 13
Elementos en el arbol: 1177
Menor Llave: 2015-06-15
Mayor Llave: 2018-09-03
```

La Altura del árbol es mayor cuando se implementa como un BST, esto sucede porque con esta implementación los datos se ordenan de manera relativa, no estricta (como sucede cuando se implementa un RBT). Por lo tanto, el árbol no quedará balanceado correctamente y por lo tanto esto se verá reflejado en la

altura del mismo. En un RBT se ordenan los datos mientras van ingresando, por lo que se balancea correctamente el árbol y la altura tendrá un valor optimo

Pregunta 2: ¿Cuántos elementos tiene el árbol (size)? ¿Qué altura tiene el árbol (height)?

BST, año 2016:



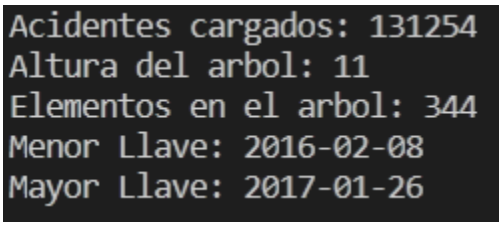
Para el 2016 se cargaron en total 344 elementos y el árbol BST tiene una altura de 14

Pregunta 3: Qué tan difícil fue hacer el cambio de una estructura de datos por otra? ¿Cuántas líneas de código tuvieron que modificar para hacer el cambio?

-> Fue necesario solo modificar una línea de código para hacer el cambio de BST a RBT, esto se hizo en el archivo model, con la función implementada al momento de crear el árbol

Pregunta 4: Cuántos elementos tiene el árbol? ¿Qué altura tiene el árbol? ¿Qué puede concluir sobre las alturas del árbol cuando se usa un BST y un RBT?

RBT:



-> Cuando se implementa un RBT se ve una clara disminución en la altura del árbol, esto se debe gracias a que cuando se construye un árbol rojo-negro este va a ordenar los elementos conforme van entrando, por lo que su distribución es mucho más acertada en comparación a los BST, por lo tanto, la altura del árbol será menor y al momento de realizar las búsquedas estas se harán mucho más rápido

Pregunta 5: Existe diferencia en el consumo de memoria? ¿Pueden proponer una relación entre el total de datos cargados y la memoria utilizada?

Existe una clara diferencia, ya que el promedio al usar los 4 archivos es casi 5 veces mayor al utilizar solamente uno, por otro lado, también se observó que esta relación puede ser proporcional, ya que por cada archivo se utilizaba 1.5 de memoria más.

Recuerden que cualquier documento solicitado durante las actividades debe incluirse en el repositorio GIT y que solo se calificará hasta el último **COMMIT** realizado dentro de la fecha límite del miércoles 14 de octubre de 2020, antes de la media noche (11:59 pm).