Universidad de Costa Rica

Facultad de Ingeniería
Escuela de Ciencias de la Computación e Informática

CI0116 – ANÁLISIS DE ALGORITMOS Y ESTRUCTURAS DE DATOS
Profesor Pablo [redacted]

Tarea Programada 2

Estudiante:
Emmanuel [redacted]

5 de julio de 2021

Índice

Tarea Programada
Estructuras de Datos

Descripción:

Las estructuras de datos hacen referencia a diferentes maneras de organizar la memoria al almacenar los datos. Este tema es de vital importancia para el mundo moderno, debido a la cantidad y variedad de datos que se manejan. Diferentes estructuras presentan diferentes ventajas al organizar los datos, proveyendo variedad de tiempos al momento de accesar, insertar, modificar y eliminar datos; es por ello que es importante su estudio, para entender sus diferencias, desventajas y fortalezas.

Enunciado:

En esta tarea el objetivo consiste en implementar las diversas estructuras de datos vistas en clase y realizar experimentos de medición asociados. Para ello deberán implementar las siguientes estructuras de datos: lista enlazada, árbol binario simple, árbol binario autobalanceado y tabla de dispersión (tabla *hash*). (+5 puntos extra si investigan, implementan y utilizan un segundo tipo de árbol autobalanceado).

Para cada una de las clases debe implementar su constructor, destructor, método de *inserción* y método de *búsqueda*. En el caso de la lista enlazada y del árbol binario simple deberá implementar además un método de *borrado*. En todos los casos deberá hacer un manejo adecuado de memoria y no poseer **ninguna** fuga.

Para cada una de las estructuras implementadas deberá evaluar el rendimiento de las estructuras para 2 diferentes casos: inserción de valores ordenados e inserción de valores aleatorios. Para cada uno de los grupos de valores insertados, deberá tomar también mediciones de la duración al realizar búsqueda aleatorias. En el caso de la lista enlazada y del árbol binario simple deberá medir además la duración del borrado de elementos aleatorios.

Deberá repetir cada una de las mediciones para una cantidad variable de inserciones/búsquedas/borrados, de tamaños: 131072 (2^{17}), 262144 (2^{18}), 524288 (2^{19}), 786432 ($2^{19} + 2^{18}$) y 1048576 (2^{20}). Los valores aleatorios a ser generados para los métodos deben estar en el intervalo de **[0, 2n[** donde **n** es la cantidad de inserciones, búsquedas o borrados a realizar. Considere que el método `rand()` varía de computadora en computadora, y en algunos casos sólo alcanza valores máximos cercanos a 32k, por lo que deberá usar su creatividad para hacerle frente a este problema.

Las mediciones de las duraciones deben hacerse con precisión en milisegundos, por lo que se le recomienda utilizar la biblioteca `<chrono>` para este fin. Además, lo ideal sería tomar 3 veces cada medición y calcular su tiempo con base en el promedio de las 3 mediciones.

Con los datos obtenidos deberá redactar un entregable escrito con los resultados de sus experimentos: tablas con los valores obtenidos y gráficos de comparación entre los

tiempos de duración de las diferentes estructuras. Deberá explicar el comportamiento observado y si dicho comportamiento es el que esperaba por parte de sus estructuras.

Evaluación:

- Lista enlazada y árbol binario simple (15 puntos c/u):
    - Constructor y destructor: 2 puntos
    - Inserción: 5 puntos
    - Búsqueda: 3 puntos
    - Borrado: 5 puntos
- Árbol balanceado y tabla de dispersión (15 puntos c/u):
    - Constructor y destructor: 2 puntos
    - Inserción: 8 puntos
    - Búsqueda: 5 puntos
- Código main (10 puntos):
    - Debe permitir ejecutar las pruebas y verificar el funcionamiento del código
- Documento escrito (30 puntos):
    - Debe incluir los gráficos y tablas con los datos obtenidos.
        - Utilice los estándares para numeración de tablas e imágenes.
    - Puede redactarlo en Word o Latex pero debe entregarse como PDF.
    - Las gráficas puede hacerlas con su herramienta favorita (e.g: Excel, Python, Drive, etc.)
    - Si menciona algo proveniente de algún libro o artículo, recuerde citar.

Se puede trabajar de manera individual o en parejas. Pero solo **una** persona por grupo debe enviar el trabajo, de lo contrario perderán 5 puntos por envío duplicado. (Los asistentes siempre agradecen cuando evitan enviar duplicados)

1. Resultados - Tablas - Algoritmos Obligatorios

Para todas esta respectivas pruebas se hicieron con dos casos, números aleatorios y numeros ordenados, con el fin de ver cómo se comportan estas distintas estructuras de datos. Los números aleatorios fueron creados en un rango $[0, 2n]$ donde n es un algún valor del conjunto $\{131072, 262144, 524288, 786432, 1048576\}$. Para cada uno de los resultados que se muestran son el número promedio de realizar tres pruebas, tal cual solicitaba el enunciado. Y los resultados se muestran en milisegundos. Para el caso de la lista enlazada y el árbol binario se solictaban hacer pruebas de *borrado, inserción y busqueda*; mientras que para el árbol AVL, árbol Rojinegro y Tabla Hash solo se pedían *busqueda e inserción*. Cabe destacar lo siguiente, realizar una prueba para los números aquí solicitados es una tarea de largas horas de procesamiento de computadora pues solo con la cantidad más pequeña los tiempos estuvieron alrededor de 20 minutos, por esta razón no se han adjuntado dichos resultados, sin embargo en el *main* se encuentran las lineas de codigo necesarias en caso de querer correr estas pruebas.

1.1. Lista Enlazada

Para **131072 números** se obtuvieron los siguientes resultados:

Tipo de Número	Inserción	Busqueda	Borrado
Ordenados	26521.3	35111.7	53896.7
Aleatorios	25400.0	37274.3	55821.7

1.2. Árbol Binario Simple

Para **131072 números** se obtuvieron los siguientes resultados:

Tipo de Número	Inserción	Busqueda	Borrado
Ordenados	25823.3.3	36497.3.7	56591.3
Aleatorios	25129.0	37011.0	56132.7

1.3. Árbol Binario AVL

Para **131072 números** se obtuvieron los siguientes resultados:

Tipo de Número	Inserción	Busqueda
Ordenados	9.0	0.66667
Aleatorios	10.6667	1.0

1.4. Tabla Hash

Para **131072 números** se obtuvieron los siguientes resultados:

Tipo de Número	Inserción	Busqueda
Ordenados	11765.0	11749.3
Aleatorios	11977.0	11892.0

2. Resultados - Gráficos - Todos los Algoritmos

Nota: Las pruebas y los resultado usados para el desarrollo de estos gráficos son resultados promedio de los datos de tener tres pruebas distintas, por lo que no es en base a una solo prueba sino a tres.

Gráfico 1: Duración de las estructuras de datos para la **inserción** de números aleatorios.

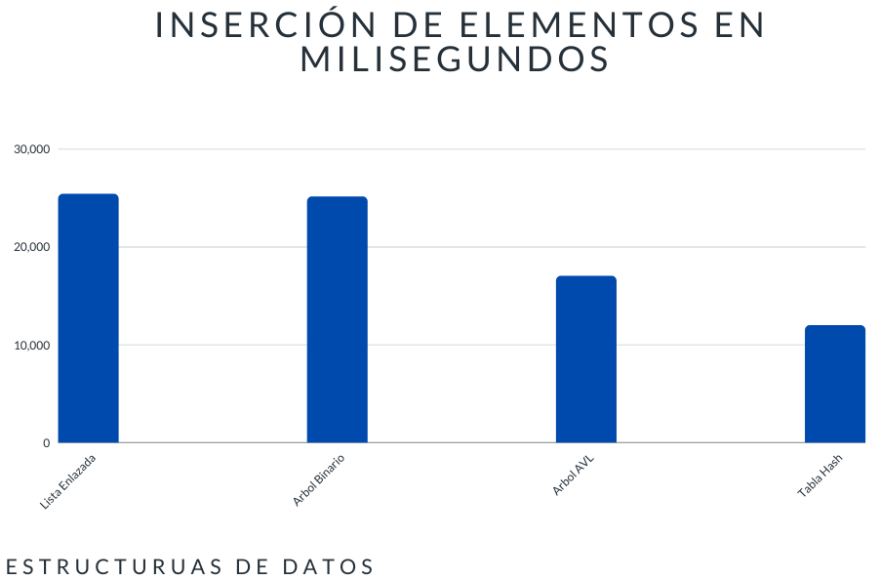


Gráfico 2: Duración de las estructuras de datos para la **busqueda** de números aleatorios.

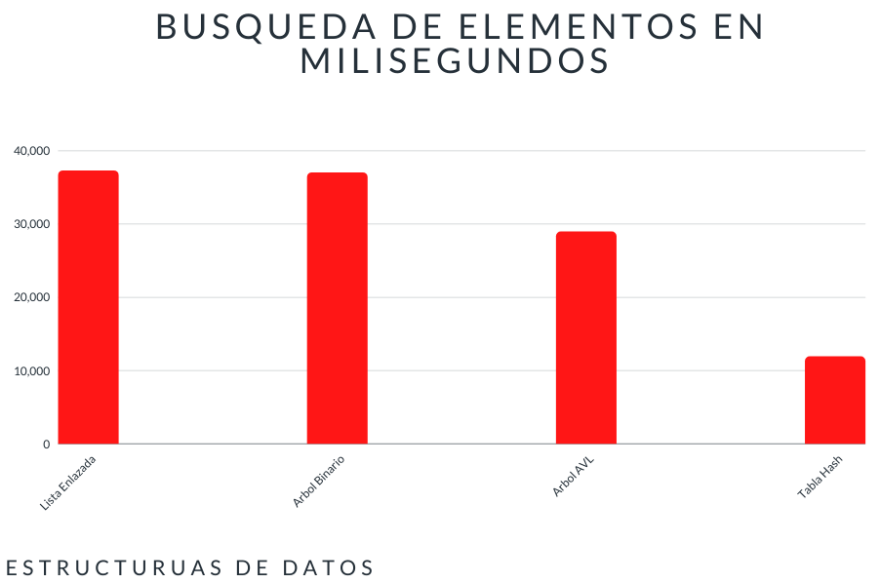
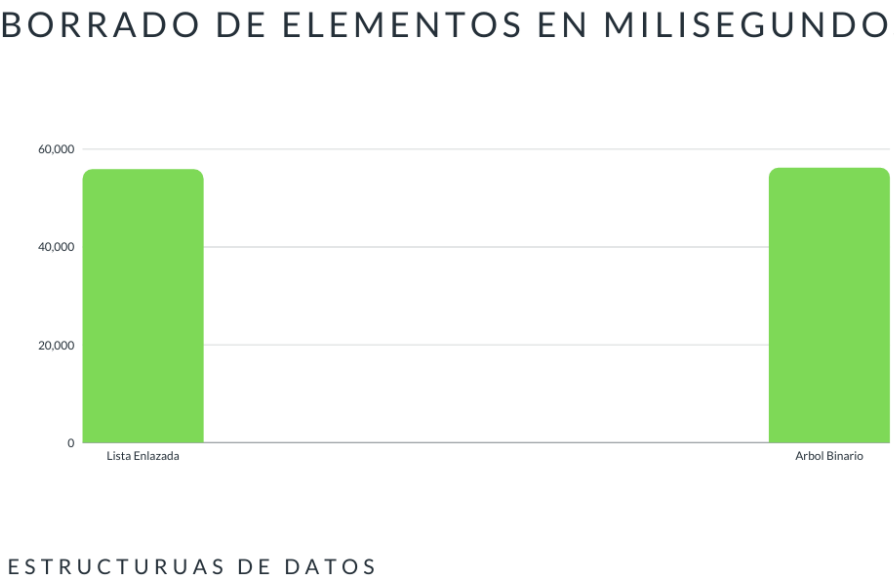


Gráfico 3: Duración de las estructuras de datos para el **borrado** de números aleatorios.



3. Conclusiones

Al ver los gráficos y los resultados del reporte adjunto podemos notar que la tabla hash, según que tan buena sea su función has, es la que nos provee de mejores tiempo para busqueda y borrado. Es sumamente importante hacer mención de que no existe un algoritmo perfecto para todas las situaciones, cada uno es necesario evaluarlo según su deseada implementación pues el que podría ser la mejor opción para ciertos datos, cirscuntancias y equipo de computación, podría ser el peor para otras sitaciones con otras circunstancias. Por poner un ejemplo, la lista enlazada es la que provee peores tiempos pero podría ser útil en casos donde no se tiene mucha memoria puesta esta no hace uso de la recursividad principalmente. Por otro lado también es importante notar que no siempre lo más complicado es lo más sencillo, el concepto de la *Tabla Hash* es un concepto sencillo pero que sin embargo al ser implementada provee los mejores tiempos, a diferencia de con otros tipos de estructuras. Según lo que hemos visto en clases este comportamiento era de esperarse pues la materia y las demostraciones matemáticas indicaban que estos algoritmo como la *Tabla Hash* tendrían un mejor desempeño. A gran escala es incluso más notorio, a la hora de realizarse estas pruebas, por dar un contraste, la *Lista Enlazada*, que obtuvo los resultado más lentos, aproximadamente duró 20 minutos corriendo; mientras que la *Tabla Hash* duró tan solo aproximadamente 10 minutos. Tómese en cuenta que estas pruebas eran con cantidades de datos muy grandes y que se repitian a si mismas tres veces pues se debían hacer de esta forma y así obtener el promedio.