CI-0116 Análisis de Algoritmos y Estructuras de Datos
II ciclo 2024
Prof. Dr. Allan Berrocal Rojas

Tarea II

Índice

1. Indicaciones generales

Esta tarea es individual y debe entregarla en la fecha que se indica en la plataforma de Mediación Virtual. Si lo desea, el(la) estudiante puede colaborar con sus compañeros(as) pero de manera profesional. Es decir, antes de discutir los problemas de la tarea con sus compañeros(as), primero dedique al menos un día de trabajo individual y formule sus propias respuestas para la tarea. Aunque colabore con algún compañero(a), todo el material que desarrolla y entrega en su tarea debe ser hecho por usted mismo(a). Cada estudiante debe estar en capacidad de defender correctamente las respuestas de su trabajo de manera verbal si la persona docente o la persona asistente del curso formula una pregunta durante la revisión del trabajo. Si la persona docente sospecha que la tarea es parcial o totalmente producto de plagio, la tarea recibirá una nota de cero y se iniciará el debido proceso estipulado en el Reglamento de Orden y Disciplina de los Estudiantes Universidad de Costa Rica. En el reporte a entregar, debe aparecer la referencia toda fuente externa de producción intelectual que utilizó para durante el desarrollo de este trabajo. Recuerde que se considera plagio presentar como propio, material parcial o totalmente creado por otras personas u obtenido de fuentes de información, como por ejemplo de libros, fuentes de Internet, sistemas de generación de texto basados en modelos de aprendizaje automático y profundo, entre otras.

2. Objetivo

El objetivo de la tarea es implementar algunas de las estructuras de datos estudiadas en el curso con el objeto de realizar experimentos con dichas estructuras y sus correspondientes operaciones. En la fase experimental se debe recolectar información sobre el desempeño de los algoritmos. Se espera que la persona estudiante sea capaz de analizar los resultados, reflexionar y comparar la eficiencia teórica de las estructuras de datos y sus algoritmos con la eficiencia observada durante los experimentos.

3. Estructuras de Datos a Implementar

Las estructuras de datos a implementar son las siguientes: lista simplemente enlazada (*Singly Linked List*), árbol de búsqueda binaria (*Binary Search Tree*), árbol rojinegro (*Red-Black Tree*), y tabla de dispersión (*Hash Table*) con el mecanismo de encadenamiento para resolver colisiones (implementando una lista doblemente enlazada *Doubly Linked List*).

4. Código Fuente de los Algoritmos

El código debe escribirse en el lenguaje de programación C++, tomando como base las plantillas publicadas junto a este enunciado:

- `SinglyLinkedList.hpp`
- `BinarySearchTree.hpp`
- `RedBlackTree.hpp`
- `ChainedHashTable.hpp`
- `DoublyLinkedList.hpp`

Los encabezados de las funciones en las plantillas ***no deben ser alterados***, ya que la persona asistente del curso usará un guion (*script*) para revisar la tarea y cualquier cambio en la interfaz de los métodos afectará la compilación. (Sin embargo, se pueden agregar métodos privados y llamarlos dentro de los métodos definidos en la plantilla si fuera necesario). Se debe usar el lenguaje en su versión estándar para evitar problemas de compilación. En particular, el código debe poder ser compilado usando `g++` y las bibliotecas estándar de C++. *Si el código no compila o no funciona correctamente, recibirá una nota de cero.*

Modifique la función `datosDeTarea()` del archivo «`Student.hpp`» para que devuelva una hilera con su número de carné, nombre completo e identificador de la tarea **[0.5 pts.]**. Por ejemplo, si su número de carné es B12345, y su nombre es ABC esta función deberá devolver la siguiente cadena: `Carné: 12345, Nombre: ABC, Tarea 2`.

Debe programar siguiendo las mejores prácticas estudiadas en la carrera incluyendo documentación de todas las clases y métodos que implementa. Para facilitar su labor, puede utilizar herramientas como doxygen.

4.1. Lista Simplemente Enlazada

Implemente la clase *lista enlazada con nodo centinela* usando la plantilla «`SinglyLinkedList.hpp`» y los siguientes métodos:

- [1 pto.] constructor
- [2 pts.] destructor
- [2 pts.] insertar [1]
- [2 pts.] buscar [2]
- [2 pts.] remover [3]

[1] La estructura de datos sí permite insertar elementos duplicados.
[2] Si el valor buscado está en la estructura de datos, el método retorna la primera aparición.
[3] Si el nodo está en la estructura de datos, la operación de borrado debe borrar todas las apariciones.

4.2. Árbol de Búsqueda Binaria

Implemente la clase *árbol de búsqueda binaria*, usando la plantilla «`BinarySearchTree.hpp`» y los siguientes métodos:

- [1 pto.] constructor
- [2 pts.] destructor
- [2 pts.] insertar [4]
- [3 pts.] remover
- [3 pts.] búsqueda iterativa
- [2 pto.] obtener máximo (iterativo)
- [2 pto.] obtener mínimo (iterativo)
- [2 pts.] obtener sucesor (iterativo)
- [3 pts.] recorrido "en orden" (iterativo)

4.3. Árbol Rojinegro

Implemente la clase *árbol rojinegro*, usando la plantilla «`RedBlackTree.hpp`» y los siguientes métodos:

- [1 pto.] constructor
- [2 pts.] destructor
- [5 pts.] insertar [5]
- [5 pts.] remover
- [3 pts.] búsqueda (iterativa)
- [1 pto.] obtener máximo (iterativo)
- [1 pto.] obtener mínimo (iterativo)
- [2 pts.] obtener sucesor (iterativo)

4.4. Tabla de Dispersión

Implemente la clase *tabla de dispersión*, usando la plantilla «`ChainedHashTable.hpp`» y los siguientes métodos:

- [1 pto.] constructor
- [1 pto.] destructor
- [3 pts.] insertar [6]
- [3 pts.] buscar
- [2 pts.] remover

Resuelva las colisiones mediante la técnica de encadenamiento utilizando una *lista doblemente enlazada*[7] que deberá programar en la plantilla «`DoublyLinkedList.hpp`» modificando la lista simplemente enlazada que creó en el punto 4.1 **[2 pts.]**. La función de dispersión a utilizar es

[4] La estructura de datos no permite insertar elementos duplicados.
[5] La estructura de datos no permite insertar elementos duplicados.
[6] La estructura de datos no permite insertar elementos duplicados.
[7] Esto para garantizar que la operación «`remove`» tenga un tiempo de duración $\Theta(1)$.

$h(k) = k \mod m$, donde m es el tamaño de la tabla (más detalles sobre esto adelante) y $\mod$ representa la operación módulo.

5. Recolección de Datos

Para cada una de las estructuras de datos que implementa, debe realizar una serie de operaciones y registrar los resultados que posteriormente utilizará para hacer análisis cuantitativos y discutir sobre las diferencias y similitudes entre la teoría y los resultados observados.

5.1. Lista Enlazada

5.1.1. Inserción Aleatoria

1. Inserte en una lista vacía $n = 1\,000\,000$ nodos cuyas llaves sean enteros seleccionados aleatoriamente en el rango $[0, 3n)$ es decir, $0 \leq x < 3n$. **[1 pto.]**.
2. Busque en la lista un total de $e = 10\,000$ elementos cuyo valor o llave es seleccionado al azar en el mismo rango $[0, 3n)$ y registre el tiempo de duración del total de búsquedas realizadas incluyendo los casos en que el elemento buscado está en la lista, y los casos en que no lo está. **[1 pto.]**
3. Elimine de la lista un total de $e = 10\,000$ elementos cuyo valor o llave es seleccionado al azar en el mismo rango $[0, 3n)$ y registre el tiempo de duración del total de eliminaciones realizadas incluyendo los casos en que el elemento buscado está en la lista y es eliminado, y los casos en que el elemento no está en la lista. **[1 pto.]**

5.1.2. Inserción Ordenada

1. Inserte en una lista vacía las llaves $0, 1, ..., n-1$, en ese orden con $n = 1\,000\,000$. **[1 pto.]**
2. Busque en la lista un total de $e = 10\,000$ elementos seleccionados al azar en el rango $[0, 3n)$ registrando el tiempo de duración del total de búsquedas realizadas incluyendo los casos en que el elemento buscado está en la lista, y los casos en que no lo está. **[1 pto.]**
3. Elimine de la lista un total de $e = 10\,000$ elementos cuyo valor o llave es seleccionado al azar en el mismo rango $[0, 3n)$ y registre el tiempo de duración del total de eliminaciones realizadas incluyendo los casos en que el elemento buscado está en la lista y es eliminado, y los casos en que el elemento no está en la lista. **[1 pto.]**

5.1.3. Mediciones

1. Repita los pasos anteriores al menos tres veces registrando en un cuadro el tiempo de duración de cada corrida y su respectivo promedio[8] **[1 pts.]**
2. Cree un gráfico que compare el tiempo obtenido en la sección 5.1.1 ítem 2 contra el tiempo obtenido en la sección 5.1.2 ítem 2. Es decir, el tiempo de duración de las operaciones de búsqueda en una lista no ordenada contra el tiempo de duración de las operaciones de búsqueda en una lista ordenada. **[1 pts.]**
3. Cree un gráfico que compare el tiempo obtenido en la sección 5.1.1 ítem 3 contra el tiempo obtenido en la sección 5.1.2 ítem 3. Es decir, el tiempo de duración de las operaciones de eliminación en una lista no ordenada contra el tiempo de duración de las operaciones de eliminación en una lista ordenada. **[1 pts.]**
4. Asegúrese de indicar en cada gráfico las unidades de tiempo utilizadas. Identifique si en alguno de los casos el tiempo de duración fue sustancialmente mayor que en el otro, e indique si esto corresponde a lo esperado. Esto lo utilizará en su reporte escrito.

5.2. Árbol de Búsqueda Binaria

1. Repita los pasos de la sección 5.1.1 con la misma cantidad de llaves pero ahora usando el árbol de búsqueda binaria como estructura de datos. **[3 pts.]**
2. Repita los pasos de la sección 5.1.2 con la misma cantidad de llaves pero ahora usando el árbol de búsqueda binaria como estructura de datos. **[3 pts.]**
   Para este paso, note que insertar n llaves ordenadas en un árbol de búsqueda binaria puede tomar demasiado tiempo si n es grande —¿por qué?—. Para evitar la larga espera, considere crear un método (`fastInsert(size_t n)` en la plantilla «`BinarySearchTree.hpp`») que produzca un árbol idéntico al que se crearía si se insertaran en él las llaves $0, 1, ..., n-1$, en ese orden, pero sin usar el método de inserción —¿cómo?—).
3. Repita los pasos de la sección 5.1.3 realizando un gráfico comparativo de los resultados de las búsquedas y las eliminaciones pero utilizando ahora el árbol de búsqueda binaria como estructura de datos de referencia. **[3 pts.]**

[8] Si el tiempo mayor es 1,5 veces más grande que el tiempo menor, se debe posiblemente a que la máquina estuvo ocupada en otras cosas mientras realizaba estas operaciones. En tal caso elimine una de las corridas y vuelva a ejecutarla. Repita esto cuantas veces sea necesario para lograr que el tiempo mayor no sea más grande que 1,5 veces el tiempo menor para cada corrida.

5.3. Árbol Rojinegro

1. Repita los pasos de la sección 5.1.1 con la misma cantidad de llaves pero ahora usando el árbol rojinegro como estructura de datos. **[3 pts.]**
2. Repita los pasos de la sección 5.1.2 con la misma cantidad de llaves pero ahora usando el árbol rojinegro como estructura de datos. **[3 pts.]**
3. Repita los pasos de la sección 5.1.3 realizando un gráfico comparativo de los resultados de las búsquedas y las eliminaciones pero utilizando ahora el árbol rojinegro como estructura de datos de referencia. **[3 pts.]**

5.4. Tabla de Dispersión

1. Repita los pasos de la sección 5.1.1 con la misma cantidad de llaves pero ahora usando como estructura de datos una tabla de dispersión de tamaño m cuyo factor de carga sea $\alpha = 1$ (es decir, $m = n$). **[3 pts.]**
2. Repita los pasos de la sección 5.1.2 con la misma cantidad de llaves pero ahora usando como estructura de datos una tabla de dispersión de tamaño m cuyo factor de carga sea $\alpha = 1$ (es decir, $m = n$). **[3 pts.]**
3. Repita los pasos de la sección 5.1.3 realizando un gráfico comparativo de los resultados de las búsquedas y las eliminaciones pero utilizando ahora la tabla de dispersión como estructura de datos de referencia. **[3 pts.]**

5.5. Comparación de Resultados

1. Haga un gráfico que muestre el tiempo de duración de las búsquedas de llaves cuando fueron insertadas de manera aleatoria en todas las estructuras de datos implementadas. **[1 pto.]**
2. Haga un gráfico que muestre el tiempo de duración de las búsquedas de llaves cuando fueron insertadas de manera ordenada en todas las estructuras de datos. **[1 pto.]**
3. Haga un gráfico que muestre el tiempo de duración de las eliminaciones de llaves cuando fueron insertadas de manera aleatoria en todas las estructuras de datos. **[1 pto.]**
4. Haga un gráfico que muestre el tiempo de duración de las eliminaciones de llaves cuando fueron insertadas de manera ordenada en todas las estructuras de datos. **[1 pto.]**

Mientras obtiene esta información, tome nota de los siguientes puntos ya que proveen insumos que podrá utilizar en su reporte escrito.

1. Identifique en cada uno de los gráficos si los resultados son los esperados.
2. Compare los tiempos de duración de las búsquedas sobre llaves aleatorias y llaves ordenadas e identifique si algunas de las estructuras de datos fueron sustancialmente mejor que otras para este ejercicio. Identifique aquellas que tardaron la mitad o menos de tiempo en realizar las búsquedas. Argumente si esto corresponde con lo esperado.
3. Compare los tiempos de duración de las eliminaciones sobre llaves aleatorias y llaves ordenadas e identifique si algunas de las estructuras de datos fueron sustancialmente mejor que otras para este ejercicio. Identifique aquellas que tardaron la mitad o menos de tiempo en realizar las búsquedas. Argumente si esto corresponde con lo esperado.

6. Reporte de Resultados

Luego de recolectar datos como se indicó en la sección anterior, usted debe discutir los resultados observados que ya tiene registrados en su hoja de datos.

Para escribir el reporte, debe utilizar como base la plantilla de artículos científicos de la IEEE. Para simplificar el proceso, el docente le brindará una plantilla que puede utilizar en **Microsoft Word** o en **LaTeX**.

El reporte escrito tiene un valor de **[20 pts.]**. Algunos de los elementos a revisar son: redacción, ortografía, cumplimiento de las secciones solicitadas, calidad y correctitud de los gráficos generados, claridad del análisis, profundidad, pertinencia y correctitud del análisis, integridad del contenido (correcto uso de referencias a material utilizado), correcto uso de la nomenclatura aplicable.

El reporte se debe entregar en **formato PDF únicamente**. NINGÚN OTRO FORMATO SERÁ ACEPTADO.

7. Descripción del Producto a Entregar

La tarea tiene una única entrega que contiene el código fuente de los algoritmos implementados en las plantillas que se le brindaron, y un informe escrito en formato PDF siguiendo los lineamientos e indicaciones para el reporte de la primera tarea programada.

Entregue su trabajo en su repositorio privado de control de versiones en una estructura de directorios **exactamente** como se muestra a continuación: **[0.5 pts.]**

- `\tareas_programadas\tp2\`
  Coloque aquí la versión del código fuente así como el documento de reporte en formato PDF.

Asegúrese de utilizar un archivo `.gitignore` para evitar subir al repositorio de control de versiones archivos temporales o autogenerados durarante el proceso de compilación. **[0.5 pts.]**

Es responsabilidad de la persona estudiante verificar que la plataforma haya recibido la tarea y que esté intacta (bajando la tarea y verificando la integridad del comprimido). En caso de que haya múltiples entregas, se considerará solamente la *última*, y si esta se entregó después de la fecha y hora límites, se aplicará la penalización acordada en la Carta al Estudiante.

Si tiene dudas sobre cómo utilizar el repositorio de control de versiones, busque al asistente del curso o a la persona docente **con antelación** para recibir ayuda. No espere hasta el día de la entrega para solicitar ayuda porque corre el riesgo de que no se le pueda atender.

8. Desgloce de la evaluación

El siguiente cuadro muestra un resumen de los elementos a evaluar en esta tarea.

Cuadro 1: Resumen de los elementos a evaluar

Elemento	Valor
Implementación de lista simplemente enlazada (sec. 4.1)	9
Datos de ejecución lista simplemente enlazada (sec. 5.1)	9
Implementación de árbol de búsqueda binaria (sec. 4.2)	20
Datos de ejecución árbol de búsqueda binaria (sec. 5.2)	9
Implementación de árbol rojinegro (sec. 4.3)	20
Datos de ejecución árbol rojinegro (sec. 5.3)	9
Implementación de tabla de dispersión (sec. 4.4)	10
Lista doblemente enlazada para la tabla de dispersión (sec. 4.4)	2
Datos de ejecución tabla de dispersión (sec. 5.4)	9
Función `datosDeTarea()` (sec. 4)	0.5
Gráficos comparativos para las estructuras de datos (sec. 5.5)	4
Reporte escrito (sec. 6)	20
Estructura de directorios	0.5
Buen uso del repositorio y `.gitignore`	0.5
Total	**122.5 pts.**