UNIVERSIDAD DE COSTA RICA

Escuela Ciencias de la Computación e Informática

CI-0116 Análisis de Algoritmos y Estructuras de Datos

Tarea II

1. Objetivo

El objetivo de la tarea es implementar las estructuras de datos y operaciones estudiadas en el segundo tercio del curso y comprobar que sus diferencias en cuanto a eficiencia teórica corresponden con la realidad.

2. Estructuras de datos

Las estructuras de datos a implementar son las siguientes: listas enlazadas con nodo centinela y árboles de búsqueda binarios para la **primera parte**, y árboles rojinegros y tablas de dispersión para la **segunda parte**. Las siguientes secciones describen las operaciones a implementar y los experimentos a realizar para evaluarlas y compararlas entre sí.

2.1. Listas enlazadas con nodo centinela (20 pts.)

1. Implemente la clase *lista enlazada con nodo centinela* usando la plantilla `llist.h`. Los métodos a implementar son los siguientes: construcción [1 pto.], destrucción [3 pts.], inserción [4 pts.], búsqueda [3 pts.] y borrado [4 pts.].

2. Inserte en una lista vacía $n = 1\,000\,000$ de nodos cuyas llaves sean enteros seleccionados aleatoriamente en el rango $[0, 2n)$. Seleccione elementos al azar en el mismo rango $[0, 2n)$ y búsquelos en la lista (estén o no en ella) registrando el número de búsquedas realizadas en un lapso de diez segundos. [1 pto.]

3. Inserte en una lista vacía las llaves $0, 1, ..., n-1$, en ese orden. Seleccione elementos al azar en el rango $[0, 2n)$, y registre el número de búsquedas que se logró hacer en un lapso de 10 segundos. [1 pto.]

4. Indique si en alguno de los dos casos (inserción de números aleatorios o inserción de números secuenciales) se realizó una cantidad de búsquedas (exitosas o fallidas, no importa) sustancialmente mayor que en el otro (más del doble), e indique si esto corresponde a lo esperado. Explique. [3 pts.]

2.2. Árboles de búsqueda binarios (30 pts.)

1. Implemente la clase *árbol de búsqueda binario*, usando la plantilla `bstree.h`. Los métodos a implementar son: construcción [1 pto.], destrucción [3 pts.], inserción [4 pts.], borrado [5 pts.], búsqueda de llave recursiva [2 pto.] e iterativa [3 pts.], búsqueda del máximo [1 pts.], mínimo [1 pts.], sucesor de un nodo [2 pts.] y recorrido del árbol en orden [3 pts.].

2. Repita el punto 2 de la sección 2.1 usando un árbol de búsqueda binario que tenga la **misma cantidad de llaves** que en esa sección. [1 pto.]

3. Repita el punto 3 de la sección 2.1 usando un árbol de búsqueda binario que tengan la **misma cantidad de llaves** que en esa sección. (Insertar n llaves ordenadas en un árbol de búsqueda binario puede tomar demasiado tiempo si n es grande —¿por qué?—. Para evitar la larga espera, considere crear un método 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?—). [1 pto.]

4. Compare el número de búsquedas realizadas en el punto 2 de las secciones anteriores e indique si alguna de las estructuras de datos (lista enlazada o árbol de búsqueda binario) fue sustancialmente mejor que la otra (es decir, si permitió realizar más del doble de búsquedas). [1 pto.]

5. Compare el número de búsquedas realizadas en el punto 3 de las secciones anteriores, e indique si alguna de las estructuras de datos (lista enlazada o árbol de búsqueda binario) fue sustancialmente mejor que la otra (es decir, si permitió realizar más del doble de búsquedas) y si esto corresponde a lo esperado. Explique. [2 pts.]

2.3. Árboles rojinegros (30 pts.)

1. Implemente la clase *árbol rojinegro* usando la plantilla `rbtree.h`. Esta plantilla incluye los mismos métodos que la clase *árbol de búsqueda binario*, **excepto la operación de borrado**: construcción [2 pts.], destrucción [3 pts.], inserción [8 pts.], búsqueda de llave recursiva [2 pts.] e iterativa [3 pts.], búsqueda del máximo [1 pto.], mínimo [1 pto.], sucesor de un nodo [2 pto.] y recorrido de un árbol en orden [3 pts.].

2. Repita el punto 2 de la sección 2.1 usando un árbol rojinegro que tenga la misma cantidad de llaves que en esa sección. [1 pto.]

3. Repita el punto 3 de la sección 2.1 usando un árbol rojinegro que tenga la misma cantidad de llaves que en esa sección. [1 pto.]

4. Compare el número de búsquedas realizadas en el punto 2 de las secciones anteriores e indique si el árbol rojinegro fue sustancialmente mejor que la mejor de las otras estructuras (es decir, si permitió realizar más del doble de búsquedas). [1 pto.]

5. Compare el número de búsquedas realizadas en el punto 3 de las secciones anteriores e indique si el árbol rojinegro fue sustancialmente mejor que la mejor de las otras estructuras (es decir, permitió realizar más del doble de búsquedas) y si esto corresponde a lo esperado. Explique. [2 pts.]

2.4. Tablas de dispersión (20 pts.)

1. Implemente la clase *tabla de dispersión* usando la plantilla `chasht.h`. Los métodos a implementar son: construcción [3 pts.], destrucción [3 pts.], inserción [4 pts.] y búsqueda [4 pts.]. Las colisiones deben ser resueltas mediante encadenamiento (puede usar la clase `list` de STL). La función de dispersión a utilizar es $h(k) = k$ mód m, donde m es el tamaño de la tabla (más detalles sobre esto adelante).

2. Repita el punto 2 de la sección 2.1 usando una tabla de dispersión de tamaño m que tenga la misma cantidad de llaves que la lista utilizada en esa sección y cuyo factor de carga sea $\alpha = 1$ (es decir, $m = n$). [1 pto.]

3. Repita el punto 3 de la sección 2.1 usando una tabla de dispersión de tamaño m que tenga la misma cantidad de llaves que la lista utilizada en esa sección y cuyo factor de carga sea $\alpha = 1$ (es decir, $m = n$). [1 pto.]

4. Compare el número de búsquedas realizadas en el punto 2 de las secciones anteriores e indique si la tabla de dispersión fue sustancialmente mejor que la mejor de las otras estructuras (es decir, si permitió realizar más del doble de búsquedas) y si esto corresponde a lo esperado. Explique. [2 pts.]

5. Compare el número de búsquedas realizadas en el punto 3 de las secciones anteriores, y diga si la tabla de dispersión fue sustancialmente mejor que la mejor de las otras estructuras (es decir, si permitió realizar más del doble de búsquedas) y si esto corresponde a lo esperado. Explique. [2 pts.]

3. Entregables

Los entregables son los mismos que para la tarea I: un informe en formato PDF y el código de las clases implementadas.