UNIVERSIDAD DE COSTA RICA

Escuela Ciencias de la Computación e Informática

CI-0116 Análisis de Algoritmos y Estructuras de Datos

II ciclo de 2018

Tarea II

Fecha limite de entrega:
Primera parte: miércoles 10 de octubre
Segunda parte: lunes 22 de octubre

1. Preludio a la tarea III

Polynizer ha contratado a los excelentes estudiantes de los grupos 3 y 4 de CI-0116 para que creen un algoritmo que le permita escoger la forma de tocar en guitarra los acordes que su aplicación infiere, de forma tal que se minimice la cantidad de movimiento requerido de la mano. El primer paso para esto es crear una colección de datos de prueba que se usarán para probar el algoritmo. Para esto busque en internet los acordes de su canción favorita, o determínelos usted mismo, si tiene la capacidad de hacerlo. Escriba los acordes sobre la letra de la canción en un archivo de texto con extensión *txt*, usando un tipo de letra monoespaciada, como se muestra en la figura 1. Este preludio es opcional y tiene un valor de 5 puntos extra que se suman a la calificación de la tarea III.

2. Objetivo

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

3. Estructuras de datos

Las estructuras de datos a implementar son las siguientes: (I) listas enlazadas con nodo centinela y (II) 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.

```
EL DÍA QUE ME QUIERAS

               Ab     Eb/G
 Acaricia mi ensueño
             Fm7    Bb7
 el suave murmullo
           Eb7sus Eb7 Gm7b5
 de tu suspirar.

      C7     Fm9
 Como ríe la vida
             Bb7
 si tus ojos negros
              Eb11 Eb7
 me quieren mirar.
```

Figura 1: Acordes de extracto de la canción *El día que me quieras*, del tanguista argentino Carlos Gardel. Los acordes corresponden a la versión interpretada por el cantante mexicano Luis Miguel.

3.1. Listas enlazadas con nodo centinela

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 [2 pts.], inserción [2 pts.], búsqueda [2 pts.] y borrado [3 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. [2 pts.]

3.2. Árboles de búsqueda binarios

1. Implemente la clase *árbol de búsqueda binario*, usando la plantilla `bstree.h`. Los métodos a implementar son: construcción [1 pto.], destrucción [2 pts.], inserción

[2 pts.], borrado [3 pts.], búsqueda de llave recursiva [1 pto.] e iterativa [1 pto.], búsqueda del máximo [1 pts.], mínimo [1 pts.], sucesor de un nodo [2 pts.] y recorrido del árbol en orden [1 pts.].

2. Repita el punto 2 de la sección 3.1 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 3.1 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.]

3.3. Árboles rojinegros

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 [1 pto.], destrucción [2 pts.], inserción [10 pts.], búsqueda de llave recursiva [1 pto.] e iterativa [1 pto.], búsqueda del máximo [1 pto.], mínimo [1 pto.], sucesor de un nodo [2 pto.] y recorrido de un árbol en orden [1 pto.].

2. Repita el punto 2 de la sección 3.1 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 3.1 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.]

3.4. Tablas de dispersión

1. Implemente la clase *tabla de dispersión* usando la plantilla `hasht.h`. Los métodos a implementar son: construcción [2 pts.], destrucción [2 pts.], inserción [2 pts.] y búsqueda [2 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 3.1 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 3.1 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.]

4. Entregables

Los entregables son los mismos que para la tarea I: un informe y el código de las clases implementadas. El cuadro 1 muestra el puntaje que vale cada una de las secciones del informe. Refiérase al enunciado de la tarea I para los detalles del contenido de cada sección del informe.

Cuadro 1: Puntaje de cada uno de los elementos del informe. Los puntos de la sección de resultados corresponden con los ítemes 2 a 5 de las listas de las secciones 3.1, 3.2, 3.3 y 3.4.

Elemento	Puntos
Título	1
Resumen	4
Introducción	5
Metodología	10
Resultados	20
Conclusiones	5
Bibliografía	2