Universidad de Talca
Facultad de Ingeniería
Departamento de Ciencias de la Computación

Algoritmos y estructuras de datos

Tarea 4

Profesor: Rodrigo Paredes, Ayudante: Erik Regla

Plazo: Martes 9 de Diciembre de 2014. Estricto!!

1. Objetivos

Es conocido que un árbol de búsqueda binaria se puede desbalancear mucho. En esta tarea vamos a considerar una variante del ABB que llamaremos ABB con mediana de tres (ABB-m3), y se espera que el alumno realice la comparación entre el ABB-m3 con el ABB estándar.

2. Descripción de la tarea

En esta tarea vamos a considerar una variante del ABB que llamaremos ABB con mediana de tres (ABB-m3). La idea es bastante simple. Los nodos internos tienen la misma estructura de un ABB, sin embargo en las hojas tendremos nodos que pueden almacenar hasta tres llaves. La idea es que el ABB-m3 resultante sea más balanceado que un ABB estándar.

La **búsqueda** opera de la siguiente forma. En la bajada por el árbol, los nodos internos permiten calcular la ruta hacia el nodo que debiese contener al elemento buscado, y cuando se llega a una hoja, el elemento se compara con las llaves que estén en el nodo. Si durante el recorrido desde la raíz a la hoja no se encuentra al objeto se declara que la búsqueda fue infructuosa. Considere la firma `nodo* buscar(int x)` en C++ o `nodo buscar(int x)` en Java, considerando programación orientada a objetos (OOP) de Java o C++. Si prefiere usar programación estructurada/procedural, use la firma `nodo* buscar(nodo *raiz, int x)` en C/C++ o `nodo buscar(nodo raiz, int x)` en Java.

Las **inserciones** se hacen en las hojas. Si en la hoja hay espacio para una nueva llave, se inserta y las llaves se almacenan en orden. Si la hoja está llena, primero se divide en tres, un nodo interno (que almacena la llave del medio), una hoja izquierda (que almacena la llave menor) y una hoja derecha (que almacena la llave mayor); y luego se inserta el nodo en la hoja correspondiente. Considere la firma `nodo* insertar(int x)` en C++ o `nodo insertar(int x)` en Java, considerando OOP. Si prefiere usar programación estructurada/procedural, use la firma `nodo* insertar(nodo *raiz, int x)` en C/C++ o `nodo insertar(nodo raiz, int x)` en Java.

Para la **eliminación**, es necesario localizar a la llave, una vez localizada se elimina y si en el subárbol quedan tres o menos llaves se compacta a una hoja. Como siempre, si la llave a eliminar

es un nodo interno, tienen que hacer los ajustes necesarios para poder hacer la eliminación. Esto es, si sólo hay llaves en un subarbol, conectar al padre de la llave a eliminar con el hijo (saltando al nodo), o si hay llaves en ambos hijos, buscar al mayor de los hijos menores (o alternativamente, el menor de los mayores) para reemplazar al elemento borrado y continuar con los ajustes. Considere la firma `nodo* borrar(int x)` en C++ o `nodo borrar(int x)` en Java, considerando OOP. Si prefiere usar programación estructurada/procedural, use la firma `nodo* borrar(nodo *raiz, int x)` en C/C++ o `nodo borrar(nodo raiz, int x)` en Java.

Adicionalmente, implementen una función para calcular la altura de un nodo, `int altura()` o `int altura(Nodo raiz)`, dependiendo si es OOP o programación estructurada.

Noten que las tres operaciones, buscar, insertar y borrar, son muy parecidas a las de un ABB estándar, salvo que las hojas pueden tener hasta tres llaves.

Experimentación.

Para ejecutar el programa deben utilizar la siguiente línea:

`tarea4 n`

El programa debe reportar para cada valor de n:

1. Tiempo de inserción de n llaves.
2. Altura del árbol.
3. Tiempo de búsqueda de 500000 llaves que pertenecen al árbol.
4. Tiempo de búsqueda de 500000 llaves que no pertenecen al árbol.
5. Tiempo de borrado de 500000 llaves que pertenecen al árbol.

Realicen pruebas para $n = 1, 2, \cdots, 10$, es decir, 1MB, 2MB, ..., 1MB $= 10^6$ enteros, graficando el tiempo para cada una de las operaciones, y la altura de los árboles obtenidos. Comparen los resultados con los que se obtienen de un ABB estándar.

Para la generación de los datos use el mismo generador de la tarea 3.

3. Restricciones

- Los programas debe ser escritos en C, C++ o Java.
- Plazo de entrega: Martes 9 de Diciembre de 2014. Si pueden venir antes a mostrar la tarea mejor. Así podrán hacerle correcciones. Estricto!!
- Queda prohibido utilizar bibliotecas o clases preconstruidas en el lenguaje que implementen árboles. USTED DEBE IMPLEMENTAR TODO.
- La tarea es individual o en parejas.

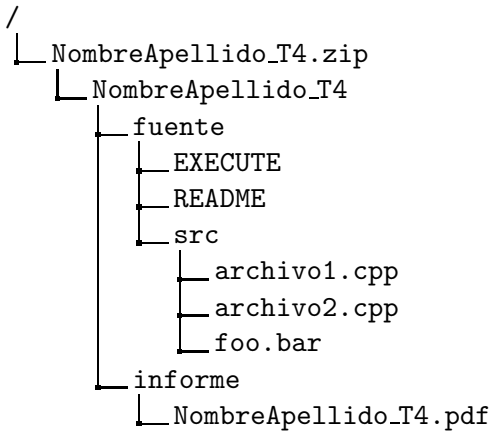

Figura 1: Estructura de archivo de entrega.

- Para la tarea tienen que entregar un informe de resultados junto al código fuente (el programa que implementa la tarea). No se aceptan códigos sin informe ni informes sin código.
- Si la tarea no compila tiene nota 1.0 tanto en el código como en el informe.
- Cualquier falta a las restricciones anteriores invalida irrevocablemente la tarea.

4. Ponderación

El informe equivale a un 30 % de la nota de tarea.

El código equivale a un 70 % de la nota de tarea.

5. Entrega de la tarea

La entrega de la tarea debe incluir (1) el código del programa que resuelve la tarea y (2) un informe escrito donde se describa y documente el programa entregado, y se incluyan ejemplos de entradas y salidas. La entrega de la tarea se realizará a través de la plataforma `lms.educandus.cl` hasta las 23:55 del día del plazo final. Generen un único archivo que contenga tanto el informe como los códigos para compilar. También se solicita que se adjunte un archivo README que explique cómo compilar la tarea y un archivo EXECUTE el cual contiene los comandos a usar para compilación y ejecución de la misma, este debe tener permisos de ejecucion a modo de poder ser utilizado. Adicionalmente, *debe de entregarse una copia impresa del informe* a subir en la plataforma `lms.educandus.cl` a más tardar el día siguiente del plazo final. La estructura del archivo a entregar debe tener el formato ejemplificado en la Figura 1.

El contenido del informe debe seguir la siguiente pauta común para todas las tareas:

- Portada. El formato de la portada se muestra en la Figura 2.

Universidad de Talca
Facultad de Ingeniería
Departamento de Ciencias de la Computación

Informe Tarea X

Nombre de la Tarea X

Fecha: <fecha entrega>
Autor: <nombre alumno>
e-mail: <correo del alumno>

Figura 2: Formato para la portada.

- Introducción: Breve descripción del problema y de su solución.
- Análisis del problema: Se expone en detalle el problema, los supuestos que se van a ocupar, las situaciones de borde y la metodología para abordar el problema.
- Solución del problema:
  - Algoritmo de solución: Se detallan los pasos a seguir para solucionar el problema. De ser necesario, se recomienda incluir figuras, tablas, etc., que permitan mejorar la comprensión de la solución propuesta.
  - Diagrama de Estados: Muestra en forma global el programa.
  - Diseño: Explicitar las Pre y Post condiciones consideradas, mostrar los invariantes empleados.
  - Implementación: Se debe mostrar el pseudo-código del programa que soluciona el problema, explicando lo que hace. Omita cualquier detalle de implementación que sea irrelevante para entender la solución del problema. Se recomienda usar nombres representativos para las variables. NOTA: El código generado para resolver la tarea debe corresponder al diseño descrito, preocúpese de comentar el código donde sea necesario para facilitar su lectura.
  - Modo de uso: Se debe explicar el modo de uso del programa y el modo de compilación. Nota la tarea debe ser compilable en los computadores de la Universidad.
- Pruebas: Se debe mostrar las pruebas realizadas y sus resultados. El número de pruebas puede variar dependiendo del problema En esta sección debe incluir las tablas y gráficos necesarios solicitados, y el análisis de los resultados. En este capítulo adjunten las conclusiones obtenidas de los resultados de la tarea.

- Anexos: De ser necesario, cualquier información adicional se debe agregar en los anexos y debe ser referenciada en alguna sección del informe de la tarea. Dentro de los anexos se puede incluir un listado con el programa completo que efectivamente fue compilado.

En este y los siguientes informes se va a valorar la ortografía y la redacción de acuerdo a las siguientes reglas de penalización.

Ortografía Se permiten hasta 10 errores ortográficos en todo el informe. Entre 11 y 20 errores descuenten 0.5 puntos al informe. Entre 21 y 30 descuentan 1.0 puntos al informe. Más de 30 descuenten 2.0 puntos al informe.

Redacción Se permiten hasta 5 ideas o párrafos mal redactados (es decir que no se entiendan). Entre 6 y 10 párrafos con problemas de redacción descuenten 0.5 puntos al informe. Entre 11 y 14 párrafos descuentan 1.0 puntos al informe. Más de 24 parrafos incomprensibles descuenten 2.0 puntos al informe.