Tarea 5: Comparación de Árboles Balanceados

Profesores: Nelson Baloian
Patricio Poblete
Auxiliares: Manuel Cáceres
Sebastián Ferrada
Sergio Peñafiel

Fecha de entrega: 6 de junio, 23:59 hrs

1. Introducción

Un árbol de búsqueda binaria (ABB) es una estructura de datos para guardar llaves y que permite realizar búsquedas eficientes sobre ellas. Su complejidad promedio en tiempos de búsqueda e inserción es $O(\log n)$. Sin embargo, es sabido que el ABB se puede degenerar a una lista enlazada si los elementos son insertados ordenadamente, lo que nos lleva a que el peor caso de búsqueda e insersión en ABB es $O(n)$.

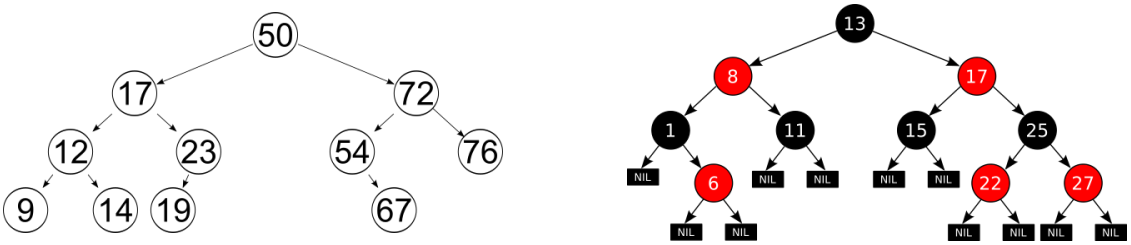
Para enfrentar este problema, existen otro tipo de ABBs llamados *balanceados* ya que se preocupan de que la cantidad de nodos en los subárboles izquierdo y derecho de cada nodo sea lo más similar posible, poniendo así un límite en la altura final del árbol y asegurando que el tiempo de búsqueda sea $O(\log n)$.

En esta tarea se considerarán dos tipos de árboles balanceados:

- **AVL** un AVL es un ABB tal que cada vez que un nodo es insertado (según el algoritmo usual de un ABB), se revisa que la condición $|\,\text{altura}\,(n.izq) - \text{altura}\,(n.der)| <= 1$ para todos los nodos n del árbol. Si la condición es violada por algún nodo, se restaura utilizando rotaciones simples y dobles.
- **Rojo-negro** un árbol rojo-negro es un ABB en el cual sus nodos tienen un bit extra de información, llamado el *color*. Cualquier nodo insertado en el árbol es de color rojo. Tras la inserción, se revisa que las siguientes condiciones sean cumplidas:
  - La raíz del árbol es negra
  - Todas las hojas (punteros `null`) son negros
  - Si un nodo es rojo, sus dos hijos deben ser negros
  - Dado un nodo, todos los caminos desde él hasta las hojas tienen la misma cantidad de nodos negros

  En caso de que alguna falle, se reparan vía rotaciones o cambios de color estratégicos.

En la figura 1 se pueden ver ejemplos de ambos tipos de árboles



(a) AVL

(b) Rojo-negro

Figura 1: Ejemplos de ABB

2. Implementación

Su trabajo consistirá en implementar tres tipos de árboles: ABB normal, AVL y árbol rojo-negro. Para lo cual, se espera que entregue tres clases (una para cada árbol) que implementen la siguiente interfaz siguiendo las reglas mencionadas en la sección previa:

```
public interface ArbolBinario{
  public void insertar(int x);
  public boolean buscar(int x);
}
```

3. Experimentación

Una vez implementados los árboles, se le pide que los compare en diversos escenarios. Los escenarios son tuplas (n, d) donde $n \in \{1000, 2000, ..., 10000\}$ es la cantidad de nodos a insertar y $d \in \{AZAR, ASC\}$ la distribución de las llaves a insertar (al azar, números ordenados ascendentemente). Para cada escenario, debe insertar n números enteros que sigan la distribución d en cada uno de los tres árboles.

Para realizar la comparación debe considerar tres indicadores:

- Altura final del árbol, medida al terminar con las inserciones.
- Tiempo promedio de inserción. Para esto, debe medir el tiempo que le toma en realizar las n inserciones y dividir ese tiempo por n.
- Tiempo promedio de búsqueda. Al terminar con las inserciones, busque al menos 1000 llaves al azar y mida el tiempo total. El tiempo promedio es entonces el tiempo total dividido por la cantidad de búsquedas realizadas.

Recuerde que si está midiendo el tiempo de búsqueda, no puede realizar otras operaciones (como medir alturas) simultáneamente, pues sus resultados estarán alterados.

Con los indicadores para los diferentes escenarios, realice gráficos que le ayuden a comparar la *performance* de los distintos árboles. Pregúntese cuál es el peor caso para cada árbol (de existir alguno). ¿Cuál es la complejidad de la búsqueda? ¿Cuál árbol tiene mejor tiempo promedio? ¿Se ajustan a la cota $O(\log n)$?

4. Condiciones de Entrega

- La tarea debe programarse en Java.
- Es obligatorio la entrega de un informe en formato pdf junto con su tarea (Ver siguiente sección). Informes en otros formatos **NO** serán considerados.
- Esta tarea es de carácter individual, cualquier caso de copia se evaluará con la nota mínima.
- No olvide subir a U-cursos todos los archivos necesarios para que su tarea funcione correctamente.
- Debe subir los archivos de código fuente (*.java). Los archivos compilados (*.class) no serán evaluados.
- Cualquier duda respecto a la tarea puede ser consultada usando el foro del curso.
- **NO** se aceptarán atrasos.

5. Informe

El informe debe describir el trabajo realizado, la solución implementada, los resultados obtenidos y las conclusiones o interpretaciones de estos. Principalmente debe ser breve, describiendo cada uno de los puntos que a continuación se indican:

- **Portada:** Indicando número de la tarea, fecha, autor, email, código del curso, etc.
- **Introducción:** Descripción breve del problema y su solución.
- **Análisis del Problema:** Exponga en detalle el problema, los supuestos que pretende ocupar, casos de borde y brevemente la metodología usada para resolverlo. Indique también sus hipótesis preliminares.
- **Solución del Problema:** Indique claramente los pasos que siguió para llegar a la solución del problema. Muestre mediante figuras y ejemplos qué es lo que realiza su código. Evite copiar todo el código fuente en el informe, sin embargo, puede mostrar las partes relevantes de éste.

- **Resultados y Conclusiones:** Muestre los gráficos realizados para cada indicador (i.e., ver cómo varía el indicador versus n) y utilícelos para ilustrar sus conclusiones. Sea detallado en explicar qué partes indican tendencias en complejidad. Sea riguroso en las visualizaciones: verifique que los datos representados se ajusten a los datos obtenidos y no a lo que ud. espera mostrar.