Ing. de Sistemas e Informática
Estructuras de Datos y Algoritmos
Jorge Mario Londoño P.

Ejercicios de Repaso:

Búsqueda, Tablas de Símbolos

Búsqueda binaria – Arreglos ordenados

. El arreglo ordenado para la tabla de símbolos no permite llaves duplicadas. Analizar desde el punto de vista de implementar las operaciones `get/put` manteniendo las llaves ordenadas si es posible operar la tabla con llaves repetidas. Como seria la implementación en este caso? Cuál sería la eficiencia de `get/put`? (Sugerencia: permitir que las búsquedas retornen "el conjunto de valores" asociados a las llaves repetidas).

. Una operación interesante en tablas de símbolos ordenadas es encontrar el sucesor y el predecesor de una llave. Dar una implementación de este método para la tabla de símbolos basada en arreglo ordenado y estimar su eficiencia.

Árboles binarios de búsqueda (BST)

1. Un árbol binario de búsqueda contiene las llaves 1..10 y se realiza la búsqueda por la llave 5. Cuál de las siguientes secuencias de comparación de llaves no es válida?

a) 10, 9, 8, 7, 6, 5
b) 4, 10, 8, 6, 5
c) 1, 10, 2, 9, 3, 8, 4, 7, 6, 5
d) 2, 7, 3, 8, 4, 5
e) 1, 2, 10, 4, 8, 5

2. Dibujar todos los posibles árboles binarios que pueden resultar de añadir las llaves 3, 4, 5, 6, 7 en un árbol binario inicialmente vacío. Las llaves se pueden insertar en cualquier orden.

3. Los algoritmos get/put para árboles binarios vistos en clase son recursivos. Por eficiencia (reducir invocaciones a métodos y consumo de memoria de pila) se suelen preferir las implementaciones no recursivas. Dar una implementación no recursiva de estos dos métodos. (Sugerencia: Utilizar una estructura auxiliar, e.g. Stack).

4. Una operación interesante en tablas de símbolos ordenadas es encontrar el sucesor y el predecesor de una llave. Explorar dos posibles soluciones para implementar estas operaciones en el BST:

a) Utilizando las operaciones `select/rank`.

b) Agregando a los nodos del árbol una referencia hacia el nodo padre.

Analizar ambas soluciones, considerando su eficiencia en número de comparaciones de llaves.

5. Como encontrar la mediana de una colección de llaves en un BST? Cuánto tiempo se requiere? Comparar con soluciones al mismo problema en arreglos, cuando se encuentran ordenados, o cuando no están en orden.

6. La implementación del texto guía de la operación de búsqueda por rangos (método keys) utiliza una estructura Queue adicional, lo que hace que el proceso no sea *in-situ*. La operación ***yield*** es una construcción que se encuentra disponible en lenguajes de programación modernos como alternativa para la implementación de iteradores. Explorar la implementación del método keys utilizando *`yield`* y determinar si con esta implementación se logra una solución *in-situ*.

Árboles de búsqueda balanceados

1. Indicar el árbol resultante de añadir las llaves Y L P M X H C R A E S en un árbol 2-3 inicialmente vacío.

2. Se elimina la llave X del árbol del númeral anterior. Iniciar el árbol 2-3 resultante.

3. Cuál es el árbol rojo-negro correspondiente al árbol 2-3 obtenido en el numeral anterior?

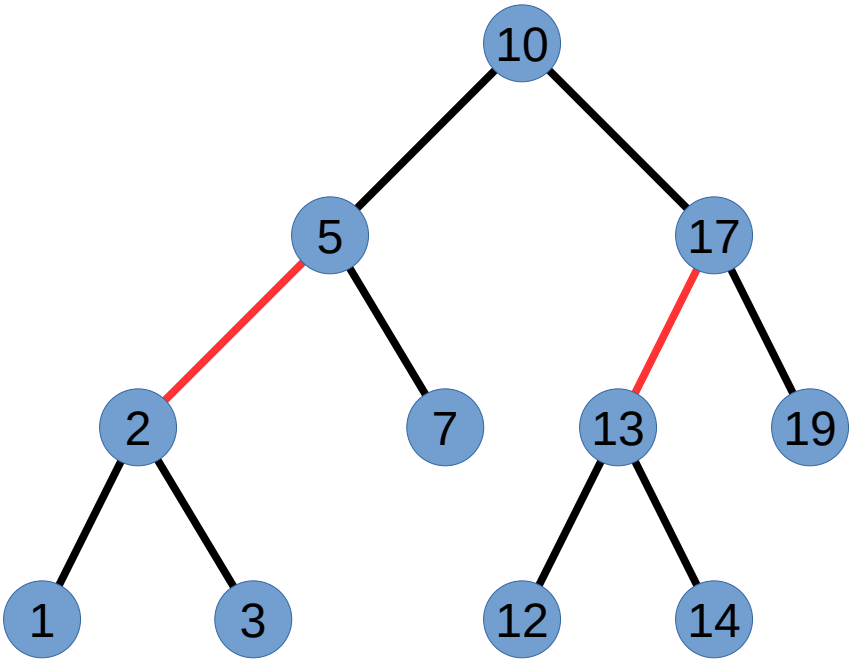
4. Considerando el mismo conjunto de llaves del numeral 1, cual sería el árbol 2-3 de máxima altura válido para estas llaves? El de mínima altura?

Árboles de búsqueda rojo-negros

1. Indicar el árbol resultante de añadir las llaves Y L P M X H C R A E S en un árbol 2-3 inicialmente vacío.

2. Cuál es el árbol rojo-negro correspondiente al árbol 2-3 obtenido en el numeral anterior?

3. Se tiene el siguiente árbol rojo-negro inicial:



a) Indicar el resultado de insertar la llave 6.

b) Indicar el resultado de insertar la llave 11.

c) Indicar el resultado de insertar la llave 20.

d) Cuál es la secuencia de comparaciones para buscar la llave 11.

e) Cuáles son las longitudes del camino negro más largo y del más corto después de realizar esta secuencia de inserciones?

f) Indicar la secuencia de operaciones realizas por la operación rank(14).

g) Indicar las comparaciones y el resultado obtenido por la operación select(5).

Tablas asociativas

1. En clase consideramos la implementación de la tabla de dispersión enlazada haciendo uso de un arreglo de Bags. En ese momento solo se consideraron las operaciones `get` y `put`.

(a) Proponer una implementación “perezosa” de la operación `delete`. (Marca elementos de la lista como borrados, pero no hace el borrado físico). Se debe así mismo modificar la operación `put` para que pueda reutilizar los nodos “borrados” de las listas.

(b) Proponer una implementación activa de la operación `delete`. Estimar su eficiencia en función del número de comparaciones realizadas.

2. Operaciones de conjuntos tales como la intersección y la diferencia normalmente requieren ciclos doblemente anidados que llevan a implementaciones cuadráticas.

(a) Explicar como podrían implementarse estas operaciones en tiempo lineal haciendo uso de tablas asociativas.

(b) Dar una implementación (pseudo-código) de la intersección y la diferencia haciendo uso de la idea propuesta en el numeral (a).

3. En clase se discutió la necesidad de hacer una “redispesión” de la tabla cuando el factor de carga se hace superior a la unidad. Proponer una implementación del algoritmo redispersion como un método de la clase SeparateChainingHashST que tome como argumento el nuevo valor M del tamaño del arreglo.

```
private void redispersion(int m)
```