Binary Search Trees

Emmanuel Ovalle Magallanes

Centro de Investigación en Matemáticas A.C
Programación Avanzada II

emmanuel.ovalle@cimat.mx

4 de marzo de 2015

Árbol binario de Búsqueda (BST)

1. Un **árbol binario** esta compuesto de **nodos**, donde cada nodo contiene: un apuntador a su **derecha**, un apuntador a su **izquierda** y un **dato**.
2. El nodo superior en el árbol se conoce como **raíz**.
3. El lado derecho e izquierdo contienen **subarboles** respectivamente.

Árbol binario de Búsqueda (BST)

1. Un **árbol binario** esta compuesto de **nodos**, donde cada nodo contiene: un apuntador a su **derecha**, un apuntador a su **izquierda** y un **dato**.
2. El nodo superior en el árbol se conoce como **raíz**.
3. El lado derecho e izquierdo contienen **subarboles** respectivamente.

Definición recursiva

Un árbol binario es vacío, o consiste de un solo nodo, donde los apuntadores izquierdo y derecho apuntan a un árbol binario.

Árbol binario de Búsqueda (BST)

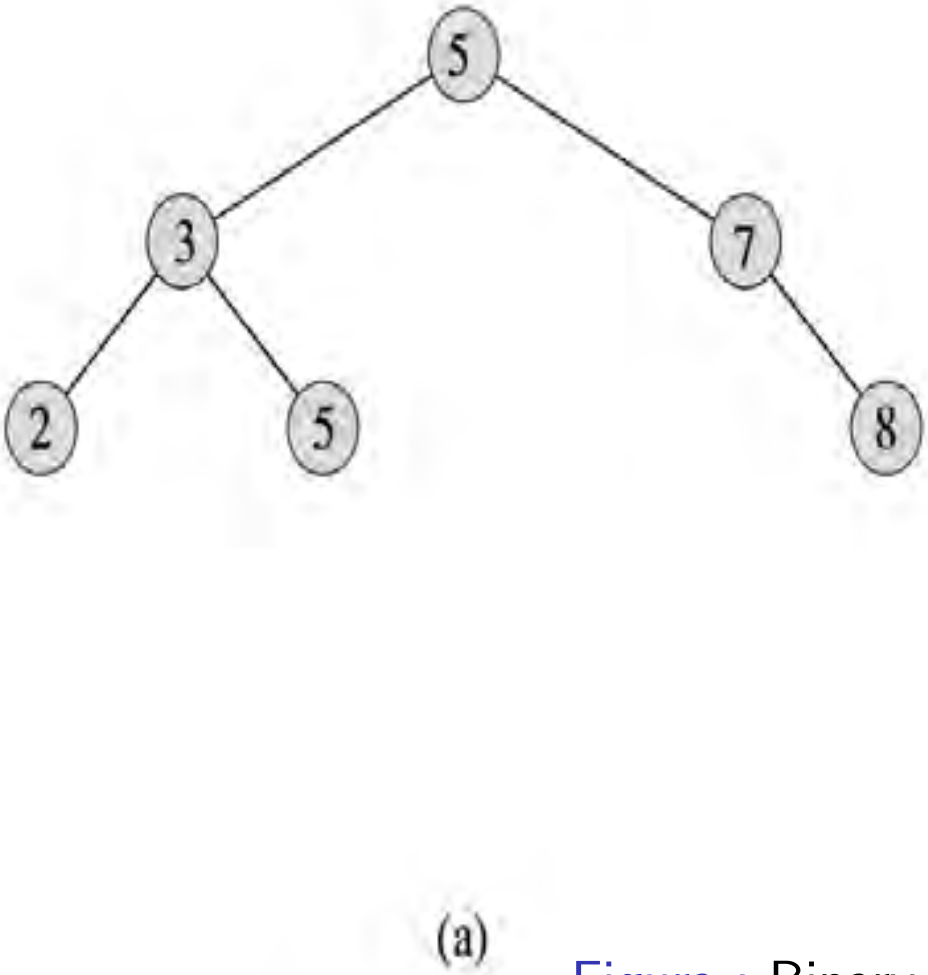
1. Un **árbol binario** esta compuesto de **nodos**, donde cada nodo contiene: un apuntador a su **derecha**, un apuntador a su **izquierda** y un **dato**.
2. El nodo superior en el árbol se conoce como **raíz**.
3. El lado derecho e izquierdo contienen **subarboles** respectivamente.

Definición recursiva

Un árbol binario es vacío, o consiste de un solo nodo, donde los apuntadores izquierdo y derecho apuntan a un árbol binario.

Un **árbol binario de búsqueda (BST)** es un tipo de árbol binario donde los nodos están ordenados: para cada nodo, todos los elementos en el subarbol izquierdo son **menor o igual** al nodo y los elementos en el subarbol derecho son **mayor** al nodo.

Árbol binario de Búsqueda (BST)



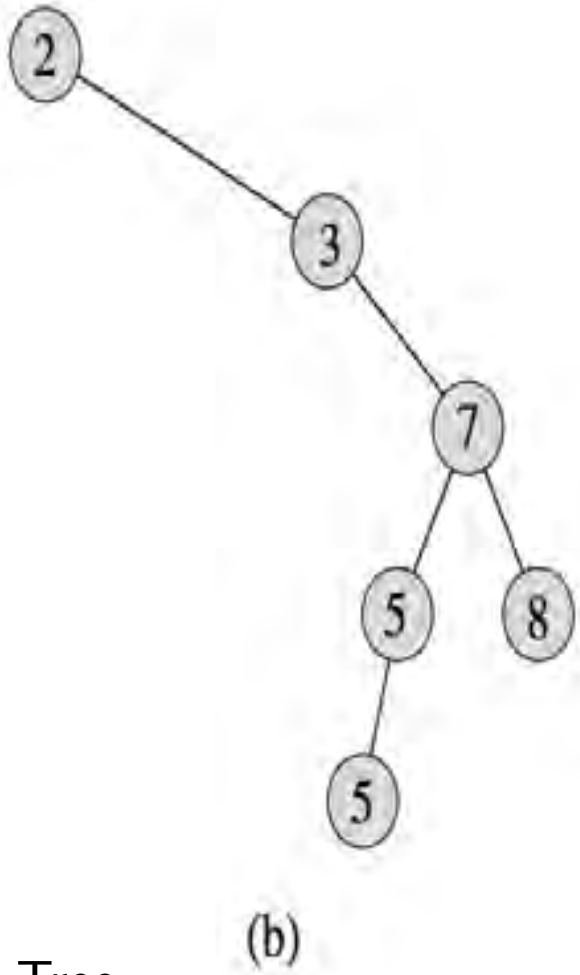


Figura : Binary Search Tree

Operaciones típicas

1. SEARCH
2. INSERT
3. DELETE
4. MINIMUM
5. MAXIMUM
6. PREDECESSOR
7. SUCCESSOR

Implementación con la STL

Es posible implementar un árbol con los contenedores: **set<T>**, **map<T1,T2>**, **multiset<T>**, **multimap<T1,T2>**.

Implementación con la STL

Es posible implementar un árbol con los contenedores: **set<T>, map<T1,T2>, multiset<T>, multimap<T1,T2>**.

set<T>

- bool empty() → Regresa true ssi. no hay elementos en el set.
- int size() → Regresa el número de elementos en el set.
- pair<iterator, bool> insert(T aValue) → Inserta un aValue en el set.
- void erase(T aValue) → Elimina un aValue del set.
- void clear() → Elimina todos los elementos del set.
- iterator find(T aValue) → Regresa un iterator a aValue (o end(), si aValue no existe en el set).
- int count(T aValue) → Regresa el número de ocurrencias de un aValue en el set.
- iterator begin() → Regresa un iteratoral primer elemento del set.

Implementación con la STL

Algunas de las operaciones de <**map**> son:

<map>

- bool empty()
- int size()
- T2 operator[](T1 aValue) → Regresa el valor de T2 asociado con aValue.
- int erase(T1 aValue) → Elimina la entrada para aValue
- void clear()
- iterator find(T1 aValue)→ Regresa un iterator al par <T1, T2> que contiene al aValue (Regresa end() si no encuentra el par).
- int count(T1 aValue) → Regresa el numero de valores cuyo valor T1 es aValue.
- iterator begin()
- iterator end()

Ejemplos

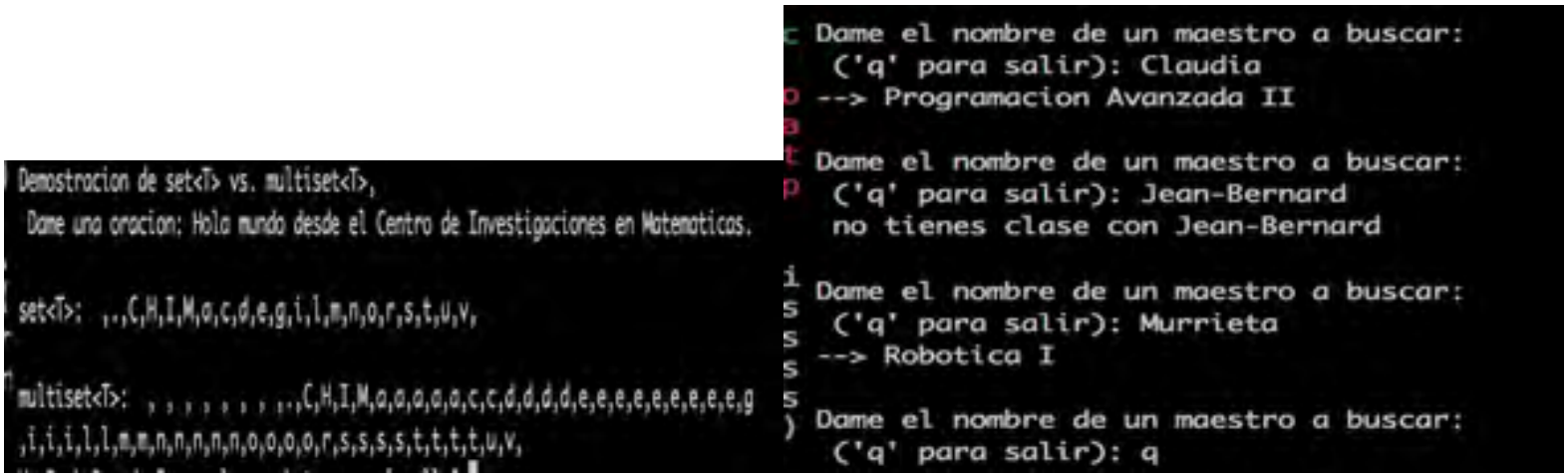


Figura : Ejemplo de set vs multiset, y de map

Referencias

Cormen, Thomas H. and Stein, Clifford and Rivest, Ronald L. and Leiserson, Charles E.(2001),

Introduction to Algorithms.

http://www.cplusplus.com/reference/set/set/.

http://cs.calvin.edu/books/c++/intro/3e/WebItems/Ch15-Web/STL-Trees.pdf