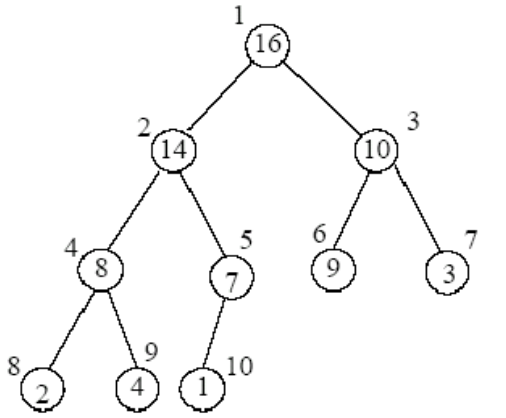
GIT Department of Computer Engineering
CSE 222/505
Spring 2013
Homework 07

Binary Search Trees
Due date: May 6th 2013 23:59

In this homework, you will implement the binary search trees using an array representation. As we saw in the lectures, any binary tree can be represented using an array. For example,



1	2	3	4	5	6	7	8	9	10
16	14	10	8	7	9	3	2	4	1

Implement the attached binary tree interface using arrays as your internal representation.

Your new class should do everything that a binary search tree can do at the same asymptotic complexity.

- Follow the rules for good software engineering steps(analysis, design, UML diagrams etc.)
- Submit your test cases for each interface method. Test each methods several times with different data.

```
/** Interface to define a search tree
 *  @author Koffman and Wolfgang
 * */

public interface SearchTree < E
    extends Comparable < E >> {
  /** Inserts item where it belongs in the tree.
      @param item The item to be inserted
      @return true If the item is inserted, false if the
      item was already in the tree.
   */
  boolean add(E item);

  /** Determine if an item is in the tree
      @param target Item being sought in tree
      @return true If the item is in the tree, false otherwise
   */
  boolean contains(E target);

  /** Find an object in the tree
      @param target The item being sought
      @return A reference to the object in the tree that compares
      equal as determined by compareTo to the target. If not found
      null is returned.
   */
  E find(E target);

  /** Removes target from tree.
      @param target Item to be removed
      @return A reference to the object in the tree that compares
      equal as determined by compareTo to the target. If not found
      null is returned.
      @post target is not in the tree
   */
  E delete(E target);

  /** Removes target from tree.
      @param target Item to be removed
      @return true if the object was in the tree, false otherwise
      @post target is not in the tree
   */
  boolean remove(E target);

  /** Returns the maximum depth of the tree.
      @return maximum depth of the tree.
   */
  int maxDepth();

}
```