

Homework # 6

due Tuesday, March 8, 3:30PM

1 Binary Search Trees for Sets

Do Exercise 11, and 12 (on page 179) with several changes (see footnote marks):

11. A binary search tree is a binary tree with special properties. It may be `Empty`. It may be a `Node` containing a left subtree, a data item x , and a right subtree. In this case, all the data items in the left subtree are smaller than x and all the elements in the right subtree are larger than x , and the left and right subtrees are also binary search trees. Write a function `makeBST` of type `('a * 'a -> bool) -> 'a list -> 'a tree` that *efficiently*¹ organizes the items in the list into a binary search tree. The tree *must* be balanced.¹ You may *not* assume that no item in the list is repeated.¹ Duplicates must be discarded.¹
12. Write a function `searchBST` of type `('a * 'a -> bool) -> 'a tree -> 'a -> bool` that searches for a given data element. You should not search every node in the tree, but only those nodes that, according to the definition, might contain the element you are looking for. The `searchBST` function must *not* require an equality type.¹ Instead it should use the comparison function, and assume that $\neg(x < y) \wedge \neg(y < x)$ means that x and y are equal.¹

Then use these functions to create “sets”:

```
type 'a set = ...;
fun makeSET (comp:'a * 'a -> bool) (l:'a list) : 'a set = ...
fun SETcontains (s: 'a set) (x: 'a) : bool = ...
```

Your `set` type will need to include a function type. One should be able to write:

```
val smallprimes = makeSET (op <) [2,2,3,5,7,11,13,17,19];
if SETcontains smallprimes 9 then
  "Oh No!"
else
  "Good.";
```

Hint: The two set functions should be one-liners.

Put all your tree and set definitions in `set.sml`.

2 Activation Records

Do Exercises 2, 3 and 7 in Chapter 12 (page 205).

¹Changed from the textbook!