Search Trees

→ use a search tree structure to efficiently implement a sorted map
→ three most fundamental methods of a map M:
1.) M[k]: return the value associated with key k in map M
2.) M[k] = v: associate value v with key k in map M
3.) del M[k]: remove from M the item with key = k

Binary trees are an excellent data struct for storing items in a map, assuming the keys have an order relation
→ binary search tree where each position p storing a key-value pair (k,v) such that
→ keys stored in left subtree of p are less than k
→ keys stored in right subtree of p are greater than k

Navigation

→ Proposition: an inorder traversal of a binary search tree visits positions in increasing order of their keys
→ with a BST, we can support:
- first(): returns position containing least key
- last(): returns position containing greatest key
- before(p): returns position containing greatest key less than position p
- after(p): returns position containing least key that is greater than that at position p

→ Algorithm after(p):

```
if right(p) is not None                    // successor in leftmost position in p's right subtree
    walk = right(p)
    while left(walk) is not None
        walk = left(walk)
else                                       // successor is nearest ancestor having p in left subtree
    walk = p
    ancestor = parent(walk)
    while ancestor is not None and walk == right(ancestor)
        walk = ancestor
        ancestor = parent(walk)
return ancestor
```