Worksheet 32

Group 23 Members

Alexander Miranda
Benjamin Tate
Shuiqiang Lin
Amy Weller
Anthony Tong

Assuming the array has n elements, what is the algorithmic execution time for step 1?

O(n log n)

What is the algorithmic execution time for step 2?

O(n)

Recall that insertion sort and quick sort are examples of algorithms that can have very different execution times depending upon the distribution of input values. Is the execution time for this algorithm dependent in any way on the input distribution?

No. As long as the tree is properly self-balancing, the worst case execution time will be O(n log n).

A sorted dynamic array bag also maintained elements in order. Why would this algorithm not work with that data structure? What would be the resulting algorithmic execution time if you tried to do this?

A dynamic array bag would be worse for this, because adding has a far slower execution time than with an AVL tree. In the worst case, the execution time of the algorithm with a sorted dynamic array bag would be O(n^2).

Can you think of any disadvantages of this algorithm?

This algorithm requires more work to implement, and may take up more memory.

You can use an indexing loop, or an iterator loop. Which is easier for step 1?

Indexing loop because the array has the index built in.

Which is easier for step 2?

Iterator loop because that would be a more effective way to traverse the AVL tree when copying out the values back into an array.

```
void treeSort (TYPE *data, int n) { /* sort values in array data */
        AVLtree tree;
        AVLtreeInit (&tree);

        for (int i = 0; i < n; i++) {
                avlTreeInsert(tree, data[i]);
        }

        AVLiter *itr = avlIterInit(tree);
        Int j = 0;
        while (hasNext(itr)) {
                data[j] = itr->value;
                iterNext(itr);
                j++;
        }
        AVLiterDelete(itr);
        AVLtreeCleanup(tree);
}
```