Exercise 2.3-6

Jack Maney

August 27, 2013

The code integrating binary search within insertion sort can be found in the class *InsertionSortWithBinarySearch*.

Let's look at a worst-case running time. Note that the method

```
int getInsertionIndex(AbstractList<T> list,T key,int begin,int end)
```

returns the index–between *begin* and *end*–of *list* at which *key* should be inserted (or -1 if no such index exists). By an argument identical to that given in Exercise 2.3-5, the worst running time of *getInsertionIndex* is $\Theta(lg(n))$.

Here is the code for the *sort* method within *InsertionSortWithBinarySearch*:

```
public static <T extends Comparable<T>> void sort(AbstractList<T> list){

        for(int j = 1; j < list.size(); j++){

                T key = list.get(j);

                int insertionIndex = getInsertionIndex(list,key,0,j-1);

                /*
                 * insertionIndex == -1 precisely when we need to keep
                 * the jth element right where it is (ie it should already be
                 * at the end of the first j+1 elements).
                 */
                if(insertionIndex >= 0){

                        for(int i = j-1; i >= insertionIndex; i--){
                                list.set(i+1,list.get(i));
                        }

                        list.set(insertionIndex, key);
                }
        }
}
```

cost	times
c_3	n
c_5	$n-1$
$lg(n)$	$n-1$
c_{14}	$n-1$
c_{16}	$1+2+\cdots+n-1$
c_{17}	$n-1$
c_{20}	$n-1$

However, since

$$1+2+\cdots+n-1=\frac{n(n-1)}{2}=\Theta(n^2),$$

we see that this algorithm still has a running time of $\Theta(n^2)$. While the binary search helped find indices at which to insert, it had no effect on moving elements over prior to insertion.

As evidence, here are two benchmark results of worst-case sorting, the first with a list of 10,000 randomly chosen integers in reverse-sorted order:

```
|= Benchmark =======================================================================================|
| -                      | unit | sum     | min   | max    | avg   | stddev | conf95        | runs   |
|============================================= TimeMeter ===========================================|
|. InsertionSortWithBinarySearchBenchmark ..........................................................|
| binarySearchBenchmark | ms   | 7003.18 | 63.87 | 166.63 | 70.03 | 11.92  | [65.76-77.67] | 100.00 |
|_ Summary for InsertionSortWithBinarySearchBenchmark ______________________________________________|
|                        | ms   | 7003.18 | 63.87 | 166.63 | 70.03 | 11.92  | [65.76-77.67] | 100.00 |
|---------------------------------------------------------------------------------------------------|
|================================== Summary for the whole benchmark ================================|
|                        | ms   | 7003.18 | 63.87 | 166.63 | 70.03 | 11.92  | [65.76-77.67] | 100.00 |
|============================================ Exceptions ===========================================|
|===================================================================================================|
```

And here are the benchmarks for 100 worst case iterations of lists of length 20,000:

```
|= Benchmark ============================================================================================|
| -                      | unit | sum      | min    | max    | avg    | stddev | conf95          | runs   |
|=============================================== TimeMeter ==============================================|
|. InsertionSortWithBinarySearchBenchmark ...............................................................|
| binarySearchBenchmark | ms   | 33720.37 | 315.45 | 573.42 | 337.20 | 40.96  | [317.61-397.23] | 100.00 |
|_ Summary for InsertionSortWithBinarySearchBenchmark ___________________________________________________|
|                        | ms   | 33720.37 | 315.45 | 573.42 | 337.20 | 40.96  | [317.61-397.23] | 100.00 |
|--------------------------------------------------------------------------------------------------------|
|==================================== Summary for the whole benchmark ===================================|
|                        | ms   | 33720.37 | 315.45 | 573.42 | 337.20 | 40.96  | [317.61-397.23] | 100.00 |
|=============================================== Exceptions =============================================|
|========================================================================================================|
```