Exercise 2.3-7

Jack Maney

March 20, 2014

The code integrating binary search within insertion sort can be found in the class *SumSearch*. Again, we consider a worst-case running time–namely, that there are no two integers in the set S whose sum is the target, x.

Again, we look at the *search* method, line by line, and compute the costs, keeping in mind that the Binary Search algorithm has a worst-case running time of $\Theta(lg(n))$ and that Merge Sort has a worst-case running time of $\Theta(nlg(n))$.

```java
public static Set<Integer> search(Set<Integer> set,int x){

                if(set.isEmpty()){
                        throw new IllegalArgumentException();
                }

                ArrayList<Integer> list = new ArrayList<>();
                list.addAll(set);
                Set<Integer> result = null;

                MergeSort.sort(list);

                for(int i = 0; i < list.size(); i++){

                        int y = list.get(i);
                        int index = BinarySearch.search(list, new Integer(x - y));

                        if(index >= 0 && index != i){
                                result = new HashSet<Integer>();
                                result.add(y);
                                result.add(list.get(index));
                                break;
                        }

                }

                return result;
        }
```

cost	times
c_3	1
c_7	1
n (line 8)	1
c_9	1
$nlg(n)$ (mergesort)	1
c_{15}	n
$lg(n)$ (binary search)	n
c_{18}, c_{19}, c_{20} and c_{21}	n

It follows that the worst-case running time is $\Theta(nlg(n))$.