Problem Set 9

Justin Ely

615.202.81.FA15 Data Structures

03 November, 2015

1)

To show that the number of comparisons needed to find the largest and second-largest values is $n + log_2(n) - 2$, I implemented a simple algorithm to count the number of comparisons done. The implementation I came up with, however, is very often around $n+log_2(n)-2$, but often above or below this value. However, getting a value below the specified requirement seems to run counter to the problem specification.

```
def find_largest(vals):
    first = vals[0]
    second = vals[1]

    count = 1
    if second > first:
        first, second = second, first

    for value in vals[2:]:
        if value > second:
            count += 1
            if value > first:
                second = first
                first = value
                count += 1
            else:
                second = value
                count += 1
        else:
            count += 1

    return (second, first)
```

2)

The number of comparisons necessary to find the largest and smallest values in a set of n distinct elements varies largely depending on the initial sorting of the data as well as the sorting method employed.

With sorted data, no comparisons are necessary and the largest and smallest values can be retrieved trivially without sorting.

For bubble sort, the number of comparisons needed for sorted data will be n-1 for the first and only pass through, at which point the largest and smallest can be accessed. For randomized data though, bubble sort will take $\sim n^2$ comparisons. An insertion sort is similar, where sorted data will need only a single pass with n-1 comparisons and shuffled data will need $\sim n^2$.

Quicksort, for random data, takes $\sim nlog_2n$ comparisons, fewer than bubble or insertion sort, but can make $\sim n^2$ comparisons for already sorted data and a naive algorithm implemenation.