Homework 1

Due 09/01/15

August 25, 2015

1. Prove that the following algorithm sorts its input data; i.e., that $data[1] \leq data[2] \leq \ldots \leq data[n]$ when the algorithm terminates. You may assume that *data* contains at least one element. Also, $\lfloor x \rfloor$ represents the *floor* function, which returns the largest integer less than or equal to the given value (e.g., $\lfloor 3.1415 \rfloor = 3$).

```
Input: data: an array of integers
Input: n: the length of data
Output: a reordering of data in (ascending) sorted order
Algorithm: ThirdSort
if n = 1 then
    return data
else if n = 2 then
    if data[1] > data[2] then
        Swap data[1] and data[2]
    end
    return data
else
    third = ⌊n/3⌋
    Call ThirdSort on data[1..n-third]
    Call ThirdSort on data[third+1..n]
    Call ThirdSort on data[1..n-third]
    return data
end
```

Hint: use strong induction on the length of *data*. You may find it useful to assign names (like A, B, and C) to the three "thirds" of the *data* array given by $data[1..third]$, $data[third+1..n\text{-}third]$, and $data[n\text{-}third+1..n]$. You may also find it helpful to simulate the algorithm on some small inputs to understand what it is doing.

Answer:

Proof. We prove the claim by induction on n, the size of *data*.

(*Base cases*) When $n = 1$, *data* must be sorted, as all arrays of size 1 are sorted. When $n = 2$, $data[1]$ and $data[2]$ are swapped if they are out-of-order, so $data[1] \leq data[2]$ when the algorithm terminates.

(*Inductive step*) Suppose that ThirdSort correctly sorts all arrays of size 1 up to k, and suppose that *data* is an array of size $k+1$. As we have already handled the cases $n = 1$ and $n = 2$, we assume that $k \geq 2$, so $n \geq 3$. As a result, ThirdSort will enter the **else** case in line 9, and $third \geq 1$.

Note that this case splits *data* into three subarrays, $data[1..third]$, $data[third{+}1..n\text{-}third]$, and $data[n\text{-}third{+}1..n]$, which we will refer to as A, B, and C, respectively. We also refer to the arrays in lines 11 and 12 of ThirdSort as AB and BC, respectively, as they are combinations of A, B, and C. Note that subarrays A and C have *third* elements each, while B has $(n - third) - third = n - 2(third)$ elements (provided that this quantity is positive). Since $third = \lfloor n/3 \rfloor$, $n - 2(third)$ is greater than or equal to *third*, so we say that $|B| \geq third = |A| = |C|$. (Since B has positive length, it is a valid subarray of *data*.)

Since the subarrays AB and BC have $n - third$ elements each and $third \geq 1$, ThirdSort must correctly sort them in lines 11–13 by the inductive hypothesis. In particular, after line 11, it must be true that every element in A is less than or equal to every element of B. After line 12, every element of C must be greater than or equal to every element of B. This sort may have moved some small elements from C to B, so the elements of B are not necessarily larger than the elements of A afterwards; however, BC contained at least $|B|$ elements larger than everything in A, and since $|B| \geq |C|$, everything in C must be larger than everything in A (as well as B). Thus, after AB is sorted in line 13, everything in A will be less than or equal to everything in B, and everything in B will be less than or equal to everything in C, so the entire *data* array will be sorted. $\square$