Homework 8 sample solution

Due 09/30/16

September 27, 2016

1. Find a recurrence that describes the worst-case complexity of the following recursive sorting algorithm. Show all work. You may assume that the floor function ($\lfloor\ \rfloor$) takes constant time.

Input: *data*: an array of integers
Input: n: the length of *data*
Output: a permutation of data such that $data[1] \leq data[2] \leq \ldots \leq data[n]$

```
1  Algorithm: ThirdSort
2  if n = 1 then
3  |   return data
4  else if n = 2 then
5  |   if data[1] > data[2] then
6  |   |   Swap data[1] and data[2]
7  |   end
8  |   return data
9  else
10 |   third = ⌊n/3⌋
11 |   Call ThirdSort on data[1..n-third]
12 |   Call ThirdSort on data[third+1..n]
13 |   Call ThirdSort on data[1..n-third]
14 |   return data
15 end
```

Answer: Lines 2–10 and line 14 all take $\Theta(1)$ time. The arrays in lines 11–13 are all size $n - third = n - \lfloor n/3 \rfloor = \lceil 2n/3 \rceil$, so these recursive calls take $T(\lceil 2n/3 \rceil)$ time. Thus, the algorithm runtime is described by the recurrence $T(n) = 3T(\lceil 2n/3 \rceil) + \Theta(1)$. $T(n) = 3T(2n/3) + \Theta(1)$ or $T(n) = 3T(n/1.5) + \Theta(1)$ would also be acceptable.

2. Use the Master Theorem to find the worst-case complexity of ThirdSort and describe how ThirdSort compares to SelectionSort.

   You may assume that $f(n)$ is regular if relevant. Recall that $\log_a(b) = \frac{\ln(b)}{\ln(a)}$

(you may need a calculator for this one). Be sure to include the value of c and the case of the Master Theorem in your answer.

Answer: $T(n) = 3T(n/1.5) + \Theta(n)$, so $a = 3$, $b = 1.5$, and $f(n) = \Theta(n)$. $c = \log_b(a) = \log_{1.5}(3) \approx 2.71$. $f(n) = O(n^{\log_{1.5}(3)-\epsilon})$ (choose any $\epsilon \leq \log_{1.5}(3)$). Thus, $T(n) = \Theta(n^c) \approx \Theta(n^{2.71})$ by the Master Theorem. (Regularity is not needed for the O case.)

ThirdSort is strictly worse than SelectionSort, which is $\Theta(n^2)$.