Matthew Kramer
U20891900

Homework 8

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.

$$T(n) = \begin{cases} \Theta(1) \text{ if } n \leq 2, \text{ else} \\ 3T(\frac{2}{3}n) + \Theta(1) \end{cases}$$

```
Input: data: an array of integers
Input: n: the length of data
Output: a permutation of data such that
        data[1] ≤ data[2] ≤ ... ≤ data[n]
Algorithm: ThirdSort
                                                  Θ(1) {
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]  T(2/3 n)   }
    Call ThirdSort on data[third+1..n]  T(2/3 n)   } 3T(2/3 n)
    Call ThirdSort on data[1..n-third]  T(2/3 n)   }
    return data                        Θ(1)
end
```

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.

$c = \log_{\frac{3}{2}} 3 = \dfrac{\ln(3)}{\ln(\frac{3}{2})} \approx 2.71$

$f(n) = \Theta(1)$ vs. $n^{2.71}$

Since $\Theta(1)$ is strictly smaller than $O(n^{2.71})$, $f(n) = O(n^{c-\epsilon})$, for some $\epsilon > 0$. Thus, the overall complexity is $T(n) = \Theta(n^{2.71})$

Compared to Selection Sort (complexity of $\Theta(n^2)$), ThirdSort will always run slower for all $n > 2$, scaling with larger n.