CS 4720/5720 Design and Analysis of Algorithms

Homework #3a

Due: Thursday, October 11, 2018

If written: due in class

If typed and submitted on Canvas: 11:59 pm

Submission requirements:

1. **5% extra credit** if you submit *digital, typed* writeups in PDF format to the "Homework 2 Writeup" assignment on the Canvas site. However, you may also turn in handwritten assignments in class – but assignments submitted in person will not receive the 5% extra credit. The due dates are the same for either submission method, but I will need handwritten assigments turned in to me in class.
2. Note: this is a "short" homework with no programming assignment, and will be worth half as much as a normal homework.

Assignment:

1. Determine whether each of the following recurrence relations can be solved using the Master Theorem. For those that can be solved with the Master Theorem, use the Master Theorem to determine the function's order of growth in terms of $\Theta(\cdot)$. For those than cannot be solved with the Master Theorem, explain why. For each, you can assume that $T(1) = 1$. Be clear about your choice of a, b, and d.
   (a) $T(n) = 5T(n/3) + n$
   (b) $T(n) = 2.7T(n/5) + n^2$
   (c) $T(n) = 2T(n-1) + n$
   (d) $T(n) = 1.1T(0.2n) + 1$
   (e) $T(n) = 2T(n/2) + n \log_2 n$
   (f) $T(n) = 2T(n/2) + \sqrt{n}$
   (g) $T(n) = 4T(n/2) + \sqrt{n^4 - n + 10}$
   (h) $T(n) = 7T(n/3) + \sum_{i=1}^{n} i$
   (i) $T(n) = 4T(n/2) + n^n$
   (j) $T(n) = 8T(n/3) + n^3$

2. For each of the following divide-and-conquer algorithms, use the Master Theorem to determine the algorithm's worst-case order of growth in terms of $\Theta(\cdot)$.

(a)

Algorithm 1

```
function Rec1($A[0..n-1]$)
    if  n == 1 then return $A[0]$
    else
        if Rec1($A[0 \,..\, \lfloor n/3 \rfloor]$)$< 0$ then return $A[0]$
        else return Rec1($A[\lfloor n/3 \rfloor \,..\, 2\lfloor n/3 \rfloor]$)
        end if
    end if
end function
```

(b)

Algorithm 2

```
function Rec2($A[0..n-1]$)
    if  n == 1 then return $A[0]$
    else return Rec2($A[0 \,..\, \lfloor n/2 \rfloor]$) + Rec2($A[\lfloor n/2 \rfloor + 1, n-1]$)
    end if
end function
```

3. Re-write the mergesort algorithm we did in class in the following way: Instead of breaking the array into 2 pieces and then recursing, break the array into k pieces and recurse on each one.

   (a) Determine worst-case efficiency of this algorithm in terms of $\Theta(\cdot)$, using the Master Theorem if you wish. For your analysis, you may assume that n is a power of k.

   (b) Is your result better or worse than the ordinary mergesort which splits into 2?

   (c) Why might we choose the ordinary $k = 2$ mergesort over one which splits into $k > 2$ pieces?