Assignment 1

Benjamin Sorenson

September 25, 2015

1. a) Show that insertion sort can sort the $\frac{n}{k}$ sublists, each of length k in $\Theta(nk)$ worst case time.

For each sublist, the worst case time is $\Theta(k^2)$. There are $\frac{n}{k}$ sublists. So, the worst case runtime is

$$\Theta\left(\frac{n}{k} \cdot k^2\right) = \Theta(n \cdot k)$$

b.) We could use the normal MERGE algorithm, but define another function like the following

```
MERGE-MANY(A, s)
    if s ≥ A.length
        RETURN
    for i from 1 to A.length by s
        MERGE(A, i, ⌊(i+s)/2⌋, i+s-1)
    MERGE-MANY(A, 2·s)
```

Setting $s = 2k$, we would merge the first $2 \cdot k$ elements of A on the first call, the next $2^2 k$ elements on the next call, ... $2^a k$ ..., halving the number of subarrays on each recursive call. We're always doing $\Theta(n)$ work on each call, but we only nee $\lg\left(\frac{n}{k}\right)$ calls.

So, that gets us to a total worst-case

runtime of $\Theta(n) \cdot \lg\left(\frac{n}{k}\right) = \Theta\left(n \lg\left(\frac{n}{k}\right)\right)$

2. For the recurrence relation $T(n) = 4T\left(\frac{n}{3}\right) + n$ show that a substitution proof with $T(k) \le ck^{\log_3 4}$ fails. Then show how to subtract off a lower-order term to make a substitution work.

Using $T(k) \le ck^{\log_3 4}$

$$T(k) \le 4 \cdot c \left(\frac{k}{3}\right)^{\log_3 4} + k$$

$$= \frac{4 \cdot c}{3^{\log_3 4}} \cdot k^{\log_3 4} + k$$

$$= c \cdot k^{\log_3 4} + k \not\le c \cdot k^{\log_3 4}$$

But, if we instead use $T(k) \le ck^{\log_3 4} - k$

$$T(k) \le 4\left[c \cdot \left(\frac{k}{3}\right)^{\log_3 4} - k\right] + k$$

$$T(k) \le \frac{4}{4} \cdot c k^{\log_3 4} - 4k + k$$

$$T(k) \le c \cdot k^{\log_3 4} - 3k \le ck^{\log_3 4} - k$$

We can, however, use $T(k) \ge ck^{\log_3 4}$ to prove the lower bound. We get

$$T(k) \ge 4 \cdot c\left(\frac{k}{3}\right)^{\log_3 4} + k$$

$$= c \cdot k^{\log_3 4} + k \ge c \cdot k^{\log_3 4}$$

and so $T(n) = \Theta(k^{\log_3 4})$

3. Give the upper and lower bounds:

a.) $T(n) = 8(n/3) + n^2$

$a = 8,\ b = 3,\ f(n) = n^2$

$n^{\log_3 8} \le n^2$

So, from case 3 of the master theorem $T(n) = \Theta(n^2)$

b.) $T(n) = 6T(n/4) + n \lg n$

$a = 6 \quad b = 4 \quad f(n) = n \lg n$

$f(n) = \Theta(n^{\log_4 6})$, so by case 1 of the master theorem $T(n) = \Theta(n \lg n)$

Note: I think the right answer is that $f(n) = \Theta(n^{\log_4 6})$ and thus $T(n) = \Theta(n^{\log_4 6} \lg n)$, but I don't know how to show $n \lg n = \Theta(n^{\log_4 6})$

c.) $T(n) = 9T(n/3) + n$

$a = 9$, $b = 3$, $f(n) = n$

$n \le n^2$ so, by case 1 of the master theorem

$T(n) = \Theta(n^2)$

4.) Let A be a sorted array. By def. $A[i] \le A[j]$ iff. $i \le j$.

A is a min-Heap iff

$A[Parent(i)] \le A[i]$ for every index i,

but $Parent(i) = \lfloor i/2 \rfloor \le i$, and

since A is sorted $A[Parent(i)] \le A[i]$ ■

5. MAX-HEAPIFY(A, i)

```
While True
    l ← LEFT(i)
    r ← RIGHT(i)
    if l ≤ heap-size[A] and A[l] > A[i]
        then largest ← l
        else largest ← i
    if r ≤ heap-size[A] and A[r] > A[largest]
        then largest ← r
    if largest = i
        BREAK
    exchange A[i] ↔ A[largest]
    i = largest
```