ALGORITHMS, FALL 2019, HOMEWORK 3

1. Use the master method for the following (state case or explain what dominates, and state the answer), or explain why it's not possible.

   (a) $T(n) = 10 \cdot T(\frac{n}{3}) + \Theta(n^2 \log^5 n)$.

   (b) $T(n) = 256 \cdot T(\frac{n}{4}) + \Theta(n^4 \log^4 n)$.

   (c) $T(n) = T(\frac{19n}{72}) + \Theta(n^2)$.

   (d) $T(n) = n \cdot T(\frac{n}{2}) + n^{\log_2 n}$.

   (e) $T(n) = 16 \cdot T(\frac{n}{4}) + n^2$.

   (f) $T(n) = 3 \cdot T(\frac{n}{2}) + n^2$.

   (g) $T(n) = T(\frac{n}{n-1}) + 1$.

   (h) $T(n) = 4 \cdot T(\frac{n}{16}) + \sqrt{n}$.

Answer:
a. $T(n) = a * T(\frac{n}{b}) + f(n)$
$a = 10, b = 3$
$n^{\log_3 10} = n^{2.09} > n^2$
Leaf level dominates.
$T(n) = \theta(n^{\log_3 10})$

b. $a = 256, b = 4$
$n^{\log_4 256} = n^4$
All levels $\approx$ same.
$T(n) = \theta(n^4 * \log^5(n))$

c. $a = 1, b = \frac{19}{72}$
$n^{\log_{\frac{19}{72}} 1} = n^1 \leq n^2$ Root level dominates. $T(n) = \theta(n^2)$

d. Master method can't be applied because $n^{\log_b a}$ needs a and b as constants.

e. $a = 16, b = 4$
$n^{\log_4 16} = n^2$
All levels $\approx$ same.
$T(n) = \theta(n^2 * \log(n))$

f. $a = 3, b = 2$
$n^{\log_2 3} \leq n^2$
Root level dominates.
$T(n) = \theta(n^2)$

g. Master method can't be applied because $n^{\log_b a}$ needs a and b as constants.

h. $a = 4, b = 16$
$n^{\log_{16} 4} = n^{0.5} = \sqrt{n}$
All levels $\approx$ same.
$T(n) = \theta(\sqrt{n} * \log(n))$

2. In class we learn how to find the number that has rank r among n elements, in $\Theta(n)$ time, using the classic algorithm with groups of size 5.

   (a) Show what happens if we form groups of size 3 instead.

   (b) Show what happens if we use groups of size k, where k represents an odd integer greater than 5.

   (c) Show what happens if we use $\sqrt{n}$ groups of size $\sqrt{n}$.

   (d) Show what happens if we use 5 groups of size $\frac{n}{5}$.

   In all cases, either prove that we still get $O(n)$ time or show that we cannot get $O(n)$ time. You may assume that division of produces an integer value. In other words you may assume that n is a power of whatever group size is used.

Answer:

2a. Using $\frac{n}{3}$ groups of 3 will cost $\theta(n)$. It will cost $T(\frac{n}{3})$ to find the median of medians. There will be $\frac{n}{6}$ items less than the median of medians. In the worst case scenario, every item in the first two columns will be smaller than the median of medians, so there will be $\frac{2n}{3}$ that can be smaller than the median of medians. The worst case scenario leads to a recurrence of $T(n) \leq T(\frac{n}{3}) + T(\frac{2n}{3}) + \theta(n)$, which says that $T(n) = \theta(n \log n)$. This concludes that groups of 3 are worse than groups of 5.

2b. As k increases, the worst case partition apporaches $\frac{3n}{4}$. The call for finding the median of medians becomes $T(\frac{n}{k})$, requiring $\theta(n)$ work to partition everything. The recurrence is $T(n) = T(\frac{3n}{4}) + T(\frac{n}{k}) + \theta(n)$. To find the median of each group, it costs $O(k \log k)$, with $\frac{n}{k}$ groups, which makes the recurrence $T(n) = T(\frac{3n}{4}) + T(\frac{n}{k}) + O(nlogk)$.

2c. With $\sqrt{n}$ groups, the recurrence is $T(n) = \theta(n) + T(\sqrt{n}) + T(\frac{3n}{4}) + \sqrt{n} * f(n)$, where $f(n)$ is the time to find a median for one partition. $f(n)$ sorted is $\theta(\sqrt{n} \log n)$, which leads to $T(n) = \theta(nlogn)$. There is no way to recurse this algorithm, so it's in non-linear time.

2d. The recurrence for $k = \frac{n}{5}$ is $T(n) = T(5) + T(\frac{3n}{4}) + \theta(n) + f(n)$. $T(5)$ is $O(1)$ because its a base case for finding the median of medians. $\theta(n)$ is the work it takes to poartition, and $f(n)$ is how long it takes to find the median of each partition. The time it takes to find the median for each partition is $f(n) = \theta(n \log n)$ if sorted, or $f(n) = 5T(\frac{n}{5})$ using recursion. Using sorting, it already becomes too much work; using recursion, the recurrence $T(n) = \theta(n) + T(\frac{3n}{4}) + 5T(\frac{n}{5})$ solves to $\theta(n \log n)$ at worst if the $T(\frac{3n}{4})$ gets taken out, and solves to something worse if it's left in. This algorithm works if $k \geq 5$ and if k is in constant time.