Algorithms & Data Structures 2018/19 Coursework

clvp22

December 26, 2018

Question 5.

We can only use master theorm for recurrences in the form,

$$T(n) = aT(n/b) + f(n)$$

for $a \geq 1$ and $b > 1$.

We have the 3 cases:

1. If $f(n) = O(n^{\log_b(a)-\epsilon})$ for some $\epsilon > 0$ then $T(n) = \Theta(n^{\log_b(a)})$.
2. If $f(n) = \Theta(n^{\log_b(a)})$ then $T(n) = \Theta(n^{\log_b(a)} \log n)$.
3. If $f(n) = \Omega(n^{\log_b(a)+\epsilon})$ for some $\epsilon > 0$ then and if $af(n/b) \leq cf(n)$ for some $c < 1$ and all n is large then $T(n) = \Theta(f(n))$

(a)

$T(n) = 9T(n/3) + n^2$, $a = 9$, $b = 3$, $f(n) = n^2$.

Trying case 2, $n^{\log_3(9)} = n^2$, $f(n)$ is $\Theta(n^2)$, so we have a case 2.

Therefore, $T(n)$ is $\Theta(n^2 \log n)$

(b)

$T(n) = 4T(n/2) + 100n$, $a = 4$, $b = 2$, $f(n) = 100n$.

Trying case 2, $n^{\log_2(4)} = n^2$, $f(n)$ is not $\Theta(n^2)$.

Trying case 1, $n^{\log_2(4)-\epsilon} = n^{2-\epsilon}$, for $\epsilon = 1$ we have n, $f(n)$ is $O(n)$, so we have a case 1.

Therefore, $T(n)$ is $\Theta(n^2)$

(c)

$T(n) = 2^n T(n/2) + n^3$, $a = 2^n$, $b = 2$, $f(n) = n^3$.

This recurrence can't be solved using master theorem because a is not constant.

(d)

$T(n) = 3T(n/3) + c \cdot n$, $a = 3$, $b = 3$, $f(n) = c \cdot n$.

Trying case 2, $n^{\log_3(3)} = n$, $f(n)$ is $\Theta(n)$, so we have a case 2.

Therefore $T(n) = \Theta(n \log n)$

(e)

$T(n) = 0.99T(n/7) + 1/(n^2)$, $a = 0.99$, $b = 7$, $f(n) = \frac{1}{n^2}$.

This recurrence can't be solved using master theorm because $a = 0.99$, so the condition $a \geq 1$ is not met.