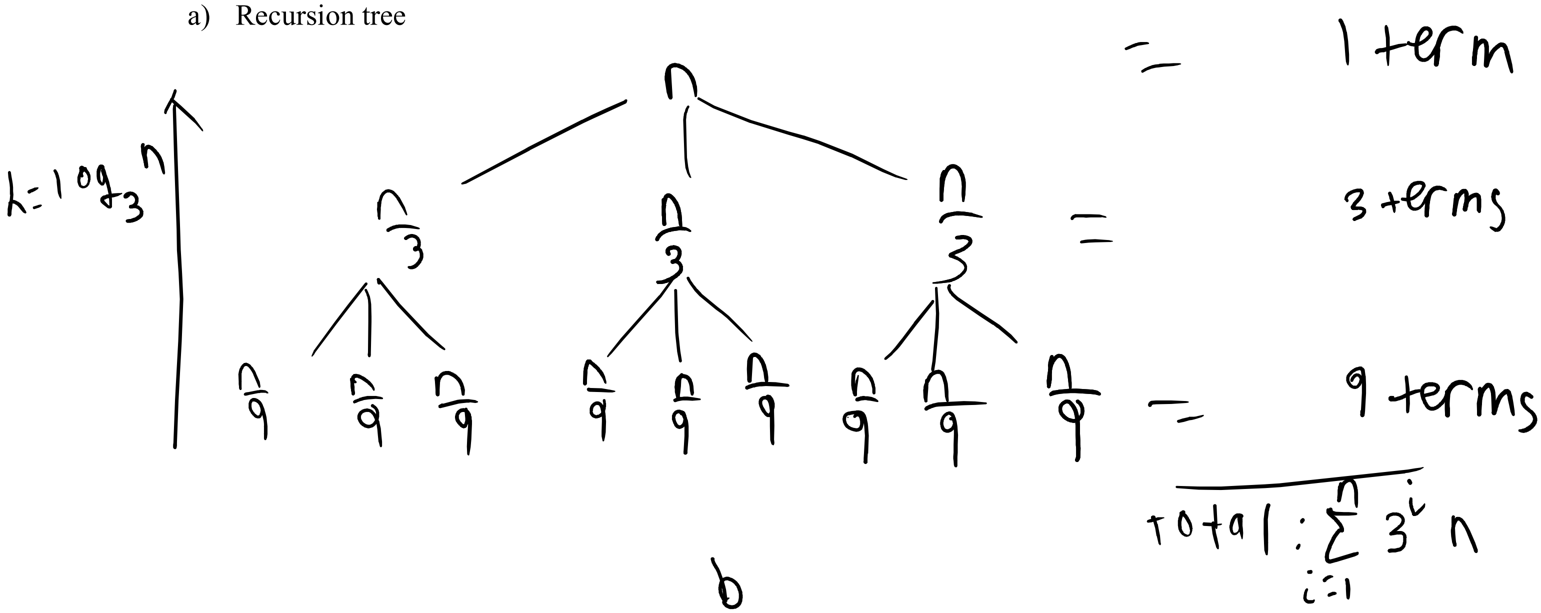
Brandon Vo

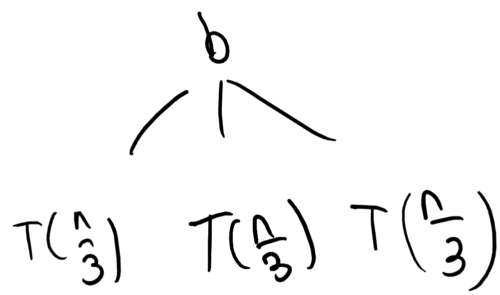
Algorithms homework related to exam 1, Summer 202

Hw2: Due Monday, June 7, at 11:59pm

1. $S(n) = 3S(\frac{n}{3}) + b$.
   (a) Use a recursion tree to show that $S(n) = \Theta(g(n))$, for some function $g(n)$.
   (b) Use induction to prove that $S(n) = \Omega(g(n))$.

a) Recursion tree





$$Assuming\ that\ T(1) = \Theta(b) = b_2$$

$$Height = \log_3 n$$

$$b\ leaves\colon\ b + 3b + 9b \ldots = Geometric\ series\colon \sum_{k=0}^{\log_3 n - 1} 3^k * b$$

$$Total\colon \sum_{k=0}^{\log_3 n - 1} 3^k * b + b_2, unravel\ the\ geometric\ series$$

$$\frac{3^{\log_3 n} - 1}{3 - 1} * b + b_2, swapping\ out\ the \log terms$$

$$\frac{n^{\log_3 3} - 1}{2} * b + b_2 = b * \frac{n-1}{2} + b_2$$

$$If\ b\ is\ constant\ time, then\ S(n) = b * \frac{n-1}{2} + b_2\ shows\ that\ S(n)\ is\ \Theta(n)\ where\ b \geq 0$$

b) Induction:

$$Assuming\ that\ the\ equation\ is\ \Omega(n), Inductive\ Hypothesis\colon\ S(k) \geq dk\ where\ k < n$$

$$With\ S(n) = 3S\left(\frac{n}{3}\right) + b, substituting\ S(n)\ with\ k = \frac{n}{3}$$

$$S(n) \geq g(k) = 3 * \frac{dn}{3} + b, using\ k = \frac{n}{3}$$

$$S(n) \geq dn + b\ where\ d > 0, n \geq k(desired\ form + leftovers)$$

$$S(n) \geq dn + b$$

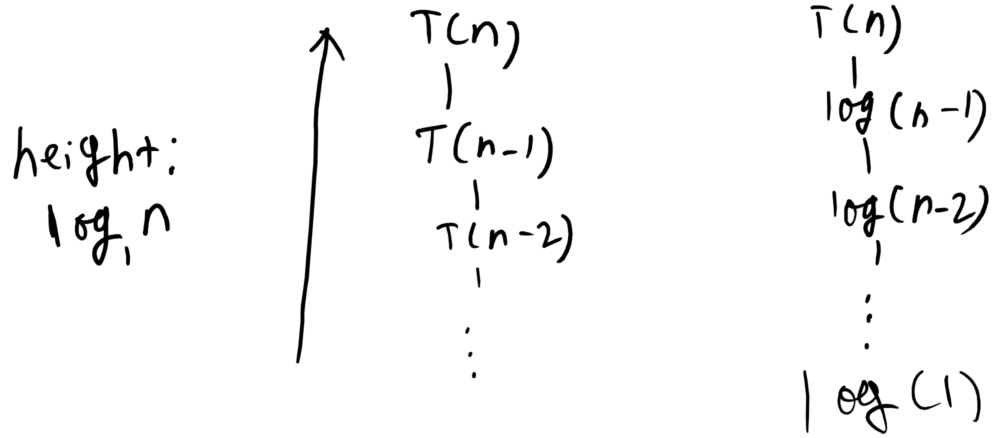
$$b\ is\ a\ constant\ which\ is\ being\ added\ with\ every\ iteration$$

$$Base\ Case\colon\ T(1) = b\ if\ b > 0$$

Brandon Vo

2. $T(n) = T(n-1) + \log n$. Assume that there is an appropriate base case so that you don't get an undefined function.

   (a) Draw a recursion tree for $T(n)$. Identify its depth and the amount of work per level. Use "exaggerate and simplify" to get an upper bound for $T(n)$, and then underestimate and simplify to get a matching lower bound. (Always using Theta notation). Your bound should be in a form that is easily recognizable (and easy to compare to simple functions like linear, quadratic, etc).



$$Total: T(n) + T(n-1) + T(n-2) + \cdots + T(1)$$

$$Total\ work: \log n + \log(n-1) + \log(n-2) + \cdots + \log(1) = \sum_{k=1}^{n} \log(n-k)$$

$$Can\ exaggerate\ and\ simplify\ to\ get\ the\ upper\ bound\ by\ raising\ all\ variables\ up\ to\ n$$

$$Total\ Work\ exaggerated\ to\ n: T(n) \leq \log n + \log n + \log n + \cdots + \log n = \log n^n$$

$$T(n) \leq n \log n$$

$$Upper\ bound: T(n) = O(n \log n)$$

$$Underestimating\ and\ simplifying\ from\ n\ to \frac{n}{2}\ terms\ for\ the\ lower\ bound$$

$$\sum_{k=\left\lceil \frac{n}{2} \right\rceil}^{n} \log k = \log\frac{n}{2} + \log\frac{n+1}{2} + \log\frac{n+2}{2} + \cdots + \log n = n * \log\frac{n^{\frac{1}{2}}}{2} = \frac{n}{2}\log\frac{n}{2}$$

$$T(n) \geq \frac{n}{2}\log\frac{n}{2} = \frac{n}{2}\log n - \frac{n}{2}\log 2\,; \log 2 = 1$$

$$T(n) \geq \frac{n}{2}\log n - \frac{n}{2}$$

$$This\ shows\ that\ the\ lower\ bound\ is\ \Omega(\mathrm{n} \log n)$$

Brandon Vo

(b) Derive the same upper bound for $T(n)$, by induction.

$$T(n) = T(n-1) + \log n$$

$$T(n-1) = T(n-2) + \log(n-1)$$

$$T(n-2) = T(n-3) + \log(n-2)$$

…

$$Base\ Case\!:\ T(1) = T(0) + \log 1 = 0$$

$$Inductive\ Hypothesis\!:\ For\ k < n, assuming\ O(n \log n)\ , T(k) \le d * k \log k$$

$$Substitution\ method\!: T(n) = T(n-1) + \log n\ , substitute\ k = n-1$$

$$T(n) \le d * (n-1) * \log(n-1) + \log n$$

$$Exaggerate\ and\ simplify\!:\ \log n > \log(n-1)$$

$$T(n) \le d * (n-1) * \log n + \log n$$

$$T(n) \le d(n \log n - \log n) + \log n$$

$$T(n) \le dn * \log n - (d * \log n - \log n), desired\ form - leftovers$$

$$Concerning\ the\ base\ case\ O(1)\!: T(1) \le 0 + \log 1 = 0\ \ d \ge 1$$

3. Solve $T(n) = T(\sqrt{n}) + 1$, using a change of variables: $n = 2^m$.

To warm up, consider the following question. (Don't submit an answer for the warmup) Think of an abstract recursive algorithm that operates on a $n \times n$ matrix: It does non-recursive work proportional to the number of elements in the matrix (say just n^2 work), and then recurses on each of the four quadrants. Assume that quadrants are nicely defined.

This could be expressed as $S(n) = 4S(\frac{n}{2}) + n^2$, meaning the parameter used in S to describe the problem is the matrix side length.

But suppose that we hadn't thought of doing that, and instead we expressed the time complexity as $T(n^2) = 4T(\frac{n^2}{4}) + n^2$. Here, the parameter used for the problem size is the total number of elements. In this case, because this parameter in T is a function of n (other than just n itself), we don't really have a direct method to deal with it. But we could set $m = n^2$, so $T(m) = 4T(\frac{m}{4}) + m$.

What's the solution in each case (in terms of n)? It should be the same.

Besides that, the lesson here is that there can be multiple ways to quantify a problem size and recurse accordingly, e.g. $S(n)$ vs $T(n^2)$. You'll need to consider this after you apply the change of variables as suggested above for the actual homework problem.

$$With\ n = 2^m\ and\ m < n$$

$$T(n) = T(\sqrt{\mathrm{n}}) + 1$$

$$T(2^m) = T\left(2^{\frac{\mathrm{m}}{2}}\right) + 1$$

$$Renaming\ for\ S(m) = T(2^m),$$

$$S(m) = S\left(\frac{m}{2}\right) + 1$$

$$Can\ now\ use\ master\ method\ to\ solve\ S(m)$$

$$Leaves\colon\ m^{\log_2 1} = m^0 = 1, Root\colon\ 1\ (Case\ 2)$$

$$S(m) = O(\mathrm{S(m)} *\ \log m) = O(m^0 * \log m) = O(\log m)$$

$$Substituting\ m = \log_2 n\,, S(m) = O(\log(\log_2 m))$$

Brandon Vo

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

a) $T(n) = 10 * T\left(\frac{n}{3}\right) + \Theta(\mathrm{n}^2 \log^5 n)$

$$Leaf\colon\ n^{\log_3 10} > n^3, Root\colon\ n^2 \log^5 n\ \ (Case\ 1)$$

$$\log_3 10 = 2.0959 > 2, leaves\ dominate\ polynomially$$

$$\Theta\left(n^{\log_3 10}\right)$$

b) $T(n) = 256 * T\left(\frac{n}{4}\right) + \Theta(\mathrm{n}^4 \log^4 n)$

$$Leaves\colon\ n^{\log_4 256} = n^4, Root\colon\ n^4 \log^4 n\ \ (Case\ 3\colon Root\ dominates\ polynomially)$$

$$\Theta(\mathrm{n}^4 \log^4 n)$$

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

$$Leaves\colon\ n^{\log_{\frac{72}{19}} 1} = n^0 = 1, Root\colon\ n^2\ (Case\ 3\colon Root\ dominates\ polynomially)$$

$$\Theta(n^2)$$

d) $T(n) = n * T\left(\frac{n}{2}\right) + n^{\log_2 n}$

$$Requirement\colon\ a\ \&\ b\ must\ be\ O(1)\ but\ A\ is\ O(n).\ \ Master\ method\ is\ not\ possible.$$

e) $T(n) = 16 * T\left(\frac{n}{4}\right) + n^2$

$$Leaves:\ n^{\log_4 16} = n^2, leaves:\ n^2\ (Case\ 2:\ Both\ are\ the\ same\ level)$$

$$\Theta(n^2 \log n)$$

f) $T(n) = 3 * T\left(\frac{n}{2}\right) + n^2$

$$Leaves:\ n^{\log_2 3} = n^{1.59}, leaves:\ n^2\ (Case\ 3:\ Root\ dominates\ polynomially)$$

$$\Theta(n^2)$$

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

$Requirement:\ a\ \&\ b\ are\ O(1)\ but\ a\ is\ O(n).\ Master\ method\ is\ not\ possible.$

h) $T(n) = 4 * T\left(\frac{n}{16}\right) + \sqrt{n}$

$$Leaves:\ n^{\log_{16} 4} = n^{1/2}, Root:\ n^{\frac{1}{2}}\ (Case\ 2:\ Both\ are\ the\ same\ level)$$

$$\Theta(n^{0.5} \log n)$$