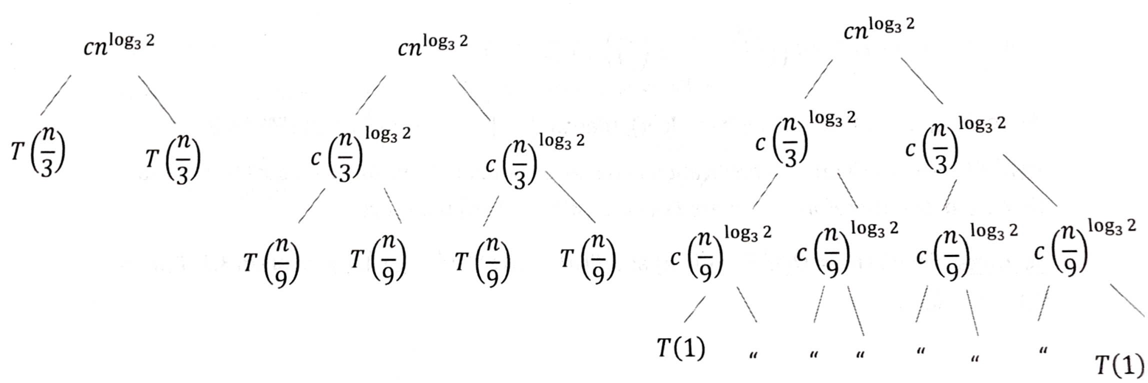
HW3

1. By Qingliu Wu

1. Master method: $T(n) = aT\left(\frac{n}{b}\right) + f(n)$. Since $T\left(\frac{n}{3}\right) + T\left(\frac{n}{3}\right) = 2T\left(\frac{n}{3}\right)$, a = 2, b = 3, and f(n) = $2^{\log_3 n} = n^{\log_3 2}$. Then, $\left[n^{\log_b a} = n^{\log_3 2}\right] = \left[f(n) = n^{\log_3 2}\right]$. Thus, this algorithm corresponds to case 2 of the Master theorem and $T(n) = \theta\left(n^{\log_b a} * \lg n\right) = \theta\left(n^{\log_3 2} * \lg n\right)$.

2. By Qingliu Wu

2. $T(n) = 2T\left(\frac{n}{3}\right) + cn^{\log_3 2}$. This gives the tree on the left. Next, $T\left(\frac{n}{3}\right) = 2T\left(\frac{n}{9}\right) + c\left(\frac{n}{3}\right)^{\log_3 2}$, giving the tree in the middle. The rightmost tree shows the tree fully expanded to n=1.



Therefore, the total cost for the 1st level is $cn^{\log_3 2}$, the 2nd level is $2c\left(\frac{n}{3}\right)^{\log_3 2} = cn^{\log_3 2}$, the 3rd level is $4c\left(\frac{n}{9}\right)^{\log_3 2} = cn^{\log_3 2}$, and so on. Thus, the total cost for each level is $cn^{\log_3 2}$. Since the subproblem size decreases by a factor of 3 every level, the subproblem size n = 1 is reached when $n/3^i = 1$, or when $i = \log_3 n$. Thus, the tree is of height $\log_3 n$. The bottom level has $2^{\log_3 n} = n^{\log_3 2}$ nodes that cost T(1) each, so T(n)= $\theta\left(n^{\log_3 2}\right)$ = for the bottom level. The total cost of the entire tree is $T(n) = \theta\left(n^{\log_3 2} * \log_3 n\right) + \theta\left(n^{\log_3 2}\right)$. Dropping the lower order term, the recursion method yields $T(n) = \theta\left(n^{\log_3 2} * \log_3 n\right)$

3. By Dangnhi Ngo

3/ Use Substitution Method

$$T(n) = T\left(\frac{n}{3}\right) + T\left(\frac{n}{3}\right) + 2^{(\log_3 n)} \qquad (n > k)$$

$$= 2T\left(\frac{n}{3}\right) + n^{(\log_3 2)}$$

* Upper bound: $T(n) \leqslant 2T(n/3) + c.n^{(\log_3 2)}$ (c : positive constant)

Guess: $T(n) = O(n^{(\log_3 2)}.\lg n)$

$\leqslant d.(n^{(\log_3 2)}.\lg n)$ (d : positive constant)

$T\left(\frac{n}{3}\right) \leqslant d.\left(\frac{n}{3}\right)^{(\log_3 2)}.\lg\left(\frac{n}{3}\right)$

$T\left(\frac{n}{3}\right) \leqslant d.\frac{n^{(\log_3 2)}}{2}.\lg\left(\frac{n}{3}\right)$

Substitution: $T(n) \leqslant \cancel{2}\left(d.\frac{n^{(\log_3 2)}}{\cancel{2}}.\lg\left(\frac{n}{3}\right)\right) + c.n^{(\log_3 2)}$

$= d\,n^{(\log_3 2)}.(\lg n - \lg 3) + c.n^{(\log_3 2)}$

$= dn^{(\log_3 2)}\lg n - dn^{(\log_3 2)}\lg 3 + c.n^{(\log_3 2)}$

$\leqslant dn^{(\log_3 2)}\lg n \quad (\text{if } (-dn^{(\log_3 2)}\lg 3 + c.n^{(\log_3 2)}) \leqslant 0)$

$\Rightarrow d\cancel{n^{(\log_3 2)}}\lg 3 \geqslant c.\cancel{n^{(\log_3 2)}}$

$\Rightarrow d \geqslant c/\lg 3$

Therefore, $T(n) = O(n^{(\log_3 2)}.\lg n)$ (1)

Lower bound: $T(n) \geqslant 2T\left(\frac{n}{3}\right) + c.n^{(\log_3 2)}$ (c: positive constant)

Guess: $T(n) = \Omega(n^{(\log_3 2)}.\lg n)$

$\geqslant d(n^{\log_3 2}.\lg n)$ (d: positive constant)

$T\left(\frac{n}{3}\right) \geqslant d\left(\frac{n}{3}\right)^{(\log_3 2)}.\lg\left(\frac{n}{3}\right)$

$T\left(\frac{n}{3}\right) \geqslant d\frac{n^{(\log_3 2)}}{2}.\lg\left(\frac{n}{3}\right)$

Substitution: $T(n) \geqslant 2\left(d\frac{n^{(\log_3 2)}}{2}.\lg\left(\frac{n}{3}\right)\right) + c.n^{(\log_3 2)}$

$= d.n^{(\log_3 2)}.(\lg n - \lg 3) + c.n^{(\log_3 2)}$

$= d.n^{(\log_3 2)}\lg n - dn^{(\log_3 2)}\lg 3 + c.n^{(\log_3 2)}$

$\geqslant dn^{(\log_3 2)}\lg n \quad (\text{if } (-dn^{(\log_3 2)}\lg 3 + c.n^{(\log_3 2)} \geqslant 0))$

$\Rightarrow c\cancel{n^{(\log_3 2)}} \geqslant d\cancel{n^{(\log_3 2)}}\lg 3$

$\Rightarrow c \geqslant d.\lg 3$

$\Rightarrow d \leqslant c/\lg 3$

Therefore, $T(n) = \Omega(n^{(\log_3 2)}.\lg n)$ (2)

From (1) and (2),

$$\boxed{T(n) = \Theta(n^{(\log_3 2)}.\lg n)}$$

4. By Logan Mann

4. $T(n) = 4T(\frac{n}{3}) + 2^{\lg n}$

$= 4T(\frac{n}{3}) + n$ $\qquad n^{\lg_2 2}$

$a = 4,\ b = 3$

$\log_b a$

$n^{\log_3 4}$ vs n

$n^{\log_3 4}$ is larger, so case 1

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

5. By Dangnhi Ngo

Recurrence

(1) $T(n) = 4T(\frac{n}{4}) + n$

$a = 4,\ b = 4 \Rightarrow \log_b a = \log_4 4 = 1$

$n^{(\log_b a)} = n^1 = n$

$f(n) = n$

Compare $n^{(\log_b a)} = n$ vs. $f(n) = n$

$\Rightarrow$ $f(n)$ is polynomially equal to $n^{(\log_b a)}$

$\Rightarrow$ Case 2 in Master Method

$T(n) = \Theta(n.\lg n)$

(2) $T(n) = 3T\left(\frac{n}{2}\right) + \sqrt{10}\,n^2$

$a = 3,\ b = 2 \Rightarrow \log_b a = \log_2 3 \approx 1.58$

Compare $n^{(\log_b a)} = n^{\log_2 3}$ vs. $f(n) = \sqrt{10}\,n^2$

$\Rightarrow$ $f(n)$ is polynomially greater than $n^{(\log_b a)}$ $\left(\sqrt{10}\,n^2 > n^{\log_2 3}\right)$

$\Rightarrow$ Case 3 in Master Method

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

(3) $T(n) = T(n-1) + 10n$

$a = 1,\ b = 1$ not in the form of aT(n/b) + f(n), can't identify a, b

$\Rightarrow$ Master Method cannot be applied, because b should be greater than 1

(4) $T(n) = 3T\left(\frac{2n}{3}\right) + n$

$a = 3,\ b = 3/2 \Rightarrow \log_b a = \log_{(3/2)} 3$

Compare $n^{(\log_b a)} = n^{(\log_{(3/2)} 3)}$ vs. $f(n) = n$

$\Rightarrow$ $f(n)$ is polynomially smaller than $n^{(\log_b a)}$ $\left(n < n^{(\log_{(3/2)} 3)}\right)$

$\Rightarrow$ Case 1 in Master Method

$T(n) = \Theta\left(n^{(\log_{(3/2)} 3)}\right)$

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

$a = 2^n,\ b = 3$

$\Rightarrow$ Master Method cannot be applied, because a should be a constant

6. By David Baumann

6

1)

Find_Value(A, p, r):	Cost	# of times
if (p == r)	c_1	1
if (A[p] == p)	c_2	1
return p	c_3	1
else	0	
return 0	c_5	1
$q = \lfloor (p+r)/2 \rfloor$	c_6	1
if (A[q] == q)	c_7	1
return q	c_8	1
if (A[q] > q)	c_9	1
return Find_Value(A, p, q)	$T(n/2)$	1
else if (A[q] < q)	c_{11}	1
return Find_Value(A, q+1, r)	$T(n/2)$	1

2) $T(n) = T(\frac{n}{2}) + \Theta(1)$, since the recursion only happens once

using Master Theorem,

$a = 1, \quad b = 2, \quad f(n) = \Theta(1)$

$\log_b a = \log_2 1 = 0$

$n^0 = 1$ vs $\Theta(1)$ => asymptotically equal, so Case 2 of Master Theorem

$\therefore \; T(n) = \Theta(1 \cdot \lg n)$

$T(n) = \Theta(\lg n)$

3) $T(n) = T(\frac{n}{2}) + \Theta(1)$

guess $T(n) = \Theta(\lg n)$

upper bound: $T(n) \leq T(\frac{n}{2}) + c$

guess $T(n) = O(\lg n)$

$\leq d \cdot \lg n$

$T(\frac{n}{2}) \leq d \cdot \lg(n/2)$

Substitute: $T(n) \leq d \cdot \lg(n/2) + c$

$\leq d \cdot \lg(n) - d \cdot \lg(2)$

$\leq d \cdot \lg(n) + c - d$

$\leq d \cdot \lg(n)$ if c<=d

$\therefore T(n) = O(\lg n)$

lower bound: $T(n) \geq T(\frac{n}{2}) + c$

guess $T(n) = \Omega(\lg n)$

$\geq d \cdot \lg n$

$T(n/2) \geq d \cdot \lg(n/2)$

Substitute: $T(n) \geq d \cdot \lg(n/2) + c$

$\geq d \cdot \lg n - d \cdot \lg 2 + c$

$\geq d \cdot \lg n + c - d$

$\geq d \cdot \lg n$ if c>=d

$\therefore T(n) = \Omega(\lg n)$

$T(n) = O(\lg n)$ and $T(n) = \Omega(\lg n)$

$\therefore T(n) = \Theta(\lg n)$