Zachary Seymour
CS 575
Theory Assignment 4
October 29, 2013

1 Use the Master Theorem to solve the recurrences

1a $T(n) = 4T(\frac{n}{2}) + n^3$

We have $\log_b a = \log_2 4 = 2$ and $f(n) = n^3$. So, $\frac{n^3}{n^2} = n = \Omega\left(n^1\right)$, which is Case 3. So $T(n) = \Theta\left(n^3\right)$.

1b $T(n) = 2T(\frac{n}{2}) + n\lg^2 n$

We have $\log_b a = \log_2 2 = 1$ and $f(n) = n\lg^2 n$. So, $\frac{n\lg^2 n}{n} = \lg^2 n = \Theta\left(\lg^2 n\right)$, which is Case 2. So $T(n) = \Theta\left(n\lg^3 n\right)$.

1c $T(n) = 3T(\frac{n}{4}) + n$

We have $\log_b a = \log_4 3$ and $f(n) = n$. So, $\frac{n}{n^{\log_4 3}} = n^{1-\log_4 3} = \Omega\left(n^{1-\log_4 3}\right)$, which is Case 3. So $T(n) = \Theta\left(n\right)$.

1d $T(n) = 2T(\frac{n}{4}) + n\lg n$

We have $\log_b a = \log_4 2 = \frac{1}{2}$ and $f(n) = n\lg n$. So, $\frac{n\lg n}{\sqrt{n}} = \sqrt{n}\lg n = \Omega\left(n\right)$, which is Case 3. So $T(n) = \Theta\left(n\lg n\right)$.

1e $T(n) = 4T(\frac{n}{2}) + n^2$

We have $\log_b a = \log_2 4 = 2$ and $f(n) = n^2$. So, $\frac{n^2}{n^2} = 1 = \Omega\left(\lg^0 n\right)$, which is Case 2. So $T(n) = \Theta\left(n^2\lg n\right)$.

1f $T(n) = 4T(\frac{n}{2}) + n^3$

We have $\log_b a = \log_2 4 = 2$ and $f(n) = n^3$. So, $\frac{n^3}{n^2} = n = \Omega(n)$, which is Case 3. So $T(n) = \Theta\left(n^3\right)$.

2 Use substitution to show that $T(n) = T(\frac{n}{2}) + T(\frac{n}{4}) + T(\frac{n}{9})$ belongs to $\Theta(n)$.

We will guess an upper bound of $kn - b$. We need to show that $T(n) \leq kn - b$ for all n. Our base case is $n = 1$. We'll assume $T(1) = 1$. So we have $1 = T(1) \leq k - b$, which holds as long as $k \geq b + 1$. Now, we will assume this holds for $\frac{n}{2}$, $\frac{n}{4}$, and $\frac{n}{9}$.

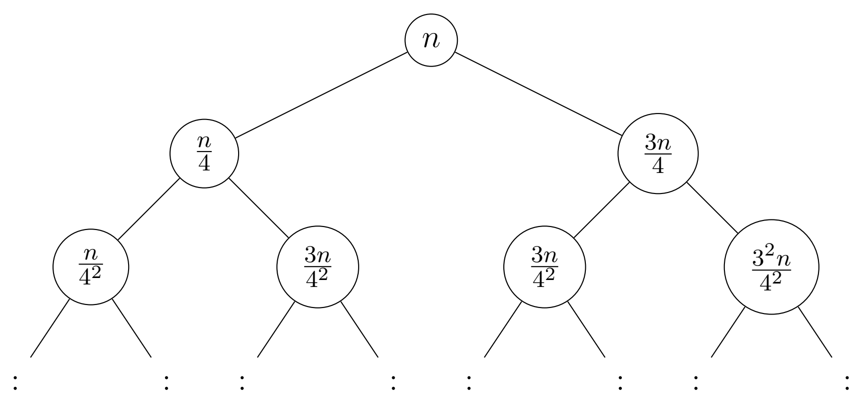
$$\begin{aligned}
T(n) &= T(\frac{n}{2}) + T(\frac{n}{4}) + T(\frac{n}{9}) \\
&= \frac{kn}{2} + \frac{kn}{4} + \frac{kn}{9} - 3b \\
&= \frac{31kn}{36} - 3b \\
&\leq kn - b
\end{aligned}$$

Thus, $T(n) = T(\frac{n}{2}) + T(\frac{n}{4}) + T(\frac{n}{9})$ belongs to $\Theta(n)$.

3

$$T(n) = \begin{cases} \Theta(1) & \text{for } n \leq 1 \\ T(\frac{n}{4}) + T(\frac{3n}{4}) & \text{otherwise} \end{cases}$$

3a Show using the recursion tree method that $T(n) = \Theta(n \lg n)$ is a good guess.



The two branches receive unequal portions of the work, with the rightmost branch being the longest, having length $\log_{4/3} n$. Thus, our guess is $\Theta(n \lg n)$.

3b Prove that $T(n) = \Theta(n \lg n)$ using the substitution method.

Proof. We will guess an upper bound of $kn \lg bn$. So, we need to show $T(n) \leq kn \lg bn$ for all n. Our base case is $n = 1$. We have $T(1) = \Theta(1) \leq k \cdot 1 \cdot \lg b = \lg b$. Now, we will assume it holds for $\frac{n}{4}$ and $\frac{3n}{4}$.

$$\begin{aligned}
T(n) &= T(\frac{n}{4}) + T(\frac{3n}{4}) \\
&= \frac{n}{4} \lg \left(b\frac{n}{4}\right) + \frac{3n}{4} \lg \left(b\frac{3n}{4}\right) \\
&= n \lg(\frac{3^{3/4}}{4} bn) \\
&\leq kn \lg bn
\end{aligned}$$

∎

Extra Credit Assume a MinHeap. Assume that each record contains a key k and data d. Modify the `insert` and `deleteMin` procedure of a minheap so that the function `location(d)` (the index in the heap) takes only $\Theta(1)$ time. Assume that d is an integer between 0 and n.

The only way I can really think to get constant time complexity is to use another array alongside the array storing the heap, where the indices, from 0 to n, are the data values.

function INSERT(v)
  last = last + 1
  bt[last] ← v
  index = percolate(last)   ▷ We would need percolate to return the true index.
  indices[v.d] ← index
end function

function DELETEMIN
  minKeyItem = bt[root]
  swap(root, last)
  indices[minKeyItem.d] = NULL
  last = last - 1
  if last > 0 **then**
    siftDown(root)
  end if
  return minKeyItem
end function