CSE 332: Data Structures and Parallelism

Section 3: Recurrences Solutions

0. Big-Oh Bounds for Recurrences

For each of the following, find a Big-Oh bound for the provided recurrence.

(a) $T(n) = \begin{cases} 1 & \text{if } n = 0 \\ T(n-1) + 3 & \text{otherwise} \end{cases}$

Solution:

Unrolling the recurrence, we get $T(n) = \underbrace{3 + \cdots + 3}_{n \text{ times}} + 1 = 3n + 1$. This is $\mathcal{O}(n)$.

(b) $T(n) = \begin{cases} 1 & \text{if } n = 1 \\ 8T(n/2) + 4n^2 & \text{otherwise} \end{cases}$

Solution:

Note that $a = 8$, $b = 2$, and $c = 2$. Since $\lg(8) = 3 > 2$, we have $T \in \Theta(n^{\lg(8)}) = \Theta(n^3)$ by Master Theorem. Then, by definition of Big-Theta, we also know $T \in \mathcal{O}(n^3)$.

(c) $T(n) = \begin{cases} 1 & \text{if } n = 1 \\ 7T(n/2) + 18n^2 & \text{otherwise} \end{cases}$

Solution:

Note that $a = 7$, $b = 2$, and $c = 2$. Since $\lg(7) = 3 > 2$, we have $T \in \Theta(n^{\lg(7)})$ by Master Theorem. Then, by definition of Big-Theta, we also know $T \in \mathcal{O}(n^{\lg(7)})$.

(d) $T(n) = \begin{cases} 1 & \text{if } n = 1 \\ T(n/2) + 3 & \text{otherwise} \end{cases}$

Solution:

Note that $a = 1$, $b = 2$, and $c = 0$. Since $\lg(1) = 0 = 2$, we have $T \in \Theta(\lg(n))$ by Master Theorem. Then, by definition of Big-Theta, we also know $T \in \mathcal{O}(\lg(n))$.

(e) $T(n) = \begin{cases} 1 & \text{if } n = 0 \\ T(n-1) + T(n-2) + 3 & \text{otherwise} \end{cases}$

Solution:

Note that this recurrence is bounded above by $T(n) = 2T(n-1) + 3$. If we unroll that recurrence, we get $3 + 2(3 + 2(3 + \cdots + 2(1)))$.

This is approximately $\sum_{i=0}^{n} 3 \times 2^i = 3(2^{n+1} - 1)$ which mean $T \in \mathcal{O}(2^n)$. We can actually find a better bound (e.g., it's not the case that $T \in \Omega(2^n)$).

1. Recurrences and Big-Oh Bounds

Consider the function f. Find a recurrence modeling the worst-case runtime of this function and then find a Big-Oh bound for this recurrence.

```
f(n) {
   if (n == 0) {
      return 0
   }

   int result = 0
   for (int i = 0; i < n; i++) {
      for (int j = 0; j < i; j++) {
         result += j

      }
   }
   return f(n/2) + result + f(n/2)
}
```

(a) Find a recurrence $T(n)$ modeling the worst-case time complexity of $f(n)$.

Solution:

We look at the three separate cases (base case, non-recursive work, recursive work). The base case is $\mathcal{O}(1)$, because we only do a return statement. The non-recursive work is $\mathcal{O}(1)$ for the assignments and `if` tests and $\sum_{i=0}^{n} i = \frac{n(n+1)}{2}$ for the loops. The recursive work is $2T\left(\frac{n}{2}\right)$.

Putting these together, we get:

$$T(n) = \begin{cases} 1 & \text{if } n = 1 \\ 2T\left(\frac{n}{2}\right) + \frac{n(n+1)}{2} & \text{otherwise} \end{cases}$$

(b) Find a Big-Oh bound for your recurrence.

Solution:

Note that $\frac{n(n+1)}{2} = \frac{n^2}{2} + \frac{n}{2} \in \mathcal{O}(n^2)$. The recursion tree has $\lg(n)$ height, each non-leaf node of the tree does $\left(\frac{n}{2^i}\right)^2$ work, each leaf node does 1 work, and each level has 2^i nodes.

So, the total work is $\sum_{i=0}^{\lg(n)} 2^i \left(\frac{n^2}{2^i}\right)^2 + 1 \cdot 2^{\lg n} = n^2 \sum_{i=0}^{\lg(n)} \left(\frac{2^i}{4^i}\right) + n < n^2 \sum_{i=0}^{\infty} \left(\frac{1}{2^i}\right) + n = \frac{n^2}{1 - \frac{1}{2}} + n.$

This expression is upper-bounded by n^2 so $T \in \mathcal{O}(n^2)$.

2. Recurrences and Closed Forms

Consider the function g. Find a recurrence modeling the worst-case runtime of this function, and then find a closed form for the recurrence.

```
g(n) {
   if (n <= 1) {
      return 1000
   }
   if (g(n/3) > 5) {
      for (int i = 0; i < n; i++) {
         println("Yay!")
      }
      return 5 * g(n/3)
   }
   else {
      for (int i = 0; i < n * n; i++) {
         println("Yay!")
      }
      return 4 * g(n/3)
   }
}
```

(a) Find a recurrence $T(n)$ modeling the worst-case time complexity of $g(n)$.

Solution:

$$T(n) = \begin{cases} 1 & \text{if } n \leq 1 \\ 2T\left(\frac{n}{3}\right) + n & \text{otherwise} \end{cases}$$

(b) Find a closed form for the above recurrence.

Solution:

The recursion tree has height $\log_3(n)$, each non-leaf level i has work $\frac{n2^i}{3^i}$, and the leaf level has work $2^{\log_3(n)}$. Putting this together, we have:

$$\begin{aligned}
\sum_{i=0}^{\log_3(n)-1} \left(\frac{n2^i}{3^i}\right) + 2^{\log_3(n)} &= n \sum_{i=0}^{\log_3(n)-1} \left(\frac{2}{3}\right)^i + n^{\log_3(2)} \\
&= n \left(\frac{1 - \left(\frac{2}{3}\right)^{\log_3(n)}}{1 - \frac{2}{3}}\right) + n^{\log_3(2)} && \text{By finite geometric series} \\
&= 3n \left(1 - \left(\frac{2}{3}\right)^{\log_3(n)}\right) + n^{\log_3(2)} \\
&= 3n \left(1 - \frac{n^{\log_3(2)}}{n}\right) + n^{\log_3(2)} \\
&= 3n - 3n^{\log_3(2)} + n^{\log_3(2)} \\
&= 3n - 2n^{\log_3(2)}
\end{aligned}$$

3. Output Complexity and Runtime Complexity

Consider the function h:

```
h(n) {
   if (n <= 1) {
      return 1
   } else {
      return h(n/2) + n + 2*h(n/2)
   }
}
```

(a) Find a recurrence $T(n)$ modeling the *worst-case runtime complexity* of $h(n)$.

Solution:

$$T(n) = \begin{cases} 1 & \text{if } n \leq 1 \\ 2T\left(\frac{n}{2}\right) + 1 & \text{otherwise} \end{cases}$$

(b) Find a closed form to your answer for (a).

Solution:

The recursion tree has height $\lg(n)$, each non-leaf level i has has work 2^i, and the leaf level has work $2^{\lg(n)}$. Putting this together, we have:

$$\begin{aligned} \left(\sum_{i=0}^{\lg n - 1} 2^i\right) + 2^{\lg(n)} = \left(\sum_{i=0}^{\lg n - 1} 2^i\right) + n &= \frac{1 - 2^{\lg n - 1 + 1}}{1 - 2} + n \\ &= 2^{\lg n} - 1 + n \\ &= (n-1) + n \\ &= 2n - 1 \end{aligned}$$

(c) Find an exact recurrence for the *output* of $h(n)$.

Solution:

$$h(n) = \begin{cases} 1 & \text{if } n \leq 1 \\ 3h\left(\frac{n}{2}\right) + n & \text{otherwise} \end{cases}$$

(d) Find a closed form to your answer for (c).

Solution:

The recursion tree has height $\lg(n)$, each non-leaf level i has has work $\left(\frac{3}{2}\right)^i \cdot n$, and the leaf level has work $1 \cdot 3^{\lg(n)}$. Putting this together, we have:

$$\begin{aligned} \sum_{i=0}^{\lg n - 1} \left(\frac{3}{2}\right)^i \cdot n + 1 \cdot 3^{\lg(n)} = n \sum_{i=0}^{\lg n - 1} \left(\frac{3}{2}\right)^i + 3^{\lg(n)} &= n \left(\frac{1 - \left(\frac{3}{2}\right)^{\lg n - 1 + 1}}{1 - \frac{3}{2}}\right) + 3^{\lg(n)} \\ &= -2n \left(1 - \left(\frac{3}{2}\right)^{\lg n}\right) + 3^{\lg(n)} \\ &= 2n \cdot 3^{\lg n} \cdot \left(\frac{1}{2}\right)^{\lg n} - 2n + 3^{\lg(n)} \\ &= 2n \cdot 3^{\lg n} \cdot \frac{1}{n} - 2n + 3^{\lg(n)} \\ &= 3 \cdot n^{\lg 3} - 2n \end{aligned}$$