CSE 332: Data Abstractions

Section 2: Heaps, Asymptotics, & Recurrences Solutions

0. Heaps

Insert 1, 3, 2, 4, 6, 8, 7, 5 into a *min heap*.
Now, insert the same values into a *max heap*.
Now, insert the same values into a *min heap*, but use Floyd's `buildHeap` algorithm.

1. Big-Oh Proofs

For each of the following, prove that $f \in \mathcal{O}(g)$.

(a) $f(n) = 7n$ $\qquad$ $g(n) = \dfrac{n}{10}$

Solution: Choose $c = 70$, $n_0 = 1$. Then, note that $7n = \frac{70n}{10} \leq 70\left(\frac{n}{10}\right)$ for all $n \geq 1$. So, $f(n) \in \mathcal{O}(g(n))$.

(b) $f(n) = 1000$ $\qquad$ $g(n) = 3n^3$

Solution: Choose $c = 3$, $n_0 = 1000$. Then, note that $1000 \leq n \leq n^3 \leq 3n^3$ for all $n \geq 1000$. So, $f(n) \in \mathcal{O}(g(n))$.

(c) $f(n) = 7n^2 + 3n$ $\qquad$ $g(n) = n^4$

Solution: Choose $c = 14$, $n_0 = 1$. Then, note that $7n^2 + 3n \leq 7(n^4 + n^4) \leq 14n^4$ for all $n \geq 1$. So, $f(n) \in \mathcal{O}(g(n))$.

(d) $f(n) = n + 2n \lg n$ $\qquad$ $g(n) = n \lg n$

Solution: Choose $c = 3$, $n_0 = 1$. Then, note that $n + 2n \lg n \leq n \lg n + 2n \lg n = 3n \lg n$ for all $n \geq 1$. So, $f(n) \in \mathcal{O}(g(n))$.

2. Is Your Program Running? Better Catch It!

For each of the following, determine the asymptotic worst-case runtime in terms of n.

(a)

```
int x = 0;
for (int i = n; i >= 0; i--) {
   if ((i % 3) == 0) {
      break;
   }
   else {
      x += n;
   }
}
```

Solution: This is $\Theta(1)$, because n, $n-1$, or $n-2$ will be divisible by three. So, the loop runs at most 3 times.

(b)

```
int x = 0;
for (int i = 0; i < n; i++) {
   for (int j = 0; j < (n * n / 3); j++) {
      x += j;
   }
}
```

Solution:

$$\sum_{i=0}^{n-1} \sum_{j=0}^{n^2/3-1} 1 = \sum_{i=0}^{n} \frac{n^2}{3} = n\left(\frac{n^2}{3}\right) = \Theta(n^3)$$

(c)

```
int x = 0;
for (int i = 0; i <= n; i++) {
   for (int j = 0; j < (i * i); j++) {
      x += j;
   }
}
```

Solution:

$$\sum_{i=0}^{n} \sum_{j=0}^{i^2-1} 1 = \sum_{i=0}^{n} i^2 = \left(\frac{n(n+1)(2n+1)}{6}\right) = \Theta(n^3)$$

3. Induction Shminduction

Prove $\sum_{i=0}^{n} 2^i = 2^{n+1} - 1$ by induction on n.

Solution:

Let $P(n)$ be the statement "$\sum_{i=0}^{n} 2^i = 2^{n+1} - 1$" for all $n \in \mathbb{N}$. We prove $P(n)$ by induction on n.

Base Case. Note that $\sum_{i=0}^{0} 2^i = 0 = 2^0 - 1$. So, $P(0)$ is true.

Induction Hypothesis. Suppose $P(k)$ is true for some $k \in \mathbb{N}$.

Induction Step. Note that

$$\begin{aligned} \sum_{i=0}^{k+1} 2^i &= \sum_{i=0}^{k} 2^i + 2^{k+1} \\ &= 2^{k+1} - 1 + 2^{k+1} && \text{[By IH]} \\ &= 2^{k+2} - 1 \end{aligned}$$

Note that this is exactly $P(k+1)$.

So, the claim is true by induction on n.

4. The Implications of Asymptotics

For each of the following, determine if the statement is true or false.

(a) $f(n) \in \Theta((g(n)) \to f(n) \in \mathcal{O}(g(n))$

Solution:

This is true. By definition of $f(n) \in \Theta((g(n))$, we have $f(n) \in \mathcal{O}(g(n))$.

(b) $f(n) \in \Theta(g(n)) \to g(n) \in \Theta(f(n))$

Solution:

This is true. By definition of $f(n) \in \Theta(g(n))$, we have $f(n) \in \mathcal{O}(g(n))$ and $f(n) \in \Omega(g(n))$. So, there exist $n_0, n_1, c_0, c_1 > 0$ such that $f(n) \le c_0 g(n)$ for all $n \ge n_0$ and $f(n) \ge c_1 g(n)$ for all $n \ge n_1$. Define $n_2 = \max(n_0, n_1$ and note that both inequalities hold for all $n \ge n_2$. Then, dividing both sides by their constants, we have:

$$g(n) \ge \frac{f(n)}{c_0}$$

$$g(n) \le \frac{f(n)}{c_1}$$

So, we've found constants $\left(\frac{1}{c_0}, \frac{1}{c_1}\right)$ and a minimum n (n_2) that satisfy the definitions of Omega and Oh. It follows that $g(n) is \Theta(f(n))$.

(c) $f(n) \in \Omega((g(n) \to g(n) \in \mathcal{O}(f(n))$

Solution:

This is true. This is basically identical to the previous part (except we only have to do half the work).

5. Asymptotic Analysis

For each of the following, determine if $f \in \mathcal{O}(g)$, $f \in \Omega(g)$, $f \in \Theta(g)$, several of these, or none of these.

(a) $f(n) = \log n$ $\qquad$ $g(n) = \log\log n$

Solution: $f(n) \in \Omega(g(n))$

(b) $f(n) = 2^n$ $\qquad$ $g(n) = 3^n$

Solution: $f(n) \in \mathcal{O}(g(n))$

(c) $f(n) = 2^{2n}$ $\qquad$ $g(n) = 2^n$

Solution: $f(n) \in \Omega(g(n))$

6. Recurrences and Closed Forms

For each of the following code snippets, find a recurrence for the worst case runtime of the function, and then find a closed form for the recurrence.

(a) Consider the function f:

```
f(n) {
   if (n == 0) {
      return 1;
   }
   return 2 * f(n - 1) + 1;
}
```

- Find a recurrence for $f(n)$.

Solution:

$$T(n) = \begin{cases} c_0 & \texttt{if } n = 0 \\ T(n-1) + c_1 & \texttt{otherwise} \end{cases}$$

- Find a closed form for $f(n)$.

Solution:

Unrolling the recurrence, we get $T(n) = \underbrace{c_1 + c_1 + \cdots + c_1}_{n \text{ times}} + c_0 = c_1 n + c_0$.

7. Big-Oh Bounds for Recurrences

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

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

Solution:

There are n terms to unroll and each one is constant. This is $\Theta(n)$.

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

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