CSE 332: Data Structures and Parallelism

Section 2: Heaps, Asymptotics, & Recurrences

0. Heaps

Insert 10, 7, 15, 17, 12, 20, 6, 32 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) $f(n) = 7n$ $\qquad$ $g(n) = \dfrac{n}{10}$

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

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

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

2. Is Your Program Running? Better Catch It!

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

(a)

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

(b)

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

(c)

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

3. Induction Shminduction

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

4. The Implications of Asymptotics

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

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

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

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

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) $f(n) = \log n$ $\quad g(n) = \log \log n$

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

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

6. Recurrences and Closed Forms

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

Consider the function f:

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

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

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

7. Big-Oh Bounds for Recurrences

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

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

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

8. Hello, elloH, lleoH, etc.

Consider the following code:

```
p(L) {
   if (L == null) {
      return [[]];
   }
   List ret = [];

   int first = L.data;
   Node rest = L.next;

   for (List part : p(rest)) {
      for (int i = 0; i <= part.size()) {
         part = copy(part);
         part.add(i, first);
         ret.add(part);
      }
   }
   return ret;
}
```

(a) Find a recurrence *for the output complexity* of $p(L)$. Then, find a Big-Oh bound for your recurrence.

(b) Now, find a recurrence *for the time complexity* of $p(L)$,