CSE 332: Data Abstractions

Section 2: Asymptotics & Recurrences

0. Big-Oh Proofs

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

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

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

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

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

1. Asymptotics Disproof

Prove that $n^2 \notin \mathcal{O}(n)$.

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;
   }
}
```

(b)

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

(c)

```
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) $f(n) \in \Theta((g(n)) \rightarrow f(n) \in \mathcal{O}(g(n))$

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

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

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

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

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

6. Summations

For each of the following, find a closed form.

(a) $\sum_{i=0}^{n} i^2$

(b) $\sum_{i=0}^{\infty} x^i$

7. 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 $f(n)$.

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

(b) Consider the function g:

```
g(n) {
   if (n == 1) {
      return 1000;
   }
   if (g(n/3) > 5) {
      return 5 * g(n/3);
   }
   else {
      return 4 * g(n/3);
   }
}
```

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

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

8. Big-Oh Bounds for Recurrences

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

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

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

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

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

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

9. 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 $p(L)$. That is, if $|L| = n$, what is the size of the output list, in terms of n? Then, find a Big-Oh bound for your recurrence.

(b) Now, find a recurrence *for the time complexity* of $p(L)$, and a Big-Oh bound for this recurrence as well.

10. MULTI-pop

Consider augmenting a standard `Stack` with an extra operation:

`multipop(k)`: Pops up to k elements from the `Stack` and returns the number of elements it popped

What is the amortized cost of a series of `multipop`'s on a `Stack` assuming `push` and `pop` are both $\mathcal{O}(1)$?