CSE 332: Data Abstractions

Section 1: Recurrences, Amortized Analysis

0. Summations

For each of the following, find a closed form.

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

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

1. 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 $g(n)$.

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

3. 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.

4. 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 $\mathcal{O}(1)$?