Quickcheck 03: Solutions

Consider the following recursive function. You may assume that the input will be a multiple of 3.

```
public int test(int n) {
    if (n <= 0) {
        return 2;
    } else {
        int curr = 0;
        for (int i = 0; i < n; i++) {
            curr += 1;
        }
        return curr + test(n - 3);
    }
}
```

(a) Write a recurrence modeling the *worst-case runtime* of `test`. When counting operations, feel free to round to a simple function (e.g. you could use n^2 instead of $n^2 - n + 3$).

Solution:

If $n \leq 0$, we just check the if-statement and return, call this 1 operation. Otherwise we go through a for-loop with n iterations and make a recursive call on input $n - 3$. We get the following recurrence.

$$T(n) = \begin{cases} 1 & \text{When } n \leq 0 \\ n + T(n-3) & \text{Otherwise} \end{cases}$$

(b) Unfold the recurrence into a summation (for $n \geq 0$).

Solution:

We unfold the recurrence until we see a pattern.

$$\begin{aligned} T(n) &= n + T(n-3) \\ &= n + (n-3) + T(n-6) \\ &= n + (n-3) + (n-6) + T(n-9) \\ &= n + (n-3) + \cdots + (n-3i+3) + T(n-3i) \end{aligned}$$

To get a closed form, we need the last $T()$ call to be a base case, so set $i = n/3$ to get $T(0)$. Plugging into the formula we get:

$$T(n) = n + (n-3) + (n-6) + \cdots + 3 + 1$$

To get a nice formula, we'd like to rewrite this using summation notation. As we said before, each term is of the form $n - 3i$ for some i, and i. Keep the last $+1$ separate since it doesn't fit the pattern. We get the following summation. Setting i to go from 0 to $n/3 - 1$ makes sure the first term is n and the last term is 3, like we need.

$$T(n) = 1 + \sum_{i=0}^{n/3-1} n - 3i$$

(c) Simplify the summation into a closed form (for $n \geq 0$).

Solution:

When you have a summation with a plus or minus sign inside of it, it's often a good trick to split into two summations.

$$T(n) = 1 + \sum_{i=0}^{n/3-1} n - 3i = \sum_{i=0}^{n/3-1} n - 3 \sum_{i=0}^{n/3-1} i$$

. The first summation is just adding n to itself $n/3$ times, which is just $n(n/3)$. The second summation has a closed form we know. Plugging in that formula we get $(n/3-1)(n/3)/2$. Combining these facts we get

$$T(n) = 1 + \frac{n^2}{3} - \frac{n/3(n/3-1)}{2}$$

A "closed form", within the context of this class, is just any expression that does not contain a summation or is recursive. This means we can stop here without needing to further simplify the expression.

If you wanted to simplify further you could:

$$T(n) = 1 + \frac{n^2}{3} - \frac{n^2}{18} + \frac{n}{6} = \frac{5n^2}{18} + \frac{n}{6} + 1$$

The final big-Oh is $O(n^2)$.