Consider the following recurrence.

\[
T(1) = d \\
T(N) = T(N/2) + cN
\]

1. Visualize the recursion tree.
2. Give a closed-form solution for \(T(N)\) by unrolling the recurrence.
3. Simplify with a familiar summation.

**Q1:** Visualize the recursion tree.

**Q2:** Give a closed-form solution for \(T(N)\) by unrolling the recurrence.

**Q3:** Simplify with a familiar summation.

**Q1:** Give a recurrence relation for \(f_3\).

```c
5
static int f3(int n) {
    if (n <= 1)
        return 1;
    return f3(n-1) + f3(n-1); 
}
```

**Q1:** Give a recurrence relation for \(f_3\).

**Q:** What is the recursive work? What is the non-recursive work?

**Q:** Give a recurrence relation for \(f_3\).
Summing Over Levels

**Approach:** Identify a pattern and then sum over all recursive levels.

What is the value of the last term in the sum for $T(N)$?

$$T(N) = c + 2c + 4c + \cdots + d$$

Give a simple asymptotic runtime bound.

---

Runtime: $g_0$

Give best and worst case runtime. Assume $k(N)$ runs in constant time and returns a boolean.

```java
static int f3(int n) {
    if (n <= 1)
        return 1;
    return f3(n-1) + f3(n-1);
}
```

**Q1:** What is the value of the last term in the sum for $T(N)$?

**Q2:** Give a simple asymptotic runtime bound.

---

```java
static void g0(int N) {
    if (N == 0)
        return;
    g0(N / 2);
    if (k(N))
        g0(N / 2);
}
```

**Q1:** Give best and worst case runtime. Assume $k(N)$ runs in constant time and returns a boolean.