Etan Soclof

"Modeling Recurrence Relations": Drill #1

```
public static int f(int n) {
    if (n == 0) {
        return 0;
    }

    int result = 0;
    for (int i = 0; i < n; i++) {
        for (int j = 0; j < i; j++) {
            result++;
        }
    }
    return 5 * f(n / 2) + 3 * result + 2 * f(n / 2);
}
```

+2 (base case)

$+ n^2/2$

+ 2T(n/2)

6.1.

1.

$$T(n) = \begin{cases} 0 & \text{if } n \text{ is 0} \\ 2T(n/2) + n^2/2 & \text{otherwise} \end{cases}$$

2.

$$W(n) = \begin{cases} 0 & \text{if } n \text{ is 0} \\ 7W(n/2) + 3\sum_{i=0}^{n-1}\sum_{j=0}^{i-1} j & \text{otherwise} \end{cases}$$

$$W(n) = \begin{cases} 0 & \text{if } n \text{ is 0} \\ 7W(n/2) + 3\sum_{i=0}^{n-1} i(i-1)/2 & \text{otherwise} \end{cases}$$

```java
public int test(int n) {
    IDictionary<Integer, Integer> dict = new AvlDictionary<>();
    populate(n, dict);
    int counter = 0;
    for (int i = 0; i < n; i++) {
        counter += dict.get(i);
    }
    return counter;
}

private void populate(int k, IDictionary<Integer, Integer> dict) {
    if (k == 0) {
        dict.put(0, k);
    } else {
        for (int i = 0; i < k; i++) {
            dict.put(i, i);
        }
        populate(k / 2, dict);
    }
}
```

log(k)

klog(k)

P(k/2)

7.

$$P(n) = \begin{cases} logn & \text{if } n \text{ is 0} \\ P(n/2) + nlogn & \text{otherwise} \end{cases}$$

$$T(n) = P(n) + nlog(n)$$

Note: Technically, in the worst case, the tree will be of size $\sum_{i=0}^{n-1} \frac{n}{2^i}$ and therefore a put and get operation will cost $log(\sum_{i=0}^{n-1} \frac{n}{2^i})$. However, $\sum_{i=0}^{n-1} \frac{n}{2^i}$ is equivalent to n multiplied by a constant and therefore in the equations I have assumed the runtime of a put and get operation to be log(n).