

## CSE 332: Data Abstractions

### QuickCheck: Recurrences Solutions (due Thursday, July 2)

#### 0. Happening Happening Happening

Consider the following code:

```
1 f(n) {
2   if (n == 0) {
3     return 0;
4   }
5
6   int result = 0;
7   for (int i = 0; i < n; i++) {
8     for (int j = 0; j < i; j++) {
9       result += j;
10
11     }
12   }
13   return f(n/2) + result + f(n/2);
14 }
```

(a) Find a recurrence for the time complexity of  $f(n)$ .

#### Solution:

We look at the three separate cases (base case, non-recursive work, recursive work):

- The base case is  $\mathcal{O}(1)$ , because we only do a return statement
- The non-recursive work is  $\mathcal{O}(1)$  for the assignments and if tests and  $= \sum_{i=0}^n i = \frac{n(n+1)}{2}$  for the for loops.
- The recursive work is  $2T(n/2)$ .

Putting these together, we get:

$$T(n) = \begin{cases} 1 & \text{if } n = 1 \\ 2T(n/2) + \frac{n(n+1)}{2} & \text{otherwise} \end{cases}$$

(b) Find a Big-Oh bound for your recurrence.

#### Solution:

The recursion tree has  $\lg(n)$  height, and each level of the tree does  $\left(\frac{n^2}{2^i}\right)$  work.

Note that  $\sum_{i=0}^{\lg(n)} \left(\frac{1}{2}\right)^i < 2$ . So,  $T(n) \in \mathcal{O}(n^2 \lg(n))$ .