CSE 332: Data Structures and Parallelism

Recurrences Solutions

Happening Happening Happening

Consider the following code:

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

   int result = 0;
   for (int i = 0; i < n; i++) {
      for (int j = 0; j < i; j++) {
         result += j;

      }
   }
   return f(n/2) + result + f(n/2);
}
```

(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 constant time, because we only do a return statement
- The non-recursive work is constant time for the assignments and if tests and $\sum_{i=0}^{n-1}\sum_{j=0}^{i-1} c = \sum_{i=0}^{n-1} c * i = \frac{c * n(n-1)}{2} = \frac{cn^2}{2} - \frac{cn}{2}$ for the for loops.
- The recursive work is $2T(n/2)$.

Putting these together, we get:

$$T(n) = \begin{cases} c_0 & \texttt{if } n = 1 \\ 2T(n/2) + c_1 * n^2 + c_2 n + c_3 & \texttt{otherwise} \end{cases}$$

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

Solution:

Note that $c_1 n^2 + c_2 n + c_3 \in \mathcal{O}(n^2)$. Since we're only looking for a big-$\mathcal{O}$ bound, we dismiss the constants and lower-order terms in the non-recursive runtime for our analysis. The recursion tree has $lg(n)$ height, each node of the tree does $\left(\frac{n}{2^i}\right)^2$ work, and each level has 2^i nodes.

Note that the total work is then $n^2 \sum_{i=0}^{\lg(n)} 2^i \left(\frac{1}{2^i}\right)^2 = n^2 \sum_{i=0}^{\lg(n)} \left(\frac{2^i}{4^i}\right) < n^2 \sum_{i=0}^{\infty} \left(\frac{1}{2^i}\right) = \frac{n^2}{1-\frac{1}{2}} \in \mathcal{O}(n^2)$.

So, $T(n) \in \mathcal{O}(n^2)$.