CSE 332: Data Structures and Parallelism

Recurrences Solutions

Happening Happening Happening

Consider the following code:

```
f(n) {
   if (n <= 1) {
      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 $\mathcal{O}(1)$, 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:

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

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

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