Note: Monday 2/21 Lecture

...Or, What Dr. Hugue swapped and merged instead of compared.

Merge Sort Trio: 3 Different Solutions, all in $\Theta(n \log n)$

During Monday's lecture I was attempting to illustrate two points. In the process, my brain sneaked in one more point while talking about point 2. So actually, I initially had correct stuff on the board which corresponded to my initial comments. Then somehow, I launched point 3 below without finishing point 2. I apologize for any confusion and hope this little note helps.

1. The first point was that we have been asking you two kinds of questions about recursions: ones where you need to get an exact solution, and ones where you need only give or prove the asymptotic complexity of the recursion. This point is illustrated for each of the following examples.

2. The second point was that the initial value or *cost* of dealing with a single element in the recurrence, typically $T(1)$, need not be one. As long as it's a constant, it has no effect on the asymptotic complexity and only changes the exact solution by a constant multiple of n. See Example 2 below.

3. The third point had to do with the practical situation where programs normally do not use the 'divide and conquer' to decompose down to a single element.

   That is, it's more likely to halt the *divide* part of the merge sort process when the number of elements is some number larger than 1; thus, the recursion tree would not go all the way down to the $T(1)$ stage, but, rather, would stop when there were say, 2^5 or 2^4 elements left per set. So, the last row of the recursion tree would not correspond to executing n copies of $T(1)$, but rather, $n/(16)$ copies of $T(16)$. This reduces the number of rows remaining in the original tree by 4 (which is $\log 16$); so, the summation would be from 5 (which is $1 + \log 16$) to $\log n$.

   Example 3 gives a general version of this result in terms of $n = b$ as the value of n such that $T(b) = A$, for some postive constant A. To check your understanding, you should be able to get the exact solution for the example above by letting $b = 16$ and $T(16) = A$ in the solution to Example 3.

How should you read this note? What you should do is check all these items using the examples given below for the three slightly different formulations of the merge sort problem. You must make sure that at least points one and two above make sense. Several readings might be needed. Then tackle point three.

Here are the specific details, most of which were on the board, but lost some of the glue required to stick the correct parts together. Essentially, each has an exact solution which can be expressed as the sum of $n \log n$ and cn for some positive constant c. That means each is in $\Theta(n \log n)$.

Example 1. Standard Merge Sort

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

Exact Solution: $T(n) = n \log n + n$

Asymptotic Complexity: $\Theta(n \log n)$

Work for Exact solution of Example 1 from adding column of Recursion Tree:

$$T(n) = T(1)n + \sum_{i=1}^{\log n} n \equiv n + n \sum_{i=1}^{\log n} 1 \equiv n \log n + n$$

Example 2. Merge-Sort with Constant Cost A for $n = 1$

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

Exact Solution: $T(n) = n \log n + An$

Asymptotic Complexity: $\Theta(n \log n)$

Work for Exact solution from column of Recursion Tree for Example 2.

$$T(n) = T(1)n + n \sum_{i=1}^{\log n} 1 = An + n \log n$$

Example 3. Early Terminating Merge-Sort, with Constant Cost A for Constant Minimim Number of Elements $n = b$

$$T(n) = \begin{cases} A & \text{if } n = b, \\ 2T(n/2) + n & \text{otherwise.} \end{cases}$$

Exact Solution: $T(n) = ((A/b) - \log b)n + nlogn$

Asymptotic Complexity: $\Theta(n \log n)$

Work for Exact solution of Example 3 from column of Recursion Tree.

$$T(n) = T(b)(n/b) + n \sum_{i=(\log b+1)}^{\log n} 1 \equiv (A/b)n + n(\log n - (\log b + 1) + 1) \equiv ((A/b) - \log b)n + n \log n.$$