Divide and conquer relations (Akra-Bazzi or Master theorem)

In divide and conquer algorithms [like merge sort], the problem is divided into smaller subproblems, each subproblem is solved recursively and a combine algorithm is used to obtain the final solution from solutions of subproblems. Assume there are a subproblems, each of size $\frac{1}{b}$ of the original problem and the algorithm to combine subproblems take polynomial time cn^k for some constants a, b, c, and k. Then we have recurrence $T(n) = aT(\frac{n}{b}) + cn^k$. By an expanding method very similar to $a_n = 2a_{\frac{n}{2}} + \dots$ (that we see previous time) we can prove the following theorem [it is a good exercise to prove it for yourself or at least see it in Manber's book]

Thm1: If $T(n) = aT(\frac{n}{b}) + cn^k$ for integers $a \geq 1, b \geq 2$ and positive constants c and k

$$T(n) = \begin{cases} 1) O(n^{\log_b a}) & \text{if } a > b^k \\ 2) O(n^k \log n) & \text{if } a = b^k \\ 3) O(n^k) & \text{if } a < b^k \end{cases}$$

E.g. for $T(n) = 2T(\frac{n}{2}) + n$, $a=2, b=2, c=1, k=1$ and thus case 2) applies since $a = b^k$. Thus $T(n) = O(n \log n)$.

This formula is very useful in many divide-and conquere algorithms and you should memorize it in this course.

A theorem which is a bit more general, i.e, instead of cn^k considers arbitrary $f(n)$ and obtains Θ instead of O, is called Master theorem, see e.g. in wikipedia. (Again conditioned on $f(n)$ we have three cases).

A Theorem which is a bit more general and gives a one formula instead of three cases above is called Akra-Bazzi theorem. In our class we always approximate $f(n)$ with appropriate cn^k and use Thm 1 instead and we obtain Θ usually (and not just O)

Recurrence Relations with Full History:

A full-history recurrence relation is one that depends on all the previous functions. We use the method of elimination of history, in which we will try to write the recurrence in such a way that most of the terms will be cancelled (we used it in computing sums before)

$T(n) = c + \sum_{i=1}^{n-1} T(i)$; $(T(1)=c)$ Now if we write $T(n+1) = c + \sum_{i=1}^{n} T(i)$ and subtract the former from the latter, we have $T(n+1) - T(n) = T(n)$,

So $T(n+1) = 2T(n)$ and $T(n+1) = T(1)\,2^n = c2^n$ as we saw in the previous session

Thus $T(n) = O(2^n)$ (exponential)

let's see a bit more complicated example:

$T(n) = n - 1 + \frac{2}{n}\sum_{i=1}^{n-1} T(i)$ for $n \geq 2$, $T(1) = 0$ [we use it in the analysis of quick sort]

Let's see $T(n+1) = (n+1) - 1 + \frac{2}{n+2}\sum_{i=1}^{n} T(i)$, we multiply both sides by n and $(n+1)$ [respectively]

$$nT(n) = n(n-1) + 2\sum_{i=1}^{n-1} T(i)$$

$$(n+1)T(n+1) = (n+1)n + 2\sum_{i=1}^{n} T(i)$$

now subtract to get $(n+1)T(n+1) - nT(n)$

$$= (n+1)n - n(n-1) + 2T(n) = 2n + 2T(n)$$

which implies $T(n+1) = \frac{2n + (n+2)T(n)}{n+1}$ $\boxed{\frac{2n}{n+1} \leq 2}$ $\leq 2 + \frac{n+2}{n+1}T(n)$

$$\leq 2 + \frac{n+1}{n}\left[2 + \frac{n}{n-1}T(n-1)\right] \leq 2 + \frac{n+1}{n}\left[2 + \frac{n}{n-1}\left[2 + \frac{n-1}{n-2}\left[\cdots \frac{4}{3}\right]\right]\right]$$

$$= 2 + 2\frac{n+1}{n} + 2\frac{n+1}{n}\frac{n}{n-1} + 2\frac{n+1}{n}\frac{n}{n-1}\frac{n-1}{n-2} + \cdots + 2\frac{n+1}{n}\cdot\frac{n}{n-1}\cdot\frac{n-1}{n-2}\cdots\frac{4}{3}$$

$$= 2 + 2\frac{n+1}{n} + 2\frac{n+1}{n-1} + 2\frac{n+1}{n-2} + \cdots + 2\frac{(n+1)}{3} = 2(n+1)\left[\frac{1}{n+1} + \frac{1}{n} + \frac{1}{n-1} + \cdots + \frac{1}{3}\right]$$

$= 2(n+1)\left(H(n+1) - \frac{3}{2}\right)$ where $H_n = 1 + \frac{1}{2} + \frac{1}{3} + \cdots \frac{1}{n}$ is called the Harmonic series and it is easy to see that (sug by Integration) $H_n = \ln n + 0.577 + O(\frac{1}{n})$

Thus $T(n)$ is $O(n \ln n)$.