

Questions

1. Solve the recurrence , where $t(0) = 1$,

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

2. Find a (non-trivial) upper bound on the recurrence

$$t(n) = 5t\left(\frac{n}{5}\right) + 2n$$

with $t(1) = 1$.

Hint: let $N = 5^{\lceil \log_5 n \rceil}$. That is, N is the smallest power of 5 that is greater than or equal to n . Then, find an upper bound on the recurrence that is in terms of N .

3. When we considered the mergesort algorithm in class, we had a base case of $n = 1$. What if we had stopped the recursion at a larger base case? For example, suppose that if the list size is 4 or less, then then you run insertion sort instead of mergesort. Since insertion sort is $O(n^2)$, one might ask whether this would change the time required to run mergesort. Does it? Does mergesort become $O(n^2)$ if you stop the recursion at $n = 4$ (or less) and use an n^2 algorithm for this case?

Answers

1. Note this is similar to the Tower of Hanoi recurrence but now we have two operations for each call. Does the result change? Let's see.

$$\begin{aligned}
 t(n) &= 2(2t(n-2) + 2) + 2 \\
 &= 2^2t(n-2) + 2^2 + 2^1 \\
 &= 2^2(2t(n-3) + 2) + 2^2 + 2^1 \\
 &= 2^3t(n-3) + 2^3 + 2^2 + 2^1 \\
 &= \dots \\
 &= 2^n t(n-n) + \sum_{i=1}^n 2^i \\
 &= 2^n t(0) + \sum_{i=0}^n 2^i - 1 \\
 &= 2^n + 2^{n+1} - 2 \\
 &= 3 \cdot 2^n - 2
 \end{aligned}$$

which is $O(2^n)$ so, no, the $O()$ behavior is the same as with Tower of Hanoi.

2.

$$\begin{aligned}
 t(n) &\leq 5t\left(\frac{N}{5}\right) + 2N \\
 &= 5^2t\left(\frac{N}{5^2}\right) + 2N + 2N \\
 &\dots \\
 &= 5^i t\left(\frac{N}{5^i}\right) + \underbrace{2N + 2N + \dots + 2N}_{i \times} \\
 &\dots \\
 &= 5^{\log_5 N} t(1) + \underbrace{2N + 2N + \dots + 2N}_{\log_5 N \times} \\
 &= N + 2N \log_5 N
 \end{aligned}$$

But $N < 5n$, so

$$t(n) \leq 5n + 10n \log_5(5n)$$

Moreover, $\log_5(5N) = 1 + \log_5 N$, and $\log_5 x = \log_5 2 \cdot \log_2 x$, and so

$$t(n) \leq 5n + 10n(1 + \log_5 n) = 5n + 10n(1 + \log_5 2 \cdot \log_2 n)$$

which you can show is $O(n \log_2 n)$.

[Note: if you find it difficult to remember that

$$\log_a x \cdot \log_b a = \log_b x ,$$

then try to remember the “trick” for deriving it, namely:

$$a^{\log_a x} = x$$

and so

$$\log_b(a^{\log_a x}) = \log_b x$$

and so

$$\log_a x \cdot \log_b a = \log_b x.$$

3. Assume n is a power of 2 (to simplify the argument). We would have

$$\begin{aligned} t(n) &= 2t(n/2) + cn \\ &= 2(2t(n/4) + cn/2) + cn \\ &= 4t(n/4) + c(n+n) \\ &= 8t(n/8) + c(n+n+n), \text{ by similar back substitution} \\ &= 16t(n/16) + c(n+n+n+n) \text{ by similar back substitution} \\ &= \dots \\ &= \frac{n}{4} t(4) + cn(\log n - 2) \quad (*) \end{aligned}$$

Here we have stopped two levels above the base case. (If we had continued to the base case, then we would have had

$$t(n) = nt(1) + cn \log n \quad)$$

But $t(n) = \frac{n}{4} t(4) + cn(\log n - 2)$ is still $O(n \log n)$. (Note: $t(4)$ is a constant. It does not depend on n .)