Iteration Method for Solving Recurrences

In this method, we first convert the recurrence into a summation. We do so by iterating the recurrence until the initial condition is reached. In the example given in the previous chapter, $T(1)$ was the time taken in the initial condition. For converting the recurrence of the previous example into summation, we would first break $T(n)$ into $T(\frac{n}{2})$ and then into $T(\frac{n}{4})$ and so on. Let's take this example and see how to apply this method.

In the previous example, we had our equation in the form:

$$T(n) = \begin{cases} T(1), & n = 1 \\ c + T\left(\frac{n}{2}\right), & \text{if } n > 1 \end{cases} \tag{1}$$

where, c is a constant

</>
We have replaced $\Theta(1)$ from the previous equation with c because $\Theta(1)$ means constant time.

Let's replace n with $n/2$ in the previous equation.

$$T\left(\frac{n}{2}\right) = c + T\left(\frac{n}{4}\right) \tag{2}$$

where, c is a constant

Now, put the value of $T\left(\frac{n}{2}\right)$ from $eq(2)$ in the $eq(1)$, we get:

$$T(n) = c + T\left(\frac{n}{2}\right)$$

$$=> T(n) = c + \underbrace{\left(c + T\left(\frac{n}{4}\right)\right)}_{\text{value of } T\left(\frac{n}{2}\right)} \tag{3}$$

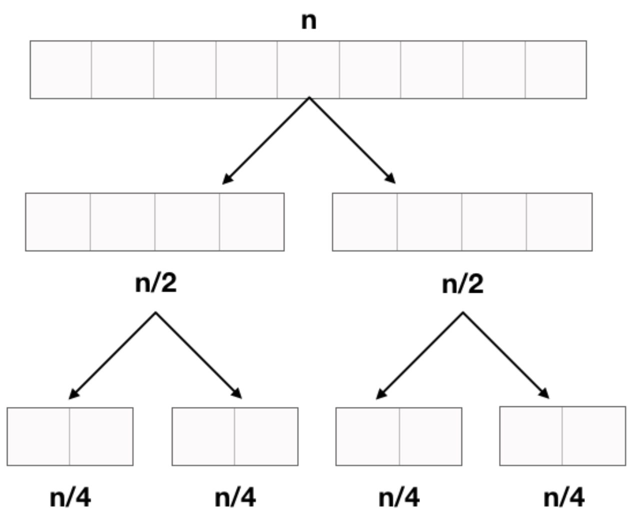
Again, let's use $T\left(\frac{n}{4}\right)$ in place of n in the $eq(1)$.

$$T\left(\frac{n}{4}\right) = c + T\left(\frac{n}{8}\right)$$

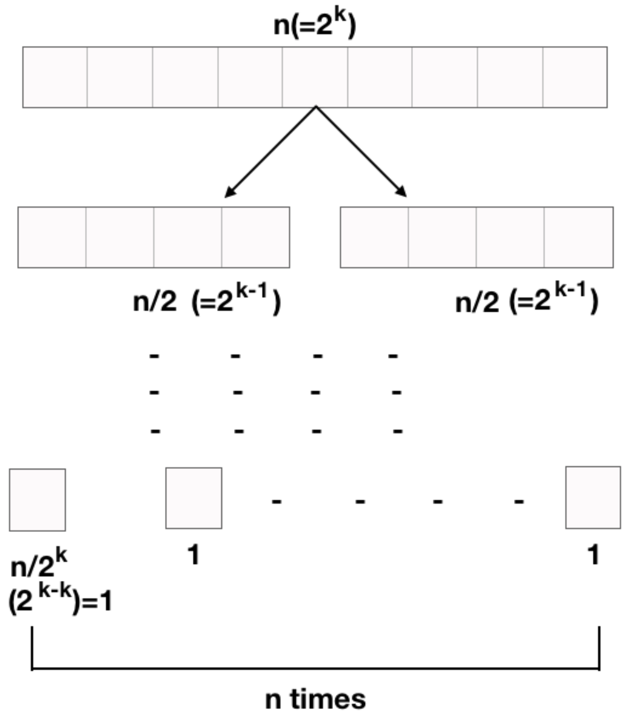
Putting this value in $eq(3)$, we get,

$$T(n) = c + c + c + T\left(\frac{n}{8}\right) \tag{4}$$

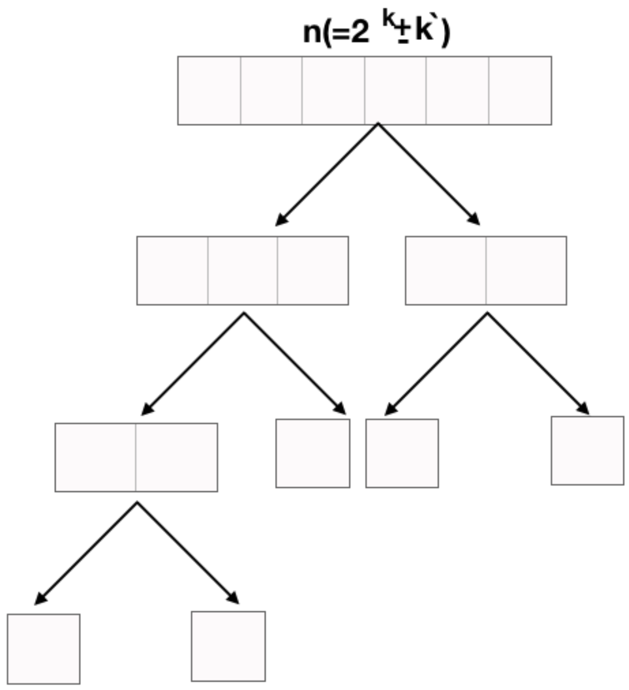
We are breaking our problem into two subproblems of half size as mentioned in the picture given below.



So, we can assume that n is in the form of 2^k.



It might be possible that n is not exactly in the form of 2^k but $2^k \pm k_o$, where k_o is some constant.



But we are concerned about the rate of the growth only and this approximation is going to give us that. So, we will proceed by assuming that n is in the form of 2^k.

Recalling the $eq(4)$,

$$T(n) = c + c + c + T\left(\frac{n}{8}\right)$$

We can also write the above equation as

$$T(n) = 3c + T\left(\frac{n}{2^3}\right)$$

Similarly, we can write

$$T(n) = \underbrace{c + c + \ldots + c}_{k \text{ times}} + T\left(\frac{n}{2^k}\right)$$

$$\text{Since } n = 2^k,$$

$$T(n) = \underbrace{c + c + \ldots + c}_{k \text{ times}} + T\left(\frac{2^k}{2^k}\right)$$

$$= kc + T(1) \qquad (5)$$

As mentioned above, $T(1)$ is the base case. So, we have reached the base case.

Also,

$$n = 2^k$$

$$=> \log_2(n) = log_2(2^k)$$

$$=> \lg(n) = k$$

Replacing the value of k in the $eq(5)$, we get

$$T(n) = c\lg(n) + T(1)$$

$$= \Theta(\lg(n))$$

Let's take a brief recap of what we did here. We first formed the recurrence equation from the code or the algorithm and then we solved that equation using the iteration method. In this method, we broke down the equation $\left(n \rightarrow \frac{n}{2} \rightarrow \frac{n}{4} \rightarrow \frac{n}{8} \rightarrow \ldots\right)$. And after reaching the base case, we back-substituted the equation (value of k) to express the equation in the form of n and initial boundary condition

Let's look at one more example of this method.

$$T(n) = n + 2T\left(\frac{n}{2}\right) \tag{6}$$

In this example also, $T(n)$ is broken down into $T\left(\frac{n}{2}\right)$, so let's replace n with $\frac{n}{2}$ in the $eq(6)$.

$$T(\frac{n}{2}) = \frac{n}{2} + 2T\left(\frac{n}{4}\right)$$

Putting this value in $eq(6)$,

$$T(n) = n + 2\left(\frac{n}{2} + 2T\left(\frac{n}{4}\right)\right)$$

$$= n + n + 4T\left(\frac{n}{4}\right)$$

Again, putting $\frac{n}{4}$ instead of n in the $eq(6)$ and getting the value of $T\left(\frac{n}{4}\right)$.

$$= n + n + 4\underbrace{\left(\frac{n}{4} + 2\left(\frac{n}{8}\right)\right)}_{\text{value of } T\left(\frac{n}{4}\right)}$$

$$= n + n + n + 8T\left(\frac{n}{8}\right) \tag{7}$$

Let's assume n is of the form of 2^k i.e., $n = 2^k$ as we did in the previous example.

For 3 terms, $eq(7)$ can be written as

$$= 3n + 2^3T\left(\frac{n}{2^3}\right)$$

Similarly, for k terms, we can write,

$$= kn + 2^kT\left(\frac{n}{2^k}\right)$$

Replacing n with 2^k,

$$= kn + 2^kT\left(\frac{2^k}{2^k}\right)$$

$$= kn + 2^kT(1)$$

$$= kn + nT(1)$$

Replacing k with $\lg(n)$ ($n = 2^k => \lg(n) = k$)

$$= n\lg(n) + nT(1)$$

Since, $n\lg(n)$ is faster growing than n

$$T(n) = \Theta(nlg(n))$$

So, we have obtained the growth order of both the algorithms discussed in the previous chapters. As stated earlier, we are going to learn two more methods to solve recurrences in the next chapters.

❝ Furious activity is no substitute for analytical thought. ❞

- Alastair Pilkington

PREV (/course/algorithms-recurrences/) (/course/algorithms-now-the-recursion/) NEXT

New Questions

setting up an ide for mac.. - **Cpp**

(/discussion/setting-up-an-ide-for-mac)

Please fill the blanks and help me am stuck - **Java**

(/discussion/fill-the-blanks-and-help-me-am-stuck)

Time complexity of the Python Function - **Python**

(/discussion/time-complexity-of-the-python-function)

How I Can find The Best Target Coupons & Latest Deals Offers? - **Other**

(/discussion/how-i-can-find-the-best-target-coupons-latest-deal)

To Calculate salaries - **Java**

(/discussion/to-calculate-salaries)

How to write a c++ program that will declares a two dimensional array say Char My element [4][5], prompt the user to initialize and display the elemen - **C**

(/discussion/how-to-write-a-c-program-that-will-declares-a-two-)

What is the difference between a local variable and an instance variable? - **Java**

(/discussion/what-is-the-difference-between-a-local-variable-an)

Ask Yours

(/add_question/)