# Recurrence Relations

**In analyzing the Towers of Hanoi, we might want to know how many moves it will take. Let T(n) stand for the number of moves it takes to solve the Towers problem for n disks. Then, we have the following formula:**

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

**This is because in order to move a tower of n disks, we first move a tower of n-1 disks, which takes T(n-1) moves. Then we move the bottom disk (this is the +1 above), and then we move a tower of n-1 disks again, which takes us T(n-1) moves again.**

**Simplifying, we get:**

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

**Unfortunately, this isn't terribly helpful to us, because it's not a formula in terms of n.**

**To get a formula in terms of n, we will use the iteration technique, which simply utilizes the fact that the formula above is true for all positive integers n. We will also use the fact that T(1) = 1, since it takes one move to move a tower of one disk.**

# Iterating to Solve the Recurrence

**$T(n) = \underline{2T(n-1) + 1}$**

**$= 2[2T(n-2) + 1] + 1$, because $T(n-1) = 2T(n-2) + 1$.**

**$= 4T(n-2) + 2 + 1$**

**$= \underline{4T(n-2) + 3}$**

**$= 4[2T(n-3) + 1] + 3$, because $T(n-2) = 2T(n-3) + 1$**

**$= 8T(n-3) + 4 + 3$**

**$= \underline{8T(n-3) + 7}$**

**The three underlined steps indicate the three iterations in our work. A pattern should emerge from these three steps. The numbers in front of T(…) are successive powers of two. The number inside the T is n – k, where k is which power of k. Finally the number at the end is one less than the same power of two. Thus, we can conjecture that**

**$= 2^kT(n – k) + 2^k – 1.$**

**Finally, we want to plug in a value of k into this expression so that we can evaluate T(n – k). This we know T(1), we want n – k = 1. Equivalently, k = n – 1.**

**Plug in k = n – 1 into our formula:**

**$= 2^{n-1}T(1) + 2^{n-1} – 1 = 2^{n-1} + 2^{n-1} – 1 = 2^n – 1.$**

**This solution is exact because we made no simplifications with respect to order notation. It's also true that $T(n) = O(2^n)$.**

## Binary Search Recurrence Relation

**Another recurrence that arises from the analysis of a recursive program is the following recurrence from binary search:**

**$T(n) = T(n/2) + 1$, since a binary search over n elements uses a comparison, and then a recursive call to an array of size n/2.**

**We use iteration:**

**$T(n) = \underline{T(n/2) + 1}$**

**$= (T(n/4) + 1) + 1$**

**$= \underline{T(n/4) + 2}$**

**$= (T(n/8) + 1) + 1$**

**$= \underline{T(n/8) + 3}$**

**We should see the pattern here and conjecture:**

**$= T(n/2^k) + k.$**

**We want a value of k that makes $n/2^k = 1$. This means that $n = 2^k$. By the definition of the logarithm, we have $k = \log_2 n$. Plugging in, we get:**

**$= T(1) + \log_2 n$**

**$= 1 + \log_2 n$, since a binary search of 1 element takes 1 step.**

**Thus the worst case run time of binary search is O(lg n).**

## Merge Sort Recurrence Relation

**Let's analyze one last recurrence using this technique, the recurrence relation we derived for Merge Sort in a prior lecture:**

**$T(n) = 2T(n/2) + O(n)$, $T(1) = 1$.**

**For now, we'll write n in place of O(n), but keep in our minds that n really means some constant times n.**

**$T(n) = \underline{2T(n/2) + n}$**

**$= 2[\ 2T(n/4) + n/2\ ] + n$, since $T(n/2) = 2T(n/4) + n/2$**

**$= 4T(n/4) + n + n$**

**$= \underline{4T(n/4) + 2n}$**

**$= 4[\ 2T(n/8) + n/4\ ] + 2n$, since $T(n/4) = 2T(n/8) + n/4$**

**$= 8T(n/8) + n + 2n$**

**$= \underline{8T(n/8) + 3n}$**

**$= 2^kT(n/2^k) + kn.$**

**Once again we want to set $k = \log_2 n$.**

**$= nT(1) + n(\log_2 n)$**

**$= n\log_2 n + n$**

**Thus, the solution to this recurrence is O(nlgn), since the n in front of the lg n wasn't literally n but just O(n).**

# Example Recurrence Relations

1. (Aug 2018 Foundation Exam) Use the iteration technique to solve the following recurrence relation in terms of n:

$$T(n) = 3T(n-1) + 1, for\ all\ integers\ n > 1$$
$$T(1) = 1$$

Please give an exact closed-form answer in terms of n, instead of a Big-Oh answer.

(Note: A useful summation formula to solve this question is $\sum_{i=0}^{n} x^i = \frac{x^{n+1}-1}{x-1}$.)

$$\boldsymbol{T(n) = 3T(n-1) + 1}$$

$$\boldsymbol{= 3(3T(n-2) + 1) + 1}$$

$$\boldsymbol{= 9T(n-2) + 3 + 1}$$

$$\boldsymbol{= 9(3T(n-3) + 1) + 3 + 1}$$

$$\boldsymbol{= 27T(n-3) + 9 + 3 + 1}$$

**After k steps, we have:** $\boldsymbol{= 3^k T(n-k) + \sum_{i=0}^{k-1} 3^i}$

**Let k = n-1, then we have that** $\boldsymbol{T(n) = 3^{n-1}T\big(n-(n-1)\big) + \sum_{i=0}^{n-2} 3^i}$

$$\boldsymbol{= 3^{n-1}T(1) + \sum_{i=0}^{n-2} 3^i}$$

$$\boldsymbol{= 3^{n-1} + \sum_{i=0}^{n-2} 3^i}$$

$$\boldsymbol{= \sum_{i=0}^{n-1} 3^i}$$

$$\boldsymbol{= \frac{3^n - 1}{3 - 1} = \frac{3^n - 1}{2}}$$

2. (Jan 2018 Foundation Exam) Using the iteration technique, find a tight Big-Oh bound for the recurrence relation defined below:

$$T(n) = 3T\left(\frac{n}{2}\right) + n^2, \text{ for } n > 1$$
$$T(1) = 1$$

Hint: You may use the fact that $\sum_{i=0}^{\infty}(\frac{3}{4})^i = 4$ and that $3^{log_2 n} = n^{log_2 3}$, and that $log_2 3 < 2$.

**Iterate the given recurrence two more times:**

$$\boldsymbol{T(n) = 3T\left(\frac{n}{2}\right) + n^2}$$
$$\boldsymbol{T(n) = 3(3T\left(\frac{n}{4}\right) + (\frac{n}{2})^2) + n^2}$$
$$\boldsymbol{T(n) = 9T\left(\frac{n}{4}\right) + \frac{3n^2}{4} + n^2}$$
$$\boldsymbol{T(n) = 9T\left(\frac{n}{4}\right) + n^2(1 + \frac{3}{4})}$$
$$\boldsymbol{T(n) = 9(3T\left(\frac{n}{8}\right) + (\frac{n}{4})^2) + n^2(1 + \frac{3}{4})}$$
$$\boldsymbol{T(n) = 27T\left(\frac{n}{8}\right) + \frac{9n^2}{16} + n^2(1 + \frac{3}{4})}$$
$$\boldsymbol{T(n) = 27T\left(\frac{n}{8}\right) + n^2(1 + \frac{3}{4} + \frac{9}{16})}$$

**In general, after the k[th] iteration, we get the recurrence**

$$\boldsymbol{T(n) = 3^k T\left(\frac{n}{2^k}\right) + n^2(\sum_{i=0}^{k-1}(\frac{3}{4})^i)}$$

**To solve the recurrence, find k such that $\frac{n}{2^k} = 1$. This occurs when $n = 2^k$ and $k = log_2 n$. Plug into the equation above for this value of k to get:**

$$\boldsymbol{T(n) = 3^{logn}T(1) + n^2\left(\sum_{i=0}^{k-1}\left(\frac{3}{4}\right)^i\right) \leq 3^{logn} + n^2\left(\sum_{i=0}^{\infty}\left(\frac{3}{4}\right)^i\right) = n^{log_2 3} + 4n^2 = O(n^2)}$$

3. (May 2017 Foundation Exam) Find the Big-Oh solution to the following recurrence relation using the iteration technique. Please show all of your work, including 3 iterations, followed by guessing the general form of an iteration and completing the solution. Full credit will only be given if all of the work is accurate (and not just for arriving at the correct answer.)

$$T(n) = 2T\left(\frac{n}{2}\right) + n^2, T(1) = 1$$

**First, iterate three times:**

$$\boldsymbol{T(n) = 2T\left(\frac{n}{2}\right) + n^2}$$
$$\boldsymbol{= 2\left[2T\left(\frac{n}{4}\right) + \left(\frac{n}{2}\right)^2\right] + n^2}$$
$$\boldsymbol{= 2\left[2T\left(\frac{n}{4}\right) + \frac{n^2}{4}\right] + n^2}$$
$$\boldsymbol{= 4T\left(\frac{n}{4}\right) + \frac{n^2}{2} + n^2}$$
$$\boldsymbol{= 4T\left(\frac{n}{4}\right) + \frac{3n^2}{2}}$$
$$\boldsymbol{= 4[2T\left(\frac{n}{8}\right) + (\frac{n}{4})^2] + \frac{3n^2}{2}}$$
$$\boldsymbol{= 4[2T\left(\frac{n}{8}\right) + \frac{n^2}{16}] + \frac{3n^2}{2}}$$
$$\boldsymbol{= 8T\left(\frac{n}{8}\right) + \frac{n^2}{4} + \frac{3n^2}{2}}$$
$$\boldsymbol{= 8T\left(\frac{n}{8}\right) + \frac{7n^2}{4}}$$

**In general, after k iterations we will have $T(n) = 2^k T\left(\frac{n}{2^k}\right) + \frac{(2^k-1)n^2}{2^{k-1}}$. We want to plug in the value of k for which $\frac{n}{2^k} = 1$,which is when $n = 2^k$. Note that for this value of k, $2^{k-1} = n/2$, since $2 \times 2^{k-1} = 2^k$:**

$$\boldsymbol{T(n) = nT(1) + \frac{(n-1)n^2}{\frac{n}{2}} = n(1) + 2n(n-1) = 2n^2 - n = O(n^2)}$$

4. (Jan 2017 Foundation Exam) Find the Big-Oh solution to the following recurrence relation using the iteration technique. Please show all of your work, including 3 iterations, followed by guessing the general form of an iteration and completing the solution. Full credit will only be given if all of the work is accurate (and not just for arriving at the correct answer.)

$$T(n) = 4T\left(\frac{n}{2}\right) + n, T(1) = 1$$

**First, iterate three times:**

$$\begin{aligned} \boldsymbol{T(n)} &= \boldsymbol{4T\left(\frac{n}{2}\right) + n} \\ &= \boldsymbol{4\left[4T\left(\frac{n}{4}\right) + \frac{n}{2}\right] + n} \\ &= \boldsymbol{16T\left(\frac{n}{4}\right) + 2n + n} \\ &= \boldsymbol{16T\left(\frac{n}{4}\right) + 3n} \\ &= \boldsymbol{16\left[4T\left(\frac{n}{8}\right) + \frac{n}{4}\right] + 3n} \\ &= \boldsymbol{64T\left(\frac{n}{8}\right) + 4n + 3n} \\ &= \boldsymbol{64T\left(\frac{n}{8}\right) + 7n} \end{aligned}$$

**In general, after k iterations, we will have $T(n) = 4^k T\left(\frac{n}{2^k}\right) + (2^k - 1)n$. We want to plug in the value of k for which $\frac{n}{2^k} = 1$,which is when $n = 2^k$. In this case, note that $n^2 = (2^k)^2 = 2^{2k} = 4^k$. Plugging in, we find:**

$$\begin{aligned} \boldsymbol{T(n)} &= \boldsymbol{n^2 T(1) + (n-1)n} \\ &= \boldsymbol{n^2 + n^2 - n} \\ &= \boldsymbol{2n^2 - n} \\ &= \boldsymbol{O(n^2)} \end{aligned}$$