RECURRENCES AND THEIR SOLUTION

A recurrence relation for the given sequence $\{T_n\}, n \geq 0$ is an equation that describes T_n using one or more previous terms $T_{n-1}, T_{n-2}, \ldots$.

Some early terms are specified explicitly and later terms are expressed as a function of their predecessors.

Examples:

1. The sequence 1, 2, 3, 4, 5, ... can be described by the recurrence
$$T_1 = 1$$
$$T_n = T_{n-1} + 1 \qquad (for\ n \geq 2)$$
2. The sequence 1, 2, 4, 8, 16 ... can be described by the recurrence
$$T_0 = 1$$
$$T_n = 2T_{n-1} \qquad (for\ n \geq 1)$$
3. The famous Fibonacci sequence 0, 1, 1, 2, 3, 5, 8, 13, 21, 34, ... is described by the recurrence
$$F_0 = 0$$
$$F_1 = 1$$
$$F_n = F_{n-1} + F_{n-2} \qquad (for\ n \geq 2)$$
4. The recurrence for the sequence 1, 3, 7, 15, 31, ... (Towers of Hanoi recurrence) is
$$T_1 = 1$$
$$T_n = 2T_{n-1} + 1 \qquad (for\ n \geq 2)$$
5. The recurrence for the merge sort is
$$T(n) = \theta(1) \qquad (if\ n = 1)$$
$$T(n) = 2T\left(\frac{n}{2}\right) + \theta(n) \quad (if\ n > 1)$$

SOLVING RECURRENCES

Recurrences arise frequently in the analysis of recursive algorithms. However, given a recurrence one cannot easily answer simple questions like "what is the value of its 100th term", or "what is the asymptotic growth rate". So, we need to solve a recurrence before answering such questions. Solving a recurrence means obtaining the closed form expression (explicit formula) that satisfies the given recurrence.

Two **general** solution techniques that are applicable to every type of recurrence are:

1. Substitution method (also known as guess-and-verify method)
2. Iteration method (also known as expansion method)

There are two other big classes of recurrences: **linear** and **divide-and-conquer** recurrences. These two classes are solvable using cookbook techniques (you follow the recipe and get the answer). A Linear Homogenous Recurrence relation can be solved systematically by following some number of steps. Most Divide-and conquer recurrences are easily solvable using **master method**.

Substitution Method

In this method we guess the solution and then verify that the guess is correct with an induction proof.

We substitute the guessed solution for the function when applying the inductive hypothesis to smaller values; hence the name "substitution" method.

Example 1: consider the recurrence for solving Towers of Hanoi problem

$$\begin{aligned} T_1 &= 1 && (for\ n = 1) \\ T_n &= 2T_{n-1} + 1 && (for\ n \geq 2) \end{aligned}$$

As a basis for making a good guess, compute the few terms from beginning:

$$\begin{aligned} T_1 &= 1 && (2^1 - 1) \\ T_2 &= 2T_1 + 1 = 2.1 + 1 = 3 && (2^2 - 1) \\ T_3 &= 2T_2 + 1 = 2.3 + 1 = 7 && (2^3 - 1) \\ T_4 &= 2T_3 + 1 = 2.7 + 1 = 15 && (2^4 - 1) \end{aligned}$$

From the pattern of values obtained above, one can naturally guess the solution as $T_n = 2^n - 1$.

Now we need to verify our guess using an induction proof. After all, our guess might be wrong.

Proving the guess using Induction:

Let the statement $T_n = 2^n - 1,\ \ n \geq 1$ satisfies the given recurrence.

Basis Step: for n=1, T_1 is true because $T_1 = 2^1 - 1 = 1$.

Inductive Step: Assuming that $T_k = 2^k - 1$, we will now prove that $T_{k+1} = 2^{k+1} - 1$.

Given $T_n = 2T_{n-1} + 1$.

Therefore, $T_{k+1} = 2T_{k+1-1} + 1$

$= 2T_k + 1$

$= 2(2^k - 1) + 1$ (from inductive hypothesis, $T_k = 2^k - 1$)

$= 2^{k+1} - 1$ ■

Example 2: We can use substitution method to establish either upper bound or lower bound on a recurrence. Now consider the following recurrence

$$\begin{aligned} &T(1) = 1, T(2) = 4, T(3) = 5 \\ &T(n) = 2T(\lfloor n/2 \rfloor) + n \end{aligned}$$

Since this recurrence is similar to merge sort recurrence, we guess that the solution is $T(n) = O(n \log n)$.

Proving the guess: We need to prove that $T(n) \leq cn\log n, \quad c > 0$.

Base Case: for n=2, $T(2) = 2T(\lfloor 2/2 \rfloor) + 2 = 2T(1) + 2 = 4 \leq cn\log n, \;\; for\; c \geq 2$.

Thus the base case holds.

[Note that for n=1, the base case T(1) fails to hold. You can remove this troublesome base case and try n=2 or n=3. Also, observe that for $n > 3$, the recurrence does not depend on T(1). Further, we don't worry about base cases, nor do we show base cases in the substitution proof. we are ultimately interested in an asymptotic solution to a recurrence, it will always be possible to choose base cases that work. However, when we want an exact solution, then we have to deal with base cases.]

Inductive Step: Assuming that $T(k)$ holds for $k < n$, in particular for $k = \lfloor n/2 \rfloor$,

i.e. $T(\lfloor n/2 \rfloor) \leq c.\lfloor n/2 \rfloor . \log(\lfloor n/2 \rfloor)$

Now given that, $T(n) = 2T(\lfloor n/2 \rfloor)+n$

$$\begin{aligned}
\Rightarrow \; T(n) &\leq 2.c.\lfloor n/2 \rfloor . \log(\lfloor n/2 \rfloor) + n \\
&\leq 2.c.\frac{n}{2}.\log\frac{n}{2} + n \\
&= cn\log\frac{n}{2} + n \\
&= cn\log n - cn\log 2 + n \\
&= cn\log n - cn + n \\
&\leq cn\log n, \;\; as\; long\; as\; c \geq 1
\end{aligned}$$

■

*Floors and ceilings can easily be removed. They don't affect the solution to the recurrence.

Example 3:

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

1. *Guess:* $T(n) = n \lg n + n$. *[Here, we have a recurrence with an exact function, rather than asymptotic notation, and the solution is also exact rather than asymptotic. We'll have to check boundary conditions and the base case.]*
2. *Induction:*

   Basis: $n = 1 \Rightarrow n \lg n + n = 1 = T(n)$

   Inductive step: Inductive hypothesis is that $T(k) = k \lg k + k$ for all $k < n$. We'll use this inductive hypothesis for $T(n/2)$.

$$\begin{aligned} T(n) &= 2T\left(\frac{n}{2}\right) + n \\ &= 2\left(\frac{n}{2}\lg\frac{n}{2} + \frac{n}{2}\right) + n \qquad \text{(by inductive hypothesis)} \\ &= n \lg \frac{n}{2} + n + n \\ &= n(\lg n - \lg 2) + n + n \\ &= n \lg n - n + n + n \\ &= n \lg n + n . \end{aligned}$$

■

Substitution method is a simple and general way to solve recurrences. But there is one big drawback: you have to guess right, which is not always easy especially when the given recurrence has a strange form.

Some tips for arriving at a good guess:

- Sometimes **identifying the pattern** of the numbers generated by expanding first few terms of the recurrence can help you to make a correct guess.
- Building a recursion tree for the given recurrence also helps in making a good guess.

Iteration Method

Here again the key step is identifying the pattern in the sequence of successive expressions. The method consists of three steps:

1. Iterate until a pattern appears, i.e. Repeatedly, apply the recurrence and simplify the result until pattern appears.
2. Verify the pattern.
3. Write T_n using early terms with known values.

Example 1: Consider the Towers of Hanoi recurrence given below:

$$T_1 = 1$$
$$T_n = 2T_{n-1} + 1 \qquad (for\ n \geq 2)$$

1. Iterate until a pattern appears.

$T_n = 2T_{n-1} + 1$

$T_n = 2(2T_{n-2} + 1) + 1$ (Applying the recurrence)

$T_n = 2^2T_{n-2} + 2 + 1$ (Simplifying the result)

$T_n = 2^2(2T_{n-3} + 1) + 2 + 1$ (Applying the recurrence)

$T_n = 2^3T_{n-3} + 2^2 + 2 + 1$ (Simplifying the result)

$T_n = 2^3(2T_{n-4} + 1) + 2^2 + 2 + 1$ (Applying the recurrence)

$T_n = 2^4T_{n-4} + 2^3 + 2^2 + 2 + 1$ (Simplifying the result)

The pattern is apparent now. The following formula seems to hold

$T_n = 2^kT_{n-k} + 2^{k-1} + 2^{k-2} + \cdots + 2^2 + 2 + 2^0$

$T_n = 2^kT_{n-k} + 2^k - 1 \qquad [\because 2^{k-1} + 2^{k-2} + \cdots + 2^2 + 2 + 2^0 = 2^k - 1]$

2. Verify the pattern by applying one more iteration. For simple enough patterns it is easy to verify that your pattern is correct.

$T_n = 2^kT_{n-k} + 2^k - 1$

$= 2^k(2T_{n-k-1} + 1) + 2^k - 1$ (Applying the recurrence)

$= 2^{k+1}T_{n-(k+1)} + 2^{k+1} - 1$ (Simplifying the result)

The final expression on the right is same as the expression on the first line, except that k is replaced by k+1. This proves that the formula is correct for all k (just like induction).

3. Finally, write T_n using early terms with known values.

We have, $T_n = 2^kT_{n-k} + 2^k - 1$

Now, choosing $k = n - 1$, expresses T_n in terms of T_1, which is equal to 1.

Thus, $T_n = 2^{n-1}T_{n-n+1} + 2^{n-1} - 1$

$or \quad T_n = 2^{n-1}T_1 + 2^{n-1} - 1$

$or \quad T_n = 2^{n-1} + 2^{n-1} - 1$

$or \quad T_n = 2.2^{n-1} - 1$

$or \quad \boldsymbol{T_n = 2^n - 1}$ ■

$\boldsymbol{Example\ 2}$: Solve the following recurrence using iteration method.

$T(0) = 1$

$T(n) = 2 + T(n-1), \qquad for\ n \geq 1$

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

$$\begin{aligned}
&= 2 + 2 + T(n-2) \\
&= 2 + 2 + 2 + T(n-3) \\
&= 2 + 2 + 2 + 2 + T(n-4) \\
&= 2.4 + T(n-4) \\
&\vdots \\
&= 2.n + T(0) \\
&= 2n + 1
\end{aligned}$$

Comparing substitution and iteration method

From the examples presented here, you might have observed that the substitution and iteration method tackle the problem from opposite directions.

In substitution method, we computed several terms at the beginning of the sequence $T_1, T_2, T_3, \ldots until$ a pattern appeared. We then generalized the n^{th} term (T_n) for the sequence.

In contrast, iteration method works backwards from the n^{th} term. Specifically, we start from the T_n and write this using earlier terms.

DIVIDE-AND-CONQUER RECURRENCES AND MASTER METHOD

A recurrence for the running time of divide-and-conquer algorithm has the following general form:

$$T(n) = aT(n/b) + f(n)$$

Here,

$T(n)$ is the running time of the algorithm on a problem of size n.

a is the number of sub-problems yielded as a result of division.

n/b is the size of each such sub-problem.

$f(n)$ is the cost of dividing the problem and combining the results of the sub-problems.

Thus, if it takes time $T(n/b)$ to solve a sub-problem of size n/b, it would take time $aT(n/b)$ to solve 'a' of them.

Master Method

Master method is a "cookbook" method for solving some of the divide-and-conquer recurrences. The master method depends on the following master theorem:

Master theorem: Let the recurrence $T(n)$ be defined as

$$T(n) = aT(n/b) + f(n),$$

$where\ \ a \geq 1\ and\ b > 1\ are\ constants\ and\ f(n) is\ an\ asymptotically\ positive\ function.$ Then, $T(n)$ has the following asymptotic bounds:

1. If $f(n) = O\left(n^{\log_b a - \epsilon}\right)$ for some constant $\epsilon > 0$, then $T(n) = \theta(n^{\log_b a})$.
2. If $f(n) = \Theta\left(n^{\log_b a}\right)$, then $T(n) = \theta\left(n^{\log_b a} \log n\right) = \theta(f(n) \log n)$.
3. If $f(n) = \Omega\left(n^{\log_b a + \epsilon}\right)$, for some constant $\epsilon > 0$, and if the regularity condition $af(n/b) \leq cf(n)$ for some constant $c < 1$ and all sufficiently large n, then $T(n) = \theta(f(n))$.

Using the master theorem is fast and easier. Just compare the function $f(n)$ with the function $n^{\log_b a}$. The larger of these two functions becomes the solution to the recurrence. However, if the two functions are of the same size, then the solution is $T(n) = \theta\left(n^{\log_b a} \log n\right) = \theta(f(n) \log n)$ i.e. multiply by a log n factor.

Examples:

1. $T(n) = 9T(n/3) + n$
   Here, $a = 9, b = 3$. $\therefore\ n^{\log_b a} = n^{\log_3 9} = n^2$. Also, $f(n) = n$.

Since, $f(n)$ *is* $\textbf{\textit{polynomially smaller}}$ *than* $n^{\log_b a}$. Thus, solution to given recurrence is $\theta\left(n^{\log_b a}\right) = \theta(n^2)$.

2. $T(n) = T(2n/3) + 1$
   Here, $a = 1, b = 3/2$. $\therefore\ n^{\log_b a} = n^{\log_{3/2} 1} = n^0 = 1$. Also, $f(n) = 1$.
   Since, $f(n)$ *and* $n^{\log_b a}$ $\textbf{\textit{have same size}}$$(both\ are\ constants)$. Thus, solution to given recurrence is $\theta\left(n^{\log_b a} \log n\right) = \theta(\log n)$.

3. $T(n) = 3T(n/4) + n \log n$
   Here, $a = 3, b = 4$. $\therefore\ n^{\log_b a} = n^{\log_4 3} = n^{0.793}$. Also, $f(n) = n \log n$.
   Since, $f(n) = n \log n$ *is* $\textbf{\textit{larger}}$ *as compared to* $n^{\log_b a} = n^{0.793}$. Thus, solution to given recurrence is $\theta\left(f(n)\right) = \theta(n \log n)$.

4. $T(n) = 2T(n/2) + \theta(n)$ (recurrence for merge sort and maximum sub-array problem)
   Here, $a = 2,\ b = 2$. $\therefore\ n^{\log_b a} = n^{\log_2 2} = n$. Also, $f(n) = \theta(n)$.
   Since, $f(n)$ *and* $n^{\log_b a}$ $\textbf{\textit{have same size}}$$(both\ linear\ functions)$. Thus, solution to given recurrence is $\theta\left(n^{\log_b a} \log n\right) = \theta(n \log n)$.

5. $T(n) = 8T(n/2) + \theta(n^2)$ (recurrence for matrix multiplication)
   Here, $a = 8,\ b = 2$. $\therefore\ n^{\log_b a} = n^{\log_2 8} = n^3$. Also, $f(n) = \theta(n^2)$.
   Since, $f(n)$ *is* $\textbf{\textit{smaller}}$ *than* $n^{\log_b a}$. Thus, solution to given recurrence is $\theta\left(n^{\log_b a}\right) = \theta(n^3)$.

6. $T(n) = 7T(n/2) + \theta(n^2)$ (recurrence for Strassen's matrix multiplication)
   Here, $a = 7,\ b = 2$. $\therefore\ n^{\log_b a} = n^{\log_2 7} = n^{2.801}$. Also, $f(n) = \theta(n^2)$.
   Since, $f(n)$ *is* $\textbf{\textit{smaller}}$ *than* $n^{\log_b a}$. Thus, solution to given recurrence is $\theta\left(n^{\log_b a}\right) = \theta\left(n^{\log_2 7}\right) = \theta(n^{2.801})$.

Now, consider the following recurrence.

$$T(n) = 2T(n/2) + n \log n$$

Here, $a = 2,\ b = 2$. $\therefore\ n^{\log_b a} = n^{\log_2 2} = n$. Also, $f(n) = n \log n$.

Since, $f(n) = n \log n$ *is* $\textbf{\textit{larger}}$ *as compared to* $n^{\log_b a} = n$. You might mistakenly think that the solution is $\theta\left(f(n)\right) = \theta(n \log n)$. But the answer is not correct. The problem is that $n \log n$ is not polynomially larger than n because the ratio $\frac{f(n)}{n^{\log_b a}} = \frac{n \log n}{n} = \log n$ is asymptotically less than n^{ϵ} for any positive constant ϵ. Thus, master method can not apply to this recurrence.

LINEAR RECURRENCE RELATIONS

One important class of recurrence relations called Linear Homogenous Recurrence Relations can be solved in a systematic way. A recurrence of the form

$$a_n = C_1 a_{n-1} + C_2 a_{n-2} + \cdots + C_k a_{n-k}$$

where $C_1,\ C_{2,}, \dots, C_k$ are real numbers and $C_k \neq 0$, is called a Linear Homogenous Recurrence Relation of degree/order k. Commonly, the value of the function a is also specified at a few points; these are called **boundary conditions.** For example, Fibonacci recurrence,

$$F_0 = 0$$
$$F_1 = 1$$
$$F_n = F_{n-1} + F_{n-2}, \qquad for\ n \geq 2$$

the most famous of all recurrence equations, which remained unsolved for almost six centuries! is a homogenous recurrence relation of degree 2. Here the boundary conditions are $F_1 = 0$, $F_2 = 1$. The recurrence is homogenous i.e. of simpler kind. Essentially, every term is product of a real and a_j (jth term of the recurrence). The recurrence $T_n = T_{n-1} + 1$, is not homogenous because in this the second term is independent of T_j.

Solving a Linear Recurrence

In general, linear recurrences tend to have exponential solutions. Let $a_n = r^n$ be the solution of recurrence

$$a_n = C_1 a_{n-1} + C_2 a_{n-2} + \cdots + C_k a_{n-k} .$$

Then, $a_n = r^n$, must satisfy the given recurrence. Substituting into the given recurrence, yields

$$r^n = C_1 r^{n-1} + C_2 r^{n-2} + \cdots + C_k r^{n-k}$$

$$or\ \ r^n = C_1 r^{n-1} + C_2 r^{n-2} + \cdots + C_{k-1} r^{n-k+1} + C_k r^{n-k}$$

Dividing both sides by r^{n-k} leaves an equation:

$$r^k = C_1 r^{k-1} + C_2 r^{k-2} + \cdots + C_{k-1} r + C_k$$

$$or\ \ r^k - C_1 r^{k-1} - C_2 r^{k-2} - \cdots - C_{k-1} r - C_k = 0.$$

This equation is known as **characteristic equation** and its roots as **characteristic roots**. Therefore, the characteristic equation corresponding to linear homogenous recurrence relation of degree 2,

$a_n = C_1 a_{n-1} + C_2 a_{n-2}$, would be $r^2 - C_1 r - C_2 = 0$.

Once you have obtained the characteristic equation corresponding to a given recurrence, proceed to find its roots. Suppose $r_{1,} r_2, \dots, r_k$ *be k* ***distinct*** *roots of the characteristic equation.* The solution to the recurrence is then given by,

$$a_n = \alpha_1 r_1{}^n + \alpha_2 r_2{}^n + \cdots + \alpha_k r_k{}^n$$

where $\alpha_1, \alpha_2, \dots$ are constants that can be obtained using initial conditions

Solving Linear Recurrences of degree, k = 2.

Use the following theorems to solve linear recurrences of degree 2.

1. Let $a_n = C_1 a_{n-1} + C_2 a_{n-2}$ be a homogenous recurrence relation of degree two. Suppose that its corresponding characteristic equation $r^2 - C_1 r - C_2 = 0$ has two **distinct** roots, say r_1 *and* r_2. Then, the solution to the given recurrence is given by
$$a_n = \alpha_1 {r_1}^n + \alpha_2 {r_2}^n,$$
where α_1 *and* α_2 are constants that can be obtained using initial conditions (boundary conditions).
Note: The above theorem does not apply if there are some indistinct (equal) roots. So, in this case use the following theorem.
2. Let $a_n = C_1 a_{n-1} + C_2 a_{n-2}$ be a homogenous recurrence relation of degree two. Suppose that its corresponding characteristic equation $r^2 - C_1 r - C_2 = 0$ has two **equal** roots, say r. Then, the solution to the given recurrence is given by
$$a_n = \alpha_1 r^n + \alpha_2 n r^n.$$

Examples: Apply following steps:

- Find the characteristic equation corresponding to the given recurrence.
- Find roots of the characteristic equation.
- Use one of the above theorems to find solution.
- Use initial conditions to find the constants $\alpha_1 \, and \, \alpha_2$.

Example 1: Find the solution of the following recurrence

$$a_n = a_{n-1} + 2a_{n-2}, \;\; a_0 = 2, \; a_1 = 7.$$

Solution: The given recurrence is linear recurrence of degree, k=2.

Here $C_1 = 1 \;\; and \; C_2 = 2.$

The characteristic equation is $r^2 - C_1 r - C_2 = 0$, that is, $r^2 - r - 2 = 0.$

Solving the characteristic equation (use quadratic formula or factorization method to solve a quadratic equation), we find the characteristic roots

$r_1 = 2 \;\; and \;\; r_2 = -1.$

∴ The solution is given by $a_n = \alpha_1 {r_1}^n + \alpha_2 {r_2}^n$, that is,

$$a_n = \alpha_1 2^n + \alpha_2 (-1)^n$$

Now, the initial conditions can be used to find the constants α_1 and α_2.

We have, $a_0 = \alpha_1 2^0 + \alpha_2 (-1)^0 = \alpha_1 + \alpha_2 = 2$ ---------- (1)

and $a_1 = \alpha_1 2^1 + \alpha_2 (-1)^1 = 2\alpha_1 - \alpha_2 = 7$---------- (2)

Solving equations (1) and (2), we get

$$\alpha_1 = 3$$

$$\alpha_2 = -1$$

Thus, the required solution to the given recurrence is

$$a_n = 3.2^n - (-1)^n$$

Example 2: Find the solution to the following recurrence

$$a_n = 6a_{n-1} - 9a_{n-2}, \ \ a_0 = 1, \ a_1 = 6.$$

Solution: The given recurrence is linear recurrence of degree, k=2.

Here $C_1 = 6 \, and \, C_2 = -9$.

The characteristic equation is $r^2 - C_1 r - C_2 = 0$, that is, $r^2 - 6r + 9 = 0$.

Solving the characteristic equation, the characteristic roots are

$r_1 = r_2 = 3 = r$ (two equal roots)

∴ The solution is given by $a_n = \alpha_1 r^n + \alpha_2 n r^n$, that is,

$$a_n = \alpha_1 3^n + \alpha_2 . n . 3^n$$

Now, $a_0 = \alpha_1 3^0 + \alpha_2 . 0 . 3^0 = \alpha_1 = 1$ ---------- (1) [$\because a_0 = 1$]

and $a_1 = \alpha_1 3^1 + \alpha_2 . 1 . 3^1 = 3\alpha_1 + 3\alpha_2 = 6$ ---------- (2) [$\because a_1 = 6$]

Substituting $\alpha_1 = 1$ in eqaution (2), we get $\alpha_2 = 1$.

Thus, the required solution is

$$a_n = 3^n + n3^n.$$

Example 3: Solve the Fibonacci recurrence give by

$$F_0 = 0$$
$$F_1 = 1$$
$$F_n = F_{n-1} + F_{n-2} \qquad (for \, n \geq 2)$$

Solution: The given recurrence is linear recurrence of degree, k=2.

Here $C_1 = 1 \, and \, C_2 = 1$.

The characteristic equation is $r^2 - C_1 r - C_2 = 0$, that is, $r^2 - r - 1 = 0$.

The characteristic roots are:

$$r_1 = \frac{1+\sqrt{5}}{2} \, and \, r_2 = \frac{1-\sqrt{5}}{2}$$

∴ The solution is given by $F_n = \alpha_1 {r_1}^n + \alpha_2 {r_2}^n$, that is,

$$F_n = \alpha_1\left(\frac{1+\sqrt{5}}{2}\right)^n + \alpha_2\left(\frac{1-\sqrt{5}}{2}\right)^n$$

Using boundary conditions $F_1 = 0, F_2 = 1$ to find the values of $\alpha_1$and α_2

$F_0 = \alpha_1\left(\frac{1+\sqrt{5}}{2}\right)^0 + \alpha_2\left(\frac{1-\sqrt{5}}{2}\right)^0 = \alpha_1 + \alpha_2 = 0$---------- (1)

$F_1 = \alpha_1\left(\frac{1+\sqrt{5}}{2}\right) + \alpha_2\left(\frac{1-\sqrt{5}}{2}\right) = 1$ ------------------ (2)

Solving equations (1) and (2), we get

$\alpha_1 = \frac{1}{\sqrt{5}}$ *and* $\alpha_2 = -\frac{1}{\sqrt{5}}$.

Thus, the solution to the given recurrence is

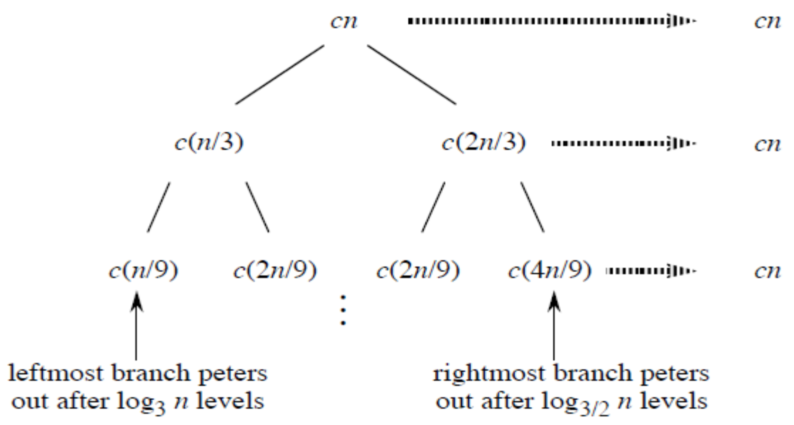
$$F_n = \frac{1}{\sqrt{5}}\left(\frac{1+\sqrt{5}}{2}\right)^n - \frac{1}{\sqrt{5}}\left(\frac{1-\sqrt{5}}{2}\right)^n$$

RECURSION TREES

Use to generate a guess. Then verify by substitution method.

Example: $T(n) = T(n/3) + T(2n/3) + \Theta(n)$. For upper bound, rewrite as $T(n) \le T(n/3) + T(2n/3) + cn$; for lower bound, as $T(n) \ge T(n/3) + T(2n/3) + cn$.

By summing across each level, the recursion tree shows the cost at each level of recursion (minus the costs of recursive calls, which appear in subtrees):



cn
cn
$c(n/3)$
$c(2n/3)$
cn
$c(n/9)$
$c(2n/9)$
$c(2n/9)$
$c(4n/9)$
cn
leftmost branch peters out after $\log_3 n$ levels
rightmost branch peters out after $\log_{3/2} n$ levels

- There are $\log_3 n$ full levels, and after $\log_{3/2} n$ levels, the problem size is down to 1.
- Each level contributes $\le cn$.
- Lower bound guess: $\ge dn \log_3 n = \Omega(n \lg n)$ for some positive constant d.
- Upper bound guess: $\le dn \log_{3/2} n = O(n \lg n)$ for some positive constant d.
- Then *prove* by substitution.

1. ***Upper bound:***

   Guess: $T(n) \le dn \lg n$.

   Substitution:

$$\begin{aligned}
T(n) &\le T(n/3) + T(2n/3) + cn \\
&\le d(n/3)\lg(n/3) + d(2n/3)\lg(2n/3) + cn \\
&= (d(n/3)\lg n - d(n/3)\lg 3) \\
&\qquad + (d(2n/3)\lg n - d(2n/3)\lg(3/2)) + cn \\
&= dn\lg n - d((n/3)\lg 3 + (2n/3)\lg(3/2)) + cn \\
&= dn\lg n - d((n/3)\lg 3 + (2n/3)\lg 3 - (2n/3)\lg 2) + cn \\
&= dn\lg n - dn(\lg 3 - 2/3) + cn \\
&\le dn\lg n \qquad \text{if } -dn(\lg 3 - 2/3) + cn \le 0\,, \\
&\qquad\qquad\qquad\qquad d \ge \frac{c}{\lg 3 - 2/3}\,.
\end{aligned}$$

   Therefore, $T(n) = O(n \lg n)$.

   Note: Make sure that the symbolic constants used in the recurrence (e.g., c) and the guess (e.g., d) are different.

2. ***Lower bound:***

   Guess: $T(n) \ge dn \lg n$.

   Substitution: Same as for the upper bound, but replacing $\le$ by $\ge$. End up needing

$$0 < d \le \frac{c}{\lg 3 - 2/3}\,.$$

   Therefore, $T(n) = \Omega(n \lg n)$.

Since $T(n) = O(n \lg n)$ and $T(n) = \Omega(n \lg n)$, we conclude that $T(n) = \Theta(n \lg n)$. ■