Recurrence

We derived the run time for merge sort in lecture 4 by constructing a *recursion tree* and then adding up the run times for each level of the recursion. The recursive equation for merge sort, however, fits into the form required by the master theorem and thus could have been solved by inspection. However when a recursive equation does not satisfy the form for the master theorem, an alternative technique known as the *substitution method* is required to find the solution (which may use the recursion tree technique as an intermediate step). This technique involves proving (similar to proof by induction) that a "guess" solves the recursive equation. This "guess" often comes from experience from other similar problems or through a recursion tree.

Recursion Trees

In order to formulate a good "guess" at a solution to use in the substitution method, a *recursion tree* can often be constructed to find an approximation to the run time. Similar to the one constructed for merge

sort, the procedure consists of:

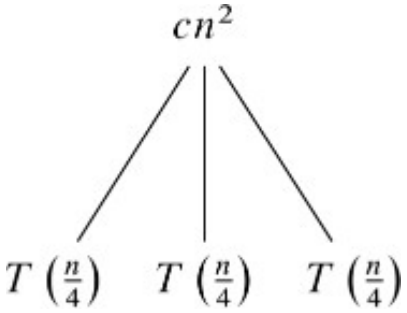
- Construct the tree based on the recurrence
- Calculate the run time for each recursive level
- Sum up all levels to compute the total run time

Example

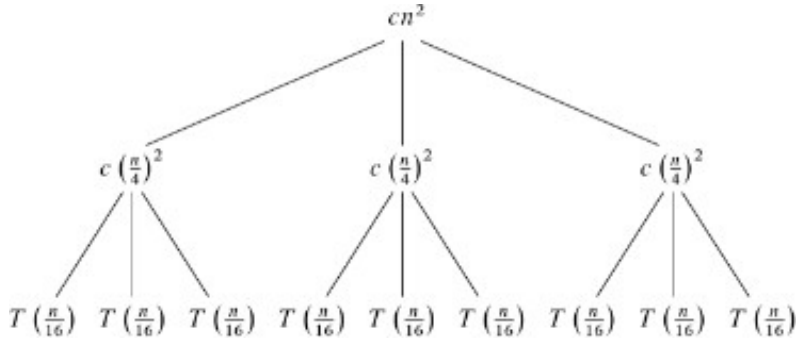
Find a "guess" for the recursive equation (which could be checked with the master theorem)

$$T(n)=3T(n/4)+\Theta(n^2)$$

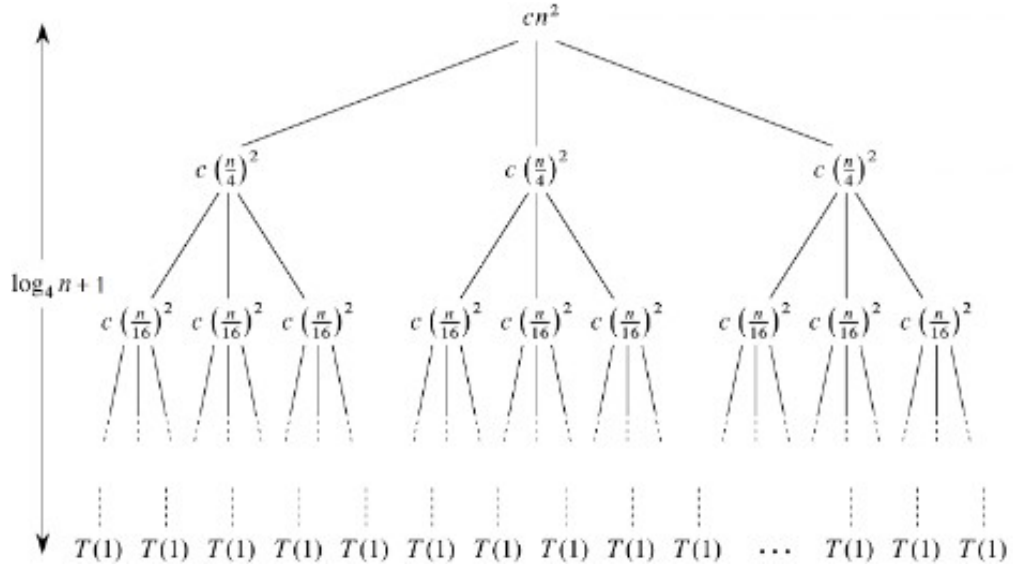
Again we will assume a bound for the asymptotic term as $\Theta(n^2) = c\,n^2$ and construct the tree as follows



Expanding the next level gives



At each subsequent level there are $n/4^i$ elements, so we continue expanding until $n/4^h = 1 < h = \log_4 n$ giving $h + 1 = \log_4 n + 1$ levels of recursion as shown below.



It can be seen from the diagram that each level has 3^i terms of $c\ (n/4^i)^2$. Hence the run time for each level is

$$3^i\left(c(n/4^i)^2\right) = cn^2(3/4^2)^i = cn^2(3/16)^i$$

The last level (where $i = \log_4 n$ and $T(1) = c$) will have

$$3^{\log_4 n} c = n^{\log_4 3} c \qquad \text{by eq. 3.16 on page 56 of CLRS}$$

$$= \Theta(n^{\log_4 3}) \qquad \text{by asymptotic reflexivity pg. 51 of CLRS}$$

Thus summing up the run time for all the levels gives

$$T(n) = cn^2 + (3/16)cn^2 + (3/16)^2 cn^2 + \ldots + (3/16)^{\log_4 n - 1} cn^2 + \Theta(n^{\log_4 3})$$

$$= \sum_{i=0}^{\log_4 n - 1} \left(\frac{3}{16}\right)^i cn^2 + \Theta(n^{\log_4 3})$$

$$\leqslant \sum_{i=0}^{\infty} \left(\frac{3}{16}\right)^i cn^2 + \Theta(n^{\log_4 3})$$

$$\leqslant cn^2 \sum_{i=0}^{\infty} \left(\frac{3}{16}\right)^i + \Theta(n^{\log_4 3})$$

$$\leqslant cn^2 \left(\frac{1}{1-(3/16)}\right) + \Theta(n^{\log_4 3}) \qquad \text{by eq. A.6 pg. 1147 of CLRS}$$

$$\leqslant \frac{16}{13} cn^2 + \Theta(n^{\log_4 3})$$

$$= O(n^2) \qquad \text{since } n^{\log_4 3} = n^{0.79} < n^2$$

Note: This strictly gives an *upper bound* which can be used as a starting "guess" for the substitution method.

Substitution Method

While the recursion tree method provides an *approximate* bound, to mathematically *prove* that the bound is correct requires the substitution method. This method essentially uses a proof technique similar to induction where the bound is verified for the recursive equation and a base case is found that holds for the recursion. To accomplish the proof, the bound is *assumed* to hold for *T(n)* and then is *substituted* into the recursive equation to demonstrate that the assumption is correct by deriving **EXACTLY** the same bound as the assumption. The technique is best demonstrated by example.

Example (to verify the recursive tree "guess" for the previous example)

Show that the solution for

$$T(n)=3T(n/4)+kn^2$$

is $T(n) = O(n^2)$.

Assume the solution holds so that by the definition of O()

$$T(n)\leq cn^2$$

for some $c > 0$. Substituting the assumption into the $T(n/4)$ term gives

$$T(n) = 3T(n/4) + kn^2 \leqslant 3c\left(\frac{n}{4}\right)^2 + kn^2$$

We must now manipulate the right side until it is **exactly** of the form cn^2 as follows

$$\begin{aligned} T(n) &\leqslant 3c\left(\frac{n}{4}\right)^2 + kn^2 \\ &\leqslant \frac{3}{16}cn^2 + kn^2 \\ &\leqslant \left(\frac{3}{16}c + k\right)n^2 \\ &\leqslant cn^2 \qquad \text{for } c \geqslant \frac{16}{13}k \end{aligned}$$

Now we need to verify a base case that holds for the recursion. Assuming $T(0) = 0$, for $n = 1$

$$T(1) = 3T(0) + k(1)^2 = k$$

Evaluating the assumption for the solution gives

$$T(1) \leqslant c(1)^2 = c$$

which holds for $c \geq k$ (which is satisfied by our choice of c above to verify the recursion). The base cases for $n = 2,3$ (since $T(2/4) = T(3/4) = T(0)$) can also be easily verified and hence we have *proven* the

assumed solution $T(n) = O(n^2)$ is correct for $c \geq (16/13)k$ and $n \geq 1$. Note that by inspection we see $T(n) = \Omega(n^2)$ (since the combine term is at least that big) and thus $T(n) = \Theta(n^2)$.

Note: Unlike inductive proofs where the induction must hold *for all n*, recursive proofs for asymptotic bounds only need to hold for $n \geq n_0$. Hence we only need to find *a* value of *n* (not *necessarily* $n = 1$) for which the base case is satisfied.

Often our guess is "too weak" to show the induction, however we can sometimes include a lower order term (which does not change the asymptotic bound) in order to prove the solution.

Example

Show that the solution for

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

is $T(n) = O(n)$.

Assume the solution holds so that by the definition of O()

$$T(n)\leqslant c\,n$$

for some $c > 0$. Substituting the assumption into the $T(n/2)$ term gives

$$T(n)=2T(n/2)+1 \leqslant 2c(n/2)+1 \leqslant cn+1$$

However we needed to show $T(n) \leq cn$ **NOT** $cn + 1$ (even though 1 seems negligible for large n)! Instead we try another "guess" which still satisfies the asymptotic bound

$$T(n) \leqslant cn-b$$

Now substituting the new assumption into the $T(n/2)$ term gives

$$\begin{aligned} T(n) &\leqslant 2(c\frac{n}{2}-b)+1 \\ &\leqslant cn-2b+1 \\ &\leqslant cn-(2b-1) \\ &\leqslant cn-b \qquad \text{for } b \geqslant 1 \end{aligned}$$

After verifying boundary conditions, we have proven that $T(n) = O(n)$.

Lastly we can sometimes perform a substitution of variables to transform a recursive equation into a form that we can more easily solve.

Example

Find the solution to the following recursive equation

$$T(n)=2T(\sqrt{n})+\lg n$$

Let $m = \lg n < n = 2^m$. Then the recursive equation can be rewritten as

$$T(2^m)=2T(2^{m/2})+m$$

Then let $S(m) = T(2^m)$ giving

$$S(m)=2S(m/2)+m$$

Which has solution $S(m) = O(m \lg m)$ (i.e. merge sort). Thus replacing $m = \lg n$ gives $T(n) = O((\lg n) \lg (\lg n))$.