RECURRENCE RELATIONS

In this lecture note we study some basic recurrence relations. These equations arise in the analysis of algorithms, among other things. We consider the following:

1. Divide-&-Conquer Recurrences and the Master Theorem.
2. Full History Recurrences.
3. The Guess-then-Verify Technique.
4. The Variable Substitution Method.

There are other method, such as the recursion tree and the generating function methods, that we will not discuss in this lecture note.

1. THE DIVIDE-AND-CONQUER RECURRENCES

Let α and β be positive integers both greater than one. Let us assume that we can divide a problem of size n into α subproblems, each of the same type, each of size n/β. Termination of the algorithm is guaranteed because β is greater than one, i.e. each of the subproblems is smaller than the original. These subproblems are solved recursively, and then their solutions are combined to get the solution of the original problem of size n. Let us assume the time needed to divide the problem into α subproblems plus combine together their solutions is $f(n)$, called the *driving function*. Of course, if the problem is sufficiently small, e.g., $n \le 1$, then we solve it directly without recursion. Let us assume for $n \le 1$ it takes a constant amount of time, say γ steps, to solve the problem. If we denote the time complexity of our algorithm by $T(n)$, then we have:

$$T(n) = \begin{cases} \alpha\, T(\frac{n}{\beta}) + f(n) & \forall n > 1 \\ \gamma & \forall n \le 1 \end{cases} \qquad \text{(I)}$$

Example 1: Consider the *merge-sort* algorithm. For simplicity, let n be a power of two. The input to the merge-sort is a list of n keys which are to be sorted into increasing order. Merge-sort splits the original list at the midpoint into two lists each with $n/2$ keys, sorts each of these two new lists recursively using merge-sort, and then merges these two sorted lists into a single sorted list containing all n keys in at most n time steps. Thus, the running time $T(n)$ for merge-sort is expressed by equation (I), where $\alpha = 2$, $\beta = 2$ and $f(n) = n$, and you may assume $\gamma = 1$. □

We solve recurrence (I) for $T(n)$ in two steps. In the first step we solve the somewhat simpler *homogeneous recurrence*:

$$h(n) = \begin{cases} \alpha\, h(\frac{n}{\beta}) & \forall n > 1 \\ 1 & \forall n \le 1 \end{cases} \qquad \text{(II)}$$

(This can be interpreted as if we have, momentarily, set the driving function to zero, i.e. the dividing and combining steps take no time, only the recursive calls remain!)

We can solve (II) for $h(n)$ rather easily by *repeated substitution* (also called the *iteration method*). Let $k = \lceil \log_\beta n \rceil$ (thus, $\beta^{k-1} < n \le \beta^k$). Observe that for any $x > 1$ we have $h(x) = \alpha\, h(x/\beta)$. Now try $x = n, n/\beta, n/\beta^2, \cdots, n/\beta^{k-1}$. Using this substitution we get:

$$h(n) = \alpha\, h(\frac{n}{\beta}) = \alpha^2\, h(\frac{n}{\beta^2}) = \cdots = \alpha^k\, h(\frac{n}{\beta^k}) = \alpha^k \qquad \text{(III)}$$

From above we have:

$$h(n) = \alpha^k = \alpha^{\lceil \log_\beta n \rceil} = \Theta(\alpha^{\log_\beta n}) = \Theta(n^{\log_\beta \alpha}) \ \dagger \qquad \text{(IV)}$$

We have solved (II) and completed the first step. In the second step let us define:

$$Q(n) = T(n)/h(n) \qquad \text{(V)}$$

That is, $T(n) = Q(n) \cdot h(n)$. Substituting this in (I) we get:

$$Q(n) \cdot h(n) = \begin{cases} \alpha\, Q(\frac{n}{\beta}) \cdot h(\frac{n}{\beta}) + f(n) & \forall n > 1 \\ \gamma & \forall n \le 1 \end{cases}$$

Using (II), we can simplify the above equation (divide by $h(n)$) to (VI) below. (This is the reason why we solved (II) first.)

$$Q(n) = \begin{cases} Q(\frac{n}{\beta}) + \frac{f(n)}{h(n)} & \forall n > 1 \\ \gamma & \forall n \le 1 \end{cases} \qquad \text{(VI)}$$

For simplicity let's first assume $n = \beta^k$, i.e., $k = \log_\beta n$. We can solve (VI) again by iterated substitution.

$$\begin{aligned} Q(n) &= Q(\beta^k) \\ &= Q(\beta^{k-1}) + \frac{f(\beta^k)}{h(\beta^k)} \\ &= Q(\beta^{k-2}) + \frac{f(\beta^{k-1})}{h(\beta^{k-1})} + \frac{f(\beta^k)}{h(\beta^k)} \\ &= \cdots \end{aligned}$$

† By taking log from each side, show that $x^{i \log_y z} = z^{i \log_y x}$ (i.e, x & z swapped).

$$= Q(\beta^0) + \sum_{i=1}^{k} \frac{f(\beta^i)}{h(\beta^i)}$$

Since $Q(\beta^0) = Q(1) = \gamma$, and from (IV) $h(\beta^i) = \beta^{i \log_\beta \alpha} = \alpha^{i \log_\beta \beta} = \alpha^i$, we have:

$$Q(n) = \gamma + \sum_{i=1}^{k} \frac{f(\beta^i)}{\alpha^i} \qquad \text{(VII)}$$

Since $T(n) = Q(n) \cdot h(n)$, from (IV) and (VII) we get:

$$T(n) = \Theta\left(n^{\log_\beta \alpha} \left(\gamma + \sum_{i=1}^{k} \frac{f(\beta^i)}{\alpha^i} \right) \right). \qquad \text{(VIII)}$$

Exercise 1: What if n is not β^k, i.e, $\log_\beta n$ is not an integer? Recall that $k = \lceil \log_\beta n \rceil$. If $f(.)$ satisfies $f(\Theta(m)) = \Theta(f(m)) \ \forall m$, we say $f(.)$ is a *faithful* function. For instance all polynomials are faithful. Starting with $Q(n)$ and using the iterated substitution again, prove that the asymptotic solution (VIII) is still valid when f is a faithful function. □

The solution to our original recurrence relation (I) is (VIII). Let us consider some examples.

Example 1 (revisited) : In (I) assume $\alpha = 2$, $\beta = 2$, $\gamma = 1$ and $f(n) = n$. That is

$$T(n) = \begin{cases} 2T(\frac{n}{2}) + n & \forall n > 1 \\ 1 & \forall n \le 1 \end{cases}$$

The solution, from (VIII), is:

$$T(n) = \Theta\left(n^{\log_2 2} \left(1 + \sum_{i=1}^{k} \frac{2^i}{2^i} \right) \right) = \Theta(n(1 + \sum_{i=1}^{k} 1)) = \Theta(n(1+k))$$

$$= \Theta(n(1 + \lceil lg\, n \rceil))$$

Therefore, $T(n) = \Theta(n\, lg\, n)$. □

There is a summation term in (VIII). We need to know ways of obtaining closed form formulas for such summations, although it turned out to be rather simple in the above example.

Power summation: $S_j(m) = 1^j + 2^j + 3^j + \cdots + m^j = \sum_{i=1}^{m} i^j$. Here are a few cases:

$$S_1(m) = m(m+1)/2 = \Theta(m^2)$$

$$S_2(m) = m(m+1)(2m+1)/6 = \Theta(m^3)$$

$$S_3(m) = (S_1(m))^2 = \Theta(m^4)$$

In fact, using integration to bound the summation, it can be shown that $S_j(m) = \Theta(m^{j+1})$ for any constant $j \ge 0$.

Harmonic numbers: The m-th harmonic number is $H_m = 1 + \frac{1}{2} + \frac{1}{3} + \cdots + \frac{1}{m}$.

An approximation formula (see, for example, *Knuth, vol 1*, or chapter 3 of [CLR]) says $H_m \cong \ln m \cong 0.7\, lg\, m$, where ln is the natural logarithm with the base $e = 2.7182\cdots$ Thus, $H_m = \Theta(lg\, m)$.

Example 2: Consider the following recurrence:

$$T(n) = \begin{cases} 4\,T(\frac{n}{2}) + n^2\, lg\, n & \forall n > 1 \\ 1 & \forall n \le 1 \end{cases}$$

The solution is

$$T(n) = \Theta\left(n^{\log_2 4}\left(1 + \sum_{i=1}^{k} \frac{(2^i)^2\, lg\,(2^i)}{4^i}\right)\right) = \Theta\left(n^2\left(1 + \sum_{i=1}^{k} lg\,(2^i)\right)\right)$$

$$= \Theta\left(n^2\left(1 + \sum_{i=1}^{k} i\right)\right) = \Theta(n^2(1 + S_1(k))) = \Theta(n^2(1 + k^2))$$

$$= \Theta(n^2(1 + (lg^2\, n)))$$

Therefore, $T(n) = \Theta(n^2 lg^2 n)$. □

Example 3: Consider the following recurrence

$$T(n) = \begin{cases} 4\,T(\frac{n}{2}) + \frac{n^2}{lg\, n} & \forall n > 1 \\ 1 & \forall n \le 1 \end{cases}$$

The solution is

$$T(n) = \Theta\left(n^{\log_2 4}\left(1 + \sum_{i=1}^{k} \frac{(2^i)^2 / lg\,(2^i)}{4^i}\right)\right)$$

$$= \Theta\left(n^2\left(1 + \sum_{i=1}^{k} \frac{1}{i}\right)\right) = \Theta\left(n^2\,(1 + H_k)\right) = \Theta(n^2\,(1 + lg\, k))$$

$$= \Theta(n^2\,(1 + lg\, lg\, n))$$

Therefore, $T(n) = \Theta(n^2\, lg\, lg\, n)$. □

Polynomial Driving Function - The Master Theorem

Often the driving function is some polynomial in n, i.e. $f(n) = \Theta(n^d)$, say, $f(n) = cn^d$ for some constants $c > 0$ and $d \ge 0$. In other words, the recurrence is:

$$T(n) = \begin{cases} \alpha T(\frac{n}{\beta}) + cn^d & \forall n > 1 \\ \gamma & \forall n \le 1 \end{cases}$$

In this case we can considerably simplify the solution given in (VIII) as follows:

$$T(n) = \begin{cases} \Theta(h(n)) = \Theta(n^{\log_\beta \alpha}) & \text{if } \alpha > \beta^d & [\text{ case 1 }] \\ \Theta(f(n)) = \Theta(n^d) & \text{if } \alpha < \beta^d & [\text{ case 2 }] \\ \Theta(f(n)\ lg\ n) = \Theta(n^d\ lg\ n) & \text{if } \alpha = \beta^d & [\text{ case 3 }] \end{cases} \qquad \text{(IX)}$$

Note that the asymptotic solution above does not depend on the two constants c and γ.

A sketch of the proof is as follows. Use equation (VIII) with $f(n) = cn^d$. The term of the summation can be simplified to

$$\sum_{i=1}^{k} \frac{f(\beta^i)}{\alpha^i} = c \sum_{i=1}^{k} \left(\frac{\beta^d}{\alpha} \right)^i$$

Let $x = \beta^d/\alpha$. The above summation depends on the value of x. If $x = 1$, that is case 3 above, then the above summation simplifies to $ck = \Theta(lg\ n)$. On the other hand, if $x \neq 1$, then the above summation forms a geometric series. If $x < 1$, that is case 1, then the geometric summation converges, and hence is no more than a constant. Thus, the above summation simplifies to $\Theta(1)$. If $x > 1$, that is case 2, then the geometric summation can be simplified by noting that

$$\sum_{i=1}^{k} x^i = \frac{x}{x-1} \cdot (x^k - 1) = \Theta(x^k) = \Theta\left(\frac{\beta^{kd}}{\alpha^k} \right) = \Theta\left(\frac{n^d}{h(n)} \right).$$

□

The case of polynomial driving function occurs quite frequently in practice. Therefore, the above simple solution is worthwhile to remember. For a more general version and its proof, see *"The Master Theorem"* in [CLR, page 62].

Exercise 2: Give a complete proof of the above solution for the case of the polynomial driving function by filling in the details in the above proof sketch. □

Example 1 (revisited again): We have $\alpha = \beta = 2$ and $f(n) = n^d$ with $d = 1$. Comparing α and β^d, we see that $\beta^d = 2 = \alpha$. Thus, case 3 above, holds, i.e. $T(n) = \Theta(n^d\ lg\ n) = \Theta(n\ lg\ n)$, as we expected. □

Exercise 3: Generalize the above Master Theorem for the case $f(n) = \Theta(n^d\ lg^t\ n)$ for some constants d , $t \ge 0$. Show that essentially the same solution holds as in (IX). That is,

$$T(n) = \begin{cases} \Theta(h(n)) & \text{if } \alpha > \beta^d \\ \Theta(f(n)) & \text{if } \alpha < \beta^d \\ \Theta(f(n)\ lg\ n) & \text{if } \alpha = \beta^d \end{cases}$$

Hint: use integration to obtain tight bounds on the summations involved in the proof. □

---

2. FULL HISTORY RECURRENCES

A second type of recurrence relation which usually arises in some average-case analyses has the following typical form:

$$T(n) = \sum_{i=0}^{n-1} T(i) + f(n) \quad \forall n \geq 0\,. \tag{X}$$

That is, $T(0) = f(0)$, $T(1) = T(0) + f(1)$, $T(2) = T(0) + T(1) + f(2)$, and so on. This recurrence appears to be more complicated, since not just one but many terms of "T" appear on the right hand side of the equation.

For $n > 0$ we may substitute $n-1$ for n and get:

$$T(n-1) = \sum_{i=0}^{n-2} T(i) + f(n-1)\,. \tag{XI}$$

Subtract (XI) from (X) to get:

$$T(n) - T(n-1) = T(n-1) + f(n) - f(n-1)\,.$$

We conclude:

$$T(n) = \begin{cases} 2\,T(n-1) + [\ f(n) - f(n-1)\ ] & \forall n \geq 1 \\ f(0) \qquad \text{for } n = 0 \end{cases} \tag{XII}$$

Now the term "T" appears only once on the right hand side. We can now solve equation (XII) by repeated substitution. The details are left to the reader.

Exercise 4: Solve the following recurrences. Recurrence (b) arises in the average-case analysis of the *QuickSelect* algorithm, and recurrence (c) arises in the average-case analysis of the *Quick-Sort* algorithm.

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

$$T(n) = \frac{1}{n}\sum_{i=0}^{n-1} T(i) + n \tag{b}$$

$$T(n) = \frac{2}{n}\sum_{i=0}^{n-1} T(i) + n \tag{c}$$

$$T(n) = \sum_{i=0}^{n-1} i \cdot T(i) + n \qquad \text{(d)}$$

Example 4: Let us solve Exercise 4(c) †. First multiply the equation by n to get:

$$n\,T(n) = 2\sum_{i=0}^{n-1} T(i) + n^2 \qquad \forall n \geq 0\,. \qquad \text{(c1)}$$

Now substitute $n-1$ for n in (c1) to get:

$$(n-1)\,T(n-1) = 2\sum_{i=0}^{n-2} T(i) + (n-1)^2 \qquad \forall n \geq 1\,. \qquad \text{(c2)}$$

Subtract (c2) from (c1) and simplify the terms to get:

$$n\,T(n) = (n+1)\,T(n-1) + 2n - 1 \qquad \forall n \geq 1\,. \qquad \text{(c3)}$$

Divide (c3) by $n(n+1)$ to get:

$$\frac{T(n)}{n+1} = \frac{T(n-1)}{n} + \frac{2n-1}{n(n+1)} \qquad \forall n \geq 1 \qquad \text{(c4)}$$

Observe that the first term on the right hand side is similar to the term on the left hand side, and note also that:

$$\frac{2n-1}{n(n+1)} = \frac{3}{n+1} - \frac{1}{n} \qquad \text{(c5)}$$

Using the above observation, define:

$$Q(n) = \frac{T(n)}{n+1} \qquad \forall n \geq 0 \qquad \text{(c6)}$$

Substitute (c5) and (c6) in (c4) to get:

$$Q(n) = \begin{cases} Q(n-1) + \dfrac{3}{n+1} - \dfrac{1}{n} & \forall n \geq 1 \\ 0 \quad \text{for } n = 0 \end{cases} \qquad \text{(c7)}$$

Now use repeated substitution to solve (c7):

$$Q(n) = \left(\frac{3}{n+1} - \frac{1}{n}\right) + \left(\frac{3}{n} - \frac{1}{n-1}\right) + \left(\frac{3}{n-1} - \frac{1}{n-2}\right) + \cdots + \left(\frac{3}{2} - \frac{1}{1}\right) + Q(0) \qquad \text{(c8)}$$

$$= 3\sum_{i=2}^{n+1} \frac{1}{i} - \sum_{i=1}^{n} \frac{1}{i} = 2H_n - \frac{3n}{n+1}$$

Finally, from (c6) and (c8) we obtain:

$$T(n) = 2\,(n+1)\,H_n - 3n \qquad \text{(c9)}$$

Since $H_n = \ln n + O(1) = \Theta(lg\ n)$, we conclude $T(n) = \Theta(n\ lg\ n)$. □

---

† Note the resemblance to the integral equation $T(x) = \dfrac{2}{x}\displaystyle\int_0^x T(y)\,dy + x$. Try to solve this integral equation similar to the (discrete) method taken for the solution of Exercise 4(c).

3. THE GUESS-THEN-VERIFY TECHNIQUE

One method that has proven to be powerful if used effectively is the *guess-then-verify* method. To illustrate the method let us start with a simple example.

Example 5: Consider the recurrence:

$$T(n) = \begin{cases} T(n-1)+n & \text{if } n \geq 1 \\ 0 & \text{if } n = 0 \end{cases}$$

This recurrence could easily be solved by the iterated substitution method. But let us guess a solution, then verify its correctness by induction on n.

First Guess: Let us guess $T(n) = O(n)$. So, we want to verify that $T(n) \leq an$ for all $n \geq n_o$, for some constants a and n_o to be determined. Let us "verify" this guess by induction on n.

The base case ($n = 0$) is $T(0) = 0 \leq a \cdot 0$, which is valid. For the inductive step ($n \geq 1$), assume the induction hypothesis $T(i) \leq a \cdot i$ for all $i < n$, then try to verify that $T(n) \leq an$. From the recurrence and the induction hypothesis for $i = n-1$, we have $T(n) = T(n-1) + n \leq a(n-1) + n = (a+1)n - a$. If $(a+1)n - a \leq an$, we would conclude $T(n) \leq an$ and complete the verification. But, there is no positive constant a that would satisfy $(a+1)n - a \leq an$ for all sufficiently large n.

The conclusion is that our "guess" was wrong! We have to make a new guess and verify it. Should we guess low or high? We should guess high. Why? Because, in the last crucial inequality that we needed to be satisfied (i.e., $(a+1)n - a \leq an$) we see that our guess, i.e., the right hand side of the inequality is too low compared to the left hand side. This would be the case even if we added some lower order terms to our guess, like $T(n) \leq an + o(n)$. Here is the key idea: *in the last crucial inequality, the dominating terms on each side should be balanced*. Once we have figured out the dominating term, we can carry out the same guess-then-verify idea in determining the lower order terms. After gaining some experience in the use of the method, we can become more efficient in guessing the right solution quickly. (See also page 55 of [CLR] on "making a good guess".) Let us carry on our guessing game for now!

Second Guess: Let us guess $T(n) = \Omega(n^3)$. So, let's verify that $T(n) \geq an^3$. The basis goes through fine again. In the inductive part we get $T(n) = T(n-1) + n \geq a(n-1)^3 + n \geq an^3$. the last crucial inequality is not balanced and shows our guess is too high. This would be the case even if we subtracted some lower order terms to our guess, like $T(n) \geq an^3 - o(n^3)$.

Third Guess: So, so far, we know that $T(n)$ is strictly above $O(n)$, and strictly below $\Omega(n^3)$. Not to drag the discussion too long, let us guess that $T(n) = O(n^2)$. We will verify that $T(n) \leq an^2 + bn + c$, for some constants a, b and c to be determined. The basis for $n = 0$ gives us $T(0) = 0 \leq a \cdot 0^2 + b \cdot 0 + c$. This is satisfied if $c \geq 0$. The inductive part gives us $T(n) = T(n-1) + n \leq a(n-1)^2 + b(n-1) + c = an^2 + (1 + b - 2a)n + (a - b + c)$. It would be sufficient to have the latter quantity to be $\leq an^2 + bn + c$. That is we want $an^2 + (1 + b - 2a)n + (a - b + c) \leq an^2 + bn + c$. At last, the dominating terms on both sides of this last crucial inequality balance up. For the inequality to hold for all $n \geq 1$, it is sufficient to match the left and right hand side term by term, that is, to have $(1 + b - 2a) \leq b$ (i.e., $a \geq 1/2$) and $(a - b + c) \leq c$ (i.e., $b \geq a$). So, all the conditions we have on the constants a, b and c from the

basis and the inductive step are $c \geq 0$ and $b \geq a \geq 1/2$. We see the choice $a = b = 1/2$ and $c = 0$ works. That is, we have verified that $T(n) \leq (n^2 + n)/2$ for all $n \geq 0$. This proves that $T(n) = O(n^2)$. In fact, it should now be easy to see from the verification steps that $T(n) = (n^2 + n)/2$. Hence, $T(n) = \Theta(n^2)$. □

Exercise 5: Solve the recurrence $T(n) = T(n-1) + n^2$, for $n \geq 1$, and and $T(0) = 0$, using the guess-then-verify technique. Keep the unsuccessful trials as scratch work and just give the final successful trial. □

Example 6: (Problem 4-4(b), page 73 of [CLR]) Consider the recurrence

$$T(n) = 3T(n/3 + 5) + n/2$$

For this recurrence to be well defined, we must have $n/3 + 5 < n$, i.e., $n \geq 8$. So we assume the boundary condition $T(n) = c$ for all $n < 8$, where c is some positive constant.

We will use the guess-then-verify technique. As a first approximation, we notice that for asymptotically large n, the recurrence is close to $T'(n) = 3T'(n/3) + \Theta(n)$. Using the Master Theorem, we know the latter recurrence has the solution $T'(n) = \Theta(n \lg n)$. Going back to the original recurrence, we now prove the following.

Claim: $T(n) = \Theta(n \lg n)$.

Proof: We need to show the following two facts:

(i) $T(n) = O(n \lg n)$, and

(ii) $T(n) = \Omega(n \lg n)$.

Here we only show the proof of (i). Part (ii) can also be proved by the guess-then-verify method. Another way to prove part (ii) is to notice that $T(n) \geq T'(n)$, then apply the lower bound on $T'(n)$ to $T(n)$.

Proof of (i):

We will show that $T(n) \leq (an - b) \lg n$, say, for all $n \geq 10$, for some suitable constants a and b. We will prove this by induction on the number of applications of the recurrence (coarsely speaking, on n).

We first notice that

$$\frac{n}{3} + 5 \leq \frac{n}{2} \quad \forall\ n \geq 30 \tag{6.1}$$

Basis ($10 \leq n < 30$):
Here we can make $T(n) \leq (an - b) \lg n$ for all n in the range $10 \leq n < 30$, if we choose the constant a large enough compared to constant b. More specifically, we must have

$$b \leq an - \frac{T(n)}{\lg n} \quad \text{for } 10 \leq n \leq 30\,. \tag{6.2}$$

Note that in the above inequality, for the bounded range $10 \leq n \leq 30$, the values of $T(n)$ and hence $T(n) / \lg n$ are bounded.

Induction Step ($n \geq 30$):

$$T(n) = 3T(n/3 + 5) + n/2$$

$$\le 3(\, a(n/3+5) - b\,)\, lg\,(n/3+5) + n/2 \quad \{\text{by the induction hyp.}\}$$

$$\le 3(\, a(n/3+5) - b\,)\, lg\, n/2 + n/2 \quad \{\text{by (6.1)}\}$$

$$= (an - b)\, lg\, n \;-\; [\,(a - 1/2)n - b] - (2b - 15a)\, lg\, n/2$$

$$\le (an - b)\, lg\, n\,.$$

The last inequality will hold if

$[(a - 1/2)n - b] \ge 0$ and
$(2b - 15a) lg\, n/2 \;\ge\; 0.$

The first inequality is implied by $a \ge 1/2$ and $b \le 30(a - 1/2)$ (which is $\le (a - 1/2)n$). The second inequality is satisfied if $b \ge \frac{15}{2}\, a$.

From the basis and the induction step, we have the following conditions for the existence of constants a and b:

$$a \;\ge\; \frac{1}{2}\,, \tag{6.3}$$

$$\frac{15}{2}\, a \le b \le \min_{10 \le n \le 30} \left(an - \frac{T(n)}{lg\, n}\,,\; 30a - 15 \right). \tag{6.4}$$

In eq. (6.4) for such a constant b to exist, we must have

$$\frac{15}{2}\, a \le \min_{10 \le n \le 30} \left(an - \frac{T(n)}{lg\, n}\,.\; 30a - 15 \right). \tag{6.5}$$

We can certainly choose a large enough to simultaneously satisfy eqs. (6.3) and (6.5), since the coefficient of a on the left hand side of eq. (6.5) is 7.5, while on the right hand side, the coefficient of a is larger (it is at least 10). This completes the proof. □

---

4. THE VARIABLE SUBSTITUTION METHOD

Consider the recurrence

$$T(n) \;=\; \begin{cases} T(\frac{n}{2}) + lg\, n & \forall n > 1 \\ 0 & \forall n \le 1 \end{cases}$$

Let us change the variable n to a new variable $k = lg\, n$ (i.e., $n = 2^k$). Now the recurrence becomes $T(2^k) = T(2^{k-1}) + k$, for $k > 0$. $T(2^k)$ is a function of k; let's rename it as $S(k)$. So, the recurrence is

$$S(k) \;=\; \begin{cases} S(k-1) + k & \forall k > 0 \\ 0 & \forall k \le 0 \end{cases}$$

Now with a simple iteration method, we see the solution is $S(k) = k + (k-1) + (k-2) + \cdots + 1 = \Theta(k^2)$. Thus, $T(n) = T(2^k) = S(k) = \Theta(k^2) = \Theta(lg^2 n)$. □

Example 6 (again): Consider the recurrence

$$T(n) = 3T(n/3 + 5) + n/2 .$$

To simplify this recurrence using the variable substitution method, let's change the variable n to k such that $n = g(k)$ and $g(k-1) = n/3 + 5 = g(k)/3 + 5$. Now rename $T(g(k))$, a function of k, as $S(k)$. We now have the following two recurrences to solve:

$$g(k) = 3g(k-1) - 15 \qquad \text{and} \tag{6.1}$$

$$S(k) = 3S(k-1) + \frac{1}{2}\, g(k) \tag{6.2}$$

You should also carefully set up their boundary conditions. First solve eq. (6.1) to obtain $g(k)$ as a function of k. Then plug $g(k)$ in eq. (6.2) and solve it for $S(k)$. We now have a solution for $S(k) = T(g(k))$ as a function of k. We need the functional inverse $k = g^{-1}(n)$ to obtain $T(n) = S(k) = S(g^{-1}(n))$ as a function of n. We leave the details as exercise for the reader. □

Exercise 6: Show the following identities hold for any real number series (a_i):

$$\sum_{i=1}^{n} a_i \;=\; n\, a_n - \sum_{i=1}^{n-1} i\,(a_{i+1} - a_i) \;=\; (n+1)\, a_n - a_1 - \sum_{i=2}^{n} i\,(a_i - a_{i-1}) .$$

[Note: this is the discrete version of the integration by parts $\int f(x)\,dx = xf(x) - \int x\, df(x)$.]

Exercise 7: Using exercise 6, derive closed form formulas for:

(a) $\displaystyle\sum_{i=1}^{n} \lfloor \lg i \rfloor$

(b) $\displaystyle\sum_{i=1}^{n} H_i \;=\; \sum_{i=1}^{n} \sum_{j=1}^{i} \frac{1}{j}$

(c) $\displaystyle\sum_{i=1}^{n} \sum_{j=1}^{n} \frac{1}{i+j}$

Exercise 8: Consider the recurrence $T(n) = T(\alpha n) + T(\beta n) + n$, where $0 < \alpha < 1$ and $0 < \beta < 1$ are constants. Show that $T(n) = \Theta(n)$ if $\alpha + \beta < 1$, and $T(n) = \Theta(n \lg n)$ if $\alpha + \beta = 1$. What if $\alpha + \beta > 1$?

Exercise 9: The exact recurrence for the worst-case number of comparisons in MergeSort is:

$$T(n) \;=\; \begin{cases} T(\lceil \frac{n}{2} \rceil) + T(\lfloor \frac{n}{2} \rfloor) + n - 1 & \text{for } n > 1 \\ 0 & \text{for } n \le 1 \end{cases}$$

Using induction on n, verify the exact solution to the recurrence is:

$$T(n) \;=\; n \lceil \lg n \rceil - 2^{\lceil \lg n \rceil} + 1 .$$

After giving it a good try, compare your solution to Parberry's article "Everything you wanted to know about the running time of mergesort but were afraid to ask" in ACM SIGACT News, 29(2), June 1998, pp. 50-57.

Exercise 10: Solve $T(n) = T(n/2 + 5) + c$, where $c > 0$ is some constant.
[Hint: substitute $n = g(k)$, where $g(k-1) = n/2 + 5$, i.e., $g(k) = 2g(k-1) - 10$, and $g(0) = 11$. Solve for $g(k)$ and substitute it in the original recurrence.]

Exercise 11: Solve $T(n) = T(n/2 + \sqrt{n}) + n$ by the guess-then-verify technique. Assume the usual boundary condition $T(O(1)) = O(1)$.

Exercise 12: Let $T(n) = \max_{1 \le k \le n/2} \{ T(k) + T(n-k) + k \}$ for $n > 2$, and $T(n) = 1$ for $n \le 2$. Show that $T(n) = O(n \ lg \ n)$.

Exercise 13: Solve the recurrence $T(n) = (T(n-1))^2$ for $n > 0$, and $T(0) = 2$.
[Hint: substitute $g(n) = lg\ (T(n))$.]

Exercise 14: Solve the recurrence $T(n) = T(lgn) + 1$ for $n>2$, and $T(n) = 2$ for $n \le 2$.