ANALYZING RECURRENCE RELATIONS

Consider a divide-and-conquer whose running time can be expressed by the recurrence

$$T(n) = aT\left(\frac{n}{b}\right) + f(n), \quad T(1) = 1.$$

where

$T(n)$ = Time to Solve a problem of size n.

$n \geq 1$

$a \geq 1$ is a constant } Independent of n.

$b > 0$ is a constant }

$f(n)$ is a function on n.

Examples: $T(n) = 2T\left(\frac{n}{2}\right) + n$ [Merge Sort]

$T(n) = T\left(\frac{n}{2}\right) + 1$ [Binary Search]

$T(n) = 3T\left(\frac{n}{2}\right) + O(n)$ [n-bit Multiplicati

$T(n) = 7T\left(\frac{n}{2}\right) + O(n^2)$ [Matrix Multiplicati

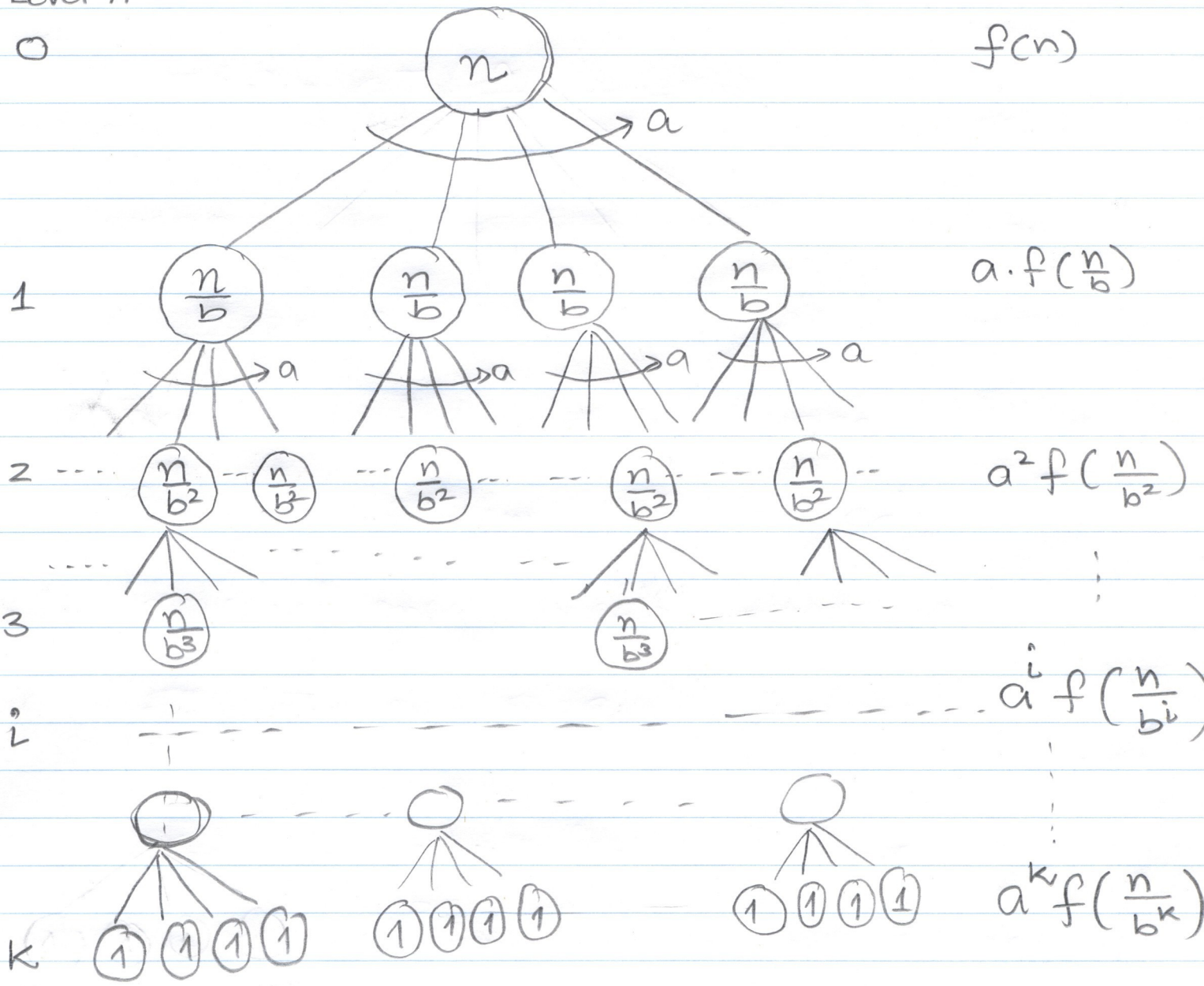
The Recurrence

$$T(n) = aT\left(\frac{n}{b}\right) + f(n)$$

means to solve the problem of size n, we subdivide it into a subproblems, each of size n/b. Each subproblem is identical to original one, except that it is of smaller size.

The quantity $f(n)$ is time to divide ~~and~~ plus time to merge the solution of the subproblems.

This can be visualized with the help of a recurrence tree.



Assume $n = b^k$ and root is at level 0.

Quantities that are of interest to us w.r.t to this tree are

(1) # Node at Level $i = a^i$

(2) Work done at each node in Level $i = f\left(\frac{n}{b^i}\right)$

~~Work done for the whole level = (# nodes at level i)~~ *

(3) # levels $= \log_b n$

(4) # leaves $= a^{\log_b n} = n^{\log_b a}$

Then $T(n) = \sum_{i=0}^{\#\text{levels}}$ Work done at level i

$$= \sum_{i=0}^{\#\text{levels}} \left(a^i\right)\left(f\left(\frac{n}{b^i}\right)\right)$$

a^i: # nodes at level i; $f\left(\frac{n}{b^i}\right)$: Work done at node i.

$$\boxed{T(n) = \sum_{i=0}^{\log_b n} a^i f\left(\frac{n}{b^i}\right)}$$

Example 1: Merge Sort

$$T(n) = 2T\left(\frac{n}{2}\right) + n; \quad T(1) = 1.$$

$$a = 2; \quad b = 2; \quad f(n) = n$$

Then

$$T(n) = \sum_{i=0}^{\text{levels}} \left(\begin{matrix}\#\text{ nodes} \\ \text{at level } i\end{matrix}\right)\left(\begin{matrix}\text{Work done} \\ \text{at each} \\ \text{node of} \\ \text{level } i\end{matrix}\right)$$

$$= \sum_{i=0}^{\log_b n} a^i f\left(\frac{n}{b^i}\right)$$

$$= \sum_{i=0}^{\log_b n} a^i \cdot \frac{n}{b^i}$$

$$= \sum_{i=0}^{\log_2 n} 2^i \cdot \frac{n}{2^i} \quad \neq$$

$$= \sum_{i=0}^{\log_2 n} n$$

$$= O(n \log n)$$

Example 2: $T(n) = 2T(\frac{n}{3}) + O(n)$

$T(1) = 1$

$a = 2; \quad b = 3; \quad f(n) \leq cn$

Then,

$$T(n) = \sum_{i=0}^{\log_b n} a^i f\left(\frac{n}{b^i}\right)$$

$$\leq \sum_{i=0}^{\log_3 n} 2^i c \frac{n}{3^i}$$

$$= cn \sum_{i=0}^{\log_3 n} \frac{2^i}{3^i}$$

$$= cn \sum_{i=0}^{\log_3 n} \left(\frac{2}{3}\right)^i \quad \text{— (I)}$$

Note that the sum of series

$1 + r + r^2 + \cdots + r^K = \frac{r^{K+1} - 1}{r - 1}$ for $r \neq 1$.

Therefore in (I) $r = 2/3$, and we obtain

$$= cn \left[\frac{\left(\frac{2}{3}\right)^{\log_3 n} - 1}{\frac{2}{3} - 1} \right]$$

$$= cn \left[\frac{1 - \left(\frac{2}{3}\right)^{\log_3 n}}{1/3} \right]$$

$$= 3cn \left[1 - \left(\frac{2}{3}\right)^{\log_3 n} \right] \leq 3cn$$

Hence, $T(n) = O(n)$.

Example 3: $T(n) = 3T(\frac{n}{2}) + O(n)$

This is the recurrence for multiplying two n-bit integers, using Gauss's trick.

Here $a=3$, $b=2$, $f(n)=cn$.

Therefore,

$$T(n) = \sum_{i=0}^{\log_b n} a^i f\left(\frac{n}{b^i}\right)$$

$$= \sum_{i=0}^{\log_2 n} 3^i c\left(\frac{n}{2^i}\right)$$

$$= cn \sum_{i=0}^{\log_2 n} \left(\frac{3}{2}\right)^i$$

$$= cn\left[1 + \frac{3}{2} + \left(\frac{3}{2}\right)^2 + \cdots + \left(\frac{3}{2}\right)^{\log_2 n}\right]$$

$$= cn\left[\frac{\left(\frac{3}{2}\right)^{\log_2 n + 1} - 1}{\frac{3}{2} - 1}\right]$$

$$= 2cn\left[\left(\frac{3}{2}\right)\left(\frac{3}{2}\right)^{\log_2 n} - 1\right]$$

$$\leq 2cn\left(\frac{3}{2}\right)\left(\frac{3^{\log_2 n}}{2^{\log_2 n}}\right) = \frac{3cn \cdot 3^{\log_2 n}}{n}$$

$$= O(n^{\log_2 3}).$$

Example 4: $T(n) = 7T\left(\frac{n}{2}\right) + O(n^2)$

This is the recurrence relation due to Strassen for matrix multiplication.

Here, $a=7$, $b=2$, $f(n) = cn^2$

Then, $T(n) = \sum_{i=0}^{\log_2 n} 7^i \cdot c \cdot \left(\frac{n}{2^i}\right)^2$

$$T(n) = \sum_{i=0}^{\log_2 n} cn^2 \cdot \frac{7^i}{2^i \cdot 2^i}$$

$$= cn^2 \sum_{i=0}^{\log_2 n} \left(\frac{7}{4}\right)^i$$

$$= cn^2 \left[1 + \frac{7}{4} + \left(\frac{7}{4}\right)^2 + \cdots + \left(\frac{7}{4}\right)^{\log_2 n}\right]$$

$$= cn^2 \left[\frac{\left(\frac{7}{4}\right)^{\log_2 n + 1} - 1}{\frac{7}{4} - 1}\right]$$

$$= \frac{4}{3} cn^2 \left[\frac{7}{4}\left(\frac{7}{4}\right)^{\log_2 n} - 1\right] \leq \frac{4}{3} cn^2 \cdot \frac{7}{4}\left(\frac{7}{4}\right)^{\log_2 n}$$

$$= \frac{7}{3} cn^2 \left(\frac{7^{\log_2 n}}{4^{\log_2 n}}\right) = \frac{7}{3} cn^2 \cdot \frac{7^{\log_2 n}}{n^2}$$

$$= O\left(n^{\log_2 7}\right).$$

Example 5: $T(n) = T\left(\frac{n}{2}\right) + 1$

This is recurrence relation for Binary Search

Here $a=1,\ b=2,\ f(n)=1$

$$T(n) = \sum_{i=0}^{\log_2 n} a^i f\left(\frac{n}{b^i}\right)$$

$$= \sum_{i=0}^{\log_2 n} 1^i \cdot 1$$

$$= \sum_{i=0}^{\log_2 n} 1$$

$$= O(\log_2 n). \quad \blacksquare$$

Substitution Method

Consider the recurrence

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

or

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

It can be seen that they are not of the form $T(n) = aT\left(\frac{n}{b}\right) + f(n)$ for $a \geq 1$, $b > 0$ constants.

Hence, we cannot apply previous technique.

Here, we use the following method.

Step 1: Guess a solution.

Step 2: Verify that the solution works, using induction on n (i.e the problem size).

Consider $T(n) = T(n-1) + n$; $T(1) = 1$.

→ Lets say that $T(n) = O(n^2)$.

Thats the Guess (Step 1).

→ Step 2: Verify $T(n) \stackrel{?}{=} O(n^2)$.

i, e. we need to check $T(n) \leq cn^2$

for some constant $c \geq 1$.

We will show that $T(n) = T(n-1) + n \stackrel{?}{\leq} cn^2$

by induction on n.

Base case: $T(1) = 1 \stackrel{?}{\leq} cn^2 = c \cdot 1^2$

i.e, $T(1) = 1 \stackrel{?}{\leq} c$ — (I)

$T(2) = T(1) + 2 \stackrel{?}{\leq} c \cdot 2^2$

i.e, $T(2) = 3 \stackrel{?}{\leq} 4c$ — (II)

(Note that we can always choose a c such that (I) and (II) are satisfied, e.g. $c = 1$. But we will choose c once we have done the last step of induction

→ Lets assume that (INDUCTION HYPOTHESIS)

$$T(k) \leq ck^2 \text{ for all values of}$$

$$1 \leq k \leq n-1$$

→ Now we show that

$$T(n) = T(n-1) + n \overset{?}{\leq} cn^2$$

Proof: Since $T(n-1) \leq c(n-1)^2$ by Induction Hypothesis for $k = n-1$, we need to show that

$$T(n) = T(n-1) + n \leq c\cdot(n-1)^2 + n \overset{?}{\leq} cn^2$$

$$\Leftrightarrow \quad c[n^2 - 2n + 1] + n \overset{?}{\leq} cn^2$$

$$\Leftrightarrow \quad cn^2 - 2cn + c + n \overset{?}{\leq} cn^2$$

$$\Leftrightarrow \quad -2cn + c + n \overset{?}{\leq} 0 \quad \text{— (III)}$$

QUESTION IS: Is there a constant c, that satisfies I, II and III.

Answere is yes, since $c = 1$ satisfies I, II and III becomes $-2n + 1 + n \overset{?}{\leq} 0$

$\Leftrightarrow 1 - n \overset{?}{\leq} 0$. This is true, since $n \geq 1$.

Example II: $T(n) = T(\frac{n}{3}) + T(\frac{2n}{3}) + n$

$T(1) = 1$.
$T(2) = 1$
$T(3) = 1$

Guess: $T(n) = O(n \log n)$.

Verification: To show that

$$T(n) = T(\tfrac{n}{3}) + T(\tfrac{2n}{3}) + n \overset{?}{\leq} cn \log n.$$

For small values of $n \overset{1,2,3,4,5,6}{=}$, we can easily check that this holds.

Assume that $T(k) \leq ck \log k$ for all values $1 \leq k \leq n-1$.

Now we show that $T(n) \leq cn \log n$.

Proof: $T(n) = T(\frac{n}{3}) + T(\frac{2n}{3}) + n$.

since $n/3 < n$ and $2n/3 < n$, we can use Induction Hypothesis for $T(\frac{n}{3})$ and $T(\frac{2n}{3})$.

i.e., we know $T(\frac{n}{3}) \leq c \cdot \frac{n}{3} \log \frac{n}{3}$

and $T(\frac{2n}{3}) \leq c \cdot \frac{2n}{3} \log \frac{2n}{3}$.

$$\Rightarrow T(n) = T\left(\frac{n}{3}\right) + T\left(\frac{2n}{3}\right) + n \overset{?}{\leq} cn\log n \quad -(I)$$

$$\Leftrightarrow c\frac{n}{3}\log_2\frac{n}{3} + c\frac{2n}{3}\log_2\frac{2n}{3} + n \overset{?}{\leq} cn\log n.$$

$$\Leftrightarrow c\frac{n}{3}\left[\log_2 n - \log_2 3\right] + c\frac{2n}{3}\left[\underbrace{\log_2 2n}_{1+\log_2 n} - \log_2 3\right] + n \overset{?}{\leq} cn\log n$$

$$\Leftrightarrow \underbrace{c\frac{n}{3}\log_2 n} - \underbrace{c\frac{n}{3}\log_2 3} + \underbrace{c\frac{2n}{3}\log_2 n} + c\frac{2n}{3} - \underbrace{c\frac{2n}{3}\log_2 3} + n \overset{?}{\leq} cn\log_2 n.$$

$$\Leftrightarrow cn\log_2 n - c\frac{n}{3}\log_2 3 + c\frac{2n}{3} + n \overset{?}{\leq} cn\log_2 n.$$

$$\Leftrightarrow -cn\log_2 3 + c\frac{2n}{3} + n \overset{?}{\leq} 0$$

Since $n \neq 0$ and $n > 0$, this implies we need to see whether

$$-c\log_2 3 + c\frac{2}{3} + 1 \overset{?}{\leq} 0$$

$$\text{or } c \overset{?}{\geq} \frac{1}{\log_2 3 - \frac{2}{3}} \approx \frac{1}{1.58 - 0.66} \approx \frac{1}{.9} \approx 1.1$$

Hence, choosing $c = 2$, will satisfy (I) and the claim that $T(n) = O(n\log n)$.

Question: Will $T(n) = T(\frac{n}{3}) + T(\frac{2n}{3}) + n \overset{?}{=} O(n)$

Lets verify this again by induction.

Then we need to eventually check whether

$$T(n) = T(\tfrac{n}{3}) + T(\tfrac{2n}{3}) + n \overset{?}{\leq} cn$$

or

$$\underbrace{c \cdot \tfrac{n}{3}}_{\text{I.H}} + \underbrace{c \cdot \tfrac{2n}{3}}_{\text{I.H}} + n \overset{?}{\leq} cn$$

$$\Leftrightarrow \quad cn + n \overset{?}{\leq} cn$$

$$\Leftrightarrow \quad n \overset{?}{\leq} 0$$

of couse that is not true and hence $T(n) \neq O(n)$. ∎