

# 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 Multiplication]

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

## 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 <sup>plus</sup> and time to merge the  
solution of the subproblems.

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

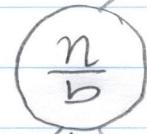
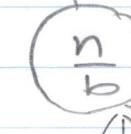
Level #

Work

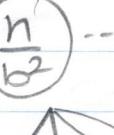
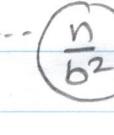
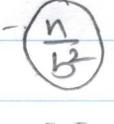
0

 $f(n)$  $\rightarrow a$ 

1

 $\rightarrow a$  $\rightarrow a$  $\rightarrow a$  $\rightarrow a$  $a \cdot f\left(\frac{n}{b}\right)$ 

2

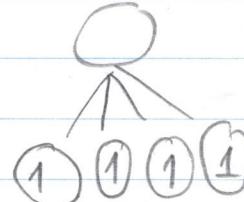
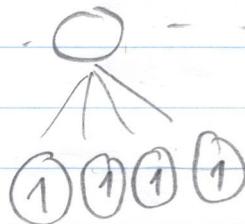
 $a^2 f\left(\frac{n}{b^2}\right)$ 

3

 $a^i f\left(\frac{n}{b^i}\right)$ 

i

K

 $a^K f\left(\frac{n}{b^K}\right)$ 

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

(4)

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 tree  
= (# nodes at level i) \* (~~

(3) # levels =  $\log_b n$

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

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

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

$\uparrow$   $\uparrow$   
# nodes at level i work done at node i.

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

# 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

$$\begin{aligned}
 T(n) &= \sum_{i=0}^{\text{levels}} (\# \text{ nodes at level } i) (\text{work done at each node of level } i) \\
 &= \sum_{i=0}^{\log_b n} a^i f\left(\frac{n}{b^i}\right) \\
 &= \sum_{i=0}^{\log_b n} a^i \cdot \frac{n}{b^i} \\
 &= \sum_{i=0}^{\log_2 n} 2^i \cdot \frac{n}{2^i} \quad \text{#} \\
 &= \sum_{i=0}^{\log_2 n} n
 \end{aligned}$$

$$= \mathcal{O}(n \log n)$$

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

$$T(1) = 1$$

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

Then,

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

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

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

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

Note that the sum of series  $1+r+r^2+\dots+r^k = \frac{r^{k+1}-1}{r-1}$  for  $r \neq 1$ .

Therefore in  $\textcircled{I}$   $r = \frac{2}{3}$ , and we

obtain

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

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

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

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

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

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

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

Therefore,

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

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

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

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

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

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

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

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

$$\text{Example 1: } T(n) = 7T\left(\frac{n}{2}\right) + O(n^2)$$

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

$$\text{Here, } a=7, b=2, f(n)=cn^2$$

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

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

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

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

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

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

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

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

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

This is recurrence relation for Binary Search

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

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

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

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

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

■

## Substitution Method

Consider the recurrence

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

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

That's 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  $\epsilon$   
once we have done the last step of induction)

→ Lets assume that (INDUCTION HYPOTHESIS)

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

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

→ Now we show that

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

Proof: Since  $T(n-1) \leq c(n-1)^2$  by

Induction Hypothesis for  $k=n-1$ ,

we need to show that

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

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

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

$$\Leftrightarrow -2cn + c + n \stackrel{?}{\leq} 0 \quad - \text{III}$$

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

Answer is Yes, since  $c=1$  satisfies I, II and III becomes  $-2n + 1 + n \stackrel{?}{\leq} 0$   
 $\Leftrightarrow 1 - n \stackrel{?}{\leq} 0$ . This is true, since  $n \geq 1$ .

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

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

$$\begin{cases} T(1) = 1 \\ T(2) = 1 \\ T(3) = 1 \end{cases}$$

Verification: To show that

$$T(n) = T\left(\frac{n}{3}\right) + T\left(\frac{2n}{3}\right) + n \stackrel{?}{\leq} cn \log n.$$

For small values of  $n = 1, 2, 3, 4, 5, 6$ , 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:  $T(n) = T\left(\frac{n}{3}\right) + T\left(\frac{2n}{3}\right) + n$ .

since  $\frac{n}{3} < n$  and  $\frac{2n}{3} < n$ , we can

use Induction hypothesis for

$T\left(\frac{n}{3}\right)$  and  $T\left(\frac{2n}{3}\right)$ .

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

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

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

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

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

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

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

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

Since  $n \neq 0$  and  $n > 0$ , this implies

we need to see whether

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

$$\text{or } c \geq \frac{1}{\log_2 3 - \frac{2}{3}} \approx \frac{1}{1.58 - 0.66} \approx \frac{1}{0.9} \approx 1.1$$

Hence, choosing  $c=2$ , will satisfy

$\textcircled{I}$  and the claim that  $T(n) = O(n \log n)$ .

Question: Will  $T(n) = T\left(\frac{n}{3}\right) + T\left(\frac{2n}{3}\right) + n$   
 $\stackrel{?}{=} \mathcal{O}(n)$

Lets verify this again by induction.

Then we need to eventually check whether

$$T(n) = T\left(\frac{n}{3}\right) + T\left(\frac{2n}{3}\right) + n \stackrel{?}{\leq} cn$$

2

$$c \cdot \underbrace{\frac{n}{3}}_{I \cdot H} + c \cdot \underbrace{\frac{2n}{3}}_{I \cdot H} + n \stackrel{?}{\leq} cn$$

$$\Leftrightarrow cn + n \leq cn ?$$

$$\Leftrightarrow n \stackrel{?}{\leq} 0$$

Of course that is not true and

hence  $T(n) \neq O(n)$ .