Recurrence

Three methods to solve recurrences:

- Substitution
- Recurrence-tree
- Master method

Assumptions
n is an integer

running time of Merger sort is really:

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

omit statements of boundary conditions
assume that T(n) is a constant for sufficiently small n.

The substitution method for solving recurrences entails two steps:
1. Guess the form of the solution.
2. Use mathematical induction to find the constants and show that the solution works.

Example

Determine an upper bound of the recurrence:

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

1 We guess that T(n) = O(n lg n).
2 Prove that T(n) <= c n lg n for c > 0.

Assume

$$T(\lfloor n/2 \rfloor) \le c \lfloor n/2 \rfloor \lg(\lfloor n/2 \rfloor)$$

Then

$$\begin{aligned} T(n) &\le 2(c \lfloor n/2 \rfloor \lg(\lfloor n/2 \rfloor)) + n \\ &\le cn \lg(n/2) + n \\ &= cn \lg n - cn \lg 2 + n \\ &= cn \lg n - cn + n \\ &\le cn \lg n , \end{aligned}$$

holds when c ≥ 1

3 Show that our solution holds for the boundary conditions.

We cannot do T(1) = 1. However, we are required only to prove
$T(n) \le c\, n \lg n$ for $n \ge n0$. Choose $n0 > 2$, then $T(2) \le c\, 2 \lg 2$, $T(3) \le c\, 3 \lg 3$.
Any choice of $c \ge 2$ suffices for the base cases of n = 2 and n = 3 to hold.

Changing variables

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

let m = lg n

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

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

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

S(m) = O (m lg m)

$$T(n) = T(2^m) = S(m) = O(m \lg m) = O(\lg n \lg \lg n).$$

Homework

1
$T(n) = T(\lceil n/2 \rceil) + 1$ is $O(\lg n)$.

2
$T(n) = 2T(\lfloor n/2 \rfloor + 17) + n$ is $O(n \lg n)$.

3 Using change of variables to solve
$T(n) = 2T(\sqrt{n}) + 1$

Recursion-tree

We will use recursion tree to generate a good guess.

Example

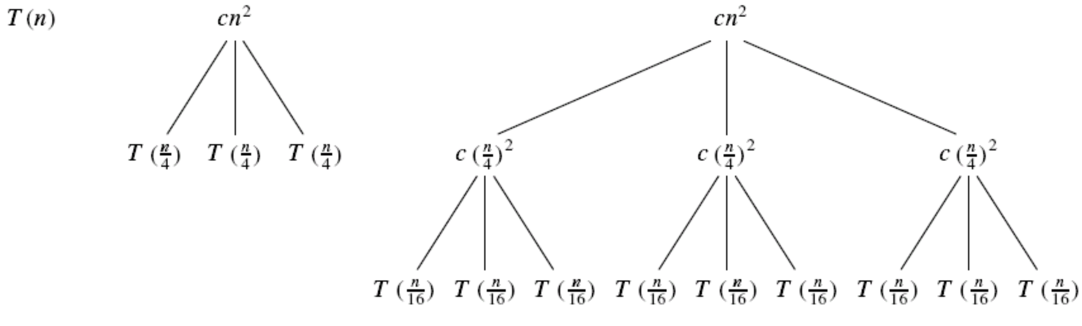
$$T(n) = 3T(\lfloor n/4 \rfloor) + \Theta(n^2).$$

We simplify it to

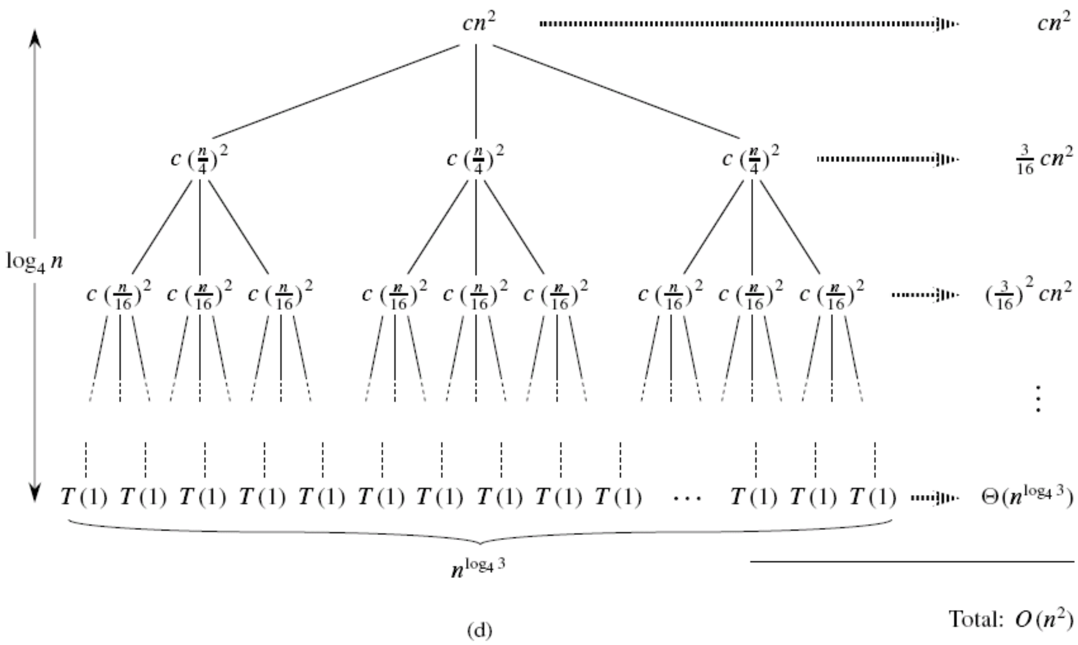
$$T(n) = 3T(n/4) + cn^2,$$

for c > 0.

we assume that *n* is an exact power of 4



(a) (b) (c)



(d)

What is the height of the tree?

The subproblem size for a node at depth i is $n/4^i$. Thus, the subproblem size hits n = 1 when $n/4^i = 1$ or, equivalently, when i = log4 n. Thus, the tree has log4 n + 1 levels (0, 1, 2, . . . , log4 n).

The number of nodes at depth *i* is 3^i.
Each node at depth i, for i = 0, 1, 2, . . . , log4 n − 1, has a cost of $c(n/4^i)^2$.

The total cost over all nodes at depth i , for i = 0, 1, 2, . . . , log4 n − 1, is $3^i c(n/4^i)^2 = (3/16)^i cn^2$. The last level, at depth log4 n, has $3^{\log_4 n} = n^{\log_4 3}$ nodes, each contributing cost T (1), for a total cost of $n^{\log_4 3}T(1)$, which is $\Theta(n^{\log_4 3})$.

the cost for the entire tree:

$$
\begin{aligned}
T(n) &= cn^2 + \frac{3}{16}cn^2 + \left(\frac{3}{16}\right)^2 cn^2 + \cdots + \left(\frac{3}{16}\right)^{\log_4 n-1} cn^2 + \Theta(n^{\log_4 3}) \\
&= \sum_{i=0}^{\log_4 n-1} \left(\frac{3}{16}\right)^i cn^2 + \Theta(n^{\log_4 3}) \\
&= \frac{(3/16)^{\log_4 n} - 1}{(3/16) - 1} cn^2 + \Theta(n^{\log_4 3}) .
\end{aligned}
$$

Use geometric series as an upper bound

$$
\begin{aligned}
T(n) &= \sum_{i=0}^{\log_4 n-1} \left(\frac{3}{16}\right)^i cn^2 + \Theta(n^{\log_4 3}) \\
&< \sum_{i=0}^{\infty} \left(\frac{3}{16}\right)^i cn^2 + \Theta(n^{\log_4 3}) \\
&= \frac{1}{1-(3/16)} cn^2 + \Theta(n^{\log_4 3}) \\
&= \frac{16}{13} cn^2 + \Theta(n^{\log_4 3}) \\
&= O(n^2) .
\end{aligned}
$$

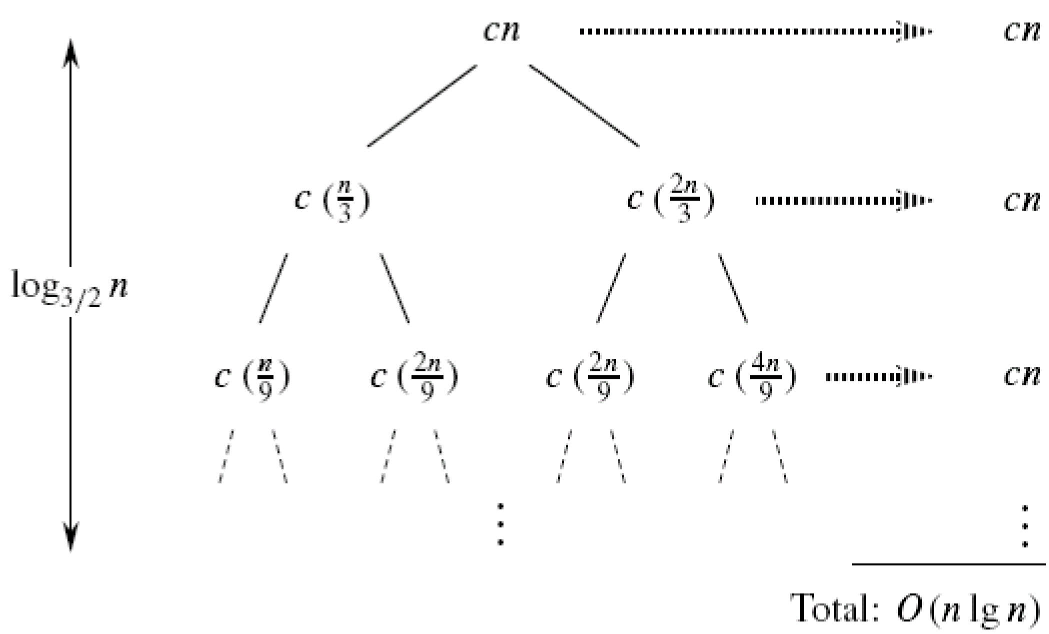
Use subsitution method to verify our guess.
We want to show that $T(n) \leq dn^2$ for some constant $d > 0$.

$$\begin{aligned}
T(n) &\le 3T(\lfloor n/4 \rfloor) + cn^2 \\
&\le 3d\lfloor n/4 \rfloor^2 + cn^2 \\
&\le 3d(n/4)^2 + cn^2 \\
&= \frac{3}{16}dn^2 + cn^2 \\
&\le dn^2 ,
\end{aligned}$$

last step holds as long as $d \ge (16/13)c$.

Example 2

T (n) = T (n/3) + T (2n/3) + O(n)



n ..(2/3)n .. $(2/3)^2$n .. .. 1. Since $(2/3)^k n = 1$ when $k = \log_{3/2} n$

We expect the solution to the recurrence to be at most the number of levels times the cost of each level, or $O(cn \log_{3/2} n) = O(n \lg n)$.

We can use the substitution method to verify that $O(n \lg n)$ is an upper bound for the solution to the recurrence. We show that $T(n) \le dn \lg n$, where d is a suitable positive constant.

$$
\begin{aligned}
T(n) &\le T(n/3) + T(2n/3) + cn \\
&\le d(n/3)\lg(n/3) + d(2n/3)\lg(2n/3) + cn \\
&= (d(n/3)\lg n - d(n/3)\lg 3) \\
&\qquad + (d(2n/3)\lg n - d(2n/3)\lg(3/2)) + cn \\
&= dn\lg n - d((n/3)\lg 3 + (2n/3)\lg(3/2)) + cn \\
&= dn\lg n - d((n/3)\lg 3 + (2n/3)\lg 3 - (2n/3)\lg 2) + cn \\
&= dn\lg n - dn(\lg 3 - 2/3) + cn \\
&\le dn\lg n\ ,
\end{aligned}
$$

where $d \geq c/(\lg 3 - (2/3))$.

Homework

1 Use recursion tree to determine a good asymptotic upper bound on the recurrence. Use the substitution method to verify your answer.

$$T(n) = 3T(\lfloor n/2 \rfloor) + n.$$

2 Draw the recursion tree for where c is a constant, and provide a tight asymptotic bound on its solution. Verify your bound by the substitution method.

$$T(n) = 4T(\lfloor n/2 \rfloor) + cn,$$

Master Method

Theorem 4.1 (Master theorem)

Let $a \geq 1$ and $b > 1$ be constants, let $f(n)$ be a function, and let $T(n)$ be defined on the nonnegative integers by the recurrence

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

where we interpret n/b to mean either $\lfloor n/b \rfloor$ or $\lceil n/b \rceil$. Then $T(n)$ can be bounded asymptotically as follows.

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

Examples

T (n) = 9T (n/3) + n .

For this recurrence, we have a = 9, b = 3, f (n) = n, and thus we have that $n^{\log b\ a} = n^{\log 3\ 9} = O(n^2)$. Since f (n) = $O(n^{\log 3\ 9 - e})$, where e = 1, we can apply case 1 of the master theorem and conclude that the solution is T (n) = $O(n^2)$.

T (n) = T (2n/3) + 1,

in which a = 1, b = 3/2, f (n) = 1, and $n^{\log b\ a} = n^{\log 3/2\ 1} = n^0 = 1$. Case 2 applies, since f (n) = $\Theta(n^{\log b\ a}) = \Theta(1)$, and thus the solution to the recurrence is T (n) = Θ(lg n).

T (n) = 3T (n/4) + n lg n ,

we have a = 3, b = 4, f (n) = n lg n, and $n^{\log b\ a} = n^{\log 4\ 3} = O(n^{0.793})$. Since f (n) = $\Omega(n^{\log 4\ 3 + e})$, where e ≈ 0.2, case 3 applies if we can show that the regularity condition holds for f (n). For sufficiently large n, a f (n/b) = 3(n/4) lg(n/4) ≤ (3/4)n lg n = c f (n) for c = 3/4. Consequently, by case 3, the solution to the recurrence is T (n) = Θ (n lg n).

T (n) = 2T (n/2) + n lg n ,

even though it has the proper form: a = 2, b = 2, f (n) = n lg n, and $n^{\log b\ a} = n$. It might seem that case 3 should apply, since f (n) = n lg n is asymptotically larger than $n^{\log b\ a} = n$. The problem is that it is not *polynomially* larger. The ratio f (n)/$n^{\log b\ a}$ = (n lg n)/n = lg n is asymptotically less than n^e for any positive constant e.

Proof of Master theorem can be seen in "Introduction to Algorithms" by CLR 2nd ed pp. 76- 81.