Algorithms

Recurrences

Recurrences

- When an algorithm contains a recursive call to itself, its running time can often described by a recurrence.
- A *recurrence* is an equation or inequality that describes a function in terms of its value of smaller inputs.

Recurrences—examples

- Summation

$$\sum_{i=1}^{n} i = n + \sum_{i=1}^{n-1} i$$

- Factorial

$$n! = n * (n-1)!$$

Recurrences—examples

- Summation

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

- Factorial

$$n! = n * (n-1)!$$

Termination conditions (boundary conditions)

- Summation

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

- Factorial

$$n! = n * (n-1)!$$
$$1! = 1$$

Recurrences—CS oriented examples

- Fibonacci number

$$F(n) = F(n-1) + F(n-2)$$

- Merge sort

$$T(n) = 2T(\frac{n}{2}) + \Theta(n)$$

Influence of the boundary conditions

- Fibonacci number

$$F(n) = F(n-1) + F(n-2)$$

- Merge sort

$$T(n) = 2T(\frac{n}{2}) + \Theta(n)$$

Recurrences

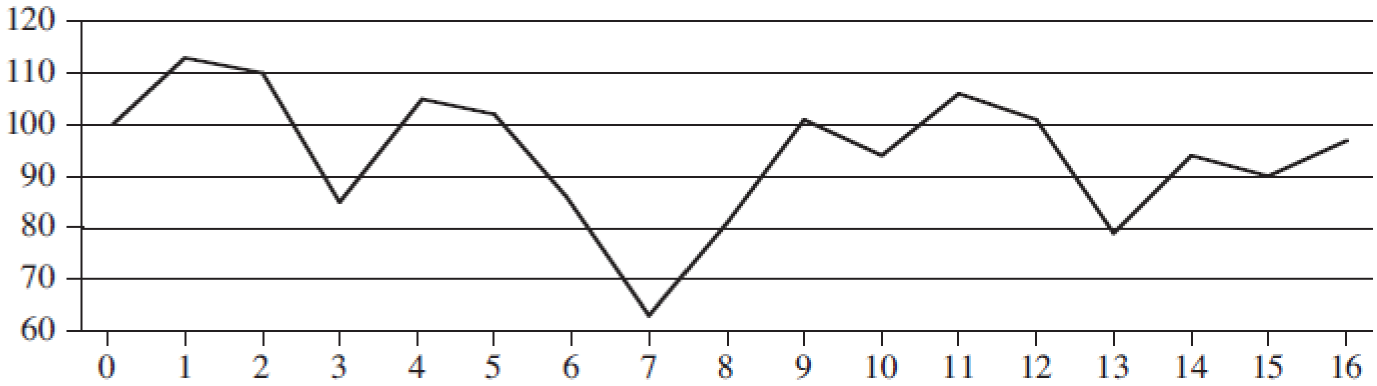
$$T(n) = aT(n/b) + f(n)$$

- *Substitution method*
- *Recursion-tree method*
- *Master method*

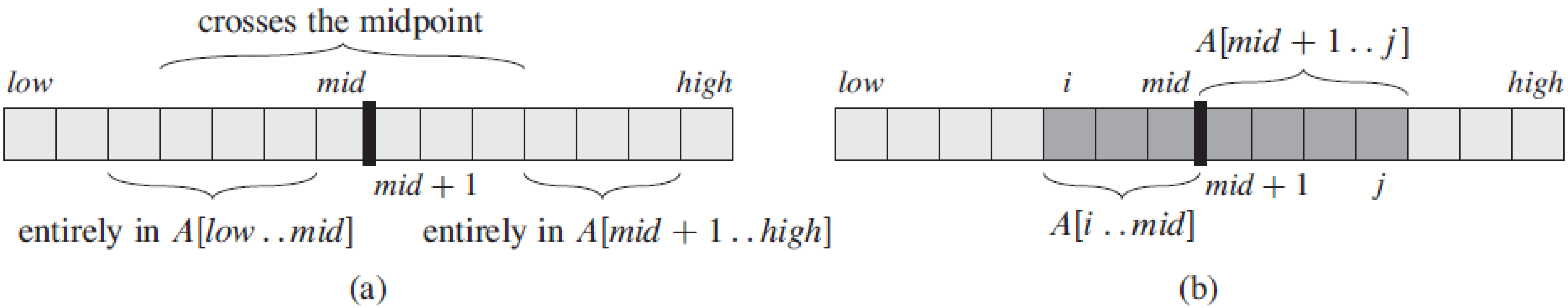
Technicalities

- Neglect certain technical details when stating and solving recurrences.
- A good example of a detail that is often glossed over is the assumption of integer arguments to functions.
- Boundary conditions is ignored.
- Omit floors, ceilings.

The maximum-subarray problem



Day	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Price	100	113	110	85	105	102	86	63	81	101	94	106	101	79	94	90	97
Change		13	−3	−25	20	−3	−16	−23	18	20	−7	12	−5	−22	15	−4	7




```
FIND-MAXIMUM-SUBARRAY(A, low, high)
if high == low
    return (low, high, A[low])            // base case: only one element
else mid = ⌊(low + high)/2⌋
    (left-low, left-high, left-sum) =
        FIND-MAXIMUM-SUBARRAY(A, low, mid)
    (right-low, right-high, right-sum) =
        FIND-MAXIMUM-SUBARRAY(A, mid + 1, high)
    (cross-low, cross-high, cross-sum) =
        FIND-MAX-CROSSING-SUBARRAY(A, low, mid, high)
    if left-sum ≥ right-sum and left-sum ≥ cross-sum
        return (left-low, left-high, left-sum)
    elseif right-sum ≥ left-sum and right-sum ≥ cross-sum
        return (right-low, right-high, right-sum)
    else return (cross-low, cross-high, cross-sum)
```

```
FIND-MAX-CROSSING-SUBARRAY(A, low, mid, high)
left-sum = -∞
sum = 0
for i = mid downto low
    sum = sum + A[i]
    if sum > left-sum
        left-sum = sum
        max-left = i
right-sum = -∞
sum = 0
for j = mid + 1 to high
    sum = sum + A[j]
    if sum > right-sum
        right-sum = sum
        max-right = j
return (max-left, max-right, left-sum + right-sum)
```

Matrix multiplication

$$c_{ij} = \sum_{k=1}^{n} a_{ik} \cdot b_{kj}$$

```
SQUARE-MATRIX-MULTIPLY(A, B)
n = A.rows
let C be a new n × n matrix
for i = 1 to n
    for j = 1 to n
        c_ij = 0
        for k = 1 to n
            c_ij = c_ij + a_ik · b_kj
return C
```

$\Theta(n^3)$

A simple divide-and-conquer algorithm

$$A = \begin{pmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{pmatrix}, \quad B = \begin{pmatrix} B_{11} & B_{12} \\ B_{21} & B_{22} \end{pmatrix}, \quad C = \begin{pmatrix} C_{11} & C_{12} \\ C_{21} & C_{22} \end{pmatrix}$$

$$\begin{pmatrix} C_{11} & C_{12} \\ C_{21} & C_{22} \end{pmatrix} = \begin{pmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{pmatrix} \cdot \begin{pmatrix} B_{11} & B_{12} \\ B_{21} & B_{22} \end{pmatrix}$$

```
SQUARE-MATRIX-MULTIPLY-RECURSIVE(A, B)
n = A.rows
let C be a new n × n matrix
if n == 1
    c11 = a11 · b11
else partition A, B, and C as in equations (4.9)
    C11 = SQUARE-MATRIX-MULTIPLY-RECURSIVE(A11, B11)
          + SQUARE-MATRIX-MULTIPLY-RECURSIVE(A12, B21)
    C12 = SQUARE-MATRIX-MULTIPLY-RECURSIVE(A11, B12)
          + SQUARE-MATRIX-MULTIPLY-RECURSIVE(A12, B22)
    C21 = SQUARE-MATRIX-MULTIPLY-RECURSIVE(A21, B11)
          + SQUARE-MATRIX-MULTIPLY-RECURSIVE(A22, B21)
    C22 = SQUARE-MATRIX-MULTIPLY-RECURSIVE(A21, B12)
          + SQUARE-MATRIX-MULTIPLY-RECURSIVE(A22, B22)
return C
```

$$T(n) = \begin{cases} \Theta(1) & \text{if } n = 1 , \\ 8T(n/2) + \Theta(n^2) & \text{if } n > 1 . \end{cases} \qquad \Theta(n^3)$$

Strassen's method

$$T(n) = \begin{cases} \Theta(1) & \text{if } n = 1 , \\ 7T(n/2) + \Theta(n^2) & \text{if } n > 1 . \end{cases}$$

$$T(n) = \Theta(n^{\lg 7}).$$

$$
\begin{array}{lllllll}
A_{11}\cdot B_{11} & + A_{11}\cdot B_{22} & + A_{22}\cdot B_{11} & + A_{22}\cdot B_{22} & & & \\
 & & - A_{22}\cdot B_{11} & & + A_{22}\cdot B_{21} & & \\
 & - A_{11}\cdot B_{22} & & & & - A_{12}\cdot B_{22} & \\
 & & & - A_{22}\cdot B_{22} & - A_{22}\cdot B_{21} & + A_{12}\cdot B_{22} & + A_{12}\cdot B_{21} \\
\hline
A_{11}\cdot B_{11} & & & & & & + A_{12}\cdot B_{21}
\end{array}
$$

$$
\begin{array}{lll}
A_{11}\cdot B_{12} & - A_{11}\cdot B_{22} & \\
 & + A_{11}\cdot B_{22} & + A_{12}\cdot B_{22} \\
\hline
A_{11}\cdot B_{12} & & + A_{12}\cdot B_{22}
\end{array}
$$

$$
\begin{array}{lll}
A_{21}\cdot B_{11} & + A_{22}\cdot B_{11} & \\
 & - A_{22}\cdot B_{11} & + A_{22}\cdot B_{21} \\
\hline
A_{21}\cdot B_{11} & & + A_{22}\cdot B_{21}
\end{array}
$$

$$
\begin{array}{lllllll}
A_{11}\cdot B_{11} & + A_{11}\cdot B_{22} & + A_{22}\cdot B_{11} & + A_{22}\cdot B_{22} & & & \\
 & - A_{11}\cdot B_{22} & & & + A_{11}\cdot B_{12} & & \\
 & & - A_{22}\cdot B_{11} & & & - A_{21}\cdot B_{11} & \\
- A_{11}\cdot B_{11} & & & & - A_{11}\cdot B_{12} & + A_{21}\cdot B_{11} & + A_{21}\cdot B_{12} \\
\hline
 & & & A_{22}\cdot B_{22} & & & + A_{21}\cdot B_{12}
\end{array}
$$

The substitution method: Mathematical induction

- The substitution method for solving recurrence has two steps:
  1. Guess the form of the solution.
  2. Use mathematical induction to find the constants and show that the solution works.
- Powerful but can be applied only in cases when it is easy to guess the form of solution.

- Can be used to establish either upper bound or lower bound on a recurrence.

General principle of the Mathematical Induction

- True for the trivial case,
  problem size =1.
- If it is true for the case of
  problem at step k,
  we can show that it is true for the case of
  problem at step k+1.

Example

Determine the upper bound on the recurrence

$$\begin{cases} T(n) = 2T(\lfloor n/2 \rfloor) + n \\ T(1) = 1 \end{cases}$$

(We may omit the initial condition later.)

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

Prove $T(n) \le cn \log n$ for some c > 0.

- Inductive hypothesis: assume this bound holds for $T(\lfloor n/2 \rfloor) \le c\lfloor n/2 \rfloor \log \lfloor n/2 \rfloor$
- The recurrence implies

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

$$\le 2[c\left\lfloor \frac{n}{2} \right\rfloor \log(\left\lfloor \frac{n}{2} \right\rfloor)] + n \le cn \log \frac{n}{2} + n$$

$$= cn \log n - cn \log 2 + n \le cn \log n \quad (\text{if } c \ge 1.)$$

- Initial condition

$$T(1) = 1$$

$$T(1) \le cn\log 1 = 0 \quad (\rightarrow\leftarrow)$$

- However

$$T(2) = 4$$

$$< cn\log 2 \quad (\text{if } c \ge 4)$$

- Making a good guess

$$T(n) = 2T(\lfloor n/2 \rfloor + 17) + n$$

We guess $T(n) = O(n \log n)$

Making guess provides loose upper bound and lower bound. Then improve the gap.

Show that the solution to $T(n) = 2T(\lfloor \frac{n}{2} \rfloor + 17) + n$ is $\boldsymbol{O}(n \lg n)$

Solution:

assume $a > 0$, $b > 0$, $c > 0$ and $T(n) \leqq an \lg n - b\lg n - c$

$$T(n) \leqq 2[(\frac{n}{2} + 17)\lg(\frac{n}{2} + 17) - b\lg(\frac{n}{2} + 17) - c] + n$$

$$\leqq (an + 34a)\lg(\frac{n}{2} + 17) - 2b\lg(\frac{n}{2} + 17) - 2c + n$$

$$\leqq an\lg(\frac{n}{2} + 17) + an\lg 2^{1/a} + (34a - 2b)\lg(\frac{n}{2} + 17) - 2c$$

$$\leqq an\lg(n)\, 2^{1/a} + (34a - 2b)\lg(n) - 2c$$

$anlg(n)\, 2^{1/a} + (34a-2b)lg(n) - 2c$

➔ $n \geqq \frac{n}{2} + 17$，$n \geqq 34$

➔ $n \geqq (\frac{n}{2} + 17)\, 2^{1/a}$，$\because 2^{1/2} \leqq 1.5 \therefore n \geqq 12$

➔ $34a-2b \leqq -b$，$b \geqq 34a$

➔ $c > 0$，$-c > -2c$

➔ $T(n) \leqq anlgn - blgn - c$，$T(n) \leqq anlgn$

➔ $T(n) = O(nlgn)$

Subtleties

$$T(n) = T(\lfloor n/2 \rfloor) + T(\lceil n/2 \rceil) + 1$$

- Guess $T(n) = O(n)$
- Assume $T(n) \le cn$

$$T(n) \le c\lfloor n/2 \rfloor + c\lceil n/2 \rceil + 1 \le cn + 1 \not\le cn$$

- However, assume $T(n) \le cn - b$

$$T(n) \le (c\lfloor n/2 \rfloor - b) + (c\lceil n/2 \rceil - b) + 1$$

$$\le cn - 2b + 1 \le cn - b \quad \text{(Choose } b \ge 1\text{)}$$

Avoiding pitfalls

$$\begin{cases} T(n) = 2T(\lfloor n/2 \rfloor) + n \\ T(1) = 1 \end{cases}$$

- Assume $T(n) \le O(n)$
- Hence $T(n) \le cn$

$$T(n) \le 2(c\lfloor n/2 \rfloor) + n \le cn + n = O(n)$$

(Since c is a constant)

- ***(WRONG!)*** You cannot find such a *c*.

Changing variables

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

Let $m = \lg n$.

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

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

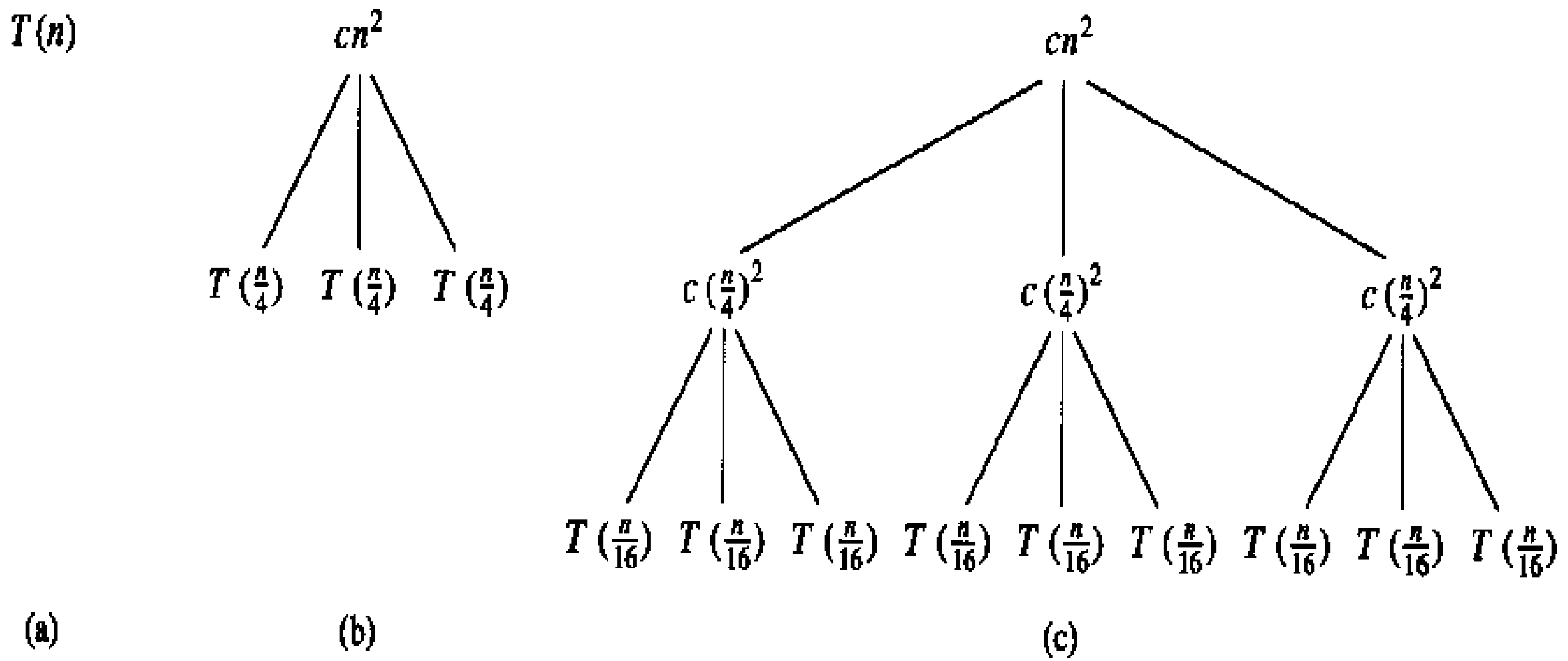
$$\Rightarrow S(m) = O(m \lg m)$$

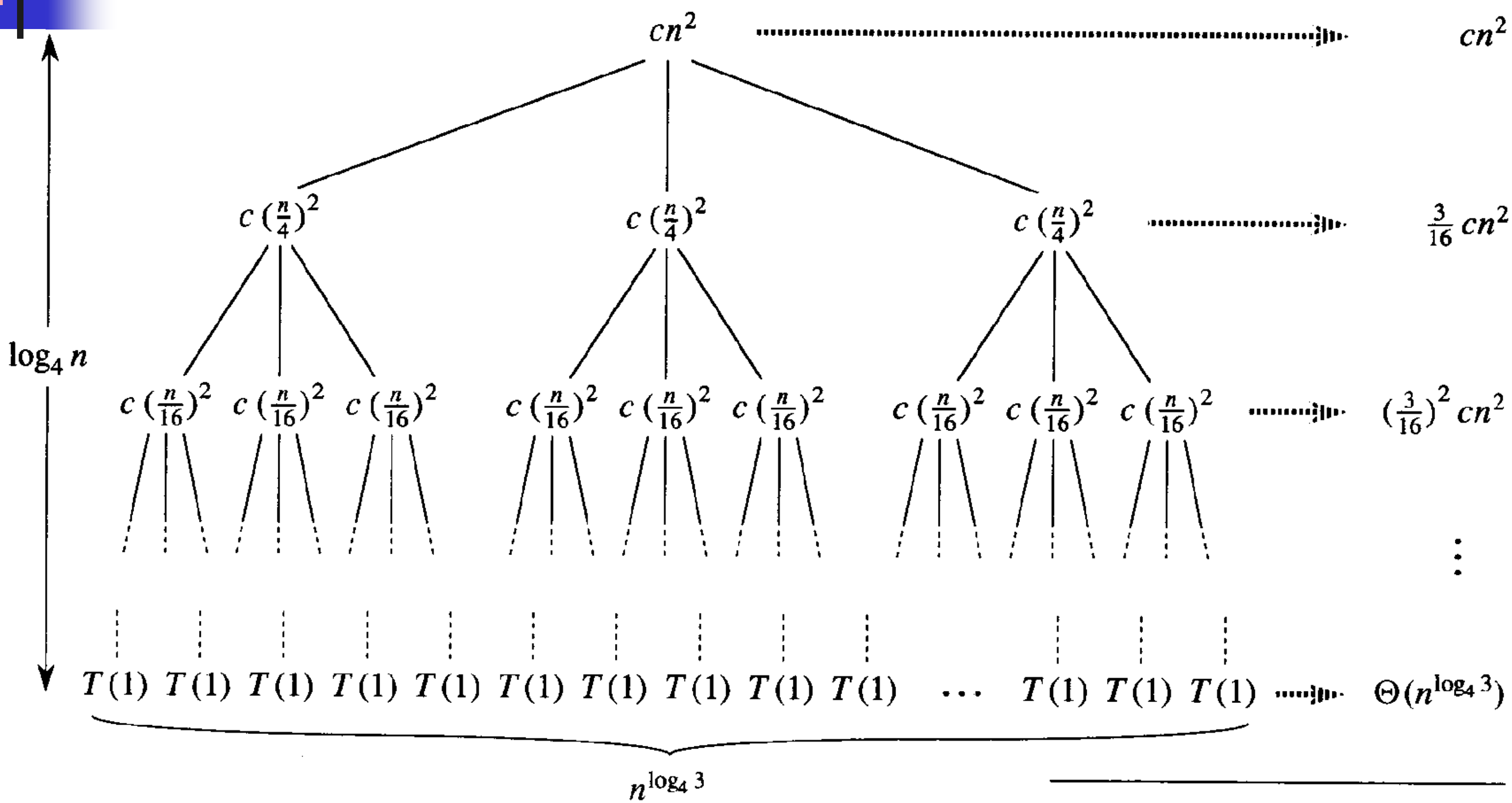
$$\Rightarrow T(n) = T(2^m) = S(m) = O(m \lg m)$$

$$= O(\lg n \lg\lg n)$$

4.4 the Recursion-tree method

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





cn^2
$c\left(\frac{n}{4}\right)^2$ $c\left(\frac{n}{4}\right)^2$ $c\left(\frac{n}{4}\right)^2$
$c\left(\frac{n}{16}\right)^2$ $c\left(\frac{n}{16}\right)^2$ $c\left(\frac{n}{16}\right)^2$ $c\left(\frac{n}{16}\right)^2$ $c\left(\frac{n}{16}\right)^2$ $c\left(\frac{n}{16}\right)^2$ $c\left(\frac{n}{16}\right)^2$ $c\left(\frac{n}{16}\right)^2$ $c\left(\frac{n}{16}\right)^2$
$\log_4 n$
$T(1)$ $T(1)$ $T(1)$ $T(1)$ $T(1)$ $T(1)$ $T(1)$ $T(1)$ $T(1)$ $T(1)$ ... $T(1)$ $T(1)$ $T(1)$
$n^{\log_4 3}$
cn^2
$\frac{3}{16}cn^2$
$\left(\frac{3}{16}\right)^2 cn^2$
$\Theta(n^{\log_4 3})$
Total: $O(n^2)$

(d)

- Subproblem size for a node at depth i $\frac{n}{4^i}$
- Total level of tree $\log_4 n + 1$
- Number of nodes at depth i 3^i
- Cost of each node at depth i $c(\frac{n}{4^i})^2$
- Total cost at depth i $3^i c(\frac{n}{4^i})^2 = (\frac{3}{16})^i cn^2$
- Last level, depth $\log_4 n$, has $3^{\log_4 n} = n^{\log_4 3}$ nodes

Prove of $3^{\log_4 n} = n^{\log_4 3}$

$$\begin{aligned}\log_4 3^{\log_4 n} &= (\log_4 n)(\log_4 3)\\ &= (\log_4 3)(\log_4 n)\\ &= \log_4 n^{\log_4 3}\end{aligned}$$

We conclude, $3^{\log_4 n} = n^{\log_4 3}$

The cost of the entire tree

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

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

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

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

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

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

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

$$= O(n^2)$$

Substitution method

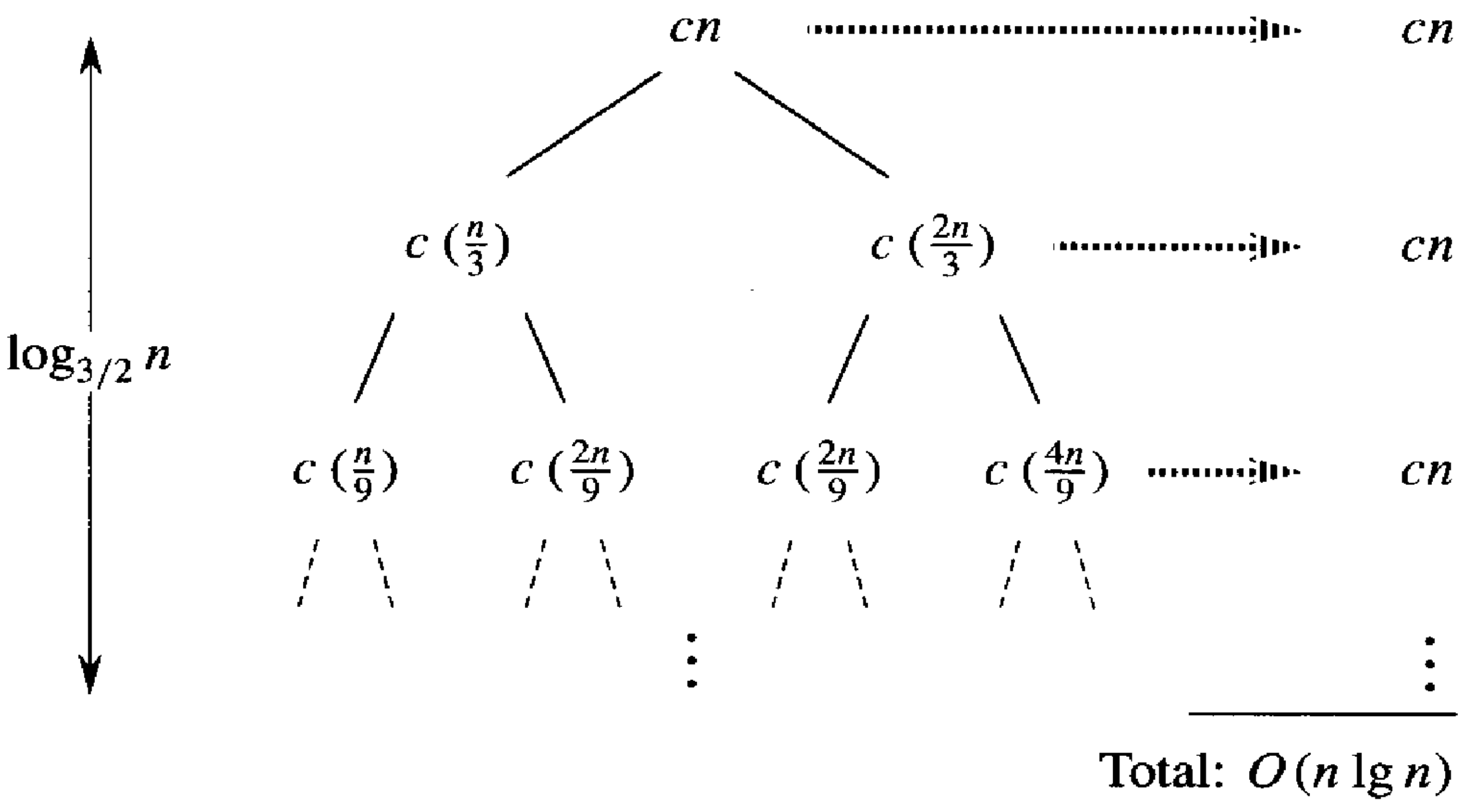
We want to Show that T(n) ≤ dn^2 for some constant $d > 0$.

Using the same constant $c > 0$ as before, we have

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

Where the last step holds as long as $d \geq (16/13)c$.

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



- Subproblem size for a node at depth i $(\frac{2}{3})^i n$
- Total level of tree $\log_{3/2} n + 1$

substitution method

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

As long as $d \ge c/(\lg 3 - (2/3))$.

4.5 Master method

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 $T(n)$ has the following asymptotic bounds:

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 $T(n) = \Theta(f(n))$. ■

Remarks 1:

- In the first case, f(n) must be polynomailly smaller than $n^{\log_b a}$
- That is, f(n) must be asymptotically smaller than $n^{\log_b a}$ by a factor of n^{ε}

- Similarly, in the third case, f(n) must be polynomailly larger than $n^{\log_b a}$
- Also, the condition $a\, f(n/b) \le c\, f(n)$ must be hold.

Remarks 2:

- The three cases in the master theorem do not cover all the possibilities.
- There is a gap between cases 1 and 2 when f(n) is smaller than $n^{\log_b a}$ but not polynomailly smaller.

- Similarly, there is a gap between cases 2 and 3 when f(n) is larger than $n^{\log_b a}$ but not polynomailly larger (or the additional condition does not hold.)

Example 1:

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

$$a = 9, b = 3, f(n) = n$$

$$n^{\log_3 9} = n^2, \quad f(n) = O(n^{\log_3 9 - 1})$$

$$Case \quad 1 \Rightarrow T(n) = \Theta(n^2)$$

Example 2:

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

$$a = 1, b = 3/2, f(n) = 1$$

$$n^{\log_{3/2} 1} = n^0 = 1 = f(n),$$

$$Case \quad 2 \Rightarrow T(n) = \Theta(\log n)$$

Example 3:

$$T(n) = 3T(n/4) + n\log n$$

$$a = 3, b = 4, f(n) = n\log n$$

$$n^{\log_4 3} = n^{0.793}, f(n) = \Omega(n^{\log_4 3+\varepsilon})$$

Case 3

Check

$$af(n/b) = 3\left(\frac{n}{4}\right)\log\left(\frac{n}{4}\right) \le \frac{3n}{4}\log n = cf(n)$$

for $c = \frac{3}{4}$, and sufficiently large n

$$\Rightarrow T(n) = \Theta(n\log n)$$

Example 4:

$T(n) = 5T(n/2) + \Theta(n^2)$

$n^{\log_2 5}$ vs. n^2

Since $\log_2 5 - \epsilon = 2$ for some constant $\epsilon > 0$, use Case 1 $\Rightarrow T(n) = \Theta(n^{\lg 5})$

Example 5:

$T(n) = 5T(n/2) + \Theta(n^3)$

$n^{\log_2 5}$ vs. n^3

Now $\lg 5 + \epsilon = 3$ for some constant $\epsilon > 0$

Check regularity condition (don't really need to since $f(n)$ is a polynomial):

$af(n/b) = 5(n/2)^3 = 5n^3/8 \leq cn^3$ for $c = 5/8 < 1$

Use Case 3 $\Rightarrow T(n) = \Theta(n^3)$

Remarks 3:

- The master theory does not apply to the recurrence $T(n) = 2T(n/2) + n\lg n,$ 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 $n^{\log_b a} = n.$
- But it is not polynomially larger.

Remarks 3 (continue)

- However, the master method augments the conditions

- Case 2 can be applied.

Example 6:

$$T(n) = 27T(n/3) + \Theta(n^3 \lg n)$$
$$n^{\log_3 27} = n^3 \text{ vs. } n^3 \lg n$$
Use Case 2 with $k = 1 \Rightarrow T(n) = \Theta(n^3 \lg^2 n)$

Example 7:

$T(n) = 27T(n/3) + \Theta(n^3/\lg n)$
$n^{\log_3 27} = n^3$ vs. $n^3/\lg n = n^3 \lg^{-1} n \neq \Theta(n^3 \lg^k n)$ for any $k \geq 0$.
Cannot use the master method.