Exercise Notes - Chpt 4

Harley Caham Combest

Fa2025 CS4413 Lecture Notes – Mk1

. . . .

4.1 Exercises

. . . .

Exercise 4.1-1. Generalize MATRIX-MULTIPLY-RECURSIVE to multiply $n \times n$ matrices when n is not a power of 2. Give a recurrence for its running time and argue it is $\Theta(n^3)$ in the worst case.

Proof. Solution.
Intuition. If n is not a power of two, the 2×2 block split is uneven. Easiest fix: pad each matrix with zeros to the next power of two $m = 2^{\lceil \lg n \rceil}$, run the usual recursive routine, and then crop the top-left $n \times n$ block. Zero padding does not change the product on the original entries, and the work pattern (8 submultiplications on half-size blocks plus $\Theta(n^2)$ additions) still scales like n^3.
Rigorous. Let $T(n)$ be the worst-case time to multiply two $n \times n$ matrices by this method.

Padding formulation. Take $m = 2^{\lceil \lg n \rceil}$ so $n \le m < 2n$, and form $m \times m$ matrices A', B' by zero-padding A, B. Running MATRIX-MULTIPLY-RECURSIVE on (A', B') obeys the textbook recurrence

$$T(m) = 8\,T\Big(\frac{m}{2}\Big) + \Theta(m^2),$$

with solution $T(m) = \Theta(m^3)$ (Master Theorem with $a = 8$, $b = 2$, $f(m) = \Theta(m^2)$). Copy/pad/crop costs are $\Theta(m^2)$ and are absorbed by $\Theta(m^3)$, hence

$$T(n) \le T(m) = \Theta(m^3) = \Theta\big((2^{\lceil \lg n \rceil})^3\big) = \Theta(n^3).$$

A matching lower bound holds because for dense inputs there are n^2 outputs, each a sum of n products, so the algorithm performs $\Omega(n^3)$ arithmetic operations in the worst case. Therefore $T(n) = \Theta(n^3)$.

Uneven-split formulation (no explicit padding). If at size n we split into blocks of sizes $\lfloor n/2 \rfloor$ and $\lceil n/2 \rceil$, the routine performs eight recursive calls on dimension $\le \lceil n/2 \rceil$ and $\Theta(n^2)$ extra work:

$$8\,T(\lfloor n/2 \rfloor) + \Theta(n^2) \;\le\; T(n) \;\le\; 8\,T(\lceil n/2 \rceil) + \Theta(n^2).$$

By monotonicity/comparison with the padded case, this also solves to $T(n) = \Theta(n^3)$. □

Exercise 4.1-2. How quickly can you multiply a $(kn) \times n$ matrix (kn rows and n columns) by an $n \times (kn)$ matrix, using MATRIX-MULTIPLY-RECURSIVE as a subroutine? Answer the same question for multiplying an $n \times (kn)$ matrix by a $(kn) \times n$ matrix. Which is asymptotically faster, and by how much?

Proof. Solution.
Intuition. The cost of dense matrix multiplication scales like (rows)$\times$(shared dim)$\times$(cols). So $(kn) \times n$ by $n \times (kn)$ costs $\Theta((kn) \cdot n \cdot (kn)) = \Theta(k^2 n^3)$, while $n \times (kn)$ by $(kn) \times n$ costs $\Theta(n \cdot (kn) \cdot n) = \Theta(kn^3)$. Thus the *second* order is faster, by a factor of k.
Rigorous. Let $T(p, q, r)$ denote the worst-case time for multiplying a $p \times q$ matrix by a $q \times r$ matrix using MATRIX-MULTIPLY-RECURSIVE (with zero-padding to powers of two as needed). At each level, the algorithm divides along the three dimensions and performs a constant number of multiplications of subblocks whose total scalar-multiplication count is $\Theta(pqr)$, plus $\Theta(pr + pq + qr)$ additions, which are asymptotically dominated by pqr when all dimensions grow. Hence

$$T(p, q, r) = \Theta(pqr).$$

Case 1: $(kn) \times n$ times $n \times (kn)$. Here $p = kn$, $q = n$, $r = kn$. Therefore

$$T(kn, n, kn) = \Theta\big((kn) \cdot n \cdot (kn)\big) = \Theta(k^2 n^3).$$

Case 2: $n \times (kn)$ times $(kn) \times n$. Here $p = n$, $q = kn$, $r = n$. Therefore

$$T(n, kn, n) = \Theta\big(n \cdot (kn) \cdot n\big) = \Theta(kn^3).$$

Conclusion. Multiplying $n \times (kn)$ by $(kn) \times n$ ($\Theta(kn^3)$) is asymptotically faster than the reverse order ($\Theta(k^2 n^3)$) by a factor of k.

□

Exercise 4.1-3.

> Thus, omitting the statement of the base case, our recurrence for the running time of MATRIX-MULTIPLY-RECURSIVE is
>
> $$T(n) = 8T(n/2) + \Theta(1) . \tag{4.9}$$

In MATRIX-MULTIPLY-RECURSIVE, the text's recurrence (4.9) with index-calculated partitioning is

$$T(n) = 8T(n/2) + \Theta(1).$$

Now suppose we *copy* elements of A, B, C into $n/2 \times n/2$ submatrices $A_{11}, A_{12}, A_{21}, A_{22}$, $B_{11}, B_{12}, B_{21}, B_{22}$, and $C_{11}, C_{12}, C_{21}, C_{22}$ before the recursive calls, and copy the results back into C afterward. How does the recurrence change, and what is its solution?

Proof. Solution.
Intuition. Replacing $O(1)$-time index arithmetic by explicit copying forces whole-matrix passes at each level (copy A, copy B, copy C into quadrants, later copy C's quadrants back). Each pass touches $\Theta(n^2)$ entries. So the *nonrecursive* work per level jumps from $\Theta(1)$ to $\Theta(n^2)$, yielding the classic recurrence whose solution is $\Theta(n^3)$.
Rigorous. Let $T(n)$ be the worst-case time with copying-based partitioning. At size n:

- **Copy in:** Scatter A, B, and C into their four $n/2 \times n/2$ blocks: $3 \cdot \Theta(n^2)$.
- **Recursive multiplies:** 8 calls on size $n/2$: $8\,T(n/2)$.
- **Combine (adds):** For each of the four C-quadrants, add two $(n/2) \times (n/2)$ products: $4 \cdot \Theta((n/2)^2) = \Theta(n^2)$.
- **Copy out:** Gather $C_{11}, C_{12}, C_{21}, C_{22}$ back into C: $\Theta(n^2)$.

Thus the nonrecursive cost per level is $\Theta(n^2)$, and the recurrence becomes

$$\boxed{T(n) = 8\,T(n/2) + \Theta(n^2)}$$

in contrast to (4.9)'s $8T(n/2) + \Theta(1)$.

By the Master Theorem ($a = 8$, $b = 2$, $f(n) = \Theta(n^2)$),

$$T(n) = \Theta\big(n^{\log_2 8}\big) = \Theta(n^3).$$

□

Exercise 4.1-4. Write pseudocode for a divide-and-conquer algorithm MATRIX-ADD-RECURSIVE that sums two $n \times n$ matrices A and B by partitioning each into four $n/2 \times n/2$ submatrices and recursively summing corresponding pairs. Assume partitioning uses $\Theta(1)$-time *index* calculations. Write a recurrence for the worst-case running time of MATRIX-ADD-RECURSIVE and solve it. What happens if you instead use $\Theta(n^2)$-time *copying* to implement partitioning?

Proof. Solution.
Intuition. Addition is local: each quadrant of C depends only on the matching quadrants of A and B. If submatrices are created by *index views* (no copying), the only work at each level is the four recursive calls, so cost obeys $T(n) = 4T(n/2) + \Theta(1) = \Theta(n^2)$. If we *copy* elements to form quadrants (and back), that adds $\Theta(n^2)$ work per level, changing the solution to $\Theta(n^2 \log n)$.
Pseudocode (index-based partitioning).

Algorithm 1 MATRIX-ADD-RECURSIVE(A, B, C, n)

```
1: if n = 1 then
2:     C[1,1] ← A[1,1] + B[1,1]
3:     return
4: end if
5: let A11, A12, A21, A22 be the four n/2 × n/2 views of A
6: let B11, B12, B21, B22 be the four n/2 × n/2 views of B
7: let C11, C12, C21, C22 be the four n/2 × n/2 views of C
8: MATRIX-ADD-RECURSIVE(A11, B11, C11, n/2)
9: MATRIX-ADD-RECURSIVE(A12, B12, C12, n/2)
10: MATRIX-ADD-RECURSIVE(A21, B21, C21, n/2)
11: MATRIX-ADD-RECURSIVE(A22, B22, C22, n/2)
```

Rigorous. Let $T(n)$ be the worst-case time for the index-based version. Creating submatrix *views* costs $\Theta(1)$, and there is no cross-quadrant combination step beyond recursive writes into C. Thus

$$T(n) = 4\,T(n/2) + \Theta(1), \qquad T(1) = \Theta(1).$$

By the Master Theorem ($a = 4$, $b = 2$, $f(n) = \Theta(1)$), $T(n) = \Theta(n^{\log_2 4}) = \Theta(n^2)$.

If instead partitioning uses $\Theta(n^2)$ copying (scatter/gather at each level), the nonrecursive work becomes $\Theta(n^2)$, giving

$$T_{\text{copy}}(n) = 4\,T_{\text{copy}}(n/2) + \Theta(n^2).$$

Here $f(n) = \Theta(n^{\log_2 4})$, so by Master Theorem (case 2),

$$T_{\text{copy}}(n) = \Theta(n^2 \log n).$$

Conclusion. With index views: $\boxed{T(n) = \Theta(n^2)}$. With copying-based partitioning: $\boxed{T(n) = \Theta(n^2 \log n)}$.

. . . .

4.2 Exercises

. . . .

Exercise 4.2-1. Use Strassen's algorithm to compute

$$\begin{pmatrix} 1 & 3 \\ 7 & 5 \end{pmatrix} \begin{pmatrix} 6 & 8 \\ 4 & 2 \end{pmatrix}.$$

Show your work.

Proof. Solution.
Intuition. For 2×2 matrices with blocks A_{ij}, B_{ij}, Strassen forms seven products $M_1, \ldots, M_7$ (fewer than the eight in the naive block method) and recombines them to get the four entries of $C = AB$.
Rigorous (Strassen's seven products). Let

$$A = \begin{pmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{pmatrix} = \begin{pmatrix} 1 & 3 \\ 7 & 5 \end{pmatrix}, \qquad B = \begin{pmatrix} B_{11} & B_{12} \\ B_{21} & B_{22} \end{pmatrix} = \begin{pmatrix} 6 & 8 \\ 4 & 2 \end{pmatrix}.$$

Compute

$$\begin{aligned}
M_1 &= (A_{11} + A_{22})(B_{11} + B_{22}) = (1+5)(6+2) = 6 \cdot 8 = 48, \\
M_2 &= (A_{21} + A_{22})B_{11} = (7+5) \cdot 6 = 12 \cdot 6 = 72, \\
M_3 &= A_{11}(B_{12} - B_{22}) = 1 \cdot (8-2) = 1 \cdot 6 = 6, \\
M_4 &= A_{22}(B_{21} - B_{11}) = 5 \cdot (4-6) = 5 \cdot (-2) = -10, \\
M_5 &= (A_{11} + A_{12})B_{22} = (1+3) \cdot 2 = 4 \cdot 2 = 8, \\
M_6 &= (A_{21} - A_{11})(B_{11} + B_{12}) = (7-1) \cdot (6+8) = 6 \cdot 14 = 84, \\
M_7 &= (A_{12} - A_{22})(B_{21} + B_{22}) = (3-5) \cdot (4+2) = (-2) \cdot 6 = -12.
\end{aligned}$$

Recombine to obtain $C = AB$:

$$\begin{aligned}
C_{11} &= M_1 + M_4 - M_5 + M_7 = 48 - 10 - 8 - 12 = 18, \\
C_{12} &= M_3 + M_5 = 6 + 8 = 14, \\
C_{21} &= M_2 + M_4 = 72 - 10 = 62, \\
C_{22} &= M_1 - M_2 + M_3 + M_6 = 48 - 72 + 6 + 84 = 66.
\end{aligned}$$

Therefore

$$AB = \begin{pmatrix} 18 & 14 \\ 62 & 66 \end{pmatrix}.$$

□

Exercise 4.2-2. Write pseudocode for Strassen's algorithm.

Proof. Solution.
Intuition. Split each matrix into four quadrants. Form seven cleverly chosen products on $(n/2) \times (n/2)$ blocks (instead of eight), then recombine to obtain the four quadrants of $C = AB$. The recurrence is $T(n) = 7T(n/2)+\Theta(n^2)$, so $T(n) = \Theta(n^{\lg 7})$.
Pseudocode (index-based partitioning, no copying).

Algorithm 2 STRASSEN-MULTIPLY(A, B, C, n) // computes $C \leftarrow A \cdot B$

if $n = 1$ **then**
  $C[1,1] \leftarrow A[1,1] \cdot B[1,1]$
  return
end if
let $A_{11}, A_{12}, A_{21}, A_{22}$ be the four $n/2 \times n/2$ *views* of A
let $B_{11}, B_{12}, B_{21}, B_{22}$ be the four $n/2 \times n/2$ *views* of B
let $C_{11}, C_{12}, C_{21}, C_{22}$ be the four $n/2 \times n/2$ *views* of C

$X_1 \leftarrow A_{11}+A_{22}$; $Y_1 \leftarrow B_{11}+B_{22}$; $M_1 \leftarrow$ STRASSEN-MULTIPLY$(X_1, Y_1, \cdot, n/2)$
$X_2 \leftarrow A_{21} + A_{22}$; $M_2 \leftarrow$ STRASSEN-MULTIPLY$(X_2, B_{11}, \cdot, n/2)$
$Y_3 \leftarrow B_{12} - B_{22}$; $M_3 \leftarrow$ STRASSEN-MULTIPLY$(A_{11}, Y_3, \cdot, n/2)$
$Y_4 \leftarrow B_{21} - B_{11}$; $M_4 \leftarrow$ STRASSEN-MULTIPLY$(A_{22}, Y_4, \cdot, n/2)$
$X_5 \leftarrow A_{11} + A_{12}$; $M_5 \leftarrow$ STRASSEN-MULTIPLY$(X_5, B_{22}, \cdot, n/2)$
$X_6 \leftarrow A_{21}-A_{11}$; $Y_6 \leftarrow B_{11}+B_{12}$; $M_6 \leftarrow$ STRASSEN-MULTIPLY$(X_6, Y_6, \cdot, n/2)$
$X_7 \leftarrow A_{12}-A_{22}$; $Y_7 \leftarrow B_{21}+B_{22}$; $M_7 \leftarrow$ STRASSEN-MULTIPLY$(X_7, Y_7, \cdot, n/2)$

$C_{11} \leftarrow M_1 + M_4 - M_5 + M_7$
$C_{12} \leftarrow M_3 + M_5$
$C_{21} \leftarrow M_2 + M_4$
$C_{22} \leftarrow M_1 - M_2 + M_3 + M_6$

Running time. With index-based partitioning and $\Theta(n^2)$-time additions/subtractions,

$$T(n) = 7T(n/2) + \Theta(n^2) \quad \Rightarrow \quad T(n) = \Theta\big(n^{\lg 7}\big).$$

□

Exercise 4.2-3. What is the largest k such that if you can multiply 3×3 matrices using k multiplications (without assuming commutativity), then you can multiply $n \times n$ matrices in $o(n^{\lg 7})$ time? What is the running time of this algorithm?

Proof. Solution.
Intuition. Split each $n \times n$ matrix into a 3×3 grid of $n/3 \times n/3$ blocks and recursively use the k-multiplication 3×3 method on these blocks. This gives the recurrence $T(n) = k\,T(n/3) + \Theta(n^2)$, hence $T(n) = \Theta(n^{\log_3 k})$. To beat Strassen's exponent $\lg 7$ we need $\log_3 k < \lg 7$.
Rigorous. (Zero-pad if $3 \nmid n$.) Using block recursion with a k-multiplication algorithm for 3×3 block matrices yields

$$T(n) = k\,T(n/3) + \Theta(n^2).$$

By the Master Theorem, $T(n) = \Theta(n^\alpha)$ with $\alpha = \log_3 k$. For $T(n) = o(n^{\lg 7})$ we require

$$\log_3 k \;<\; \lg 7 \quad\Longleftrightarrow\quad k \;<\; 3^{\lg 7} \;=\; 7^{\lg 3},$$

(using $a^{\log_b c} = c^{\log_b a}$). Numerically $3^{\lg 7} \approx 21.8498$, so the largest integer k is

$$\boxed{k = 21}.$$

The running time is then

$$\boxed{T(n) = \Theta\big(n^{\log_3 21}\big)} \qquad \text{with } \log_3 21 \approx 2.77124 \;<\; \lg 7 \approx 2.80735.$$

□

Exercise 4.2-4. V. Pan discovered ways to multiply 68×68 matrices using 132,464 multiplications, 70×70 using 143,640, and 72×72 using 155,424. Which method gives the best asymptotic running time in a divide-and-conquer matrix-multiplication algorithm? How does it compare with Strassen's algorithm?

Proof. Solution.
Intuition. If we split an $n \times n$ matrix into a $b \times b$ grid of blocks and can multiply $b \times b$ block matrices using a multiplications, the recurrence is

$$T(n) = a\,T(n/b) + \Theta(n^2),$$

so $T(n) = \Theta\big(n^{\alpha}\big)$ with exponent $\alpha = \log_b a$. Smaller α means faster asymptotics. Compare the three α's and pick the minimum; then compare that to Strassen's $\lg 7 \approx 2.807355$.
Rigorous. For each (b, a) pair:

$$\alpha(b, a) = \log_b a = \frac{\ln a}{\ln b}.$$

$$\begin{aligned}
(b, a) = (68, 132{,}464) &\;\Rightarrow\; \alpha \approx 2.795128, \\
(b, a) = (70, 143{,}640) &\;\Rightarrow\; \alpha \approx \mathbf{2.795123}, \quad \text{(smallest)} \\
(b, a) = (72, 155{,}424) &\;\Rightarrow\; \alpha \approx 2.795147.
\end{aligned}$$

Hence the 70×70 method yields the best asymptotic bound:

$$T(n) = \Theta\big(n^{\,2.795123\ldots}\big).$$

Strassen's algorithm runs in $\Theta\big(n^{\lg 7}\big)$ with $\lg 7 \approx 2.807355$, so Pan's best option improves the exponent by about

$$\lg 7 - 2.795123 \approx 0.012232,$$

i.e., it is asymptotically faster by a factor of $n^{0.012232}$ compared with Strassen.

□

Exercise 4.2-5. Show how to multiply $(a + bi)$ and $(c + di)$ using only three multiplications of real numbers. The algorithm should output the real part $ac - bd$ and the imaginary part $ad + bc$ separately.

Proof. Solution.
Intuition. Compute one clever product of sums to capture the cross terms:

$$(a + b)(c + d) = ac + ad + bc + bd.$$

If we also know ac and bd, then subtracting them from the clever product leaves $ad + bc$. Thus three real multiplications suffice.
Algorithm (three real multiplications).

Algorithm 3 COMPLEX-MUL-3MULTS(a, b, c, d) // returns $(\text{real}, \text{imag})$

1: $s_1 \leftarrow a + b$; $s_2 \leftarrow c + d$
2: $p_1 \leftarrow s_1 \cdot s_2$ ▷ $(a + b)(c + d)$
3: $p_2 \leftarrow a \cdot c$
4: $p_3 \leftarrow b \cdot d$
5: $\text{real} \leftarrow p_2 - p_3$ ▷ $ac - bd$
6: $\text{imag} \leftarrow p_1 - p_2 - p_3$ ▷ $(ac{+}ad{+}bc{+}bd) - ac - bd = ad + bc$
7: **return** $(\text{real}, \text{imag})$

Rigorous. Correctness follows from

$$p_1 = (a + b)(c + d) = ac + ad + bc + bd, \tag{1}$$
$$\text{real} = p_2 - p_3 = ac - bd, \tag{2}$$
$$\text{imag} = p_1 - p_2 - p_3 = ad + bc. \tag{3}$$

The procedure uses exactly three real multiplications (p_1, p_2, p_3) and a constant number of additions/subtractions.

□

Exercise 4.2-6. Suppose you have a $\Theta(n^\alpha)$-time algorithm for *squaring* $n \times n$ matrices, where $\alpha \geq 2$. Show how to use that algorithm to multiply two different $n \times n$ matrices in $\Theta(n^\alpha)$ time.

Proof. Solution.
Intuition. Embed A and B into a $2n \times 2n$ upper-block-triangular matrix

$$M = \begin{pmatrix} A & B \\ 0 & 0 \end{pmatrix}.$$

Then $M^2 = \begin{pmatrix} A^2 & AB \\ 0 & 0 \end{pmatrix}$, so the top-right block of M^2 is exactly AB. One squaring on size $2n$ plus $O(n^2)$ copying yields $\Theta((2n)^\alpha) = \Theta(n^\alpha)$ time.
Algorithm (reduce multiplication to one squaring).

Algorithm 4 MULTIPLY-VIA-SQUARE(A, B) // returns $C = AB$

1: Form the $2n \times 2n$ block matrix $M \leftarrow \begin{pmatrix} A & B \\ 0 & 0 \end{pmatrix}$.
2: Compute $S \leftarrow$ SQUARE(M) ▷ $S = M^2$ using the given $\Theta(m^\alpha)$ squaring algorithm on size $m = 2n$
3: Output $C \leftarrow S_{12}$ ▷ the top-right $n \times n$ block of S

Rigorous. Correctness: Block multiplication gives

$$M^2 = \begin{pmatrix} A & B \\ 0 & 0 \end{pmatrix} \begin{pmatrix} A & B \\ 0 & 0 \end{pmatrix} = \begin{pmatrix} A^2 & AB \\ 0 & 0 \end{pmatrix},$$

so $S_{12} = AB$.

Running time: Building M and extracting S_{12} each cost $\Theta(n^2)$. The squaring step on a $(2n) \times (2n)$ matrix costs $\Theta\big((2n)^\alpha\big) = \Theta(n^\alpha)$. Since $\alpha \geq 2$, the $\Theta(n^2)$ overhead is dominated, giving total time

$$\Theta(n^\alpha) + \Theta(n^2) = \Theta(n^\alpha).$$

□

. . . .

4.3 Exercises

. . . .

Exercise 4.3-1. Use the substitution method to show that each of the following recurrences (defined on the reals) has the asymptotic solution specified.

1. $T(n) = T(n-1) + n$ has solution $T(n) = O(n^2)$.
2. $T(n) = T(n/2) + \Theta(1)$ has solution $T(n) = O(\lg n)$.
3. $T(n) = 2T(n/2) + n$ has solution $T(n) = \Theta(n \lg n)$.
4. $T(n) = 2T(n/2 + 17) + n$ has solution $T(n) = O(n \lg n)$.
5. $T(n) = 2T(n/3) + \Theta(n)$ has solution $T(n) = \Theta(n)$.
6. $T(n) = 4T(n/2) + \Theta(n)$ has solution $T(n) = \Theta(n^2)$.

Proof. Solution. (Assume T is nondecreasing and base cases are absorbed into constants. Floor/ceiling effects do not change asymptotics.)

(a) Intuition. Unrolling adds $n + (n-1) + \cdots + 1 = \Theta(n^2)$ to a constant base.

(a) Rigorous (upper bound). Assume $T(1) \le C$ and for some $c \ge 1$ and all $m < n$, $T(m) \le cm^2$. Then

$$T(n) = T(n-1) + n \le c(n-1)^2 + n = cn^2 - 2cn + c + n \le cn^2$$

whenever $n(1-2c) + c \le 0$, which holds for all $n \ge n_0 := \lceil c/(2c-1) \rceil$. Thus $T(n) = O(n^2)$.

(b) Intuition. Each step halves the size and adds $O(1)$; depth is $\Theta(\lg n)$.

(b) Rigorous (upper bound). Write $T(n) \le T(n/2) + d$ for some $d > 0$ and assume $T(m) \le c \lg m$ for $m < n$. Then

$$T(n) \le c \lg(n/2) + d = c \lg n - c + d \le c \lg n$$

if $c \ge d$. Hence $T(n) = O(\lg n)$.

(c) Intuition. At each level the total nonrecursive work is $\Theta(n)$ across $O(\lg n)$ levels: $n \lg n$. The same reasoning gives a matching lower bound.

(c) Rigorous (upper and lower). Assume $T(1) \le C$. For the upper bound, suppose $T(m) \le c\, m \lg m$ for $m < n$. Then

$$T(n) = 2T(n/2) + n \le 2\Big(c\frac{n}{2} \lg \frac{n}{2}\Big) + n = cn(\lg n - 1) + n \le cn \lg n$$

if $c \ge 1$. For the lower bound, suppose $T(m) \ge c' m \lg m$ for $m < n$. Then

$$T(n) \ge 2\Big(c'\frac{n}{2} \lg \frac{n}{2}\Big) + n = c'n(\lg n - 1) + n \ge c'n \lg n$$

if $0 < c' \leq 1$. Hence $T(n) = \Theta(n \lg n)$.

(d) Intuition. The $+17$ shift inside the argument is a constant-offset in size; absorb it by a change of variables to recover (c).

(d) Rigorous (upper bound). Define $S(n) := T(n+34)$. Then

$$S(n) = T(n+34) = 2\,T\left(\frac{n+34}{2} + 17\right) + (n+34) = 2\,T\left(\frac{n}{2} + 34\right) + n + 34 \quad (4)$$

$$= 2S(n/2) + n + 34. \quad (5)$$

As in (c), by the same substitution proof (with an additive constant) we obtain $S(n) = O(n \lg n)$, hence $T(n) = O(n \lg n)$.

(e) Intuition. Here the nonrecursive work $\Theta(n)$ dominates the branching factor 2 on thirds, since $n^{\log_3 2} = n^{0.63\ldots}$ is smaller than n; total is linear.

(e) Rigorous (both bounds). Let $T(n) \leq 2T(n/3) + cn$ for a constant $c > 0$. Assume $T(m) \leq dm$ for $m < n$. Then

$$T(n) \leq 2d\frac{n}{3} + cn = \left(\frac{2d}{3} + c\right) n \leq dn$$

if $d \geq 3c$. For the lower bound, if $T(n) \geq 2T(n/3) + c'n$ with $c' > 0$ and $T(m) \geq em$ for $m < n$, then

$$T(n) \geq \left(\frac{2e}{3} + c'\right) n \geq en$$

whenever $e/3 \leq c'$. Thus $T(n) = \Theta(n)$.

(f) Intuition. Four half-size subproblems already cost n^2; the extra $+\Theta(n)$ is lower order.

(f) Rigorous (both bounds). Upper: with $T(n) \leq 4T(n/2) + cn$ and hypothesis $T(m) \leq dm^2$ for $m < n$,

$$T(n) \leq 4d\left(\frac{n}{2}\right)^2 + cn = dn^2 + cn \leq (d + c)\, n^2.$$

Choosing d large enough gives $T(n) \leq Dn^2$ for some constant D. Lower: with $T(n) \geq 4T(n/2) + c'n$ and $T(m) \geq em^2$ for $m < n$,

$$T(n) \geq 4e\left(\frac{n}{2}\right)^2 + c'n = en^2 + c'n \geq en^2.$$

Hence $T(n) = \Theta(n^2)$.

□

Exercise 4.3-2. The solution to the recurrence $T(n) = 4T(n/2) + n$ turns out to be $T(n) = \Theta(n^2)$. Show that a substitution proof with the assumption $T(n) \leq c\,n^2$ fails. Then show how to subtract a lower-order term to make a substitution proof work.

Proof. Solution.
Intuition. Plugging $T(n) \leq cn^2$ into the recurrence reproduces cn^2 *plus* a leftover $+n$, which cannot be absorbed by the same quadratic bound. Strengthen the induction by subtracting a linear slack: prove $T(n) \leq cn^2 - dn$ for some constant $d > 0$. The extra $-dn$ cancels the stray $+n$ after the recursive step.
Rigorous.

Why $T(n) \leq cn^2$ fails. Assume the inductive hypothesis $T(m) \leq cm^2$ for all $m < n$. Then

$$T(n) = 4T(n/2) + n \;\leq\; 4 \cdot c\left(\frac{n}{2}\right)^2 + n \;=\; cn^2 + n \;>\; cn^2$$

for every $n > 0$. Thus the bound $T(n) \leq cn^2$ cannot be proved by simple induction.

Strengthened hypothesis with a subtractive term. We prove by induction (for all sufficiently large n) that

$$T(n) \;\leq\; cn^2 - dn$$

for suitable constants $c, d > 0$.

Inductive step. Assume $T(m) \leq cm^2 - dm$ for all $m < n$. Then

$$\begin{aligned} T(n) &= 4T(n/2) + n \\ &\leq 4\left(c\left(\frac{n}{2}\right)^2 - d\frac{n}{2}\right) + n = cn^2 - 2dn + n. \end{aligned}$$

To reach $T(n) \leq cn^2 - dn$ it suffices that

$$-2dn + n \;\leq\; -dn \quad\Longleftrightarrow\quad (1-d)n \leq 0,$$

which holds for every $n > 0$ provided we choose $d \geq 1$.

Base case. Pick $d = 1$ and choose c large enough that the inequality holds for the finitely many small n (e.g., ensure $T(1) \leq c - 1$ and $T(2) \leq 4c - 2$). Such a c exists because $T(1), T(2)$ are constants.

Therefore $T(n) \leq cn^2 - n$ for all $n \geq n_0$, which implies $T(n) = O(n^2)$. A matching lower bound $T(n) = \Omega(n^2)$ follows, for example, by noting that with $S(n) = T(n)/n^2$ the recurrence yields

$$S(n) = S(n/2) + \frac{1}{n},$$

so $S(n) \geq S(1) + \sum_{i=0}^{\lfloor \lg n \rfloor - 1} 2^{-i} \geq S(1) + 1$. Hence $T(n) = \Theta(n^2)$.

□

Exercise 4.3-3. The recurrence $T(n) = 2T(n-1)+1$ has the solution $T(n) = O(2^n)$. Show that a substitution proof fails with the assumption $T(n) \leq c\,2^n$, where $c > 0$ is constant. Then show how to subtract a lower-order term to make a substitution proof work.

Proof. Solution.
Intuition. Each step roughly doubles the previous value, so 2^n growth is natural. But a direct induction with $T(n) \leq c2^n$ leaves a stray $+1$ that cannot be absorbed. Strengthen the hypothesis by subtracting a constant slack: prove $T(n) \leq c2^n - d$ for some $d > 0$. That constant cancels the leftover $+1$ after the recursive step.
Rigorous.

The naive bound $T(n) \leq c2^n$ fails. Assume as induction hypothesis that $T(m) \leq c2^m$ for all $m < n$. Then

$$T(n) = 2T(n-1) + 1 \;\leq\; 2 \cdot c\,2^{\,n-1} + 1 \;=\; c\,2^n + 1 \;>\; c\,2^n,$$

so the inequality cannot be closed with the same c.

Strengthened hypothesis with a subtractive term. We prove (for all sufficiently large n) that

$$T(n) \;\leq\; c\,2^n - d$$

for suitable constants $c, d > 0$.

Inductive step. Assume $T(m) \leq c2^m - d$ for all $m < n$. Then

$$T(n) = 2T(n-1) + 1 \;\leq\; 2(c2^{\,n-1} - d) + 1 = c2^n - 2d + 1 \;\leq\; c2^n - d,$$

provided $d \geq 1$.

Base case. Choose $d = 1$ and take c large enough that $T(1) \leq c2-1$ (and similarly for finitely many small n if desired). Such a c exists because $T(1)$ is a constant.

Therefore $T(n) \leq c2^n - 1$ for all $n \geq n_0$, hence $T(n) = O(2^n)$. (Indeed, a matching lower bound $T(n) = \Omega(2^n)$ follows by a similar inductive argument, so $T(n) = \Theta(2^n)$.)

□

. . . .

4.4 Exercises

. . . .

Exercise 4.4-1. For each recurrence, sketch its recursion tree to guess a good asymptotic *upper* bound, then verify by substitution.

1. $T(n) = T(n/2) + n^3$ has solution $T(n) = O(n^3)$.

   Intuition (tree). Cost per level: $n^3, (n/2)^3, (n/4)^3, \ldots$; a geometric series dominated by the root: $\sum_{i \geq 0} (n/2^i)^3 = O(n^3)$.

   Rigorous (substitution). Assume $T(m) \leq cm^3$ for $m < n$ and $T(1) \leq C$. Then

   $$T(n) \leq c\Big(\frac{n}{2}\Big)^3 + n^3 = \frac{c}{8}n^3 + n^3 \leq cn^3$$

   if $c \geq \frac{8}{7}$. Thus $T(n) = O(n^3)$.

2. $T(n) = 4T(n/3) + n$ has solution $T(n) = O\big(n^{\log_3 4}\big)$.

   Intuition (tree). Level-i has 4^i nodes of size $n/3^i$; level cost $4^i \cdot (n/3^i) = n \cdot (4/3)^i$. The leaf level occurs at depth $\log_3 n$ and contributes about $n^{\log_3 4}$. Summing gives $O(n^{\log_3 4})$.

   Rigorous (substitution with slack). Let $\alpha = \log_3 4$. Prove $T(n) \leq cn^\alpha - dn$ for constants $c, d > 0$. Assuming the claim for $m < n$,

   $$T(n) \leq 4\Big(c\Big(\tfrac{n}{3}\Big)^\alpha - d\tfrac{n}{3}\Big) + n = cn^\alpha - \frac{4d}{3}n + n \leq cn^\alpha - dn$$

   whenever $d \geq 3$. Choosing such d and large enough c for base cases yields $T(n) = O(n^\alpha)$.

3. $T(n) = 4T(n/2) + n$ has solution $T(n) = O(n^2)$.

   Intuition (tree). Level-i cost: $4^i \cdot (n/2^i) = n \cdot 2^i$, which grows geometrically; leaves at depth $\log_2 n$ contribute $n \cdot 2^{\log_2 n} = n^2$. Total $O(n^2)$.

   Rigorous (substitution with slack). Show $T(n) \leq cn^2 - dn$. Using the hypothesis for $m < n$,

   $$T(n) \leq 4\left(c\Big(\tfrac{n}{2}\Big)^2 - d\tfrac{n}{2}\right) + n = cn^2 - 2dn + n \leq cn^2 - dn$$

   for $d \geq 1$. Hence $T(n) = O(n^2)$.

4. $T(n) = 3T(n-1) + 1$ has solution $T(n) = O(3^n)$.

   Intuition (tree). Depth n; each level triples the number of subproblems, so the total work is dominated by the leaves: $\Theta(3^n)$.

   Rigorous (substitution with constant slack). Prove $T(n) \le c\,3^n - d$. Assuming it for $m < n$,

$$T(n) \le 3(c\,3^{n-1} - d) + 1 = c\,3^n - 3d + 1 \le c\,3^n - d$$

   if $d \ge \frac{1}{2}$. Picking $d = 1$ and c large enough for the base case gives $T(n) = O(3^n)$.

□

Exercise 4.4-2.

. . . . . .

To figure out how many leaves there really are, let's write a recurrence $L(n)$ for the number of leaves in the recursion tree for $T(n)$. Since all the leaves in $T(n)$ belong either to the left subtree or the right subtree of the root, we have

$$L(n) = \begin{cases} 1 & \text{if } n < n_0 , \\ L(n/3) + L(2n/3) & \text{if } n \geq n_0 . \end{cases} \tag{4.15}$$

. . . . . .

Use the substitution method to prove that recurrence (4.15) has the asymptotic lower bound $L(n) = \Omega(n)$. Conclude that $L(n) = \Theta(n)$.

$$L(n) = \begin{cases} 1, & n < n_0, \\ L(n/3) + L(2n/3), & n \geq n_0. \end{cases} \tag{4.15}$$

Proof. Solution.
Intuition. Splitting n into $n/3$ and $2n/3$ preserves the total "mass" n each level; following the $2n/3$ branch gives $\Theta(\lg n)$ levels, so the number of leaves grows linearly.
Rigorous (lower bound by substitution). Pick $c = \frac{1}{n_0}$. We prove by induction that $L(n) \geq c\,n$ for all $n \geq 1$.

Base: If $n < n_0$, then $L(n) = 1 \geq cn$ since $cn \leq \frac{n_0 - 1}{n_0} < 1$.

Step: For $n \geq n_0$, assume $L(m) \geq c\,m$ for all $m < n$. Then

$$L(n) = L(n/3) + L(2n/3) \;\geq\; c\frac{n}{3} + c\frac{2n}{3} \;=\; c\,n.$$

Hence $L(n) = \Omega(n)$.
Conclusion. A symmetric induction shows $L(n) \leq d\,n$ for some constant $d > 0$: if $L(m) \leq d\,m$ for $m < n$, then $L(n) \leq d(n/3) + d(2n/3) = dn$. Thus $L(n) = O(n)$ and, combined with the lower bound,

$$\boxed{L(n) = \Theta(n)}.$$

□

Exercise 4.4-3.

. . . . . .

Let's find an asymptotic upper bound for another, more irregular, example. Figure 4.2 shows the recursion tree for the recurrence

$$T(n) = T(n/3) + T(2n/3) + \Theta(n) \,. \tag{4.14}$$

. . . . . .

Let

$$T(n) = T(n/3) + T(2n/3) + \Theta(n). \tag{4.14}$$

Use the substitution method to prove that $T(n) = \Omega(n \lg n)$. Conclude that $T(n) = \Theta(n \lg n)$.

Proof. Solution.
Intuition. Each level of the recursion tree does $\Theta(n)$ total work (the two subproblems have sizes $n/3$ and $2n/3$), and the depth is $\Theta(\lg n)$ (following the $2n/3$ branch). Hence $\Theta(n)$ per level for $\Theta(\lg n)$ levels $\Rightarrow \Theta(n \lg n)$ total.

Rigorous (lower bound by substitution). Fix a constant $c > 0$ so that the additive term in (4.14) is $\geq cn$ for all n. We prove by induction that for some $d > 0$,

$$T(n) \;\geq\; d\, n \lg n \qquad \text{for all large } n.$$

Assume $T(m) \geq d\, m \lg m$ for all $m < n$. Then

$$\begin{aligned} T(n) &\geq T(n/3) + T(2n/3) + cn \\ &\geq d\frac{n}{3}\lg\frac{n}{3} \;+\; d\frac{2n}{3}\lg\frac{2n}{3} \;+\; cn \\ &= d\, n \lg n \;+\; n\Big[c - d\Big(\lg 3 - \tfrac{2}{3}\lg 2\Big)\Big]\,. \end{aligned}$$

Taking binary logarithms ($\lg 2 = 1$), the bracket equals $c - d(\lg 3 - 2/3)$. Choose

$$0 < d \;\leq\; \frac{c}{\lg 3 - 2/3},$$

so the bracket is nonnegative. Then $T(n) \geq d\, n \lg n$, proving $T(n) = \Omega(n \lg n)$.

Matching upper bound (for the conclusion). Similarly, with the $\Theta(n)$ term upper-bounded by cn and the inductive hypothesis $T(m) \le a\, m \lg m$ for $m < n$,

$$\begin{aligned} T(n) &\le a\frac{n}{3}\lg\frac{n}{3} \;+\; a\frac{2n}{3}\lg\frac{2n}{3} \;+\; cn \\ &= a\, n \lg n \;+\; n\Big[c - a\Big(\lg 3 - \tfrac{2}{3}\Big)\Big] \;\le\; a\, n \lg n \end{aligned}$$

if $a \ge c/(\lg 3 - 2/3)$. Hence $T(n) = O(n \lg n)$.

Conclusion. Combining the lower and upper bounds yields

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

□

Exercise 4.4-4. Use a recursion tree to justify a good guess for the solution to the recurrence

$$T(n) = T(\alpha n) + T\big((1-\alpha)n\big) + \Theta(n),$$

where α is a constant with $0 < \alpha < 1$.

Proof. Solution.
Intuition (recursion tree). At each level the two subproblem sizes sum to n (namely αn and $(1-\alpha)n$), so the *total* nonrecursive work at that level is $\Theta(n)$. The longest root-to-leaf path follows the larger branch, of size βn where $\beta = \max\{\alpha, 1-\alpha\} < 1$, and therefore has depth $\Theta(\log_{1/\beta} n) = \Theta(\lg n)$. Hence the tree has $\Theta(\lg n)$ levels with $\Theta(n)$ work per level:

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

Rigorous (substitution). Let the $\Theta(n)$ term be bounded as $c_1 n \le \Theta(n) \le c_2 n$ for fixed $c_1, c_2 > 0$. Define the binary-entropy-like constant

$$H_\alpha \;:=\; -\alpha \lg \alpha - (1-\alpha)\lg(1-\alpha) \;>\; 0.$$

Lower bound. We prove $T(n) \ge d\, n \lg n$ for a suitable $d > 0$. Assuming $T(m) \ge d\, m \lg m$ for all $m < n$,

$$\begin{aligned} T(n) &\ge T(\alpha n) + T((1-\alpha)n) + c_1 n \\ &\ge d\,\alpha n \lg(\alpha n) + d\,(1-\alpha)n \lg((1-\alpha)n) + c_1 n \\ &= d\, n \lg n + n\big[c_1 + d\big(\alpha \lg \alpha + (1-\alpha)\lg(1-\alpha)\big)\big] \\ &= d\, n \lg n + n\,(c_1 - d\, H_\alpha). \end{aligned}$$

Choose $0 < d \le c_1 / H_\alpha$ to make the bracket nonnegative. Then $T(n) \ge d\, n \lg n$.

Upper bound. We prove $T(n) \le a\, n \lg n$ for a suitable $a > 0$. Assuming $T(m) \le a\, m \lg m$ for all $m < n$,

$$\begin{aligned} T(n) &\le a\,\alpha n \lg(\alpha n) + a\,(1-\alpha)n \lg((1-\alpha)n) + c_2 n \\ &= a\, n \lg n + n[c_2 - a\, H_\alpha] \;\le\; a\, n \lg n \end{aligned}$$

whenever $a \ge c_2 / H_\alpha$.

Conclusion. Combining bounds yields

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

□

. . . .

4.5 Exercises

. . . .

Exercise 4.5-1. Use the master method to give *tight* asymptotic bounds for the recurrences

$$\text{(for all parts)} \qquad T(n) = 2T(n/4) + f(n),$$
$$\text{where} \quad a = 2,\ b = 4,\ n^{\log_b a} = n^{\log_4 2} = n^{1/2}.$$

1. $f(n) = 1$.

   Intuition. Work per level shrinks geometrically; leaves dominate at size about $n^{1/2}$ total.

   Rigorous (Master, Case 1). $f(n) = \Theta(n^0) = O(n^{1/2-\varepsilon})$ with $\varepsilon = \frac{1}{2}$. Hence

   $$\boxed{T(n) = \Theta\big(n^{1/2}\big)}.$$

2. $f(n) = \sqrt{n}$.

   Intuition. Each level contributes $\Theta(\sqrt{n})$; there are $\Theta(\lg n)$ levels.

   Rigorous (Master, Case 2 with $k = 0$). $f(n) = \Theta\big(n^{1/2} \log^0 n\big)$, so

   $$\boxed{T(n) = \Theta\big(n^{1/2} \log n\big)}.$$

3. $f(n) = \sqrt{n}\ \lg^2 n$.

   Intuition. As in (b) but with an extra $\lg^2 n$ factor; we gain one more $\lg n$ from the tree height.

   Rigorous (Master, Case 2 with $k = 2$). $f(n) = \Theta\big(n^{1/2} \log^2 n\big)$, hence

   $$\boxed{T(n) = \Theta\big(n^{1/2} \log^3 n\big)}.$$

4. $f(n) = n$.

   Intuition. The nonrecursive work already exceeds the leaf cost by a polynomial factor; it dominates.

   Rigorous (Master, Case 3). Since $f(n) = \Omega\big(n^{1/2+\varepsilon}\big)$ with $\varepsilon = \frac{1}{2}$ and the regularity condition holds:

   $$a\, f(n/b) = 2 \cdot (n/4) = n/2 \le c\, n \quad \text{with } c = \tfrac{1}{2} < 1,$$

   we obtain

   $$\boxed{T(n) = \Theta(n)}.$$

5. $f(n) = n^2$.

   Intuition. Per-level work explodes; the root already dominates the sum.

   Rigorous (Master, Case 3). Here $f(n) = \Omega\big(n^{1/2+\varepsilon}\big)$ with $\varepsilon = \frac{3}{2}$ and

$$a\, f(n/b) = 2 \cdot (n/4)^2 = n^2/8 \le c\, n^2 \quad (c = 1/8 < 1),$$

   so

$$\boxed{T(n) = \Theta(n^2)}.$$

□

Exercise 4.5-2. Professor Caesar designs a divide-and-conquer matrix-multiplication algorithm that splits into a subproblems of size $n/4$ and whose divide+combine work is $\Theta(n^2)$. For which largest integer a could his algorithm be asymptotically faster than Strassen's?

Proof. Solution.
Intuition. The recurrence is

$$T(n) = a\,T(n/4) + \Theta(n^2).$$

Compare $n^{\log_4 a}$ to n^2. If $a < 16$ we get $\Theta(n^2)$; if $a = 16$ we get $\Theta(n^2 \log n)$; if $a > 16$ we get $\Theta(n^{\log_4 a})$. To beat Strassen's $n^{\lg 7}$, we need $\log_4 a < \lg 7$, i.e. $a < 49$. Thus the largest integer is $a = \boxed{48}$.
Rigorous. Apply the Master Theorem with a subproblems, $b = 4$, and $f(n) = \Theta(n^2)$. Let $\alpha = \log_4 a$.

- If $a < 16$ ($\alpha < 2$), the *regularity* condition holds: $a\,f(n/4) = a(n/4)^2 = (a/16)n^2 \le c\,n^2$ with $c = \frac{a}{16} < 1$. Hence $T(n) = \Theta(n^2)$ (Case 3).
- If $a = 16$ ($\alpha = 2$), $T(n) = \Theta(n^2 \log n)$ (Case 2).
- If $a > 16$ ($\alpha > 2$), $T(n) = \Theta(n^\alpha)$ (Case 1).

Strassen's exponent is $\lg 7 \approx 2.80735$. For $a > 16$ we need $\alpha < \lg 7$, i.e.

$$\frac{\lg a}{\lg 4} < \lg 7 \iff \lg a < 2 \lg 7 = \lg(7^2) \iff a < 49.$$

Therefore the largest integer a that can still be asymptotically faster than Strassen's is

$$\boxed{a = 48}.$$

(Indeed: $a \le 15 \Rightarrow T(n) = \Theta(n^2)$; $a = 16 \Rightarrow \Theta(n^2 \log n)$; $17 \le a \le 48 \Rightarrow T(n) = \Theta(n^{\log_4 a})$ with exponent $< \lg 7$; $a = 49$ ties Strassen; $a \ge 50$ is slower.) □

Exercise 4.5-3. Use the master method to show that the solution to the binary-search recurrence

$$T(n) = T(n/2) + \Theta(1)$$

is $T(n) = \Theta(\lg n)$.

Proof. Solution.
Intuition. Each recursive call halves the size, and we do only constant extra work per level. There are $\Theta(\lg n)$ levels until the size drops to the base case.
Rigorous (Master Theorem). Match $T(n) = a\,T(n/b) + f(n)$ with $a = 1$, $b = 2$, $f(n) = \Theta(1)$. Then

$$n^{\log_b a} = n^{\log_2 1} = n^0 = 1.$$

Since $f(n) = \Theta(1) = \Theta\big(n^{\log_b a} \log^0 n\big)$, this is *Case 2* of the master theorem with $k = 0$, giving

$$T(n) = \Theta\big(n^{\log_b a} \log^{k+1} n\big) = \Theta(\log n).$$

□

Exercise 4.5-4. Let $f(n) = \lg n$. Argue that although $f(n/2) < f(n)$, the *regularity* condition

$$a\, f(n/b) \;\le\; c\, f(n) \qquad \text{with } a = 1,\ b = 2$$

does not hold for any constant $c < 1$. Then argue that for any $\varepsilon > 0$, the case-3 condition of the master theorem,

$$f(n) = \Omega\big(n^{\log_b a + \varepsilon}\big),$$

does not hold.

Proof. Solution.

Intuition. For $f(n) = \lg n$, the ratio $\dfrac{f(n/2)}{f(n)} = \dfrac{\lg(n/2)}{\lg n} = 1 - \dfrac{1}{\lg n}$ tends to 1, not to a value strictly less than 1. So no fixed $c < 1$ can bound it for all large n. Also $\lg n$ grows slower than any positive power n^{ε}, so it cannot be $\Omega(n^{\varepsilon})$.

Rigorous.

Failure of the regularity condition. With $a = 1, b = 2$, the condition requires

$$\lg(n/2) \;\le\; c \lg n \qquad \text{for all sufficiently large } n.$$

Equivalently,

$$\frac{\lg(n/2)}{\lg n} = 1 - \frac{1}{\lg n} \;\le\; c.$$

But $1 - \dfrac{1}{\lg n} \xrightarrow[n\to\infty]{} 1$. Hence, for any fixed $c < 1$, the inequality fails for all sufficiently large n. Thus the regularity condition cannot hold.

Failure of the case-3 growth requirement. Here $\log_b a = \log_2 1 = 0$, so case 3 would need

$$f(n) = \Omega\big(n^{0+\varepsilon}\big) = \Omega\big(n^{\varepsilon}\big) \quad \text{for some } \varepsilon > 0.$$

But for every $\varepsilon > 0$,

$$\lim_{n\to\infty} \frac{\lg n}{n^{\varepsilon}} = 0$$

(e.g., by standard growth-rate facts or L'Hôpital after converting to natural logs), which means $\lg n = o(n^{\varepsilon})$ and therefore not $\Omega(n^{\varepsilon})$. Hence the case-3 hypothesis does not apply.

Remark. These facts explain why case 3 cannot be used when $f(n) = \lg n$ and $a = 1, b = 2$. (Indeed, $f(n) = \Theta(n^{\log_b a} \log^1 n)$ matches *case 2*, leading to $T(n) = \Theta(\lg^2 n)$ for the recurrence $T(n) = T(n/2) + \lg n$.) □

Exercise 4.5-5. Show that for suitable constants a, b, and ε, the function

$$f(n) = 2^{\lceil \lg n \rceil}$$

satisfies all the hypotheses of *case 3* of the master theorem *except* the regularity condition.

Proof. Solution.
Intuition. Since $2^{\lceil \lg n \rceil} \in [n, 2n)$, we have $f(n) = \Theta(n)$. Thus $f(n) = \Omega(n^{\varepsilon})$ for any fixed $\varepsilon \in (0, 1]$. However, for any choice $b \in (1, 2)$ there are infinitely many n with $\lceil \lg(n/b) \rceil = \lceil \lg n \rceil$, so $f(n/b) = f(n)$ and the regularity inequality $a\, f(n/b) \le c\, f(n)$ with $c < 1$ fails.
Rigorous.

Choose constants. Let

$$a = 1, \qquad b \in (1, 2) \text{ (e.g., } b = \tfrac{3}{2}), \qquad \varepsilon = \tfrac{1}{2}.$$

(i) Case-3 growth condition holds. Because

$$n \;\le\; 2^{\lceil \lg n \rceil} \;<\; 2n \qquad \text{for all } n \ge 1,$$

we have $f(n) \ge n/1 = \Omega(n^{1/2}) = \Omega\big(n^{\log_b a + \varepsilon}\big)$ since $\log_b a = \log_b 1 = 0$.

(ii) Regularity condition fails. Case 3 also requires a constant $c < 1$ such that

$$a\, f(n/b) \;\le\; c\, f(n) \qquad \text{for all sufficiently large } n.$$

Here $a = 1$, so we need $f(n/b) \le c\, f(n)$. Fix any integer $k \ge 0$ and set $n_k := 2^k$. Then

$$\lceil \lg n_k \rceil = \lceil \lg 2^k \rceil = k, \quad \Rightarrow \quad f(n_k) = 2^k.$$

Since $b \in (1, 2)$, we have $\lg(n_k/b) = k - \lg b \in (k - 1, k)$ and hence

$$\left\lceil \lg\Big(\frac{n_k}{b}\Big) \right\rceil = k \quad \Rightarrow \quad f(n_k/b) = 2^k = f(n_k).$$

Thus for every $c < 1$ and all such n_k,

$$f(n_k/b) = f(n_k) \;\not\le\; c\, f(n_k),$$

so the regularity condition fails on an infinite subsequence of n.

Conclusion. With $a = 1$, any fixed $b \in (1, 2)$, and $\varepsilon = \frac{1}{2}$, the function $f(n) = 2^{\lceil \lg n \rceil}$ satisfies the case-3 growth hypothesis $f(n) = \Omega\big(n^{\log_b a + \varepsilon}\big)$ but *violates* the regularity condition $a\, f(n/b) \le c\, f(n)$ for every $c < 1$. □

4.6 Exercises

Exercise 4.6-1. Show that

$$\sum_{j=0}^{\lfloor \log_b n \rfloor} \big(\log_b n - j\big)^k \;=\; \Omega\big((\log_b n)^{k+1}\big),$$

for constant $b > 1$ and fixed integer $k \geq 0$.

Proof. Solution.
Intuition. Let $L = \lfloor \log_b n \rfloor$. The sum is just $\sum_{i=0}^{L} i^k$, whose growth is on the order of the $(k{+}1)$-st power of L. A crude lower bound: take only the largest half of the terms—there are $\Theta(L)$ of them, each at least $(L/2)^k$, giving $\Omega(L^{k+1})$.
Rigorous. Let $L = \lfloor \log_b n \rfloor$ and change variables $i = L - j$:

$$\sum_{j=0}^{L} \big(\log_b n - j\big)^k \;\geq\; \sum_{j=0}^{L} (L-j)^k \;=\; \sum_{i=0}^{L} i^k \;\geq\; \sum_{i=\lceil L/2 \rceil}^{L} i^k.$$

For $i \in [\lceil L/2 \rceil,\, L]$, we have $i \geq L/2$, hence

$$\sum_{i=\lceil L/2 \rceil}^{L} i^k \;\geq\; \left(L - \left\lceil \frac{L}{2} \right\rceil + 1\right) \left(\frac{L}{2}\right)^k \;\geq\; \frac{L}{2} \cdot \left(\frac{L}{2}\right)^k \;=\; \frac{1}{2^{k+1}}\, L^{k+1}.$$

Therefore,

$$\sum_{j=0}^{\lfloor \log_b n \rfloor} \big(\log_b n - j\big)^k \;\geq\; \frac{1}{2^{k+1}} \left\lfloor \log_b n \right\rfloor^{k+1} \;=\; \Omega\big((\log_b n)^{k+1}\big).$$

□

Exercise 4.6-2. Show that *case 3 of the master theorem is overstated* in the sense that the *regularity condition*

$$a\,f(n/b) \;\le\; c\,f(n) \qquad \text{for some constant } c < 1$$

already implies the existence of $\varepsilon > 0$ such that

$$f(n) = \Omega\big(n^{\log_b a+\varepsilon}\big).$$

(Compare with Lemma 4.3, where case 3 does not separately assume $f(n) = \Omega(n^{\log_b a+\varepsilon})$.)

Proof. Solution.

Intuition. From $af(n/b) \le cf(n)$ we get $f(n) \ge \frac{a}{c} f(n/b)$. Iterate this inequality down the geometric sequence $n, b^{-1}n, b^{-2}n, \ldots$. After $t \approx \log_b n$ steps, $f(n) \gtrsim (a/c)^t \cdot$ (constant) $= n^{\log_b(a/c)}$. Since $c < 1$, $\log_b(a/c) = \log_b a - \log_b c = \log_b a + \underbrace{\log_b(1/c)}_{=:\ \varepsilon>0}$.

Rigorous. Assume there exists $n_0 \ge 1$ so that for all $n \ge n_0$ the regularity condition

$$a\,f(n/b) \;\le\; c\,f(n) \qquad (0 < c < 1) \tag{$*$}$$

holds and f is nonnegative. Then for all $n \ge n_0$,

$$f(n) \;\ge\; \frac{a}{c}\, f(n/b). \tag{1}$$

Fix $n \ge n_0$ and choose the largest integer $t \ge 0$ with $n/b^t \ge n_0$. Iterating (1) t times yields

$$f(n) \;\ge\; \left(\frac{a}{c}\right)^t f\!\left(\frac{n}{b^t}\right).$$

Set $m := n/b^t \in [n_0,\, bn_0)$, so $f(m) \ge f(n_0) =: K \ge 0$. Thus

$$f(n) \;\ge\; K\left(\frac{a}{c}\right)^t \;=\; K\, b^{t\log_b(a/c)} \;=\; K\left(\frac{n}{m}\right)^{\log_b(a/c)} \;\ge\; K'\, n^{\log_b(a/c)}$$

for a constant $K' > 0$ (since $m < bn_0$ is bounded above by a constant multiple of 1). Let

$$\varepsilon \;:=\; -\log_b c \;=\; \log_b(1/c) \;>\; 0.$$

Then $\log_b(a/c) = \log_b a + \varepsilon$, and we have shown

$$\boxed{\; f(n) \;=\; \Omega\big(n^{\log_b a+\varepsilon}\big) \;}.$$

Conclusion. The regularity condition alone forces polynomial growth above $n^{\log_b a}$ by a positive margin ε. Hence case 3 of the master theorem need not additionally assume $f(n) = \Omega(n^{\log_b a+\varepsilon})$—that property follows from regularity. □

Exercise 4.6-3. For

$$g(n) = \sum_{j=0}^{\lfloor \log_b n \rfloor} a^j \, f\Big(\frac{n}{b^j}\Big) \tag{4.19}$$

prove that if

$$f(n) = \Theta\Big(\frac{n^{\log_b a}}{\lg n}\Big),$$

then

$$g(n) = \Theta\big(n^{\log_b a} \lg\lg n\big).$$

Conclude that a master recurrence $T(n) = a\,T(n/b) + f(n)$ has solution $T(n) = \Theta\big(n^{\log_b a} \lg\lg n\big)$.

Proof. Solution.

Intuition. Let $s = \log_b a$. Each summand simplifies to

$$a^j \, f\Big(\frac{n}{b^j}\Big) \asymp a^j \, \frac{(n/b^j)^s}{\lg(n/b^j)} = \frac{n^s}{\lg n - j \lg b}.$$

Hence $g(n)$ is n^s times a harmonic sum over the grid $\lg n$, $\lg n - \lg b$, …, which totals $\Theta(\lg\lg n)$.

Rigorous. Write $s = \log_b a$ and $L = \lfloor \log_b n \rfloor$. There exist constants $0 < c_1 \le c_2$ such that

$$c_1 \, \frac{x^s}{\lg x} \;\le\; f(x) \;\le\; c_2 \, \frac{x^s}{\lg x} \quad \text{for all } x \ge 2.$$

Then

$$c_1 \, n^s \sum_{j=0}^{L-1} \frac{1}{\lg n - j \lg b} \;\le\; g(n) \;\le\; c_2 \, n^s \sum_{j=0}^{L-1} \frac{1}{\lg n - j \lg b} \;+\; O(n^s),$$

where we isolated the (at most one) last term $j = L$, which contributes $O(n^s)$.

Let $c = \lg b > 0$. The inner sum equals

$$\sum_{j=0}^{L-1} \frac{1}{\lg n - jc} = \frac{1}{c} \sum_{i=1}^{L} \frac{1}{i} = \frac{1}{c} \, H_L = \Theta(\ln L) = \Theta(\lg L) = \Theta(\lg\lg n),$$

since $L = \Theta(\log_b n)$ and changing log bases only affects constants. Therefore

$$g(n) = \Theta\big(n^s \lg\lg n\big) = \Theta\big(n^{\log_b a} \lg\lg n\big).$$

Master recurrence. By Lemma 4.3, $T(n) = a\,T(n/b) + f(n)$ solves to $T(n) = \Theta(g(n))$. Hence

$$\boxed{T(n) = \Theta\big(n^{\log_b a} \lg\lg n\big)}.$$

□

. . . .

4.7 Exercises

. . . .

Exercise 4.7-1.

. . . . . .

In particular, we'll look at the class of algorithmic divide-and-conquer recurrences originally studied by M. Akra and L. Bazzi [13]. These ***Akra-Bazzi*** recurrences take the form

$$T(n) = f(n) + \sum_{i=1}^{k} a_i T(n/b_i) , \tag{4.22}$$

where k is a positive integer; all the constants $a_1, a_2, \ldots, a_k \in \mathrm{R}$ are strictly positive; all the constants $b_1, b_2, \ldots, b_k \in \mathrm{R}$ are strictly greater than 1; and the driving function $f(n)$ is defined on sufficiently large nonnegative reals and is itself nonnegative.

. . . . . .

Consider an Akra–Bazzi recurrence $T(n)$ on the reals as given in recurrence (4.22), and define $T'(n)$ as

$$T'(n) \;=\; c\,f(n) \;+\; \sum_{i=1}^{k} a_i\, T'\!\left(\frac{n}{b_i}\right),$$

where $c > 0$ is constant. Prove that whatever the implicit initial conditions for $T(n)$ might be, there exist initial conditions for $T'(n)$ such that $T'(n) = cT(n)$ for all $n > 0$. Conclude that we can drop the asymptotics on a driving function in any Akra–Bazzi recurrence without affecting its asymptotic solution.

Proof. Solution.
Intuition. The recurrence is linear in both T and f. Scaling f by c yields a solution scaled by c. If we also scale the base values by c, the entire solution scales: $T' = cT$.

Rigorous. Let $n_0 > 0$ be the threshold for which initial values are specified. Define the initial conditions for T' by

$$T'(x) \;:=\; c\,T(x) \qquad \text{for all } x \in (0, n_0]. \tag{$\dagger$}$$

We prove by strong induction on $n \geq n_0$ that $T'(n) = c\,T(n)$. Assume $T'(x) = c\,T(x)$

for all $x < n$ with $x > 0$. Since $b_i > 1$, we have $n/b_i < n$, hence

$$T'(n) = \sum_{i=1}^{k} a_i\, T'\!\left(\frac{n}{b_i}\right) + c\, f(n) = \sum_{i=1}^{k} a_i \left[c\, T\!\left(\frac{n}{b_i}\right) \right] + c\, f(n)$$
$$= c\left(\sum_{i=1}^{k} a_i\, T\!\left(\frac{n}{b_i}\right) + f(n) \right) = c\, T(n).$$

Together with (†), this holds for all $n > 0$.

Conclusion. Multiplying the driving term f by a constant $c > 0$ multiplies the entire solution by c (after scaling base values accordingly). Therefore the multiplicative constant hidden in $\Theta(\cdot)$ on the driving function does not change the Θ-class of the Akra–Bazzi solution. □

Exercise 4.7-2.

The polynomial-growth condition

If the driving function $f(n)$ in equation (4.22) is well behaved in the following sense, it's okay to drop floors and ceilings.

> A function $f(n)$ defined on all sufficiently large positive reals satisfies the ***polynomial-growth condition*** if there exists a constant $\hat{n} > 0$ such that the following holds: for every constant $\phi \geq 1$, there exists a constant $d > 1$ (depending on ϕ) such that $f(n)/d \leq f(\psi n) \leq df(n)$ for all $1 \leq \psi \leq \phi$ and $n \geq \hat{n}$.

This definition may be one of the hardest in this textbook to get your head around. To a first order, it says that $f(n)$ satisfies the property that $f(\Theta(n)) = \Theta(f(n))$, although the polynomial-growth condition is actually somewhat stronger (see Exercise 4.7-4). The definition also implies that $f(n)$ is asymptotically positive (see Exercise 4.7-3).

Show that $f(n) = n^2$ satisfies the polynomial-growth condition but that $f(n) = 2^n$ does not.

The polynomial-growth condition (textbook). A function $f(n)$ defined on all sufficiently large positive reals satisfies the *polynomial-growth condition* if there exists a constant $\hat{n} > 0$ such that the following holds: for every constant $\phi \geq 1$, there exists a constant $d > 1$ (depending on ϕ) such that

$$\frac{f(n)}{d} \;\leq\; f(\psi n) \;\leq\; d\,f(n) \qquad \text{for all } \; 1 \leq \psi \leq \phi \text{ and all } \; n \geq \hat{n}. \tag{PGC}$$

Proof. Solution.

Intuition. Polynomials scale by fixed powers: $f(\psi n) = \psi^k f(n)$, which stays within a ψ-dependent constant factor of $f(n)$ for any bounded ψ. Exponentials scale as $f(\psi n) = \big(f(n)\big)^{\psi}$; even for fixed $\psi > 1$, the ratio to $f(n)$ blows up with n, so no *n-independent* constant factor can contain it.

Rigorous.

(i) $f(n) = n^2$ satisfies (PGC). For any $\psi \geq 1$,

$$f(\psi n) = (\psi n)^2 = \psi^2 n^2 = \psi^2 f(n).$$

Given an arbitrary $\phi \geq 1$, choose $d := \phi^2 > 1$ and any $\hat{n} > 0$. For all $n \geq \hat{n}$ and all $1 \leq \psi \leq \phi$,

$$\frac{f(n)}{d} = \frac{n^2}{\phi^2} \;\leq\; \psi^2 n^2 = f(\psi n) \;\leq\; \phi^2 n^2 = d\, f(n).$$

Thus (PGC) holds for $f(n) = n^2$.

(ii) $f(n) = 2^n$ does not satisfy (PGC). Fix any $\phi > 1$ and suppose, for contradiction, that there exist $d > 1$ and $\hat{n}$ satisfying (PGC). Take $\psi := \phi$. Then for all $n \geq \hat{n}$,

$$\frac{f(\phi n)}{f(n)} = \frac{2^{\phi n}}{2^n} = 2^{(\phi-1)n} \;\leq\; d.$$

But $2^{(\phi-1)n} \to \infty$ as $n \to \infty$, contradicting the existence of an n-independent constant d. Hence no such d exists, and (PGC) fails for $f(n) = 2^n$.

Conclusion. $f(n) = n^2$ satisfies the polynomial-growth condition (with $d = \phi^2$), whereas $f(n) = 2^n$ violates it because the required bounding constant d cannot be independent of n. □

Exercise 4.7-3. Let $f(n)$ be a function that satisfies the polynomial-growth condition. Prove that $f(n)$ is asymptotically positive, that is, there exists a constant $n_0 \geq 0$ such that $f(n) \geq 0$ for all $n \geq n_0$.

Proof. Solution.
Intuition. Plug $\psi = 1$ into (PGC). Then for all large n we must have $\frac{f(n)}{d} \leq f(n) \leq df(n)$ with $d > 1$. If $f(n)$ were negative, the right inequality would read $f(n) \leq df(n)$ with $df(n) < f(n)$—impossible. Hence $f(n)$ cannot be negative for sufficiently large n.

Rigorous. Fix $\phi = 1$ in (PGC). Then there exists $d > 1$ and $\hat{n} > 0$ such that for all $n \geq \hat{n}$,

$$\frac{f(n)}{d} \;\leq\; f(n) \;\leq\; d\,f(n). \tag{1}$$

Suppose, for contradiction, that there is some $n \geq \hat{n}$ with $f(n) < 0$. Since $d > 1$ and $f(n) < 0$, we have $d\,f(n) < f(n)$, which violates the right-hand inequality in (1). Therefore no such n exists, and we must have $f(n) \geq 0$ for all $n \geq \hat{n}$.

Set $n_0 := \hat{n}$. Then $f(n) \geq 0$ for all $n \geq n_0$; i.e., f is *asymptotically positive*.

Conclusion. Any function satisfying the polynomial-growth condition is eventually nonnegative. □

Exercise 4.7-4. Give an example of a function $f(n)$ that does not satisfy the polynomial-growth condition but for which $f(\Theta(n)) = \Theta(f(n))$.

The polynomial-growth condition (textbook). There exists $\hat{n} > 0$ such that for every $\phi \geq 1$ there is a constant $d > 1$ (depending on ϕ) with

$$\frac{f(n)}{d} \;\leq\; f(\psi n) \;\leq\; d\, f(n) \qquad \text{for all } \; 1 \leq \psi \leq \phi \text{ and } n \geq \hat{n}. \tag{PGC}$$

Construction. Define a piecewise exponent via the fractional part of $\log_2 n$:

$$e(n) \;:=\; \begin{cases} 2, & \{\log_2 n\} \in [0, \frac{1}{2}), \\ 10, & \{\log_2 n\} \in [\frac{1}{2}, 1), \end{cases} \qquad f(n) \;:=\; n^{\,e(n)}.$$

Intuition. Across each dyadic scale $[2^k, 2^{k+1})$, f is polynomial but its exponent flips between 2 and 10 depending on where n falls within the interval. Doubling preserves the fractional part of $\log_2 n$, so $f(2n)$ is always a fixed constant factor times $f(n)$ (hence $f(2n) = \Theta(f(n))$). But multiplying by a different constant (e.g. 3/2) shifts the fractional part, and infinitely often moves n from the "exponent 2" zone to the "exponent 10" zone, making $f(\frac{3}{2}n)/f(n)$ blow up with n. Therefore (PGC) fails.

Rigorous.

1) $f(\Theta(n)) = \Theta(f(n))$ (witness $\times 2$). For all $n > 0$,

$$\{\log_2(2n)\} = \{\log_2 n + 1\} = \{\log_2 n\},$$

hence $e(2n) = e(n)$ and therefore

$$f(2n) = (2n)^{e(2n)} = 2^{\,e(n)}\, n^{e(n)} = \Theta\big(f(n)\big),$$

with constants between 2^2 and 2^{10}, independent of n.

2) (PGC) fails. Fix $\psi = \frac{3}{2}$ (so any $\phi \geq \frac{3}{2}$ must cover this ψ). Let $n_k := 2^k$, for which $\{\log_2 n_k\} = 0 \in [0, \frac{1}{2})$ and thus $e(n_k) = 2$. Then

$$\log_2\big(\tfrac{3}{2}\, n_k\big) = \log_2(3) + k - 1, \quad \text{whose fractional part is } \{\log_2 3\} \approx 0.585 \in [\tfrac{1}{2}, 1),$$

so $e(\frac{3}{2}\, n_k) = 10$. Consequently,

$$\frac{f(\frac{3}{2}\, n_k)}{f(n_k)} = \frac{(\frac{3}{2}\, n_k)^{10}}{n_k^2} = \Big(\frac{3^{10}}{2^{10}}\Big)\, n_k^8 = \Theta\big(2^{8k}\big) \;\xrightarrow[k\to\infty]{}\; \infty.$$

Thus no constant d can satisfy $f(\psi n) \le d\, f(n)$ for all large n with this fixed $\psi = \frac{3}{2}$, so (PGC) fails.

Conclusion. The function $f(n) = n^{e(n)}$ above satisfies $f(2n) = \Theta\big(f(n)\big)$ (hence exhibits the informal property $f(\Theta(n)) = \Theta(f(n))$), yet it violates the polynomial-growth condition. This shows (PGC) is strictly stronger than the first-order rule of thumb $f(\Theta(n)) = \Theta(f(n))$. □

Exercise 4.7-5. Use the Akra–Bazzi method to solve the following recurrences.

(a) $T(n) = T(n/2) + T(n/3) + T(n/6) + n \lg n$.

(b) $T(n) = 3T(n/3) + 8T(n/4) + n^2 \lg n$.

(c) $T(n) = (2/3)T(n/3) + (1/3)T(2n/3) + \lg n$.

(d) $T(n) = (1/3)T(n/3) + 1/n$.

(e) $T(n) = 3T(n/3) + 3T(2n/3) + n^2$.

Akra–Bazzi template. For

$$T(n) = \sum_{i=1}^{k} a_i \, T\left(\frac{n}{b_i}\right) + f(n),$$

let p be the unique real satisfying $\sum_{i=1}^{k} a_i \, b_i^{-p} = 1$. Then

$$T(n) = \Theta\Big(n^p \Big(1 + \int_1^n \frac{f(u)}{u^{p+1}} \, du\Big)\Big),$$

whenever the side conditions hold.

(a) $T(n) = T(n/2) + T(n/3) + T(n/6) + n \lg n$.
Solve for p: $2^{-p} + 3^{-p} + 6^{-p} = 1$ gives $p = 1$ (since $1/2 + 1/3 + 1/6 = 1$).
Integral: $\displaystyle\int_1^n \frac{u \lg u}{u^2} \, du = \int_1^n \frac{\lg u}{u} \, du = \Theta((\lg n)^2)$.
Answer: $\boxed{T(n) = \Theta\big(n(\lg n)^2\big)}$.

(b) $T(n) = 3T(n/3) + 8T(n/4) + n^2 \lg n$.
Solve for p: $3 \cdot 3^{-p} + 8 \cdot 4^{-p} = 1$ has the unique root $p \in (1, 2)$.
Integral: $\displaystyle\int_1^n \frac{u^2 \lg u}{u^{p+1}} \, du = \int_1^n u^{1-p} \lg u \, du = \Theta\big(n^{2-p} \lg n\big)$ (since $1 - p \in (-1, 0)$).
Answer: $n^p \cdot n^{2-p} \lg n = \boxed{\Theta\big(n^2 \lg n\big)}$.

(c) $T(n) = (2/3)T(n/3) + (1/3)T(2n/3) + \lg n$.
Rewrite $T(2n/3) = T\big(n/(3/2)\big)$. Thus $a_1 = 2/3, b_1 = 3$ and $a_2 = 1/3, b_2 = 3/2$.
Solve for p: $(2/3)\, 3^{-p} + (1/3)\, (3/2)^{-p} = 1$ has $p = 0$ (plug in).
Integral: $\displaystyle\int_1^n \frac{\lg u}{u^1} \, du = \Theta((\lg n)^2)$.

Answer: $\boxed{T(n) = \Theta\big((\lg n)^2\big)}$.

(d) $T(n) = (1/3)T(n/3) + 1/n$.
Solve for p: $(1/3)\,3^{-p} = 1 \Rightarrow p = -1$.
Integral: $\displaystyle\int_1^n \frac{1/u}{u^{p+1}}\,du = \int_1^n \frac{1}{u}\,du = \Theta(\lg n)$.
Answer: $n^p(1 + \Theta(\lg n)) = n^{-1}\Theta(\lg n)$, i.e. $\boxed{T(n) = \Theta\big((\lg n)/n\big)}$.

(e) $T(n) = 3T(n/3) + 3T(2n/3) + n^2$.
Rewrite $T(2n/3) = T\big(n/(3/2)\big)$, with $a_1 = 3, b_1 = 3$ and $a_2 = 3, b_2 = 3/2$.
Solve for p: $3 \cdot 3^{-p} + 3 \cdot (3/2)^{-p} = 1$ gives $p = 3$ (check: $1/9 + 24/27 = 1$).
Integral: $\displaystyle\int_1^n \frac{u^2}{u^{p+1}}\,du = \int_1^n u^{-2}\,du = \Theta(1)$.
Answer: $\boxed{T(n) = \Theta\big(n^3\big)}$.

Exercise 4.7-6. Use the Akra–Bazzi method to prove the continuous master theorem.

Statement (Continuous Master Theorem). Let

$$T(n) \;=\; a\,T(n/b) \;+\; f(n) \qquad (a>0,\; b>1),$$

where f satisfies the polynomial–growth condition. Let $p = \lg_b a$. Then

$$\boxed{T(n) \;=\; \Theta\Big(n^p\Big(1+\int_1^n \frac{f(u)}{u^{p+1}}\,du\Big)\Big).} \tag{CMT}$$

Equivalently (by elementary integral estimates), the following cases hold:

$$T(n) = \begin{cases} \Theta\big(n^p\big), & \text{if } f(n) = O\big(n^{p-\varepsilon}\big) \text{ for some } \varepsilon>0, \\ \Theta\big(n^p \log^{k+1} n\big), & \text{if } f(n) = \Theta\big(n^p \log^k n\big) \text{ for some } k \ge -1, \\ \Theta\big(f(n)\big), & \text{if } f(n) = \Omega\big(n^{p+\varepsilon}\big) \text{ and regularity holds.} \end{cases}$$

Proof. Solution.
Intuition. The Akra–Bazzi theorem already gives the exact growth for general

$$T(n) = \sum_{i=1}^{k} a_i\,T(n/b_i) + f(n)$$

as $T(n) = \Theta\big(n^p(1+\int_1^n f(u)/u^{p+1}\,du)\big)$, where p solves $\sum_i a_i b_i^{-p} = 1$. Specializing to the single-branch case ($k=1$, $a_1=a$, $b_1=b$) gives the continuous master theorem in one line. The familiar three-case form follows from elementary evaluation of the integral depending on how f compares to u^p.

Rigorous (from Akra–Bazzi). Apply the Akra–Bazzi theorem to the recurrence

$$T(n) = a\,T(n/b) + f(n), \qquad a>0,\; b>1,$$

with p determined by $a\,b^{-p}=1$, i.e. $p=\lg_b a$. The theorem yields

$$T(n) \;=\; \Theta\Big(n^p\Big(1+\int_1^n \frac{f(u)}{u^{p+1}}\,du\Big)\Big),$$

provided f satisfies the polynomial–growth condition and the side conditions of the theorem (which hold under the textbook hypotheses). This is exactly (CMT).

To obtain the usual "three-case" presentation, bound the integral by simple calculus.

Case 1: $f(n) = O(n^{p-\varepsilon})$ for some $\varepsilon > 0$. Then for large u, $\frac{f(u)}{u^{p+1}} = O(u^{-1-\varepsilon})$, and $\int_1^\infty u^{-1-\varepsilon}\,du < \infty$, hence $\int_1^n \frac{f(u)}{u^{p+1}}\,du = O(1)$. Therefore $T(n) = \Theta(n^p)$.

Case 2: $f(n) = \Theta\big(n^p \log^k n\big)$ with $k \geq -1$. Then

$$\int_1^n \frac{f(u)}{u^{p+1}}\,du \;=\; \Theta\bigg(\int_1^n \frac{\log^k u}{u}\,du\bigg) \;=\; \Theta\big(\log^{k+1} n\big),$$

(using that $\int (\log^k u)/u\,du = \frac{1}{k+1}\log^{k+1} u$ for $k > -1$, and $\Theta(\log\log n)$ when $k = -1$). Hence $T(n) = \Theta\big(n^p \log^{k+1} n\big)$ (interpreting $k = -1$ as $\Theta(n^p \log\log n)$).

Case 3: $f(n) = \Omega\big(n^{p+\varepsilon}\big)$ for some $\varepsilon > 0$ and the regularity condition $a\,f(n/b) \leq c\,f(n)$ for some $c < 1$. Then for large u,

$$\int_1^n \frac{f(u)}{u^{p+1}}\,du \;\geq\; \int_{n/2}^n \frac{f(u)}{u^{p+1}}\,du \;=\; \Omega\Big(\frac{f(n)}{n^{p+1}} \cdot n\Big) \;=\; \Omega\Big(\frac{f(n)}{n^p}\Big),$$

by polynomial growth, and the Akra–Bazzi framework (or the regularity condition) also ensures $T(n) = \Theta\big(n^p \int_1^n f(u)/u^{p+1} du\big) = \Theta(f(n))$.

Conclusion. The Akra–Bazzi theorem specialized to one branch ($k = 1$) is the continuous master theorem. Evaluating the integral establishes the familiar three-case form. □

. . . .

Ch4 Problems

. . . .

Problem 4-1. Give asymptotically tight upper and lower bounds for $T(n)$ in each of the following recurrences. Justify your answers.

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

   Intuition. Two halves, cost n^3. Since n^3 grows faster than the splitting term $n^{\log_2 2} = n$, the n^3 dominates.

   Rigorous. By the Master Theorem, $a = 2$, $b = 2$, $f(n) = n^3$. We have $n^{\log_2 2} = n$. Since $n^3 = \Omega(n^{1+\varepsilon})$, and the regularity condition holds, Case 3 applies.

   $$T(n) = \Theta(n^3).$$

2. $T(n) = T(8n/11) + n$.

   Intuition. The problem shrinks by factor $8/11 < 1$, so the recursion depth is $\Theta(\log n)$. At each level, we spend $\Theta(n)$ shrinking geometrically. The top level dominates.

   Rigorous. By the Akra–Bazzi theorem, $a = 1$, $b = 11/8 > 1$, so p solves $ab^{-p} = 1$, i.e. $(11/8)^{-p} = 1$. Thus $p = 0$. Then

   $$T(n) = \Theta\Big(n^0\Big(1 + \int_1^n \tfrac{u}{u^{0+1}}du\Big)\Big) = \Theta\Big(\int_1^n 1\,du\Big) = \Theta(n).$$

3. $T(n) = 16T(n/4) + n^2$.

   Intuition. We split into 16 subproblems of size $n/4$. This yields work exponent $\log_4 16 = 2$. The combine cost is also n^2. Balanced. Expect $n^2 \log n$.

   Rigorous. Master Theorem: $a = 16$, $b = 4$, $f(n) = n^2$. Then $n^{\log_4 16} = n^2$. Since $f(n) = \Theta(n^2)$, Case 2 applies:

   $$T(n) = \Theta(n^2 \log n).$$

4. $T(n) = 4T(n/2) + n^2 \log n$.

   Intuition. Now $a = 4$, $b = 2$, so $n^{\log_2 4} = n^2$. The extra $\log n$ factor in $f(n)$ means $f(n) = \Theta(n^2 \log n)$. This matches Case 2 with $k = 1$.

   Rigorous. Master Theorem with $a = 4$, $b = 2$, gives $n^{\log_2 4} = n^2$. Compare with $f(n) = n^2 \log n = \Theta(n^2 \log^1 n)$. Case 2:

   $$T(n) = \Theta(n^2 \log^2 n).$$

5. $T(n) = 8T(n/3) + n^2$.

   Intuition. $a = 8$, $b = 3$. The critical exponent is $\log_3 8 \approx 1.89$. Since n^2 grows slightly faster than $n^{1.89}$, the combine term dominates, so $T(n) = \Theta(n^2)$.

   Rigorous. $n^{\log_3 8} \approx n^{1.893}$. Since $f(n) = n^2 = \Omega(n^{1.893+\varepsilon})$ with $\varepsilon \approx 0.1$, and regularity holds, Case 3 applies:

$$T(n) = \Theta(n^2).$$

6. $T(n) = 7T(n/2) + n^2 \log n$.

   Intuition. $a = 7$, $b = 2$, so $n^{\log_2 7} \approx n^{2.807}$. The combine cost $n^2 \log n$ is smaller than $n^{2.807}$. So the branching dominates.

   Rigorous. $f(n) = O(n^{2.807-\varepsilon})$ (take $\varepsilon \approx 0.8$). Thus Case 1 applies:

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

7. $T(n) = 2T(n/4) + \sqrt{n}$.

   Intuition. $a = 2$, $b = 4$. Then $n^{\log_4 2} = n^{1/2}$. Combine term is also $\sqrt{n}$. Balanced $\rightarrow$ expect $\sqrt{n} \log n$.

   Rigorous. Master Theorem with $a = 2$, $b = 4$: $n^{\log_4 2} = n^{1/2}$. Since $f(n) = \Theta(n^{1/2})$, Case 2 applies:

$$T(n) = \Theta(\sqrt{n} \log n).$$

8. $T(n) = T(n-2) + n^2$.

   Intuition. This decreases linearly in n rather than geometrically. We expand for about $n/2$ steps, each costing about n^2. That sums to $\Theta(n^3)$.

   Rigorous. Unroll: $T(n) = n^2 + (n-2)^2 + (n-4)^2 + \cdots + O(1)$. This is $\sum_{i=1}^{n/2} \Theta((n-2i)^2) = \Theta(\sum_{j=1}^{n} j^2) = \Theta(n^3)$. Thus

$$T(n) = \Theta(n^3).$$

Problem 4-2. Parameter-passing costs.

Throughout this book, we assume that parameter passing during procedure calls takes constant time, even if an N-element array is being passed. This assumption is valid in most systems because a pointer to the array is passed, not the array itself. This problem examines the implications of three parameter-passing strategies:

1. Arrays are passed by pointer. Time $= \Theta(1)$.
2. Arrays are passed by copying. Time $= \Theta(N)$, where N is the size of the array.
3. Arrays are passed by copying only the subrange that might be accessed by the called procedure. Time $= \Theta(n)$ if the subarray contains n elements.

Consider the following three algorithms:

1. The recursive binary-search algorithm for finding a number in a sorted array (see Exercise 2.3-6).
2. The MERGE-SORT procedure from Section 2.3.1.
3. The MATRIX-MULTIPLY-RECURSIVE procedure from Section 4.1.

Give nine recurrences $T_{a1}(N, n)$, $T_{a2}(N, n)$, $\ldots$, $T_{c3}(N, n)$ for the worst-case running times of each of the three algorithms above when arrays and matrices are passed using each of the three parameter-passing strategies above. Solve your recurrences, giving tight asymptotic bounds.

Solution.

Intuition (one paragraph). When passing by *pointer*, calls are cheap, so you see the textbook costs. When passing by *copying the entire array/matrix*, *every* call pays a fixed overhead depending on the original container size ($\Theta(N)$ or $\Theta(N^2)$), and because the number of calls grows with the recursion tree, this overhead can snowball (e.g., $\Theta(Nn)$ for merge sort, $\Theta(N^2n^3)$ for recursive matrix multiply). When copying *just the active subrange*, the overhead scales with the current subproblem size ($\Theta(n)$ or $\Theta(n^2)$) and thus matches the usual combine work, leaving asymptotics unchanged up to constants/logs.

Rigorous. Below, N is the full array (or matrix side-length) and n is the current subproblem size. All bounds are worst-case.

1. **Recursive Binary Search.**

   (1) By pointer.

$$T_{a1}(N,n) \;=\; T_{a1}(N,n/2) + \Theta(1) \quad \Rightarrow \quad T_{a1}(N,n) = \Theta(\log n).$$

   (2) Copy whole array.

$$T_{a2}(N,n) \;=\; T_{a2}(N,n/2) + \Theta(N).$$

   Depth $= \Theta(\log n)$, so $T_{a2}(N,n) = \Theta(N \log n)$.

   (3) Copy subrange.

$$T_{a3}(N,n) \;=\; T_{a3}(N,n/2) + \Theta(n) \quad \Rightarrow \quad T_{a3}(N,n) = \Theta(n)$$

   since $\sum_{i\geq 0} n/2^i = \Theta(n)$.

2. **Merge Sort.**

   (1) By pointer.

$$T_{b1}(N,n) \;=\; 2T_{b1}(N,n/2) + \Theta(n) \quad \Rightarrow \quad T_{b1}(N,n) = \Theta(n \log n).$$

   (2) Copy whole array.

$$T_{b2}(N,n) \;=\; 2T_{b2}(N,n/2) + \Theta(n) + \Theta(N).$$

   At level i there are 2^i calls, so the copy overhead that level is $\Theta(N\,2^i)$. Summing levels $i = 0$ to $\lfloor \log n \rfloor$ gives total copy cost $\Theta\big(N \sum_i 2^i\big) = \Theta(Nn)$. Adding the usual $\Theta(n \log n)$ merging cost,

$$T_{b2}(N,n) = \Theta(Nn + n \log n) = \Theta(Nn)$$

   (and if $n = N$ initially, this is $\Theta(n^2)$).

   (3) Copy subrange.

$$T_{b3}(N,n) \;=\; 2T_{b3}(N,n/2) + \Theta(n) + \Theta(n) \;=\; 2T_{b3}(N,n/2) + \Theta(n)$$

$$\Rightarrow \quad T_{b3}(N,n) = \Theta(n \log n).$$

3. **Matrix-Multiply-Recursive.**

(1) By pointer.

$$T_{c1}(N,n) \;=\; 8T_{c1}(N,n/2) + \Theta(n^2) \quad \Rightarrow \quad T_{c1}(N,n) = \Theta(n^3).$$

(2) Copy whole matrix.

$$T_{c2}(N,n) \;=\; 8T_{c2}(N,n/2) + \Theta(n^2) + \Theta(N^2).$$

At level i there are 8^i calls, so the copy overhead that level is $\Theta(N^2\,8^i)$. Summing to depth $\log n$ gives total overhead $\Theta\big(N^2 \sum_i 8^i\big) = \Theta(N^2 n^3)$. Including arithmetic work (which also sums to $\Theta(n^3)$),

$$T_{c2}(N,n) = \Theta(N^2n^3 + n^3) = \Theta(N^2n^3)$$

(and with $n = N$ initially, $\Theta(n^5)$).

(3) Copy subrange.

$$T_{c3}(N,n) \;=\; 8T_{c3}(N,n/2) + \Theta(n^2) + \Theta(n^2) \;=\; 8T_{c3}(N,n/2) + \Theta(n^2)$$

$$\Rightarrow \quad T_{c3}(N,n) = \Theta(n^3).$$

Summary (tight bounds).

	Pointer (1)	Copy All (2)	Copy Subrange (3)
Binary Search	$\Theta(\log n)$	$\Theta(N \log n)$	$\Theta(n)$
Merge Sort	$\Theta(n \log n)$	$\Theta(Nn)$	$\Theta(n \log n)$
Matrix Mult.	$\Theta(n^3)$	$\Theta(N^2n^3)$	$\Theta(n^3)$

□

Problem 4-3. Solving recurrences with a change of variables.

Sometimes, a little algebraic manipulation can make an unknown recurrence similar to one you have seen before. Let's solve the recurrence

$$T(n) = 2T(\sqrt{n}) + \Theta(\lg n) \tag{4.25}$$

by using the change-of-variables method.

1. Define $m = \lg n$ and $S(m) = T(2^m)$. Rewrite recurrence (4.25) in terms of m and $S(m)$.
2. Solve your recurrence for $S(m)$.
3. Use your solution for $S(m)$ to conclude that $T(n) = \Theta(\lg n \lg\lg n)$.
4. Sketch the recursion tree for recurrence (4.25), and use it to explain intuitively why the solution is $T(n) = \Theta(\lg n \lg\lg n)$.

Solve the following recurrences by changing variables:

1. $T(n) = 2T(\sqrt{n}) + \Theta(1)$.
2. $T(n) = 3T(\sqrt[3]{n}) + \Theta(n)$.

Solution.

Intuition (one paragraph). When the subproblem size is a *root* of n, taking logarithms turns the multiplicative shrinkage into an additive shrinkage. Setting $m = \lg n$ converts $n \mapsto \sqrt{n}$ into $m \mapsto m/2$ (and $n \mapsto \sqrt[3]{n}$ into $m \mapsto m/3$). After this change, the recurrences become standard divide-and-conquer forms in the m-variable, solvable by the Master Theorem. We then substitute back $m = \lg n$.

(a) Change of variables for (4.25). Let $m = \lg n$ and $S(m) = T(2^m)$. Then $\sqrt{n} = 2^{m/2}$ and $\lg n = m$. Hence

$$S(m) \;=\; 2\,S(m/2) \;+\; \Theta(m).$$

(b) Solve for $S(m)$. Apply the Master Theorem with $a = 2$, $b = 2$, and $f(m) = \Theta(m)$. Since $m^{\log_2 2} = m$, this is the balanced case (Case 2):

$$S(m) \;=\; \Theta\big(m \log m\big).$$

(c) Substitute back. With $m = \lg n$,

$$T(n) \;=\; S(\lg n) \;=\; \Theta\big((\lg n)\,\lg\lg n\big).$$

(d) Recursion-tree sketch and intuition. In the n-domain, each level replaces n by $\sqrt{n}$; there are

$$L = \min\{\,\ell :\; n^{1/2^\ell} \le O(1)\,\} = \Theta(\lg\lg n)$$

levels. The cost contributed by the level at depth i is $\Theta(\lg(n^{1/2^i})) \;=\; \Theta\big(\frac{1}{2^i}\lg n\big)$. Summing over levels gives

$$\sum_{i=0}^{L} \Theta\Big(\tfrac{1}{2^i}\lg n\Big) = \Theta\big(\lg n\big)\cdot \sum_{i=0}^{\Theta(\lg\lg n)} \tfrac{1}{2^i} = \Theta(\lg n)\cdot\Theta(1)$$

per branch, and there are 2^i branches at depth i. The total work at depth i is $\Theta(2^i \cdot \frac{1}{2^i}\lg n) = \Theta(\lg n)$, which is the same for each of the $\Theta(\lg\lg n)$ levels, giving $\Theta(\lg n \lg\lg n)$ overall.

(e) $T(n) = 2T(\sqrt{n}) + \Theta(1)$**.** Let $m = \lg n$, $S(m) = T(2^m)$. Then

$$S(m) = 2S(m/2) + \Theta(1).$$

Master Theorem with $a = 2$, $b = 2$, $f(m) = \Theta(1)$ (Case 1 with $\varepsilon = 1$) yields

$$S(m) = \Theta(m) \quad\Rightarrow\quad T(n) = S(\lg n) = \Theta(\lg n).$$

(f) $T(n) = 3T(\sqrt[3]{n}) + \Theta(n)$**.** Let $m = \lg n$, $S(m) = T(2^m)$. Then $\sqrt[3]{n} = 2^{m/3}$ and $n = 2^m$, so

$$S(m) = 3S(m/3) + \Theta(2^m).$$

Here $m^{\log_3 3} = m$, while $f(m) = \Theta(2^m)$ dominates superpolynomially. The Master Theorem (Case 3 with regularity $3\cdot 2^{m/3} \le c\,2^m$ for some $c < 1$ and large m) gives

$$S(m) = \Theta(2^m) \quad\Rightarrow\quad T(n) = S(\lg n) = \Theta(n).$$

Conclusion.

$$\boxed{T(n) = \Theta(\lg n\,\lg\lg n) \text{ for (4.25)};\quad T(n) = \Theta(\lg n) \text{ for (e)};\quad T(n) = \Theta(n) \text{ for (f)}.}$$

□

Problem 4-4. More recurrence examples.

Give asymptotically tight upper and lower bounds for $T(n)$ in each of the following recurrences. Justify your answers.

1. $T(n) = 5T(n/3) + n \lg n$.
2. $T(n) = 3T(n/3) + n/\lg n$.
3. $T(n) = 8T(n/2) + n^3\sqrt{n}$.
4. $T(n) = 2T(n/2 - 2) + n/2$.
5. $T(n) = 2T(n/2) + n/\lg n$.
6. $T(n) = T(n/2) + T(n/4) + T(n/8) + n$.
7. $T(n) = T(n-1) + 1/n$.
8. $T(n) = T(n-1) + \lg n$.
9. $T(n) = T(n-2) + 1/\lg n$.
10. $T(n) = \sqrt{n}\,T(\sqrt{n}) + n$.

Solution.

Intuition (one paragraph). Where a fixed fraction of n appears, we use the Master (or Akra–Bazzi) theorem. When the decrement is by a constant (e.g., $n-1$, $n-2$), we unroll and sum. Borderline terms like $n/\lg n$ yield an extra $\lg\lg n$ factor via the continuous master theorem. For (j), divide by n after the $m = \lg n$ change of variables to get a simple "halve-the-argument, add 1" recurrence.

1. $T(n) = 5T(n/3) + n\lg n$**.** *Rigorous.* Master theorem with $a = 5$, $b = 3$, so $n^{\log_3 5} \approx n^{1.464}$. Since $f(n) = n\lg n = O(n^{\log_3 5 - \varepsilon})$ for some $\varepsilon > 0$, Case 1 applies:
$$\boxed{T(n) = \Theta\big(n^{\log_3 5}\big)}.$$
2. $T(n) = 3T(n/3) + \dfrac{n}{\lg n}$**.** *Rigorous.* Here $p = \log_3 3 = 1$. By Akra–Bazzi/continuous master,
$$T(n) = \Theta\Big(n\Big(1 + \int_1^n \frac{(u/\lg u)}{u^2}\,du\Big)\Big) = \Theta\big(n \cdot \lg\lg n\big).$$
Thus $\boxed{T(n) = \Theta(n\lg\lg n)}$.

3. $T(n) = 8T(n/2) + n^3\sqrt{n}$. *Rigorous.* $n^{\log_2 8} = n^3$ and $f(n) = n^{3.5} = \Omega(n^{3+\varepsilon})$. Regularity: $8\,f(n/2) = 8\,(n/2)^{3.5} = (8/2^{3.5})n^{3.5} < c\,n^{3.5}$ for $c < 1$. Case 3:

$$\boxed{T(n) = \Theta(n^{7/2})}.$$

4. $T(n) = 2T(n/2 - 2) + n/2$. *Rigorous.* The -2 is a lower-order perturbation; Akra–Bazzi handles $g(n) = n/2 - 2$. The balance exponent solves $2 \cdot (1/2)^p = 1 \Rightarrow p = 1$. With $f(n) = \Theta(n)$, Case 2 gives

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

5. $T(n) = 2T(n/2) + \dfrac{n}{\lg n}$. *Rigorous.* $p = \log_2 2 = 1$. By the continuous master theorem (border case $k = -1$),

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

6. $T(n) = T(n/2) + T(n/4) + T(n/8) + n$. *Rigorous.* Akra–Bazzi with $a_i = 1$, $b_i \in \{2, 4, 8\}$. p solves $(1/2)^p + (1/4)^p + (1/8)^p = 1$. Let $x = 2^{-p}$; then $x + x^2 + x^3 = 1$, whose root is $x \approx 0.543689 \Rightarrow p = \log_2(1/x) \approx 0.879$. Using AB,

$$T(n) = \Theta\Big(n^p\Big(1 + \int_1^n \frac{u}{u^{p+1}}\,du\Big)\Big) = \Theta\big(n^p \cdot n^{1-p}\big) = \boxed{\Theta(n)}.$$

7. $T(n) = T(n-1) + 1/n$. *Rigorous.* Unroll: $T(n) = T(1) + \sum_{k=2}^{n} 1/k = \Theta(\lg n)$. So $\boxed{T(n) = \Theta(\lg n)}$.

8. $T(n) = T(n-1) + \lg n$. *Rigorous.* Unroll: $T(n) = T(1) + \sum_{k=2}^{n} \lg k = \Theta(\lg(n!)) = \Theta(n \lg n)$. Thus $\boxed{T(n) = \Theta(n \lg n)}$.

9. $T(n) = T(n-2) + \dfrac{1}{\lg n}$. *Rigorous.* Unroll for about $n/2$ terms:

$$T(n) = T(\Theta(1)) + \sum_{j=0}^{\lfloor n/2 \rfloor - 1} \frac{1}{\lg(n-2j)} = \Theta\left(\sum_{m \asymp n/2}^{n} \frac{1}{\lg m}\right) = \boxed{\Theta\Big(\frac{n}{\lg n}\Big)}.$$

10. $T(n) = \sqrt{n}\,T(\sqrt{n}) + n$. *Intuition.* Change variables to $m = \lg n$ and normalize by 2^m.

Rigorous. Let $S(m) = T(2^m)$. Then

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

Divide by 2^m and define $U(m) = S(m)/2^m$:

$$U(m) = U(m/2) + 1.$$

Depth is $\Theta(\lg m)$; hence $U(m) = \Theta(\lg m)$ and

$$T(n) = S(\lg n) = 2^{\lg n}\, \Theta(\lg \lg n) = \boxed{\Theta(n \lg \lg n)}.$$

Conclusion.

(a) $\Theta(n^{\log_3 5})$	(f) $\Theta(n)$
(b) $\Theta(n \lg \lg n)$	(g) $\Theta(\lg n)$
(c) $\Theta(n^{7/2})$	(h) $\Theta(n \lg n)$
(d) $\Theta(n \lg n)$	(i) $\Theta\big(n/\lg n\big)$
(e) $\Theta(n \lg \lg n)$	(j) $\Theta(n \lg \lg n)$

□

Problem 4-5. Fibonacci numbers.

This problem develops properties of the Fibonacci numbers, which are defined by recurrence (3.31) on page 69. We'll explore the technique of generating functions to solve the Fibonacci recurrence. Define the *generating function* (or *formal power series*) $\mathcal{F}$ as

$$\mathcal{F}(z) \;=\; \sum_{i=0}^{\infty} F_i z^i \;=\; 0 + z + z^2 + 2z^3 + 3z^4 + 5z^5 + 8z^6 + 13z^7 + 21z^8 + \cdots ,$$

where F_i is the ith Fibonacci number.

1. Show that $\mathcal{F}(z) = z + z\,\mathcal{F}(z) + z^2\,\mathcal{F}(z)$.

2. Show that
$$\mathcal{F}(z) = \frac{z}{1 - z - z^2} = \frac{1}{\sqrt{5}}\left(\frac{1}{1 - \phi z} - \frac{1}{1 - \hat{\phi} z}\right),$$
where ϕ is the golden ratio, and $\hat{\phi}$ is its conjugate.

3. Show that
$$\mathcal{F}(z) = \sum_{i=0}^{\infty} \frac{1}{\sqrt{5}}\left(\phi^i - \hat{\phi}^{\,i}\right) z^i.$$
You may use without proof the generating-function version of $\sum_{k=0}^{\infty} x^k = 1/(1-x)$.

4. Use part (c) to prove that $F_i = \phi^i/\sqrt{5}$ for $i > 0$, rounded to the nearest integer. (Hint: observe that $|\hat{\phi}| < 1$.)

5. Prove that $F_{i+2} \geq \phi^i$ for $i \geq 0$.

Solution.

Intuition. Pack the Fibonacci recurrence $F_i = F_{i-1} + F_{i-2}$ into one object $\mathcal{F}(z) = \sum F_i z^i$. Shifting indices multiplies by z (one step) or z^2 (two steps), so the recurrence becomes a simple algebraic identity in z. Solving for $\mathcal{F}(z)$ gives a rational function. Factoring its denominator by the roots of $x^2 = x + 1$ (namely ϕ and $\hat{\phi}$) yields a partial-fraction form whose geometric-series expansions give Binet's formula and the "nearest-integer" rule. The inequality in (e) follows from the fact that ϕ satisfies $\phi^2 = \phi + 1$ and Fibonacci numbers satisfy the same linear recurrence.

(a) $\mathcal{F}(z) = z + z\,\mathcal{F}(z) + z^2\,\mathcal{F}(z)$**.** Using $F_0 = 0$, $F_1 = 1$, and $F_i = F_{i-1} + F_{i-2}$ for $i \geq 2$,

$$\mathcal{F}(z) = \sum_{i\geq 0} F_i z^i = F_0 + F_1 z + \sum_{i\geq 2}(F_{i-1} + F_{i-2})z^i = z + \sum_{i\geq 2} F_{i-1}z^i + \sum_{i\geq 2} F_{i-2}z^i.$$

Recognize the shifts:

$$\sum_{i\geq 2} F_{i-1}z^i = z\sum_{j\geq 1} F_j z^j = z\mathcal{F}(z), \qquad \sum_{i\geq 2} F_{i-2}z^i = z^2\sum_{k\geq 0} F_k z^k = z^2\mathcal{F}(z),$$

hence $\mathcal{F}(z) = z + z\,\mathcal{F}(z) + z^2\,\mathcal{F}(z)$.

(b) Closed form and factorization. From (a),

$$\mathcal{F}(z)\big(1 - z - z^2\big) = z \quad \Rightarrow \quad \mathcal{F}(z) = \frac{z}{1 - z - z^2}.$$

Let $\phi = \frac{1+\sqrt{5}}{2}$ and $\hat{\phi} = \frac{1-\sqrt{5}}{2}$ be the roots of $x^2 - x - 1 = 0$. Then $1 - z - z^2 = (1 - \phi z)(1 - \hat{\phi} z)$, so partial fractions give

$$\frac{z}{1 - z - z^2} = \frac{A}{1 - \phi z} + \frac{B}{1 - \hat{\phi} z}.$$

Solve for A, B by clearing denominators:

$$z = A(1 - \hat{\phi} z) + B(1 - \phi z) = (A + B) - (\hat{\phi} A + \phi B)z.$$

Matching coefficients: $A + B = 0$ and $-(\hat{\phi} A + \phi B) = 1$. With $B = -A$, we get $-(\hat{\phi} - \phi)A = 1$, i.e. $A = \frac{1}{\phi - \hat{\phi}} = \frac{1}{\sqrt{5}}$ and $B = -\frac{1}{\sqrt{5}}$. Therefore

$$\mathcal{F}(z) = \frac{1}{\sqrt{5}}\left(\frac{1}{1 - \phi z} - \frac{1}{1 - \hat{\phi} z}\right).$$

(c) Series expansion. Using $\frac{1}{1-\alpha z} = \sum_{i\geq 0}(\alpha z)^i$ as a formal power series,

$$\mathcal{F}(z) = \frac{1}{\sqrt{5}}\sum_{i\geq 0}\big(\phi^i - \hat{\phi}^i\big)z^i.$$

(d) Nearest-integer rule. Comparing coefficients of z^i in (c) yields Binet's formula:

$$F_i = \frac{\phi^i - \hat{\phi}^i}{\sqrt{5}} \quad (i \geq 0).$$

Since $|\hat{\phi}| < 1$, we have $|\hat{\phi}^i| < 1$ and, in fact,

$$\frac{|\hat{\phi}^i|}{\sqrt{5}} \leq \frac{1}{\sqrt{5}} < \frac{1}{2} \qquad (i \geq 1).$$

Thus $F_i = \frac{\phi^i}{\sqrt{5}}$ plus an error of magnitude $< \frac{1}{2}$, so for $i > 0$,

$$\boxed{F_i = \phi^i/\sqrt{5} \text{ rounded to the nearest integer.}}$$

(e) Inequality $F_{i+2} \geq \phi^i$ for $i \geq 0$. We prove by induction using $F_{k+2} = F_{k+1} + F_k$ and $\phi^2 = \phi + 1$. Base $i = 0$: $F_2 = 1 \geq \phi^0 = 1$. Base $i = 1$: $F_3 = 2 \geq \phi \approx 1.618$. Inductive step: assume $F_{k+1} \geq \phi^{k-1}$ and $F_k \geq \phi^{k-2}$ (true for $k \geq 2$). Then

$$F_{k+2} = F_{k+1} + F_k \;\geq\; \phi^{k-1} + \phi^{k-2} = \phi^{k-2}(\phi + 1) = \phi^{k-2}\phi^2 = \phi^k.$$

Thus $F_{i+2} \geq \phi^i$ for all $i \geq 0$. □

Problem 4-6. Chip testing.

Professor Diogenes has n supposedly identical integrated-circuit chips that in principle are capable of testing each other. The professor's test jig accommodates two chips at a time. When the jig is loaded, each chip tests the other and reports whether it is good or bad. A good chip always reports accurately whether the other chip is good or bad, but the professor cannot trust the answer of a bad chip. Thus, the four possible outcomes of a test are as follows:

Chip A says	*Chip B says*	*Conclusion*
B is good	A is good	both are good, or both are bad
B is good	A is bad	at least one is bad
B is bad	A is good	at least one is bad
B is bad	A is bad	at least one is bad

(a) Show that if at least $n/2$ chips are bad, the professor cannot necessarily determine which chips are good using any strategy based on this kind of pairwise test. Assume that the bad chips can conspire to fool the professor.

Now you will design an algorithm to identify which chips are good and which are bad, assuming that more than $n/2$ of the chips are good. First, you will determine how to identify one good chip.

(b) Show that $\lfloor n/2 \rfloor$ pairwise tests are sufficient to reduce the problem to one of nearly half the size. That is, show how to use $\lfloor n/2 \rfloor$ pairwise tests to obtain a set with at most $\lceil n/2 \rceil$ chips that still has the property that more than half of the chips are good.

(c) Show how to apply the solution to part (b) recursively to identify one good chip. Give and solve the recurrence that describes the number of tests needed to identify one good chip.

You have now determined how to identify one good chip.

(d) Show how to identify all the good chips with an additional $\Theta(n)$ pairwise tests.

Solution.

Intuition. If bad chips are at least half and collude, they can "mirror" the behavior of the good half, so transcripts of any testing strategy cannot distinguish which half is truly good. When the good chips are a strict majority, we can pair chips arbitrarily and discard any pair that disagrees (since it contains a bad chip). If both in a pair say "good," we keep just one representative; this preserves the strict majority among survivors and halves the instance size with only $\lfloor n/2 \rfloor$ tests. Recursing yields one certified good chip. A single known-good chip then classifies everyone else in one pass.

(a) Impossibility at $\geq n/2$ bad. Assume n is even and partition the chips into two sets G and B of size $n/2$. Consider two worlds:

- *World 1:* G are good, B are bad. Good chips report truthfully; bad chips collude to mimic the behavior: when tested against members of B they both say "good," and against G they say "bad."
- *World 2:* B are good, G are bad. Now the (truly) good set B produces the same pattern of reports (truthful), while the bad set G colludes to mirror as above.

Under any sequence of pairwise tests, the multiset of outcomes (who accuses whom) is identical in the two worlds; hence no strategy can distinguish which side is the good set. Therefore, with at least $n/2$ bad chips, one cannot necessarily determine the good chips.

(b) Halving step with $\lfloor n/2 \rfloor$ tests. Pair the chips arbitrarily; if n is odd, leave one unpaired. For each pair (x, y) (one test):

if either says "bad" about the other, discard *both*; if both say "good," keep *one* of them.

Let S be the set of survivors plus the possible unpaired chip. Then:

- $|S| \leq \lceil n/2 \rceil$ (at most one survivor per pair, plus possibly one unpaired).
- If good chips are $> n/2$ initially, then good chips are a strict majority in S. Indeed, any pair containing a bad chip is either discarded or contributes at most one survivor; pairs of two goods contribute exactly one survivor (a good). Counting shows goods cannot lose their strict majority.

Exactly $\lfloor n/2 \rfloor$ tests are used.

(c) Recursing to find one good chip. Apply (b) recursively until one chip remains; call it g. Because the strict majority is preserved at every reduction, g must be good.

Let $T(n)$ be the number of tests to find *one* good chip when $> n/2$ are good. From (b),

$$T(n) \;=\; T\big(\lceil n/2 \rceil\big) + \big\lfloor n/2 \big\rfloor, \qquad T(1) = 0.$$

Solution: $T(n) = \Theta(n)$. Proof by induction with the guess $T(n) \leq cn$ for $c \geq 1$:

$$T(n) \leq c\lceil n/2 \rceil + \lfloor n/2 \rfloor \leq c\frac{n}{2} + \frac{n}{2} \leq cn$$

for $c \geq 1$. A matching lower bound $\Omega(n)$ follows because we do at least one test per pair across $\Theta(\log n)$ levels summing to $\sum \lfloor n/2^i \rfloor = \Theta(n)$.

(d) Classifying all chips with $\Theta(n)$ more tests. With a known good chip g, test g against every other chip once. Each test is truthful (since g is good), so one pass of $n-1$ tests labels all chips. Hence the additional work is $\Theta(n)$, and the entire algorithm uses $\Theta(n)$ tests overall. $\square$

Problem 4-7. Monge arrays.

An $m \times n$ array A of real numbers is a *Monge array* if for all i, j, k, ℓ such that $1 \le i < k \le m$ and $1 \le j < \ell \le n$, we have

$$A[i, j] + A[k, \ell] \;\le\; A[i, \ell] + A[k, j].$$

In other words, whenever we pick two rows and two columns of a Monge array and consider the four elements at the intersections of the rows and the columns, the sum of the upper-left and lower-right elements is less than or equal to the sum of the lower-left and upper-right elements. For example, the following array is Monge:

$$\begin{bmatrix} 10 & 17 & 13 & 28 & 23 \\ 17 & 22 & 16 & 29 & 23 \\ 24 & 28 & 22 & 34 & 24 \\ 11 & 13 & 6 & 17 & 7 \\ 45 & 44 & 32 & 37 & 23 \\ 75 & 66 & 51 & 53 & 34 \end{bmatrix}.$$

1. Prove that an array is Monge if and only if for all $i = 1, 2, \ldots, m-1$ and $j = 1, 2, \ldots, n-1$,

   $$A[i, j] + A[i+1, j+1] \;\le\; A[i, j+1] + A[i+1, j].$$

   (*Hint:* For the "if" part, use induction separately on rows and on columns.)

2. The following array is not Monge. Change one element in order to make it Monge. (*Hint:* Use part (a).)

   $$\begin{bmatrix} 37 & 23 & 22 & 32 \\ 21 & 6 & 7 & 10 \\ 53 & 34 & 30 & 31 \\ 32 & 13 & 9 & 6 \\ 43 & 21 & 15 & 8 \end{bmatrix}$$

3. Let $f(i)$ be the index of the column containing the leftmost minimum element of row i. Prove that $f(1) \le f(2) \le \cdots \le f(m)$ for any $m \times n$ Monge array.

4. Here is a description of a divide-and-conquer algorithm that computes the leftmost minimum element in each row of an $m \times n$ Monge array A:

Construct a submatrix A' of A consisting of the even-numbered rows of A. Recursively determine the leftmost minimum for each row of A'. Then compute the leftmost minimum in the odd-numbered rows of A.

Explain how to compute the leftmost minimum in the odd-numbered rows of A (given that the leftmost minimum of the even-numbered rows is known) in $O(m+n)$ time.

5. Write the recurrence for the running time of the algorithm in part (d). Show that its solution is $O(m + n \log m)$.

Solution.

Intuition. The Monge inequality is "local": checking it on all 2×2 adjacent blocks forces it to hold for all larger rectangles by telescoping along rows and columns. That locality gives a powerful *monotonicity* property—leftmost row minima never move left as you go down the rows. With that monotonicity, a divide-and-conquer can find minima in even rows first, then scan only short ranges to fill in odd rows, yielding $O(m + n \log m)$.

(a) Adjacent 2×2 blocks suffice. *($\Rightarrow$)* If A is Monge, the stated inequality holds for all choices, in particular for adjacent indices.

($\Leftarrow$) Suppose the 2×2 inequality holds for all adjacent blocks. Fix $i < k$ and $j < \ell$. By applying the adjacent inequality successively along columns we obtain

$$A[i,j] + A[i+1,\ell] \;\le\; A[i,\ell] + A[i+1,j],$$

then along rows (again telescoping) we derive

$$A[i,j] + A[k,\ell] \;\le\; A[i,\ell] + A[k,j].$$

Formally, induct on $k - i$ (number of row steps) holding j, ℓ fixed, and inside that induction induct on $\ell - j$ (number of column steps). Hence the global Monge property holds.

(b) One-entry fix. Check adjacent 2×2 inequalities. The only violating block is

$$\begin{bmatrix} 23 & 22 \\ 6 & 7 \end{bmatrix} \quad \text{since} \quad 23 + 7 > 22 + 6.$$

Changing a single entry to repair this and preserve all others works, e.g.

$$\boxed{\text{change } A[1,3] \text{ from 22 to 24}}$$

(1-indexed). With this change all adjacent inequalities hold, so the array becomes Monge.

(c) Monotonicity of leftmost minima. Let r and $r+1$ be consecutive rows, and suppose for contradiction that $f(r) > f(r+1)$. Put $j = f(r+1)$ and $\ell = f(r)$. Then

$$A[r+1, j] \le A[r+1, \ell] \quad \text{and} \quad A[r, \ell] \le A[r, j].$$

Adding yields

$$A[r+1, j] + A[r, \ell] \;\le\; A[r+1, \ell] + A[r, j],$$

which *violates* the Monge inequality (with $i = r$, $k = r+1$, $j = j$, $\ell = \ell$). Hence $f(r) \le f(r+1)$. By induction, $f(1) \le \cdots \le f(m)$.

(d) Filling odd rows in $O(m+n)$. After recursively computing $f(2), f(4), \ldots$ for the even rows, use (c): the indices are nondecreasing. For each odd row i between even rows $i-1$ and $i+1$, its leftmost minimum must lie in the interval $[\, f(i-1),\, f(i+1)\,]$. Scan that interval to find the minimum of row i. Process odd rows in increasing order; maintain a single cursor per row so total work over all odd rows is proportional to the total lengths of these disjoint/overlapping intervals. Summed over all rows the scanning costs $O(m+n)$, because each column index is advanced at most a constant number of times across the whole pass.

(e) Running time. Let $T(m, n)$ be the time to compute all leftmost row minima. We recurse on the $\lceil m/2 \rceil$ even-numbered rows (same n columns), then fill odd rows in $O(m+n)$:

$$T(m, n) \;=\; T(\lceil m/2 \rceil,\, n) \;+\; O(m+n).$$

Unfolding for $\lceil \log_2 m \rceil$ levels gives

$$T(m, n) = \sum_{t=0}^{\lceil \log m \rceil - 1} O\big(m/2^t + n\big) = O\big(m + n \log m\big).$$

□