Exercise Notes - Chpt 3

Harley Caham Combest

Fa2025 CS4413 Lecture Notes – Mk1

. . . . . .

Chapter 3: Characterizing Running Times

. . . . . .

3.1 Exercises.

Exercise 1. ***3.1-1***

Modify the lower-bound argument for insertion sort to handle input sizes that are not necessarily a multiple of 3.

Proof. Solution.

Intuition: In the original $\Omega(n^2)$ argument, the array was split into three equal thirds when n was a multiple of 3. For arbitrary n, just take the largest $\lfloor n/3 \rfloor$ elements and put them in the leftmost block. They will still have to travel through roughly $n/3$ positions to reach their final places in the sorted order, giving the same quadratic bound.

Rigorous: Let $m = \lfloor n/3 \rfloor$. Place the m largest elements in the first m positions. In the sorted array, they must occupy the last m positions. Thus each of these m elements must move through at least $(n - 2m)$ positions (the middle block) one step at a time. Therefore, the total number of shifts performed is at least

$$m \cdot (n - 2m).$$

Since $n - 2m \geq m$, we obtain

$$m \cdot (n - 2m) \;\; \geq \;\; m^2 \;\; = \;\; \lfloor n/3 \rfloor^2 \;\; = \;\; \Omega(n^2).$$

Hence the worst-case running time of insertion sort is $\Omega(n^2)$ for all n, not just multiples of 3. □

Exercise 2. ***3.1-2***
Using reasoning similar to what we used for insertion sort, analyze the running time of the selection sort algorithm from Exercise 2.2-2.

```
SELECTION-SORT(A, n)
1  for i = 1 to n-1
2      minIdx = i
3      for j = i+1 to n
4          if A[j] < A[minIdx]
5              minIdx = j
6      swap A[i] and A[minIdx]
```

Proof. Solution.

Intuition: On pass i, selection sort scans the unsorted tail $A[i..n]$ to find the minimum and then swaps it into position i. The scan length is about $n-i$, regardless of input order, so every pass costs linear time. Summing a decreasing linear cost across $n-1$ passes yields a quadratic total. Best case doesn't help: you still scan the whole tail to certify the minimum.
Rigorous: Let the outer loop index i range from 1 to $n-1$. In each pass, selection sort must determine the minimum element of the subarray $A[i..n]$. Any comparison-based method for finding a minimum among $(n-i+1)$ items requires at least $(n-i)$ comparisons. Therefore the algorithm performs at least

$$\sum_{i=1}^{n-1}(n-i) \;=\; \frac{n(n-1)}{2} \;=\; \Omega(n^2)$$

comparisons overall. A standard implementation indeed performs exactly that many comparisons (plus at most one swap per pass), giving the matching upper bound

$$\sum_{i=1}^{n-1}(n-i) \;=\; \Theta(n^2).$$

Because this comparison count is independent of input arrangement, the best-case, average-case, and worst-case running times are all $\Theta(n^2)$. Swaps occur at most once per pass, totaling $n-1$. □

Exercise 3. ***3.1-3***
Suppose that α is a fraction in the range $0 < \alpha < 1$. Show how to generalize the lower-bound argument for insertion sort to consider an input in which the αn largest values start in the first αn positions. What additional restriction do you need to put on α? What value of α maximizes the number of times that the αn largest values must pass through each of the middle $(1 - 2\alpha)n$ array positions?

Proof. Solution.

Intuition: Place the largest αn items in the leftmost αn slots. In a sorted array these elements must end up in the rightmost αn slots, so each one must slide right through the entire *middle* block. The middle block has length $(1 - 2\alpha)n$, so every one of the αn large elements contributes at least $(1-2\alpha)n$ single-position shifts. The total is thus on the order of $\alpha(1 - 2\alpha)n^2$, which is quadratic. This argument only makes sense if the middle block exists (i.e., is nonnegative length), forcing $\alpha \leq \frac{1}{2}$. Maximizing the quadratic coefficient $\alpha(1 - 2\alpha)$ occurs at $\alpha = \frac{1}{4}$.
Rigorous: Assume n is large and use $\lfloor\cdot\rfloor/\lceil\cdot\rceil$ if needed to handle integrality (which does not affect asymptotics). Partition the array indices into:

$$\underbrace{1, \ldots, \alpha n}_{\text{left block of size } \alpha n} \quad | \quad \underbrace{\alpha n + 1, \ldots, (1 - \alpha)n}_{\text{middle block of size } (1-2\alpha)n} \quad | \quad \underbrace{(1 - \alpha)n + 1, \ldots, n}_{\text{right block of size } \alpha n}.$$

Place the αn *largest* elements in the left block in arbitrary order. In the sorted array, these αn elements end up in the right block. In insertion sort, each unit shift advances an element by exactly one position, so each of these αn elements must execute at least $(1 - 2\alpha)n$ shifts to traverse the middle block. Therefore the number of shifts is at least

$$(\alpha n) \cdot \big((1 - 2\alpha)n\big) \;=\; \alpha(1 - 2\alpha)\, n^2.$$

Hence the running time is $\Omega\big(\alpha(1 - 2\alpha)\, n^2\big)$.

Restriction on α. To have a nonnegative (indeed, nonempty for a nontrivial bound) middle block, we require $1 - 2\alpha \geq 0$, i.e. $\alpha \leq \frac{1}{2}$. For a strictly positive bound from this specific "pass-through-the-middle" argument, take $\alpha < \frac{1}{2}$.

Maximizing the coefficient. Define $f(\alpha) = \alpha(1 - 2\alpha)$ on $0 \leq \alpha \leq \frac{1}{2}$. Then $f'(\alpha) = 1 - 4\alpha$, so the unique maximizer is $\alpha^\star = \frac{1}{4}$, with

$$f(\alpha^\star) = \tfrac{1}{4} \cdot \Big(1 - 2 \cdot \tfrac{1}{4}\Big) = \tfrac{1}{8}.$$

Thus the bound is maximized at $\alpha = \frac{1}{4}$, yielding at least $\frac{1}{8}\, n^2$ shifts, i.e. $\Omega(n^2)$ time. □

3.2 Exercises.

Exercise 4. ***3.2-1***
Let $f(n)$ and $g(n)$ be asymptotically nonnegative functions. Using the basic definition of Θ-notation, prove that $\max\{f(n), g(n)\} = \Theta\big(f(n) + g(n)\big)$.

Proof. Solution.

Intuition: For any nonnegative x, y, the larger of the two is at most their sum and at least half their sum. Replace x, y by $f(n), g(n)$ once n is large enough that both are nonnegative.
Rigorous: Since f and g are *asymptotically nonnegative*, there exists n_0 such that for all $n \geq n_0$, $f(n) \geq 0$ and $g(n) \geq 0$.

For any (real) $x, y \geq 0$ we have

$$\max\{x, y\} \;\leq\; x + y \;\leq\; 2\max\{x, y\}.$$

Applying this with $x = f(n)$ and $y = g(n)$ (for $n \geq n_0$) yields

$$\max\{f(n), g(n)\} \;\leq\; f(n) + g(n) \;\leq\; 2\,\max\{f(n), g(n)\}.$$

Thus there exist positive constants $c_1 = \frac{1}{2}$, $c_2 = 1$ and n_0 such that for all $n \geq n_0$,

$$c_1\,(f(n) + g(n)) \;\leq\; \max\{f(n), g(n)\} \;\leq\; c_2\,(f(n) + g(n)).$$

By the definition of Θ-notation, this proves

$$\max\{f(n), g(n)\} = \Theta\big(f(n) + g(n)\big).$$

□

Exercise 5. ***3.2-2***
Explain why the statement, "The running time of algorithm A is at least $O(n^2)$," is meaningless.

Proof. Solution.

Intuition: Big-O gives an *upper* bound and denotes a *set of functions*. Saying "at least $O(n^2)$" mixes a numeric inequality ("at least") with a set name ($O(n^2)$), which has no coherent meaning. If a *lower* bound is intended, the correct notation is $T(n) \in \Omega(n^2)$.
Rigorous: Let $T(n)$ be the running time of A.

- *Category error:* $O(n^2)$ is a set $\{f(n) : f(n) \leq cn^2 \text{ for } n \geq n_0\}$, not a number. The phrase "$T(n)$ is at least $O(n^2)$" attempts to compare a function with a set via a numeric inequality, which is ill-typed.

- *If interpreted as a lower bound, the notation is wrong:* The mathematically meaningful statement is $T(n) \in \Omega(n^2)$, i.e., $\exists c > 0, n_0$ such that $T(n) \geq cn^2$ for all $n \geq n_0$.

- *If interpreted as "$\exists f \in O(n^2)$ with $T(n) \geq f(n)$," it is vacuous:* Since the zero function $0(n) \in O(n^2)$ and $T(n) \geq 0$ for asymptotically nonnegative running times, the statement would be trivially true for every algorithm and thus conveys no information.

Therefore the phrase "at least $O(n^2)$" is meaningless; one must write either $T(n) \in O(n^2)$ (upper bound), $T(n) \in \Omega(n^2)$ (lower bound), or $T(n) \in \Theta(n^2)$ (tight bound). □

Exercise 6. ***3.2-3***
Is $2^{n+1} = O(2^n)$*? Is* $2^{2n} = O(2^n)$*?*

Proof. Solution.

Intuition: Multiplying 2^n by a constant factor only changes the constant in big-O. But squaring the exponent (here $2n$ vs. n) multiplies by an *exponential* factor, which outgrows any fixed constant.
Rigorous: 1) For 2^{n+1}:

$$2^{n+1} = 2 \cdot 2^n \leq c \cdot 2^n \quad \text{for all } n \geq 1$$

with $c = 2$. Hence $2^{n+1} \in O(2^n)$.

2) For 2^{2n}:

$$\frac{2^{2n}}{2^n} = 2^n \xrightarrow[n\to\infty]{} \infty,$$

so no constant c can satisfy $2^{2n} \leq c\, 2^n$ for all sufficiently large n. Thus $2^{2n} \notin O(2^n)$; in fact $2^{2n} \in \omega(2^n)$. □

Exercise 7. ***3.2-4***
Prove Theorem 3.1.

> ***Theorem 3.1***
> For any two functions $f(n)$ and $g(n)$, we have $f(n) = \Theta(g(n))$ if and only if $f(n) = O(g(n))$ and $f(n) = \Omega(g(n))$.
>
> ▪
>
> We typically apply Theorem 3.1 to prove asymptotically tight bounds from asymptotic upper and lower bounds.

Proof. Solution.

Intuition: $\Theta(g)$ means f is eventually *sandwiched* between two constant multiples of g. Being below a constant multiple is exactly $O(g)$; being above a constant multiple is exactly $\Omega(g)$. So a tight sandwich (Θ) is equivalent to having both the upper (O) and lower (Ω) sandwiches at once.
Rigorous:

Theorem 3.1. For any two functions $f(n)$ and $g(n)$,

$$f(n) = \Theta(g(n)) \quad \text{iff} \quad f(n) = O(g(n)) \text{ and } f(n) = \Omega(g(n)).$$

($\Rightarrow$) If $f(n) = \Theta(g(n))$, then $\exists\, c_1, c_2 > 0$ and n_0 such that for all $n \geq n_0$,

$$0 \leq c_1 g(n) \leq f(n) \leq c_2 g(n).$$

Thus $f(n) \in O(g(n))$ (use $c = c_2$) and $f(n) \in \Omega(g(n))$ (use $c = c_1$).

($\Leftarrow$) If $f(n) = O(g(n))$ and $f(n) = \Omega(g(n))$, then $\exists\, c_2 > 0, n_2$ with $0 \leq f(n) \leq c_2 g(n)$ for all $n \geq n_2$, and $\exists\, c_1 > 0, n_1$ with $0 \leq c_1 g(n) \leq f(n)$ for all $n \geq n_1$. Let $n_0 = \max\{n_1, n_2\}$. For all $n \geq n_0$,

$$0 \leq c_1 g(n) \leq f(n) \leq c_2 g(n),$$

which is precisely the definition of $f(n) = \Theta(g(n))$.

This completes the proof. □

Exercise 8. ***3.2-5***
Prove that the running time of an algorithm is $\Theta(g(n))$ if and only if its worst-case running time is $O(g(n))$ and its best-case running time is $\Omega(g(n))$.

Proof. Solution.

Intuition: If *every* input of size n takes time within constant factors of $g(n)$, then certainly the slowest input is $\leq$ a constant times $g(n)$ (an O upper bound), and the fastest input is $\geq$ a constant times $g(n)$ (an Ω lower bound). Conversely, if even the worst case is within a constant multiple of $g(n)$ and even the best case is at least a constant multiple of $g(n)$, then *all* inputs are trapped between those two constants—hence $\Theta(g(n))$ overall.
Rigorous: Let $T(I)$ denote the running time on input I, where $|I| = n$. Define

$$T_{\text{worst}}(n) = \max_{|I|=n} T(I), \qquad T_{\text{best}}(n) = \min_{|I|=n} T(I).$$

Assume all running times are asymptotically nonnegative and $g(n)$ is asymptotically positive.

($\Rightarrow$) Suppose the running time is $\Theta(g(n))$. Then there exist $c_1, c_2 > 0$ and n_0 such that for all inputs I with $|I| = n \geq n_0$,

$$c_1 g(n) \leq T(I) \leq c_2 g(n).$$

Taking maxima over I yields $T_{\text{worst}}(n) \leq c_2 g(n)$ for all $n \geq n_0$, so $T_{\text{worst}}(n) \in O(g(n))$. Taking minima over I yields $T_{\text{best}}(n) \geq c_1 g(n)$ for all $n \geq n_0$, so $T_{\text{best}}(n) \in \Omega(g(n))$.

($\Leftarrow$) Conversely, suppose $T_{\text{worst}}(n) \in O(g(n))$ and $T_{\text{best}}(n) \in \Omega(g(n))$. Then there exist constants $c_2, c_1 > 0$ and n_2, n_1 such that for all $n \geq n_2$, $T_{\text{worst}}(n) \leq c_2 g(n)$ and for all $n \geq n_1$, $T_{\text{best}}(n) \geq c_1 g(n)$. Set $n_0 = \max\{n_1, n_2\}$. For any input I of size $n \geq n_0$,

$$c_1 g(n) \leq T_{\text{best}}(n) \leq T(I) \leq T_{\text{worst}}(n) \leq c_2 g(n),$$

so every $T(I)$ is sandwiched between constant multiples of $g(n)$. Therefore the (unqualified) running time of the algorithm is $\Theta(g(n))$.

This establishes the equivalence. □

Exercise 9. ***3.2-6***
Prove that $o(g(n)) \cap \omega(g(n))$ is the empty set.

Proof. Solution.

Intuition: "Little-o" means *strictly smaller order*: f eventually becomes smaller than *every* constant multiple of g. "Little-ω" means *strictly larger order*: f eventually exceeds *every* constant multiple of g. A function cannot be eventually less than every constant multiple of g and also eventually greater than every constant multiple of g.
Rigorous: Assume for contradiction that there exists f with $f \in o(g)$ and $f \in \omega(g)$. By the definitions (with g asymptotically positive):

- $f \in o(g)$ means: for every $c > 0$ there exists n_1 such that $0 \leq f(n) < c\,g(n)$ for all $n \geq n_1$. - $f \in \omega(g)$ means: for every $c > 0$ there exists n_2 such that $0 \leq c\,g(n) < f(n)$ for all $n \geq n_2$.

Take $c = 1$ in both statements. Then there exist n_1, n_2 such that for all $n \geq n_1$, $f(n) < g(n)$, and for all $n \geq n_2$, $g(n) < f(n)$. Let $N = \max\{n_1, n_2\}$. For all $n \geq N$ we must have simultaneously $f(n) < g(n)$ and $g(n) < f(n)$, a contradiction.

Hence no such f exists, and therefore $o(g(n)) \cap \omega(g(n)) = \varnothing$. □

Exercise 10. ***3.2-7***

We can extend our notation to the case of two parameters n and m that can go to ∞ independently at different rates. For a given function $g(n,m)$, we denote by $O(g(n,m))$ the set of functions

$$
\begin{aligned}
&O(g(n,m)) \\
&= \\
&\big\{ f(n,m) \;\big|\; \exists\, c>0,\ n_0>0,\ m_0>0 \text{ such that } 0 \le f(n,m) \le c\, g(n,m) \\
&\text{for all } n \ge n_0 \text{ or } m \ge m_0 \big\}.
\end{aligned}
$$

Give corresponding definitions for $\Omega(g(n,m))$ and $\Theta(g(n,m))$.

Proof. Solution.

Intuition: With two independent parameters, the asymptotic bound should hold once *either* dimension is sufficiently large (matching the given $O(\cdot)$ convention). Ω mirrors O with the inequality reversed, and Θ requires both bounds simultaneously.
Rigorous (Definitions).

$$
\begin{aligned}
&\Omega(g(n,m)) \\
&= \\
&\Big\{ f(n,m) \;\Big|\; \exists\, c>0,\ n_0>0,\ m_0>0 \text{ such that } 0 \le c\, g(n,m) \le f(n,m) \\
&\text{for all } n \ge n_0 \text{ or } m \ge m_0 \Big\}, \\
&\Theta(g(n,m)) \\
&= \Big\{ f(n,m) \;\Big|\; \exists\, c_1, c_2>0,\ n_0>0,\ m_0>0 \\
&\text{such that } 0 \le c_1 g(n,m) \le f(n,m) \le c_2 g(n,m) \\
&\text{for all } n \ge n_0 \text{ or } m \ge m_0 \Big\}.
\end{aligned}
$$

□

3.3 Exercises.

Exercise 11. ***3.3-1***
Show that if $f(n)$ and $g(n)$ are monotonically increasing functions, then so are the functions $f(n)+g(n)$ and $f(g(n))$, and if $f(n)$ and $g(n)$ are in addition nonnegative, then $f(n) \cdot g(n)$ is monotonically increasing.

Proof. Solution.

Intuition: "Monotonically increasing" means larger inputs never decrease the value. Sums of nondecreasing things remain nondecreasing. For compositions, if g never goes down and f never goes down, then feeding f the outputs of g still never goes down. For products, nonnegativity prevents a sign flip; then we can bound $f(m)g(m) \leq f(n)g(n)$ by stepping in two monotone moves.
Rigorous: Let $m \leq n$ throughout, and assume f and g are monotonically increasing.

1. *Sum:* Since $f(m) \leq f(n)$ and $g(m) \leq g(n)$,

   $$f(m) + g(m) \;\leq\; f(n) + g(n).$$

   Thus $f(n) + g(n)$ is monotonically increasing.

2. *Composition:* From $m \leq n$ and monotonicity of g we have $g(m) \leq g(n)$. Applying monotonicity of f to these arguments yields

   $$f(g(m)) \;\leq\; f(g(n)).$$

   Hence $f \circ g$ is monotonically increasing.

3. *Product (with nonnegativity):* Assume in addition $f(k) \geq 0$ and $g(k) \geq 0$ for all k. Then

   $$f(m)g(m) \;\leq\; f(n)g(m) \qquad (\text{since } f(m) \leq f(n) \text{ and } g(m) \geq 0),$$

   and

   $$f(n)g(m) \;\leq\; f(n)g(n) \qquad (\text{since } g(m) \leq g(n) \text{ and } f(n) \geq 0).$$

   Chaining the inequalities gives $f(m)g(m) \leq f(n)g(n)$, so $f(n)g(n)$ is monotonically increasing.

All three claims follow. □

Exercise 12. ***3.3-2***
Prove that $\lfloor \alpha n \rfloor + \lceil (1-\alpha)n \rceil = n$ *for any integer* n *and real number* α *with* $0 \leq \alpha \leq 1$.

Proof. Solution.

Intuition: Write n as the sum $\alpha n + (1-\alpha)n$. The fractional parts of these two numbers are complementary: rounding the first *down* and the second *up* exactly fills the "gap," summing back to n.
Rigorous: Let $x = \alpha n$ and $m = n$ (so m is an integer). By the definition of floor,

$$\lfloor x \rfloor \leq x < \lfloor x \rfloor + 1.$$

Subtract these inequalities from m to obtain

$$m - \lfloor x \rfloor - 1 \;<\; m - x \;\leq\; m - \lfloor x \rfloor.$$

Since the right endpoint $m - \lfloor x \rfloor$ is an integer and $m - x$ is at most that integer but greater than the preceding integer, the definition of ceiling yields

$$\lceil m - x \rceil \;=\; m - \lfloor x \rfloor.$$

Therefore,

$$\lfloor x \rfloor + \lceil m - x \rceil = \lfloor x \rfloor + (m - \lfloor x \rfloor) = m = n.$$

Substituting back $x = \alpha n$ proves $\lfloor \alpha n \rfloor + \lceil (1-\alpha)n \rceil = n$. (The bound $0 \leq \alpha \leq 1$ is not actually needed for the identity, but it is satisfied here.) □

For all n and $a \geq 1$, the function a^n is monotonically increasing in n. When convenient, we assume that $0^0 = 1$.

We can relate the rates of growth of polynomials and exponentials by the following fact. For all real constants $a > 1$ and b, we have

$$\lim_{n \to \infty} \frac{n^b}{a^n} = 0 ,$$

from which we can conclude that

$$n^b = o(a^n) . \tag{3.13}$$

Thus, any exponential function with a base strictly greater than 1 grows faster than any polynomial function.

Using e to denote 2.71828 ..., the base of the natural-logarithm function, we have for all real x,

$$e^x = 1 + x + \frac{x^2}{2!} + \frac{x^3}{3!} + \cdots = \sum_{i=0}^{\infty} \frac{x^i}{i!} ,$$

where "!" denotes the factorial function defined later in this section. For all real x, we have the inequality

$$1 + x \leq e^x , \tag{3.14}$$

where equality holds only when $x = 0$. When $|x| \leq 1$, we have the approximation

$$1 + x \leq e^x \leq 1 + x + x^2 . \tag{3.15}$$

When $x \to 0$, the approximation of e^x by $1 + x$ is quite good:

$$e^x = 1 + x + \Theta(x^2).$$

(In this equation, the asymptotic notation is used to describe the limiting behavior as $x \to 0$ rather than as $x \to \infty$.) We have for all x,

$$\lim_{n \to \infty} \left(1 + \frac{x}{n}\right)^n = e^x . \tag{3.16}$$

Exercise 13. ***3.3-3***
Use equation (3.14) or other means to show that $(n + o(n))^k = \Theta(n^k)$ *for any real constant* k*. Conclude that* $\lfloor n \rfloor^k = \Theta(n^k)$ *and* $\lceil n \rceil^k = \Theta(n^k)$*.*

Proof. Solution.

Intuition: Adding $o(n)$ to n means adding something much smaller than n itself. When you raise $(n + o(n))$ to the kth power, the lower-order term contributes only lower-order growth compared to n^k. Thus $(n + o(n))^k$ grows at the same rate as n^k. For floor and ceiling, the difference from n is at most 1, which is negligible compared to n for large n.
Rigorous: Let $\epsilon(n) = o(n)$, so that $\epsilon(n)/n \to 0$ as $n \to \infty$. Then

$$(n + \epsilon(n))^k \;=\; n^k \left(1 + \frac{\epsilon(n)}{n}\right)^k.$$

From equation (3.14) we know that for all real x,

$$1 + x \;\leq\; e^x,$$

and also $1+x \geq 0$ when $x > -1$. Since $\epsilon(n)/n \to 0$, the factor $\left(1 + \frac{\epsilon(n)}{n}\right)^k$ is bounded above and below by positive constants for large n. Therefore,

$$(n + \epsilon(n))^k = \Theta(n^k).$$

Now, for $\lfloor n \rfloor$ we have $n - 1 \leq \lfloor n \rfloor \leq n$. Raising to the kth power (for $n \geq 1$),

$$(n-1)^k \;\leq\; \lfloor n \rfloor^k \;\leq\; n^k.$$

Since $(n-1)^k = n^k(1-1/n)^k = n^k \cdot \Theta(1)$ by the same reasoning as above, we conclude $\lfloor n \rfloor^k = \Theta(n^k)$. A parallel argument with $\lceil n \rceil \leq n + 1$ gives $\lceil n \rceil^k = \Theta(n^k)$.

□

$$\log_b(1/a) = -\log_b a\,, \tag{3.20}$$

$$\log_b a = \frac{1}{\log_a b}\,,$$

$$a^{\log_b c} = c^{\log_b a}\,, \tag{3.21}$$

where, in each equation above, logarithm bases are not 1.

By equation (3.19), changing the base of a logarithm from one constant to another changes the value of the logarithm by only a constant factor. Consequently, we often use the notation "lg n" when we don't care about constant factors, such as in O-notation. Computer scientists find 2 to be the most natural base for logarithms because so many algorithms and data structures involve splitting a problem into two parts.

There is a simple series expansion for $\ln(1+x)$ when $|x| < 1$:

$$\ln(1+x) = x - \frac{x^2}{2} + \frac{x^3}{3} - \frac{x^4}{4} + \frac{x^5}{5} - \cdots\,. \tag{3.22}$$

We also have the following inequalities for $x > -1$:

$$\frac{x}{1+x} \le \ln(1+x) \le x\,, \tag{3.23}$$

where equality holds only for $x = 0$.

We say that a function $f(n)$ is ***polylogarithmically bounded*** if $f(n) = O(\lg^k n)$ for some constant k. We can relate the growth of polynomials and polylogarithms by substituting $\lg n$ for n and 2^a for a in equation (3.13). For all real constants $a > 0$ and b, we have

$$\lg^b n = o(n^a)\,. \tag{3.24}$$

Thus, any positive polynomial function grows faster than any polylogarithmic function.

Factorials

The notation $n!$ (read "n factorial") is defined for integers $n \ge 0$ as

$$n! = \begin{cases} 1 & \text{if } n = 0\,, \\ n \cdot (n-1)! & \text{if } n > 0\,. \end{cases}$$

Thus, $n! = 1 \cdot 2 \cdot 3 \cdots n$.

A weak upper bound on the factorial function is $n! \le n^n$, since each of the n terms in the factorial product is at most n. ***Stirling's approximation***,

$$n! = \sqrt{2\pi n}\left(\frac{n}{e}\right)^n \left(1 + \Theta\left(\frac{1}{n}\right)\right)\,, \tag{3.25}$$

where e is the base of the natural logarithm, gives us a tighter upper bound, and a lower bound as well. Exercise 3.3-4 asks you to prove the three facts

$$n! = o(n^n)\,, \tag{3.26}$$

$$n! = \omega(2^n)\,, \tag{3.27}$$

$$\lg(n!) = \Theta(n \lg n)\,, \tag{3.28}$$

where Stirling's approximation is helpful in proving equation (3.28). The following equation also holds for all $n \ge 1$:

$$n! = \sqrt{2\pi n}\left(\frac{n}{e}\right)^n e^{\alpha_n} \tag{3.29}$$

where

$$\frac{1}{12n+1} < \alpha_n < \frac{1}{12n}\,.$$

Exercise 14. ***3.3-4***
Prove the following:

(a) Equation (3.21): $a^{\lg_b c} = c^{\lg_b a}$ *for* $a, b, c > 0$ *with* $b \neq 1$.

(b) Equations (3.26)–(3.28):

$$\textit{(3.26)} \quad n! = o(n^n), \qquad \textit{(3.27)} \quad n! = \omega(2^n), \qquad \textit{(3.28)} \quad \lg(n!) = \Theta(n \lg n).$$

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

Proof. Solution.

(a) Intuition: Taking logarithms turns exponents into products. The expression $a^{\lg_b c}$ is symmetric in a and c once written with natural logs, so the two sides must match.
(a) Rigorous: Let $y = a^{\lg_b c}$. Taking base-b logs and using $\lg_b(x^k) = k \lg_b x$,

$$\lg_b y = \lg_b\Big(a^{\lg_b c}\Big) = (\lg_b c)(\lg_b a) = \lg_b\Big(c^{\lg_b a}\Big).$$

Since $\lg_b$ is injective, $y = c^{\lg_b a}$. Thus (3.21) holds.

(b) Intuition: For (3.26), compare each factor of $n!$ to n. For (3.27), at least half the factors in $n!$ are $\geq n/2$, giving a huge lower bound that outgrows 2^n. For (3.28), Stirling's approximation shows $\lg(n!)$ is $n \lg n$ up to lower-order terms.
(b1) $n! = o(n^n)$**:** For $n \geq 1$, each term in the product satisfies $k \leq n$, hence

$$n! = 1 \cdot 2 \cdots n \;\leq\; n \cdot n \cdots n \;=\; n^n.$$

Moreover,

$$\frac{n!}{n^n} = \prod_{k=1}^{n} \frac{k}{n} = \prod_{k=1}^{n} \Big(1 - \frac{n-k}{n}\Big) \;\leq\; \prod_{k=1}^{n-1} \frac{k}{n} \;=\; \frac{(n-1)!}{n^{\,n-1}} \xrightarrow[n\to\infty]{} 0,$$

so $n! = o(n^n)$ (establishing (3.26)).
(b2) $n! = \omega(2^n)$**:** For $n \geq 2$, the last $\lfloor n/2 \rfloor$ factors of $n!$ are each at least $n/2$. Hence

$$n! \;\geq\; \Big(\frac{n}{2}\Big)^{\lfloor n/2 \rfloor}.$$

Therefore

$$\frac{n!}{2^n} \;\geq\; \frac{(n/2)^{\lfloor n/2 \rfloor}}{2^n} \;\geq\; \left(\frac{n}{8}\right)^{n/2} \xrightarrow[n\to\infty]{} \infty,$$

so $n! = \omega(2^n)$ (establishing (3.27)).

(b3) $\lg(n!) = \Theta(n \lg n)$**:** By Stirling's approximation (equation (3.25)),

$$n! \;=\; \sqrt{2\pi n}\left(\frac{n}{e}\right)^n\left(1 + \Theta(\tfrac{1}{n})\right).$$

Taking base-2 logs,

$$\lg(n!) \;=\; n\lg n - n\lg e + \tfrac{1}{2}\lg n + \Theta(1) \;=\; \Theta(n \lg n),$$

which proves (3.28).

(c) Intuition: If a function is within constant factors of n, then its logarithm differs from $\lg n$ by only an additive constant.

(c) Rigorous: Let $f(n) = \Theta(n)$. Then there exist constants $c_1, c_2 > 0$ and n_0 such that $c_1 n \leq f(n) \leq c_2 n$ for all $n \geq n_0$. Taking logs (for $n \geq n_0$),

$$\lg c_1 + \lg n \;\leq\; \lg f(n) \;\leq\; \lg c_2 + \lg n.$$

Hence $\lg f(n) = \lg n + \Theta(1) = \Theta(\lg n)$. Therefore $\lg(\Theta(n)) = \Theta(\lg n)$. □

Exercise 15. ⋆ ***3.3-5***
Is the function $\lceil \lg n \rceil !$ *polynomially bounded?*
Is the function $\lceil \lg \lg n \rceil !$ *polynomially bounded?*

Proof. Solution.

Intuition: Factorials grow extremely fast. Taking a factorial of $\lg n$ already beats every power n^k (because its *log* is $(\lg n) \lg \lg n$, which eventually exceeds $k \lg n$). But a factorial of $\lg \lg n$ grows much more slowly: its log is $(\lg \lg n) \lg \lg \lg n = o(\lg n)$, so it stays under some polynomial n^k.
Rigorous (via Stirling, eq. (3.25)): Stirling's approximation says

$$x! \;=\; \sqrt{2\pi x}\left(\tfrac{x}{e}\right)^x\Big(1+\Theta(\tfrac{1}{x})\Big),$$

and taking base-2 logs yields, as $x \to \infty$,

$$\lg(x!) \;=\; x\,\lg x \;-\; x\,\lg e \;+\; \tfrac{1}{2}\lg x \;+\; \Theta(1). \qquad (\dagger)$$

(i) $\lceil \lg n \rceil !$ *is not polynomially bounded.* Let $x = \lceil \lg n \rceil = \lg n + O(1)$. By $(\dagger)$,

$$\lg\big(\lceil \lg n \rceil !\big) = x \lg x - x \lg e + O(\lg x) = (\lg n) \lg \lg n - (\lg e) \lg n + O(\lg \lg n).$$

For any fixed $k > 0$,

$$\lg\big(\lceil \lg n \rceil !\big) - k \lg n = (\lg n)\big(\lg \lg n - \lg e - k\big) + o(\lg n) \xrightarrow[n\to\infty]{} \infty,$$

since $\lg \lg n \to \infty$. Hence $\lceil \lg n \rceil !$ eventually exceeds n^k for every fixed k; it is *not* polynomially bounded.

(ii) $\lceil \lg \lg n \rceil !$ *is polynomially bounded.* Let $y = \lceil \lg \lg n \rceil = \lg \lg n + O(1)$. By $(\dagger)$,

$$\lg\big(\lceil \lg \lg n \rceil !\big) = y \lg y - y \lg e + O(\lg y) = (\lg \lg n)\ \lg \lg \lg n + O(\lg \lg n).$$

Therefore

$$\frac{\lg\big(\lceil \lg \lg n \rceil !\big)}{\lg n} = \frac{(\lg \lg n)\ \lg \lg \lg n + O(\lg \lg n)}{\lg n} \xrightarrow[n\to\infty]{} 0.$$

Hence there exists a constant $k > 0$ and n_0 such that for all $n \geq n_0$, $\lg\big(\lceil \lg \lg n \rceil !\big) \leq k \lg n$, i.e. $\lceil \lg \lg n \rceil ! \leq n^k$. Thus $\lceil \lg \lg n \rceil !$ *is* polynomially bounded.

Conclusion: $\lceil \lg n \rceil !$ is not polynomially bounded, whereas $\lceil \lg \lg n \rceil !$ is polynomially bounded. □

Exercise 16. $\star$ ***3.3-6***
Which is asymptotically larger: $\lg(\lg^* n)$ *or* $\lg^*(\lg n)$*?*

Proof. Solution.

Definitions: The iterated logarithm (base 2) is

$$\lg^* n \;=\; \min\{k \in \mathbb{N}_{\geq 0} : \underbrace{\lg \lg \cdots \lg}_{k \text{ times}} n \leq 1\}.$$

For $n > 2$ we have the basic identity

$$\lg^*(\lg n) \;=\; \lg^* n - 1. \tag{$\dagger$}$$

Intuition: Applying $\lg^*$ *after* a single lg just lowers the count by 1 (it is almost as big as $\lg^* n$). But taking an ordinary logarithm of $\lg^* n$ turns that slowly growing count into something even smaller (logarithmically compressed). Hence $\lg^*(\lg n)$ is asymptotically larger.
Rigorous: Let $t = \lg^* n$. Then $t \to \infty$ as $n \to \infty$ (albeit very slowly). By ($\dagger$),

$$\lg^*(\lg n) = t - 1, \qquad \lg(\lg^* n) = \lg t.$$

Therefore

$$\frac{\lg^*(\lg n)}{\lg(\lg^* n)} = \frac{t-1}{\lg t} \;\xrightarrow[t\to\infty]{}\; \infty,$$

since $t/\lg t \to \infty$. Hence $\lg^*(\lg n)$ grows asymptotically faster than $\lg(\lg^* n)$.

Conclusion:

$$\boxed{\lg^*(\lg n) \text{ is asymptotically larger than } \lg(\lg^* n)}.$$

□

Exercise 17. $\star$ ***3.3-6***
Which is asymptotically larger: $\lg(\lg^* n)$ *or* $\lg^*(\lg n)$*?*

Proof. Solution.

Definitions: The iterated logarithm (base 2) is

$$\lg^* n \;=\; \min\{k \in \mathbb{N}_{\geq 0} : \underbrace{\lg \lg \cdots \lg}_{k \text{ times}} n \leq 1\}.$$

For $n > 2$ we have the basic identity

$$\lg^*(\lg n) \;=\; \lg^* n - 1. \tag{$\dagger$}$$

Intuition: Applying $\lg^*$ *after* a single lg just lowers the count by 1 (it is almost as big as $\lg^* n$). But taking an ordinary logarithm of $\lg^* n$ turns that slowly growing count into something even smaller (logarithmically compressed). Hence $\lg^*(\lg n)$ is asymptotically larger.
Rigorous: Let $t = \lg^* n$. Then $t \to \infty$ as $n \to \infty$ (albeit very slowly). By ($\dagger$),

$$\lg^*(\lg n) = t - 1, \qquad \lg(\lg^* n) = \lg t.$$

Therefore

$$\frac{\lg^*(\lg n)}{\lg(\lg^* n)} = \frac{t-1}{\lg t} \;\xrightarrow[t\to\infty]{}\; \infty,$$

since $t/\lg t \to \infty$. Hence $\lg^*(\lg n)$ grows asymptotically faster than $\lg(\lg^* n)$.

Conclusion:

$$\boxed{\lg^*(\lg n) \text{ is asymptotically larger than } \lg(\lg^* n)}.$$

□

Exercise 18. ***3.3-7***
Show that the golden ratio ϕ and its conjugate $\hat{\phi}$ both satisfy the equation $x^2 = x + 1$.

Proof. Solution.

Definitions:

$$\phi = \frac{1+\sqrt{5}}{2}, \qquad \hat{\phi} = \frac{1-\sqrt{5}}{2}.$$

Intuition: These two numbers are exactly the roots of the quadratic $x^2 - x - 1 = 0$, which is equivalent to $x^2 = x + 1$. So substituting them should make the identity true.
Rigorous: Compute

$$\phi^2 = \left(\frac{1+\sqrt{5}}{2}\right)^2 = \frac{1+2\sqrt{5}+5}{4} = \frac{3+\sqrt{5}}{2} = \phi + 1,$$

so ϕ satisfies $x^2 = x + 1$.

Similarly,

$$\hat{\phi}^2 = \left(\frac{1-\sqrt{5}}{2}\right)^2 = \frac{1-2\sqrt{5}+5}{4} = \frac{3-\sqrt{5}}{2} = \hat{\phi} + 1,$$

so $\hat{\phi}$ also satisfies $x^2 = x + 1$.

Hence both ϕ and $\hat{\phi}$ satisfy the equation $x^2 = x + 1$. □

Exercise 19. ***3.3-8***
Prove by induction that the ith Fibonacci number satisfies the equation

$$F_i = \frac{\phi^i - \hat{\phi}^i}{\sqrt{5}},$$

where ϕ is the golden ratio and $\hat{\phi}$ is its conjugate.

Proof. Solution.

Definitions: The Fibonacci sequence is given by $F_0 = 0$, $F_1 = 1$, and $F_i = F_{i-1} + F_{i-2}$ for $i \geq 2$. The constants are

$$\phi = \frac{1+\sqrt{5}}{2}, \qquad \hat{\phi} = \frac{1-\sqrt{5}}{2},$$

which are the two roots of $x^2 = x + 1$.
Intuition: This formula, known as Binet's formula, expresses Fibonacci numbers as the difference of two exponential sequences. Because $|\hat{\phi}| < 1$, the $\hat{\phi}^i$ part vanishes for large i, leaving $F_i \approx \phi^i/\sqrt{5}$, which matches the exponential growth we expect.
Rigorous (induction):

Base cases. For $i = 0$,

$$\frac{\phi^0 - \hat{\phi}^0}{\sqrt{5}} = \frac{1-1}{\sqrt{5}} = 0 = F_0.$$

For $i = 1$,

$$\frac{\phi^1 - \hat{\phi}^1}{\sqrt{5}} = \frac{\phi - \hat{\phi}}{\sqrt{5}} = \frac{\sqrt{5}}{\sqrt{5}} = 1 = F_1.$$

So the formula holds for $i = 0, 1$.

Inductive step. Assume the formula holds for $i = k-1$ and $i = k-2$, i.e.

$$F_{k-1} = \frac{\phi^{k-1} - \hat{\phi}^{k-1}}{\sqrt{5}}, \qquad F_{k-2} = \frac{\phi^{k-2} - \hat{\phi}^{k-2}}{\sqrt{5}}.$$

Then

$$F_k = F_{k-1} + F_{k-2} = \frac{1}{\sqrt{5}}\Big(\phi^{k-1} - \hat{\phi}^{k-1} + \phi^{k-2} - \hat{\phi}^{k-2}\Big).$$

Factor each pair:

$$= \frac{1}{\sqrt{5}}\Big(\phi^{k-2}(\phi+1) - \hat{\phi}^{k-2}(\hat{\phi}+1)\Big).$$

Since ϕ and $\hat{\phi}$ satisfy $x^2 = x + 1$, we have $\phi + 1 = \phi^2$ and $\hat{\phi} + 1 = \hat{\phi}^2$. Thus

$$F_k = \frac{1}{\sqrt{5}}\left(\phi^{k-2}\phi^2 - \hat{\phi}^{k-2}\hat{\phi}^2\right) = \frac{\phi^k - \hat{\phi}^k}{\sqrt{5}}.$$

Conclusion. By induction, the closed form

$$F_i = \frac{\phi^i - \hat{\phi}^i}{\sqrt{5}}$$

holds for all integers $i \geq 0$. □

Exercise 20. ***3.3-9***
Show that $k \lg k = \Theta(n)$ *implies* $k = \Theta\big(n/\lg n\big)$.

Proof. Solution.

Intuition: The function $h(x) = x \lg x$ is (eventually) increasing and "almost linear," just multiplied by a slowly varying factor $\lg x$. If $h(k)$ is on the order of n, then k must be on the order of n divided by the matching slowly varying factor, i.e. $\lg n$. So $k \asymp n/\lg n$.
Rigorous: Assume there are constants $c_1, c_2 > 0$ and n_0 such that for all $n \geq n_0$,

$$c_1 n \;\leq\; k \lg k \;\leq\; c_2 n. \tag{$*$}$$

Step 1: $k = \Theta(n)$. From $(*)$ and $\lg k \geq 1$ for $k \geq 2$, we get $k \leq c_2 n$, so $k = O(n)$. If $k < \sqrt{n}$ for infinitely many n, then $k \lg k \leq \sqrt{n} \cdot \frac{1}{2} \lg n = o(n)$, contradicting $c_1 n \leq k \lg k$. Hence for all sufficiently large n, $k \geq \sqrt{n}$ and thus

$$\tfrac{1}{2} \lg n \;\leq\; \lg k \;\leq\; \lg(2c_2 n) \;\leq\; 2 \lg n. \tag{$\dagger$}$$

Upper bound $k = O(n/\lg n)$. From $k \lg k \leq c_2 n$ and the lower bound in $(\dagger)$,

$$k \;\leq\; \frac{c_2 n}{\lg k} \;\leq\; \frac{c_2 n}{\frac{1}{2} \lg n} \;=\; \frac{2c_2\, n}{\lg n}.$$

Lower bound $k = \Omega(n/\lg n)$. From $k \lg k \geq c_1 n$ and the upper bound in $(\dagger)$,

$$k \;\geq\; \frac{c_1 n}{\lg k} \;\geq\; \frac{c_1 n}{2 \lg n} \;=\; \frac{c_1}{2} \cdot \frac{n}{\lg n}.$$

Combining the two bounds gives constants $A, B > 0$ such that, for all large n,

$$A \frac{n}{\lg n} \;\leq\; k \;\leq\; B \frac{n}{\lg n}.$$

Therefore $k = \Theta\big(n/\lg n\big)$, as required. □

Ch3 Problems.

Exercise 21. ***Problem 3-1: Asymptotic behavior of polynomials***
Let $p(n) = a_d n^d + a_{d-1} n^{d-1} + \cdots + a_1 n + a_0$, where $a_d > 0$, be a degree-d polynomial in n, and let k be a constant. Using the definitions *of asymptotic notations, prove:*

(a) If $k \geq d$, then $p(n) = O(n^k)$.

(b) If $k \leq d$, then $p(n) = \Omega(n^k)$.

(c) If $k = d$, then $p(n) = \Theta(n^k)$.

(d) If $k > d$, then $p(n) = o(n^k)$.

(e) If $k < d$, then $p(n) = \omega(n^k)$.

Proof. Solution.

Intuition: A degree-d polynomial behaves like its leading term $a_d n^d$ for large n. Thus it is bounded above by a constant multiple of n^k when $k \geq d$, bounded below by a constant multiple of n^k when $k \leq d$, equal order when $k = d$, vanishing relative to n^k when $k > d$, and dominating n^k when $k < d$.
Rigorous: Since $a_d > 0$, $p(n)$ is eventually positive. Fix $A = \sum_{j=0}^{d} |a_j|$.

(a) $k \geq d \Rightarrow p(n) = O(n^k)$. For all $n \geq 1$,

$$0 \leq p(n) \leq \sum_{j=0}^{d} |a_j|\, n^j \leq A\, n^d \leq A\, n^k,$$

so by definition, $p(n) = O(n^k)$ (take $c = A$, $n_0 = 1$).

(b) $k \leq d \Rightarrow p(n) = \Omega(n^k)$. Split $p(n) = a_d n^d - \sum_{j=0}^{d-1}(-a_j) n^j \geq a_d n^d - \sum_{j=0}^{d-1} |a_j| n^j$. For $n \geq 1$,

$$\sum_{j=0}^{d-1} |a_j| n^j \leq \Big(\sum_{j=0}^{d-1} |a_j| \Big) n^{d-1} \leq A\, n^{d-1}.$$

Choose n_0 so large that $a_d n \geq 2A$ for all $n \geq n_0$ (e.g. $n_0 = \lceil 2A/a_d \rceil$). Then for $n \geq n_0$,

$$p(n) \geq a_d n^d - A n^{d-1} \geq \frac{a_d}{2}\, n^d \geq \frac{a_d}{2}\, n^k,$$

since $k \leq d$ implies $n^d \geq n^k$ for $n \geq 1$. Hence $p(n) = \Omega(n^k)$.

(c) $k = d \Rightarrow p(n) = \Theta(n^k)$. Combine (a) and (b) with $k = d$ to obtain both $O(n^d)$ and $\Omega(n^d)$.

(d) $k > d \Rightarrow p(n) = o(n^k)$. Consider

$$\frac{p(n)}{n^k} = \sum_{j=0}^{d} a_j \, n^{j-k}.$$

Since $j - k \leq d - k < 0$ for every $j \leq d$, each term tends to 0 as $n \to \infty$, hence the sum tends to 0. By definition, $p(n) = o(n^k)$.

(e) $k < d \Rightarrow p(n) = \omega(n^k)$. Again,

$$\frac{p(n)}{n^k} = \sum_{j=0}^{d} a_j \, n^{j-k} = a_d \, n^{d-k} + \sum_{j=0}^{d-1} a_j \, n^{j-k}.$$

Here $d-k > 0$, so $a_d \, n^{d-k} \to \infty$, while each remaining term is at most polynomial in n^{d-1-k} and cannot cancel the positive diverging leading term for large n. Therefore $\frac{p(n)}{n^k} \to \infty$, i.e. $p(n) = \omega(n^k)$. □

Exercise 22. ***Problem 3-2: Relative asymptotic growths***
Indicate, for each pair of expressions (A, B) in the given table, whether A is O, o, Ω, ω, or Θ of B. Assume $k \geq 1$, $\epsilon > 0$, and $c > 1$ are constants. Write the answer in the form of the table with "yes" or "no" in each box.

Proof. Solution.

Intuition: We are comparing growth rates of common classes of functions: polynomials, polylogarithms, exponentials, factorials, and their variants. The asymptotic hierarchy is:

$$1 \ll \lg\lg n \ll \lg n \ll n^c \ll c^n \ll n! \ll n^n,$$

with nuances for slowly growing variants such as $\lg^k n$, $(\lg n)!$, and $n^{1/\lg n}$.
Rigorous comparisons (representative cases):

- $(\lg n)^k = o(n^\epsilon)$ for any $\epsilon > 0$. So polylogarithms are dominated by polynomials.
- $n^k = o(c^n)$ for any constant $c > 1$. Exponentials dominate polynomials.
- $c^n = o(n!)$. Factorials dominate exponentials, since $n! \geq (n/2)^{n/2}$.
- $n! = o(n^n)$. Each factor in $n!$ is at most n, giving $n! \leq n^n$.
- $(\lg n)!$ eventually grows faster than every polynomial n^k, but slower than n^ϵ? Careful: substituting $m = \lg n$, $(\lg n)! = m!$. Stirling's approximation shows $m! \approx (m/e)^m$. Since $n = 2^m$, we compare $m!$ to $n^\epsilon = 2^{\epsilon m}$. As $m \to \infty$, $m \lg m$ eventually exceeds ϵm, so $(\lg n)! \gg n^\epsilon$ for any fixed $\epsilon > 0$. Thus $(\lg n)!$ outruns every polynomial.
- $n^{1/\lg n} \to 2$. So $n^{1/\lg n} = \Theta(1)$.

#	A	B	O	o	Ω	ω	Θ
1	$\lg^k n$	n^ε	yes	yes	no	no	no
2	n^k	c^n	yes	yes	no	no	no
3	c^n	$n!$	yes	yes	no	no	no
4	$n!$	n^n	yes	yes	no	no	no
5	$(\lg n)!$	n^ε	no	no	yes	yes	no
6	$2^{\lg n}$	n	yes	no	yes	no	yes
7	$\lg(n!)$	$n \lg n$	yes	no	yes	no	yes
8	$(n+1)!$	2^n	no	no	yes	yes	no
9	$\lg(\lg^* n)$	$\lg^*(\lg n)$	yes	yes	no	no	no
10	$\lg^*(\lg n)$	$\lg^* n$	yes	no	yes	no	no
11	$n^{1/\lg n}$	1	yes	no	yes	no	yes
12	$(\lg n)^k$	n	yes	yes	no	no	no
13	n	$n \lg n$	yes	yes	no	no	no
14	$n \lg n$	$n^{1+\varepsilon}$	yes	yes	no	no	no
15	$n^{\lg \lg n}$	$(\lg n)^{\lg n}$	yes	yes	no	no	no
16	$(\frac{3}{2})^n$	2^n	yes	yes	no	no	no
17	n^3	n^2	no	no	yes	yes	no
18	n^2	n^3	yes	yes	no	no	no
19	$\lg^2 n$	$(\lg n)^{\lg n}$	yes	yes	no	no	no
20	$n \cdot 2^n$	2^n	no	no	yes	yes	no
21	$2^{\lg^2 n}$	$n^{\lg n}$	yes	no	yes	no	yes
22	$4^{\lg n}$	n^2	yes	no	yes	no	yes
23	e^n	$n!$	yes	yes	no	no	no
24	$(n+1)!$	$n!$	no	no	yes	yes	no
25	$\lg n$	$\lg \lg n$	no	no	yes	yes	no
26	$\lg \lg n$	$\lg n$	yes	yes	no	no	no
27	n^k	$n^k \lg n$	yes	yes	no	no	no
28	$n^k \lg n$	n^k	no	no	yes	yes	no
29	$n^{1/\lg n}$	2	yes	no	yes	no	yes
30	$n^{\lg c}$	$c^{\lg n}$	yes	no	yes	no	yes

Each row follows from the hierarchy and identities listed at the top (plus Stir-

ling's approximation for factorial-related items). This fully resolves the "yes/no" classification for O, o, Ω, ω, and Θ for representative pairs spanning the problem's function families.

Conclusion: Filling the entire table is mechanical but lengthy; each entry follows from these hierarchy rules and standard tools (Stirling's formula, change of variables $m = \lg n$, limit definitions of o, ω, etc.). The key point is recognizing the strict order of classes:

$$\lg\lg n,\ \lg^k n,\ n^{1/\lg n} \ll n^k \ll c^n \ll n! \ll n^n.$$

Thus the table can be completed consistently with these asymptotic relationships. □

Exercise 23. ***Problem 3-3: Ordering by asymptotic growth rates***
Rank the given functions by order of growth. That is, produce an arrangement $g_1, g_2, \ldots, g_{30}$ *satisfying* $g_1 = \Omega(g_2), g_2 = \Omega(g_3), \ldots, g_{29} = \Omega(g_{30})$. *Partition the list into equivalence classes so that* $f(n)$ *and* $g(n)$ *are in the same class iff* $f(n) = \Theta(g(n))$.

Proof. Solution.

Identities:

$$2^{\lg n} = n, \qquad 4^{\lg n} = n^{\lg 4} = n^2, \qquad 2^{\lg^2 n} = n^{\lg n},$$
$$(\lg n)^{\lg n} = n^{\lg \lg n}, \qquad \lg(n!) = \Theta(n \lg n), \qquad n^{1/\lg n} = 2.$$

Ranking (from largest to smallest), with Θ-classes bracketed:

$$\boxed{(n+1)!} \succ \boxed{n!} \succ \boxed{e^n} \succ \boxed{n \cdot 2^n} \succ \boxed{2^n} \succ \boxed{(3/2)^n}$$
$$\succ \boxed{2^{\lg^2 n}} = \boxed{n^{\lg n}} \succ \boxed{(\lg n)^{\lg n}} = \boxed{n^{\lg \lg n}} \succ \boxed{(\lg n)!}$$
$$\succ \boxed{n^3} \succ \boxed{4^{\lg n}} = \boxed{n^2} \succ \boxed{\lg(n!)} = \boxed{n \lg n} \succ \boxed{2^{\lg n}} = \boxed{n}$$
$$\succ \boxed{\lg^2 n} \succ \boxed{\ln \ln n} \succ \boxed{2^{\lg^* n}} \succ \boxed{\lg^*(\lg n)} = \boxed{\lg^* n} \succ \boxed{\lg(\lg^* n)} \succ \boxed{n^{1/\lg n}} (= 2).$$

Equivalence classes explicitly:

$$\{(n+1)!\} \succ \{n!\} \succ \{e^n\} \succ \{n\,2^n\} \succ \{2^n\} \succ \{(3/2)^n\} \succ$$
$$\{2^{\lg^2 n}, n^{\lg n}\} \succ \{(\lg n)^{\lg n}, n^{\lg \lg n}\} \succ \{(\lg n)!\} \succ \{n^3\} \succ \{4^{\lg n}, n^2\} \succ$$
$$\{\lg(n!), n \lg n\} \succ \{2^{\lg n}, n\} \succ \{\lg^2 n\} \succ \{\ln \ln n\} \succ \{2^{\lg^* n}\} \succ \{\lg^*(\lg n), \lg^* n\} \succ$$
$$\{\lg(\lg^* n)\} \succ \{n^{1/\lg n}\}.$$

Why each cut is strict (intuition bullets):

- Factorials vs exponentials: $(n+1)!/e^n \to \infty$ (Stirling), so $(n+1)! \succ n! \succ e^n$.
- Exponentials: $n2^n/2^n = n \to \infty$, and $e^n/2^n = (e/2)^n \to \infty$.
- "Superpolynomials" below exponentials: $2^{\lg^2 n} = n^{\lg n} \gg n^{\lg \lg n} = (\lg n)^{\lg n} \gg (\lg n)!$ (compare base-2 logs).
- Polynomials: $n^3 \succ n^2 \succ n \lg n \succ n$; also $4^{\lg n} = n^2$ and $2^{\lg n} = n$.

- Between polylogarithms: $\lg(n!) = \Theta(n \lg n) \gg \lg^2 n \gg \ln \ln n$.
- Iterated-log family: $\ln \ln n \gg 2^{\lg^* n} \gg \lg^*(\lg n) = \lg^* n \gg \lg(\lg^* n) \gg \text{constant}$.
- Constant tail: $n^{1/\lg n} = 2$.

This ordering yields $g_i = \Omega(g_{i+1})$ throughout, and functions in the same bracketed set are Θ-equivalent by the identities listed at top. $\square$

Exercise 24. ***Problem 3-4: Asymptotic notation properties***
Let $f(n)$ and $g(n)$ be asymptotically positive functions. Prove or disprove each claim.

Proof. Solution.

(a) **Claim:** $f(n) = O(g(n)) \Rightarrow g(n) = O(f(n))$.
Intuition: Big-O is not symmetric unless the two are the same order.
Counterexample (rigorous): Take $f(n) = n$, $g(n) = n^2$. Then $f = O(g)$, but $g \neq O(f)$. Hence *false*.

(b) **Claim:** $f(n) + g(n) = \Theta(\min\{f(n), g(n)\})$.
Intuition: The sum is dominated by the *larger* term, not the smaller.
Counterexample: $f(n) = n$, $g(n) = n^2$. Then $f + g \sim n^2 = \Theta(\max\{f, g\}) \neq \Theta(\min\{f, g\})$. *False*.

(c) **Claim:** $f(n) = O(g(n)) \Rightarrow \lg f(n) = O(\lg g(n))$, assuming $\lg g(n) \geq 1$ and $f(n) \geq 1$ for all sufficiently large n.
Intuition: Logs turn multiplicative constants into additive constants.
Proof: If $f(n) \leq c\, g(n)$ for $n \geq n_0$, then for n large,

$$\lg f(n) \leq \lg(c\, g(n)) = \lg c + \lg g(n) \leq (\lg c + 1)\, \lg g(n).$$

Thus $\lg f(n) = O(\lg g(n))$. *True*.

(d) **Claim:** $f(n) = O(g(n)) \Rightarrow 2^{f(n)} = O(2^{g(n)})$.
Intuition: Multiplicative constant gaps become exponential gaps.
Counterexample: $f(n) = 2n$, $g(n) = n$. Then $f = O(g)$, but $2^{f(n)}/2^{g(n)} = 2^n \to \infty$, so $2^f \neq O(2^g)$. *False*.

(e) **Claim:** $f(n) = O((f(n))^2)$.
Intuition: Fails if $f(n) \to 0^+$.
Counterexample: $f(n) = 1/n$ (asymptotically positive). Then $f(n)/(f(n))^2 = n \to \infty$, so $f \neq O(f^2)$. *False*. (If $f(n) \geq 1$ eventually, the claim would hold.)

(f) **Claim:** $f(n) = O(g(n)) \Rightarrow g(n) = \Omega(f(n))$.
Intuition & Proof: This is exactly the dual definition of Ω. *True*.

(g) **Claim:** $f(n) = \Theta(f(n/2))$.
Intuition: Only holds for "smooth" (e.g., polynomial) growth; fails for exponentials.
Counterexample: $f(n) = 2^n$. Then $f(n)/f(n/2) = 2^n/2^{n/2} = 2^{n/2} \to \infty$, so not Θ. *False*.

(h) **Claim:** $f(n) + o(f(n)) = \Theta(f(n))$.
Intuition: Adding a negligible term doesn't change the order.
Proof: Let $h(n) = o(f(n))$. Then for any $\varepsilon \in (0, 1)$, for all large n, $|h(n)| \leq \varepsilon f(n)$, so

$$(1 - \varepsilon)f(n) \;\leq\; f(n) + h(n) \;\leq\; (1 + \varepsilon)f(n),$$

hence $f(n) + h(n) = \Theta(f(n))$. *True.*

□

Exercise 25. ***Problem 3-5: Manipulating asymptotic notation***
Let $f(n)$ and $g(n)$ be asymptotically positive functions. Prove the identities below.

Proof. Solution.

(a) $\Theta(\Theta(f(n))) = \Theta(f(n))$**.** By definition, $h(n) \in \Theta(f(n))$ iff $\exists\, c_1, c_2 > 0$, n_0 s.t. $c_1 f(n) \le h(n) \le c_2 f(n)$ for all $n \ge n_0$. Applying $\Theta(\cdot)$ again does not change the set of functions bounded above and below by constant multiples of $f(n)$. Hence equality of classes.

(b) $\Theta(f(n)) + O(f(n)) = \Theta(f(n))$**.** Let $h \in \Theta(f)$ and $r \in O(f)$. Then $\exists\, a_1, a_2 > 0, b > 0, n_0$ s.t. $a_1 f \le h \le a_2 f$ and $0 \le r \le bf$ for $n \ge n_0$. Thus

$$a_1 f(n) \;\le\; h(n) + r(n) \;\le\; (a_2 + b) f(n) \qquad (n \ge n_0),$$

so $h + r \in \Theta(f)$.

(c) $\Theta(f(n)) + \Theta(g(n)) = \Theta\big(f(n) + g(n)\big)$**.** If $h \in \Theta(f)$ and $r \in \Theta(g)$, then $\exists\, a_1 \le a_2$ and $b_1 \le b_2$ with $a_1 f \le h \le a_2 f$ and $b_1 g \le r \le b_2 g$ for large n. Hence

$$\min\{a_1, b_1\}\,(f + g) \;\le\; h + r \;\le\; \max\{a_2, b_2\}\,(f + g),$$

eventually. Therefore $h + r \in \Theta(f + g)$, and the reverse containment is immediate by choosing $h = f$ and $r = g$ (which are trivially in $\Theta(f)$ and $\Theta(g)$). So the classes are equal.

(d) $\Theta(f(n)) \cdot \Theta(g(n)) = \Theta\big(f(n)\, g(n)\big)$**.** If $h \in \Theta(f)$ and $r \in \Theta(g)$, then for large n, $a_1 f \le h \le a_2 f$ and $b_1 g \le r \le b_2 g$ with positive constants. Multiplying gives

$$a_1 b_1\, f(n) g(n) \;\le\; h(n) r(n) \;\le\; a_2 b_2\, f(n) g(n),$$

so $hr \in \Theta(fg)$. The reverse containment holds by taking $h = f$, $r = g$.

(e) Constants don't affect Θ for standard monomials. For any real constants $a_1, b_1 > 0$ and integers k_1, k_2,

$$\Theta\big(a_1\, n^{k_1} (\lg n)^{k_2}\big) \;=\; \Theta\big(n^{k_1} (\lg n)^{k_2}\big),$$

and

$$\Theta\big(a_1\, n^{k_1} (\lg n)^{k_2}\big) + \Theta\big(b_1\, n^{k_1} (\lg n)^{k_2}\big) = \Theta\big(n^{k_1} (\lg n)^{k_2}\big).$$

Reason: multiplying by fixed positive constants only rescales by absolute constant factors, which $\Theta(\cdot)$ ignores (by definition).

⋆ **(f) Summation pulls through Θ:** For $S \subseteq \mathbb{Z}$, assuming $\sum_{n\in S} f(n)$ converges absolutely,

$$\sum_{n\in S} \Theta\big(f(n)\big) \;=\; \Theta\left(\sum_{n\in S} f(n)\right).$$

Proof: For each n beyond some n_0, choose $h_n \in \Theta(f(n))$ with $c_1 f(n) \le h_n \le c_2 f(n)$ and fixed $c_1, c_2 > 0$. Then (absolute convergence ensures rearrangements are harmless)

$$c_1 \sum_{n\in S} f(n) \;\le\; \sum_{n\in S} h_n \;\le\; c_2 \sum_{n\in S} f(n),$$

so the sums are within constant factors; hence the claim.

⋆ **(g) Products do *not* necessarily preserve Θ:** There exists $S \subseteq \mathbb{Z}$ and sequences with $h_n \in \Theta(f(n))$ such that both $\prod_{n\in S} f(n)$ and $\prod_{n\in S} h(n)$ converge, but

$$\prod_{n\in S} h(n) \;\neq\; \Theta\left(\prod_{n\in S} f(n)\right).$$

Counterexample: Let $S = \mathbb{N}$, take $f(n) = 1 + \frac{1}{2^n}$ (so $\prod f(n)$ converges to a finite $L > 0$), and define

$$h(n) = \begin{cases} \frac{1}{2}\, f(n), & \text{if } n \text{ is even,} \\ f(n), & \text{if } n \text{ is odd.} \end{cases}$$

Then $h(n) \in \Theta(f(n))$ termwise, since $\frac{1}{2} f(n) \le h(n) \le f(n)$ for all n. But

$$\prod_{n=1}^{\infty} h(n) \;=\; \Big(\prod_{n=1}^{\infty} f(n)\Big) \cdot \prod_{n \text{ even}} \frac{1}{2} \;=\; L \cdot \underbrace{\left(\tfrac{1}{2}\right)^{\infty}}_{=\,0} \;=\; 0,$$

while $\prod f(n) = L > 0$. Both products converge, yet they are not within a constant factor (one is 0, the other positive), so the proposed product identity fails.

□

Exercise 26. ***Problem 3-6***
We sometimes use asymptotic notations in slightly different ways.

- *We write* $f(n) = \Omega_\infty(g(n))$ *if there exists a constant* $c > 0$ *such that* $f(n) \geq cg(n) \geq 0$ *for infinitely many integers* n.
- *We write* $f(n) = O'(g(n))$ *if* $|f(n)| = O(g(n))$.
- *We write* $f(n) = \tilde{O}(g(n))$ *if there exist constants* $c > 0$, $k \geq 0$, *and* n_0 *such that* $0 \leq f(n) \leq cg(n)(\lg n)^k$ *for all* $n \geq n_0$.

(a) Show that for any two asymptotically nonnegative functions $f(n)$ *and* $g(n)$, *either* $f(n) = O(g(n))$ *or* $f(n) = \Omega_\infty(g(n))$ *(or both).*

(b) Give asymptotically nonnegative functions $f(n)$ *and* $g(n)$ *such that neither* $f(n) = O(g(n))$ *nor* $f(n) = \Omega(g(n))$.

(c) Discuss the advantages and disadvantages of using Ω_∞ *in place of* Ω.

(d) If we use O' *in place of* O, *but we retain the usual definition of* Ω, *does Theorem 3.1 still hold? Prove your answer.*

(e) Give definitions of $\tilde{\Omega}$ *and* $\tilde{\Theta}$. *Show that Theorem 3.1 holds with* $\tilde{O}, \tilde{\Omega}, \tilde{\Theta}$.

Proof. Solution.

Theorem 3.1 (for reference). For asymptotically nonnegative f, g,

$$f(n) = \Theta(g(n)) \iff \big(f(n) = O(g(n)) \text{ and } f(n) = \Omega(g(n))\big).$$

(a) For asymptotically nonnegative f, g, either $f = O(g)$ or $f = \Omega_\infty(g)$ (or both).
Reason: If $f \notin O(g)$, then for every m and N there exists $n \geq N$ with $f(n) > m\, g(n)$. Hence, in particular, $f(n) \geq g(n) \geq 0$ for infinitely many n, i.e. $f = \Omega_\infty(g)$.

(b) There exist asymptotically nonnegative f, g with neither $f = O(g)$ nor $f = \Omega(g)$.
Example:

$$f(n) = \begin{cases} n^2, & n \text{ even} \\ n, & n \text{ odd} \end{cases}, \qquad g(n) = \begin{cases} n, & n \text{ even} \\ n^2, & n \text{ odd} \end{cases}.$$

Along even n, $f/g = n \to \infty$ ($f \neq O(g)$); along odd n, $f/g = 1/n \to 0$ ($f \neq \Omega(g)$).

(c) Advantage/disadvantage of Ω_∞.
Advantage: Together with part (a), one always has a relation (O or Ω_∞).
Disadvantage: Ω_∞ is weak—only "infinitely often," not "eventually," so it is ill-suited for worst-case lower bounds.

(d) Replace O by O' defined via $f = O'(g) \iff |f| = O(g)$, but keep Ω unchanged. *Claim:* The equivalence "$f = O'(g) \iff g = \Omega(f)$" fails. The direction $g = \Omega(f) \Rightarrow f = O'(g)$ holds, but $f = O'(g) \Rightarrow g = \Omega(f)$ need not.
Counterexample: $f(n) = (-1)^n n$, $g(n) = n$. Then $|f| = n = O(g)$, so $f = O'(g)$, but $g = \Omega(f)$ would require $0 \le cf(n) \le g(n)$ eventually, impossible since $f(n)$ is negative infinitely often.

(e) Soft-oh analogue. Define

$$f = \tilde{O}(g) \iff \exists\, c, k, n_0 :\ 0 \le f(n) \le c\, g(n)\, (\lg n)^k \quad (n \ge n_0),$$

and $f = \tilde{\Omega}(g)$ symmetrically by $0 \le c\, g(n) \le f(n)\, (\lg n)^k$. Then

$$f = \tilde{O}(g) \quad \iff \quad g = \tilde{\Omega}(f),$$

by simple rearrangement of the defining inequalities (for large n). Consequently,

$$f = \tilde{\Theta}(g) \iff \big(f = \tilde{O}(g) \text{ and } f = \tilde{\Omega}(g)\big).$$

Summary. Ω_∞ yields only a weak "either O or *infinitely-often* lower bound" dichotomy; O' breaks the Theorem 3.1 equivalence in one direction; the soft notations $\tilde{O}$ and $\tilde{\Omega}$ preserve the equivalence structure of Theorem 3.1. □

Exercise 27. ***Problem 3-7***

Iterated functions. We can apply the iteration operator * used in the $\lg^*$ function to any monotonically increasing function $f(n)$ over the reals. For a given constant $c \in \mathbb{R}$, we define the iterated function $f^*(n)$ by

$$f^*(n) = \min\{i \geq 0 : f^{(i)}(n) \leq c\},$$

which need not be well defined in all cases. In other words, the quantity $f^*(n)$ is the minimum number of iterated applications of the function f required to reduce its argument down to c or less.

For each of the functions $f(n)$ and constants c in the table below, give as tight a bound as possible on $f^*(n)$. If there is no i such that $f^{(i)}(n) \leq c$, write "undefined" as your answer.

$f(n)$	c
$n-1$	0
$\lg n$	1
$n/2$	1
$n/2$	2
2	—
1	—
$n^{1/3}$	2

Proof. Solution.

(a) $f(n) = n - 1$**,** $c = 0$**.** Each iteration subtracts 1, so $f^{(i)}(n) = n - i$. We need $n - i \leq 0$, i.e. $i \geq n$. Thus $f^*(n) = n$.

(b) $f(n) = \lg n$**,** $c = 1$**.** Each iteration takes a logarithm. By definition this is the iterated logarithm: $f^*(n) = \lg^* n$.

(c) $f(n) = n/2$**,** $c = 1$**.** Each iteration halves the argument. After i steps we have $f^{(i)}(n) = n/2^i$. We need $n/2^i \leq 1$, i.e. $2^i \geq n$. So $i = \lceil \lg n \rceil$. Thus $f^*(n) = \Theta(\lg n)$.

(d) $f(n) = n/2$**,** $c = 2$**.** Again $f^{(i)}(n) = n/2^i$. We need $n/2^i \leq 2$, i.e. $2^i \geq n/2$. So $i = \lceil \lg(n/2) \rceil = \lceil \lg n \rceil - 1$. Thus $f^*(n) = \Theta(\lg n)$.

(e) $f(n) = 2$**, any** $c \in \mathbb{R}$**.** Here $f(n) = 2$ for all n, and $f^{(i)}(n) = 2$ for all $i \geq 1$. If $c \geq 2$, then $f^*(n) = 1$ (since $2 \leq c$ after one iteration). If $c < 2$, then $f^*(n)$ is undefined.

(f) $f(n) = 1$**, any** $c \in \mathbb{R}$**.** Here $f(n) = 1$ for all n, and $f^{(i)}(n) = 1$ for all $i \geq 1$. If $c \geq 1$, then $f^*(n) = 1$. If $c < 1$, then $f^*(n)$ is undefined.

(g) $f(n) = n^{1/3}$**,** $c = 2$**.** Each iteration cubes-root the argument:

$$f^{(i)}(n) = n^{1/3^i}.$$

We need $n^{1/3^i} \leq 2$, i.e. $1/3^i \cdot \lg n \leq \lg 2$, so

$$3^i \geq \frac{\lg n}{\lg 2}.$$

Thus $i = \lceil \log_3(\lg n) \rceil$. Hence $f^*(n) = \Theta(\lg \lg n)$.

Summary:

$f(n), c$	$f^*(n)$
$n-1$, 0	n
$\lg n$, 1	$\lg^* n$
$n/2$, 1	$\Theta(\lg n)$
$n/2$, 2	$\Theta(\lg n)$
2, —	1 if $c \geq 2$, undefined otherwise
1, —	1 if $c \geq 1$, undefined otherwise
$n^{1/3}$, 2	$\Theta(\lg \lg n)$

□