Problem 1 (20 pts)

Put the following functions in order in terms of o-notation:

1. $\sqrt{n}$
2. $2^{\log_3 n}$
3. $(\log n)^2$
4. 3^n
5. n^3
6. $8^{n/2}$

Prove that your relation is correct for each adjacent pair. In particular, if your functions are ordered as $f_1, f_2, f_3, f_4, f_5, f_6$; then show that $f_1 \in o(f_2)$ and $f_2 \in o(f_3)$ and so on.

Solution

Step 1. Ordering

The functions ordered from slowest to fastest growth are as follows:

$$(\log n)^2, \quad \sqrt{n}, \quad 2^{\log_3 n}, \quad n^3, \quad 8^{n/2}, \quad 3^n$$

In o-notation, we write

$$(\log n)^2 \in o(\sqrt{n}) \in o(2^{\log_3 n}) \in o(n^3) \in o(8^{n/2}) \in o(3^n).$$

Step 2. Simplification

Before proving the relations, we simplify the functions to make the proofs easier.
Using the identity

$$a^{\log_b n} = n^{\log_b a}$$

we have

$$2^{\log_3 n} = n^{\log_3 2} \approx n^{0.63}$$

Also,

$$8^{n/2} = (8^{1/2})^n = (2\sqrt{2})^n$$

Step 3. Proofs

- 1. $(\log n)^2 \in o(\sqrt{n})$: We need to show that

$$\lim_{n\to\infty} \frac{(\log n)^2}{\sqrt{n}} = 0.$$

Let $n = 2^m$. Then,

$$\log n = m \quad \text{and} \quad \sqrt{n} = 2^{m/2}.$$

Thus,

$$\frac{(\log n)^2}{\sqrt{n}} = \frac{m^2}{2^{m/2}},$$

and as $m \to \infty$, the exponential term in the denominator dominates the polynomial numerator, so the limit is 0.

- 2. $\sqrt{n} \in o(2^{\log_3 n})$: Since $2^{\log_3 n} = n^{\log_3 2}$, we have

  $$\frac{\sqrt{n}}{2^{\log_3 n}} = \frac{n^{1/2}}{n^{\log_3 2}} = n^{\,1/2-\log_3 2}.$$

  Because $1/2 - \log_3 2 < 0$,

  $$\lim_{n\to\infty} n^{\,1/2-\log_3 2} = 0.$$

- 3. $2^{\log_3 n} \in o(n^3)$: Since $2^{\log_3 n} = n^{\log_3 2}$, we have

  $$\frac{2^{\log_3 n}}{n^3} = \frac{n^{\log_3 2}}{n^3} = n^{\,\log_3 2-3}.$$

  Because $\log_3 2 - 3 < 0$,

  $$\lim_{n\to\infty} n^{\,\log_3 2-3} = 0.$$

- 4. $n^3 \in o(8^{n/2})$: Since the exponential function $8^{n/2}$ grows much faster than the polynomial n^3, we have

  $$\lim_{n\to\infty} \frac{n^3}{8^{n/2}} = 0,$$

- 5. $8^{n/2} \in o(3^n)$: We have

  $$\frac{8^{n/2}}{3^n} = \left(\frac{8^{1/2}}{3}\right)^n = \left(\frac{2\sqrt{2}}{3}\right)^n.$$

  Since $\frac{2\sqrt{2}}{3} < 1$,

  $$\lim_{n\to\infty} \left(\frac{2\sqrt{2}}{3}\right)^n = 0.$$

Problem 2 (30 pts)

Consider the function $f(n) = n \cdot (n \mod 2) + \log n$.

(a) Show that $f(n) \in O(n)$ and $f(n) \in \Omega(\log n)$.

(b) Show that neither $f(n) \in \Theta(n)$ nor $f(n) \in \Theta(\log n)$.

(c) Suppose for some function $g(n)$, we have $f(n) \notin O(g(n))$. Is it always true that $f(n) \in \omega(g(n))$? Justify your answer with either a proof or a counter-example.

Solution

(a)

$\underline{f(n) \in O(n)}$: For any $n \geq 2$:

$$\begin{aligned} \text{If } n \text{ is even:} \quad & f(n) = \log n & \leq n, \\ \text{If } n \text{ is odd:} \quad & f(n) = n + \log n & \leq n + n = 2n. \end{aligned}$$

Thus, $f(n) \leq 2n$ for all $n \geq 2$. Choosing $C = 2$ and $n_0 = 2$ gives $f(n) \in O(n)$.

$\underline{f(n) \in \Omega(\log n)}$: For all n:

$$\begin{aligned} \text{If } n \text{ is even:} \quad & f(n) = \log n, \\ \text{If } n \text{ is odd:} \quad & f(n) = n + \log n & \geq \log n. \end{aligned}$$

Thus, let $c = 1$ we have $f(n) \geq \log n$ for all n, so $f(n) \in \Omega(\log n)$.

(b)

We prove by contradiction.

Assum $f(n) \in \Theta(n)$, then there exist constants $c_1, c_2 > 0$ and n_0 such that for all $n \geq n_0$,

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

However, when n is even, $f(n) = \log n$ and the inequality $c_1 n \leq \log n$ fails for large n (since $\log n / n \to 0$). Thus, we've reached a contradiction and $f(n) \notin \Theta(n)$.

Similarly, assume $f(n) \in \Theta(\log n)$, then there exist constants $c_1, c_2 > 0$ such that for all large n,

$$c_1 \log n \leq f(n) \leq c_2 \log n.$$

But for odd n, $f(n) = n + \log n$ grows like n, which is much larger than any constant multiple of $\log n$. Thus, we've reached a contradiction and $f(n) \notin \Theta(\log n)$.

(c)

It is not always true. We will show a counterexample.

Counterexample: Define

$$g(n) = \begin{cases} n, & \text{if } n \text{ is even,} \\ \log n, & \text{if } n \text{ is odd.} \end{cases}$$

Then:

$$\begin{aligned} \text{If } n \text{ is even:} \quad & f(n) = \log n, \quad g(n) = n, \quad \tfrac{f(n)}{g(n)} = \tfrac{\log n}{n} \to 0, \\ \text{If } n \text{ is odd:} \quad & f(n) = n + \log n, \quad g(n) = \log n, \quad \tfrac{f(n)}{g(n)} \sim \tfrac{n}{\log n} \to \infty. \end{aligned}$$

Thus, there is no constant C such that $f(n) \leq C\, g(n)$ for all large n (so $f(n) \notin O(g(n))$). However, $f(n)$ also does not satisfy the definition of $\omega(g(n))$ because for even n, $\lim_{n\to\infty} f(n)/g(n)$ is 0 but not ∞.

Problem 3 (20 pts)

Let $f(n)$ and $g(n)$ be non-negative functions.

(a) Using the formal definition of $\Theta()$, prove that $\max(f(n), g(n)) = \Theta(f(n) + g(n))$, where

$$\max(a, b) = \begin{cases} a & \text{if } a \geq b \\ b & \text{otherwise} \end{cases}.$$

(b) Can we also show that $\min(f(n), g(n)) = \Theta(f(n) + g(n))$, where

$$\min(a, b) = \begin{cases} a & \text{if } a \leq b \\ b & \text{otherwise} \end{cases}?$$

If yes, show how the proof from part (a) needs to be adapted. If no, provide a counter-example.

Solution

(a)

Since both $f(n)$ and $g(n)$ are nonnegative,

$$\max(f(n), g(n)) = \begin{cases} f(n), & \text{if } f(n) \geq g(n) \\ g(n), & \text{otherwise} \end{cases}.$$

In either case,

$$f(n) \leq \max(f(n), g(n)) \quad \text{and} \quad g(n) \leq \max(f(n), g(n)).$$

Adding these two inequalities we have:

$$f(n) + g(n) \leq 2\max(f(n), g(n)) \quad \Longrightarrow \quad \max(f(n), g(n)) \geq \frac{1}{2}\big(f(n) + g(n)\big).$$

Also, clearly,

$$\max(f(n), g(n)) \leq f(n) + g(n).$$

Thus, we have

$$\frac{1}{2}\big(f(n) + g(n)\big) \leq \max(f(n), g(n)) \leq f(n) + g(n).$$

This shows

$$\max(f(n), g(n)) = \Theta(f(n) + g(n)),$$

with constants $c_1 = \frac{1}{2}$ and $c_2 = 1$.

(b)

It is always true that

$$\min(f(n), g(n)) \leq f(n) + g(n),$$

however, for the equality $\min(f(n), g(n)) = \Theta(f(n)+g(n))$ to hold we would also require a constant $c > 0$ such that

$$\min(f(n), g(n)) \geq c\,(f(n) + g(n))$$

for all sufficiently large n. We will show a counterexample.

Counterexample: Let

$$f(n) = n \quad \text{and} \quad g(n) = 1 \quad \text{for all } n.$$

Then,

$$\min(f(n), g(n)) = \min(n, 1) = 1,$$

and

$$f(n) + g(n) = n + 1.$$

Givin $\min(f(n), g(n)) = \Theta(f(n) + g(n))$, there exist constants $c_1, c_2 > 0$ and n_0 such that for all $n \geq n_0$,

$$c_1(n+1) \leq 1 \leq c_2(n+1).$$

While the upper bound $1 \leq c_2(n+1)$ holds for any $c_2 \geq 0$ and large n, the lower bound

$$1 \geq c_1(n+1)$$

cannot hold for any fixed $c_1 > 0$ as $n \to \infty$. Thus, the equality does not hold in general and we cannot say that

$$\min(f(n), g(n)) = \Theta(f(n) + g(n)).$$

Problem 4 (30 pts)

You are given the coefficients $\alpha_0, \alpha_1, \ldots, \alpha_n$ of a polynomial

$$\begin{aligned} P(x) &= \sum_{k=0}^{n} \alpha_k x^k \\ &= \alpha_0 + \alpha_1 x + \alpha_2 x^2 + \cdots + \alpha_n x^n, \end{aligned}$$

and you want to evaluate this polynomial for a given value of x. *Horner's rule* says to evaluate the polynomial according to this parenthesization:

$$P(x) = \alpha_0 + x\Bigg(\alpha_1 + x\Bigg(\alpha_2 + \cdots + x\left(\alpha_{n-1} + x\alpha_n\right)\cdots\Bigg)\Bigg).$$

The procedure Horner implements Horner's rule to evaluate $P(x)$, give the coefficients $\alpha_0, \alpha_1, \ldots, \alpha_n$ in an array $A[0:n]$ and the value of x.

Horner(A, n, x)

```
1: p ← 0
2: for i = n to 0 do
3:     p ← A[i] + x · p
4: end for
5: return p
```

For this problem, assume that addition and multiplication can be done in constant time.

(a) In terms of Θ-notation, what is the running time of this procedure?

(b) Write pseudocode to implement the naive polynomial-evaluation algorithm that computes each term of the polynomial from scratch. What is the running time of this algorithm? How does it compare to Horner?

(c) Consider the following loop invariant for the prcedure Horner:
At the start of each iteration of the **for** loop of lines 2-3,

$$p = \sum_{k=0}^{n-(i+1)} A[k+i+1] \cdot x^k.$$

Interpret a summation with no terms as equaling 0. Following the structure of the loop-invariant proof presented in class, use this loop invariant to show that, at termination, $p = \sum_{k=0}^{n} A[k] \cdot x^k$.

Solution

(a)

The for loop in Horner iterates from $i = n$ down to 0, i.e., it makes $n + 1$ iterations. In each iteration, a constant number of operations is performed (one multiplication and one addition).

Thus, the total running time is proportional to $n+1$. In Θ–notation we have:

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

(b)

A naive method to evaluate

$$P(x) = \sum_{k=0}^{n} A[k]\, x^k$$

is to compute each term $A[k] \cdot x^k$ from scratch and sum them.

NaivePolyEval(A, n, x):

```
p ← 0
for k = 0 to n do
    term ← 1
    for j = 1 to k do
        term ← term · x
    end for
    p ← p + A[k] · term
end for
return p
```

In this algorithm, for each k the inner loop runs k times. Therefore, the total number of multiplications is

$$\sum_{k=0}^{n} k = \frac{n(n+1)}{2} = \Theta(n^2).$$

Thus, the running time of the naive algorithm is $\Theta(n^2)$, which is asymptotically slower than Horner's $\Theta(n)$ method.

(c)

We consider the following loop invariant:

$$\text{At the start of each iteration of the for-loop (lines 2–3), } p = \sum_{k=0}^{n-(i+1)} A[k+i+1] \cdot x^k.$$

We want to show that when the loop terminates, $p = \sum_{k=0}^{n} A[k] \cdot x^k$.

Initialization:

When the loop begins, $i = n$. Then the invariant claims that:

$$p = \sum_{k=0}^{n-(n+1)} A[k+n+1] \cdot x^k.$$

Since the upper limit is $n-(n+1) = -1$, the summation is empty and by convention equals 0. This matches the initialization $p \leftarrow 0$, so the invariant holds initially.

Maintenance:

Suppose the invariant holds at the start of an iteration for some index i, so that:

$$p = \sum_{k=0}^{n-(i+1)} A[k+i+1] \cdot x^k.$$

In the body of the loop, we update p as follows:

$$p \leftarrow A[i] + x \cdot p.$$

Substituting the invariant expression into this update gives:

$$p = A[i] + x\left(\sum_{k=0}^{n-(i+1)} A[k+i+1] \cdot x^k\right) = A[i] + \sum_{k=0}^{n-(i+1)} A[k+i+1] \cdot x^{k+1}.$$

Reindex the summation by letting $j = k+1$. Then when $k = 0$, $j = 1$, and when $k = n-(i+1)$, $j = n-i$. This gives:

$$p = A[i] + \sum_{j=1}^{n-i} A[j+i] \cdot x^j = \sum_{j=0}^{n-i} A[j+i] \cdot x^j,$$

where we have written $A[i]$ as the term corresponding to $j = 0$. This is exactly the loop invariant for the next iteration (with i replaced by $i-1$).

Termination:
The loop terminates when $i = -1$. At this point, the invariant gives:

$$p = \sum_{k=0}^{n-((-1)+1)} A[k+(-1)+1] \cdot x^k = \sum_{k=0}^{n} A[k] \cdot x^k.$$

Hence, upon termination, $p = P(x)$.

This completes the loop invariant proof that Horner correctly evaluates the polynomial.