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

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.

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.

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, $p = \sum_{k=0}^{n} A[k] \cdot x^k$.