Learning Journal – Unit 2: Algorithm Analysis

This week, I spent my time trying to understand algorithm analysis and how to analyze and describe the performance of algorithms using asymptotic notations. Mostly, I read through the theoretical concepts in the learning guide and textbook, prepared and posted in the discussions forum, solved real algorithmic problems, and responded to my classmates' posts to come to a better understanding of algorithms.

In my discussion post, I worked through two algorithm analysis problems. The first one compared a polynomial-time algorithm with an exponential-time algorithm, helping me understand at what input size one starts outperforming the other. It was eye-opening to realize how even a polynomial algorithm with a large constant can eventually become more efficient than an exponential algorithm as the input grows. Through step-by-step calculations, I determined that Algorithm A, which runs in $O(n^3)$ time, overtakes Algorithm B, running in $O(2^n)$, at around $n = 29$. This reinforced the powerful concept that exponential growth, though seemingly manageable at small sizes, becomes impractical very quickly.

The second problem was about figuring out the time complexity of nested loops. Outer loop ran n, inner loop ran at max i, thus making the time complexity $= \Theta(n^2)$ In this example, I used the idea of tight bounds with Big Theta notation to show that the runtime of the algorithm is upper and lower bounded by a quadratic function. Going through this practical example helped me realize the differences between best-case, worst-case and average-case complexities and how to arrive at asymptotic notations for these scenarios. It also taught me how to come up with the running time based on real code, which is an essential skill for a software developer.

Throughout the week, I reflected on the theoretical content, particularly the role of upper, lower, and tight bounds in evaluating algorithms. I found Shaffer's (2011) explanation of these concepts particularly helpful, especially his breakdown of asymptotic analysis rules and simplification strategies. His point that Big-O notation provides an upper bound on growth rate but does not indicate how tight that bound is made me realize the importance of using Big Omega (Ω) and Big Theta (Θ) when applicable, to express algorithm performance more accurately (Shaffer, 2011).

One part that challenged me was interpreting when exactly exponential functions begin to dominate polynomial ones. The math was clear, but the broader implication—that exponential algorithms can become unusable very quickly—really made me think about how essential it is to choose efficient algorithms, especially in real-world systems that need to be scaled. I also appreciated how Shaffer (2011) emphasized trade-offs in algorithm design, noting that sometimes more efficient algorithms are complex to implement, and the cost of implementation outweighs performance gains for small inputs. This helped me consider the practical aspect of algorithm choice, not just theoretical efficiency.

I feel more confident now in my ability to analyze algorithms and determine their time complexity using formal notation. I'm gaining valuable skills in logical reasoning and mathematical abstraction—both of which are critical in computer science. I also realize that I learn best by actively working through examples and comparing different approaches. The discussion activity helped me articulate my thoughts and validate them through peer interaction, which strengthened my conceptual understanding.

One important takeaway this week is the insight that algorithm efficiency is not just about reducing lines of code—it's about understanding how performance scales as input size increases.

This is crucial in both academic exercises and real-world software systems. I'm now thinking more about how to balance simplicity and performance when writing or choosing algorithms and how theoretical understanding guides practical decision-making.

Wordcount: 597

Reference:

Shaffer, C. (2011). A Practical Introduction to Data Structures and Algorithm Analysis. Blacksburg: Virginia. Tech. https://people.cs.vt.edu/shaffer/Book/JAVA3e20130328.pdf