Erdianti Wiga Putri A

13522053 – K01

**Rangkuman Algoritma dan Struktur Data**

**Chapter 3 – Big O Notation**

Big O Notation measures the number of steps an algorithm takes, regardless of how long each step actually takes. O(1) means that the algorithm always takes the same number of steps, no matter the size of the data. Operations like reading from an array and adding or removing elements from the end of an array are examples of O(1) operations. Big O Notation signifies how the number of steps an algorithm takes changes with the data size. O(N) represents a linear relationship where each additional data element adds an extra step, forming a diagonal line on a graph. In contrast, O(1) denotes constant time, with the number of steps remaining consistent regardless of data size, represented as a horizontal line. Even if an algorithm takes more than one step, if it remains constant for any data size, it is still considered O(1). While a 100-step O(1) algorithm is less efficient than a one-step algorithm, it is more efficient than any O(N) algorithm, as depicted by the graph where the O(N) algorithm surpasses the O(1) algorithm only for datasets exceeding a certain threshold. O(log N) means that the algorithm takes as many steps as it takes to keep halving the data elements until we remain with one. The algorithm that prints all items from a list is described as O(N) because the number of steps directly correlates with the number of elements in the list, with each element requiring a print statement. Conversely, the algorithm for printing "Hello world!" is denoted as O(1) as it consistently takes one step. For the algorithm determining whether a number is prime, the efficiency is expressed as O(N) since the number of steps increases linearly with the magnitude of the input number, each step corresponding to a division check.