Uzmabanu Kapadia
EMPLID: 23864352
HW 2-2 Time and space complexity

Time Complexity

Time complexity is a way to measure how an algorithm takes to finish based on size of the input. It gives an upper limit on the running time of the algorithm in based on the input size. To express this time complexity, the big O notation (O(f(n))) is used, in which f(n) represents the change in the growth rate in relation to the input size (n). Some commonly known time complexities include O(1) for constant time, O(n) for linear time, O(log n) for logarithmic time, O(n log n) for linearithmic, and O(n^2) for quadratic time. To determine the time complexity of an algorithm, you analyze the number of basic operations such as assignments, comparison, and arithmetic operations that the algorithm performs based on the input size. After that, this function is simplified and expressed using the Big O notation, focusing on the term that grows the fastest with the input size.

Space Complexity

Space complexity refers to the amount of memory an algorithm uses based on the input size. It also gives an upper limit on the on the memory needed by the algorithm, including both the auxiliary space and the input space. Similarly to time complexity, space complexity is also represented using the Big O notation. To calculate the space complexity of an algorithm, you examine the total memory used by the algorithm, which consists of variables, data structures, recursive function call stack, and any temporary storage. This memory usage is expressed as a function of the input size and simplify it to determine the space complexity in Big O notation.

The analysis of the time and space complexity aids in comparing various algorithms to be able to choose the most efficient one for the given task. It also gives insights into how the performance of an algorithm scales as its input size increases, helping in making informed decisions on selecting and optimizing algorithms.