Best case time complexity, Average case time complexity, Worst case time complexity and Lower bound

We talk about the **Best case time complexity,** the **Average case time complexity** and the **Worst case time complexity** in the context of an algorithm **A** to solve a problem **P**. For example, you know those values for the quick sort.

We talk about lower bound in the context of a problem **P**. What we say applies to all algorithms **A** (present and future) to solve **P**.

For example, we proved that “The problem **P** of sorting n items using comparison” has a lower bound of Ω(n log n). What we are saying is “any sorting algorithm **A** to sort n item through comparison (present or future) will have worst case time complexity O(n log n).

We have the following relationship.

Let **P** be a problem and **A** be an algorithm to solve **P**. Then

The Worst case time complexity A $\geq$ The Lower Bound of (P).

If Worst case time complexity A is equal to the Lower Bound of (P), then A is an optimal algorithm to solve P

Example:

P is the problem of sorting n items using comparison.

Merge sort (**A**) an algorithm that solves **P**.

Worst case time complexity of **A** (merge sort) is O(n log n).

Lower bound of **P** (the problem of sorting n items using comparison) is Ω(n log n).

Since both time complexity expressions are same (in this case **n log n**), we say Merge Sort is an **optimal algorithm**. Note that Quick Sort is **not** an optimal algorithm!

f is O(g) means f(n) <= cg(n) for large values of n.

f is Ω(g) means f(n) >= cg(n) for large values of n.

f is Θ(g) means f(n) = cg(n) for large values of n.