Algorithm Analysis

In this lesson, we'll cover algorithm analysis.

We'll cover the following

- Algorithm analysis
- Cases
- Big-O notation

Algorithm analysis

Analyzing the run-time complexity of your solution is the most essential step in solving a problem to determine whether your solution is going to execute in time or if the judge is going to throw a TLE (Time Limit Exceeded) verdict.

Cases

There are three cases when it comes to algorithm analysis:

1. Best Case
2. Average Case
3. Worst Case

When participating in online competitions, we are always interested in worst-case analysis. The test suite for the problem will still contain input, which forces your solution to its worst-case performance.

Big-O notation

The Big-O notation is used to define the upper bound of an algorithm. We will get into the complexity analysis of algorithms in their corresponding lessons. Here we'll see the properties of the Big-O notation.

Complexity is expressed in terms of the input size. Take for example, searching for integer x through a list of N integers. This will require anywhere between `1` (Best

case => x is the first element) and `N` (Worst case => x is the last element)

operations. We say that this is an `O(N)` solution, i.e., linear in input size (N). This will become clearer once we go through some samples.

In the next lesson, we will see how to calculate run-time complexities for simpler cases.