

In general, the elapsed time is the time from the starting point to ending point of an event. Following are various ways to find elapsed time in Java −

* The currentTimeMillis() method returns the current time in milliseconds. To find the elapsed time for a method you can get the difference between time values before and after the execution of the desired method.
* The nanoTime() method returns the current time in nano seconds. To find the elapsed time for a method you can get the difference between time values before and after the execution of the desired method.
* The now() method of the Instant class returns the current time and the Duration.between() methods returns the difference between the given two time values to get the elapsed time retrieve the time values before and after the execution of the desired method and retrieve the duration using the Duration.between() method.
* The Apache commons library provides a class known as Stopwatch to it provides the start() stop() and getTime() methods to find the time taken for the execution of a method.

From <<https://www.tutorialspoint.com/how-to-measure-execution-time-for-a-java-method>>

For parameters -->O(a+b)

#Big O Cheat Sheet:

-Big OsO(1) Constant- no loops

O(log N) Logarithmic- usually searching algorithms have log n if they are sorted (Binary Search)

O(n) Linear- for loops, while loops through n items

O(n log(n)) Log Liniear- usually sorting operations

O(n^2) Quadratic- every element in a collection needs to be compared to ever other element. Two

nested loops

n^3 Cubic

O(2^n) Exponential- recursive algorithms that solves a problem of size N

O(n!) Factorial- you are adding a loop for every element

Iterating through half a collection is still O(n)

Two separate collections: O(a \* b)

-What can cause time in a function?-

Operations (+, -, \*, /)

Comparisons (<, >, ==)

Looping (for, while)

Outside Function call (function())

-Rule BookRule 1: Always worst Case

Rule 2: Remove Constants

Rule 3: Different inputs should have different variables. O(a+b). A and B arrays nested would be

O(a\*b)

+ for steps in order

\* for nested steps

Rule 4: Drop Non-dominant terms

-What causes Space complexity?-

Variables

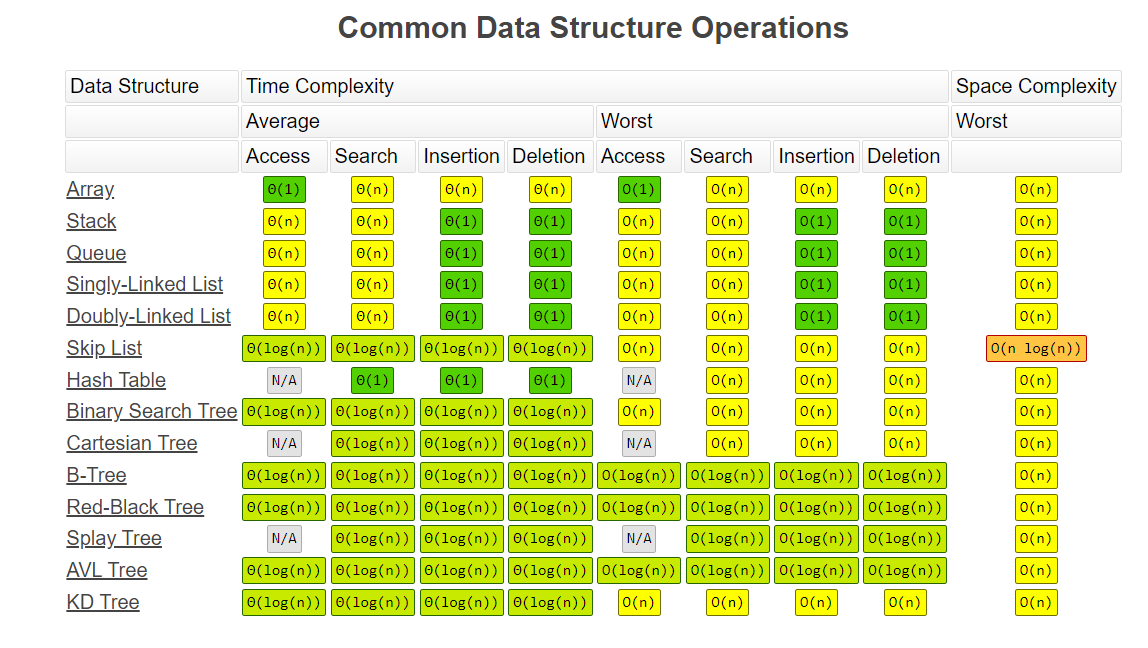
Data Structures

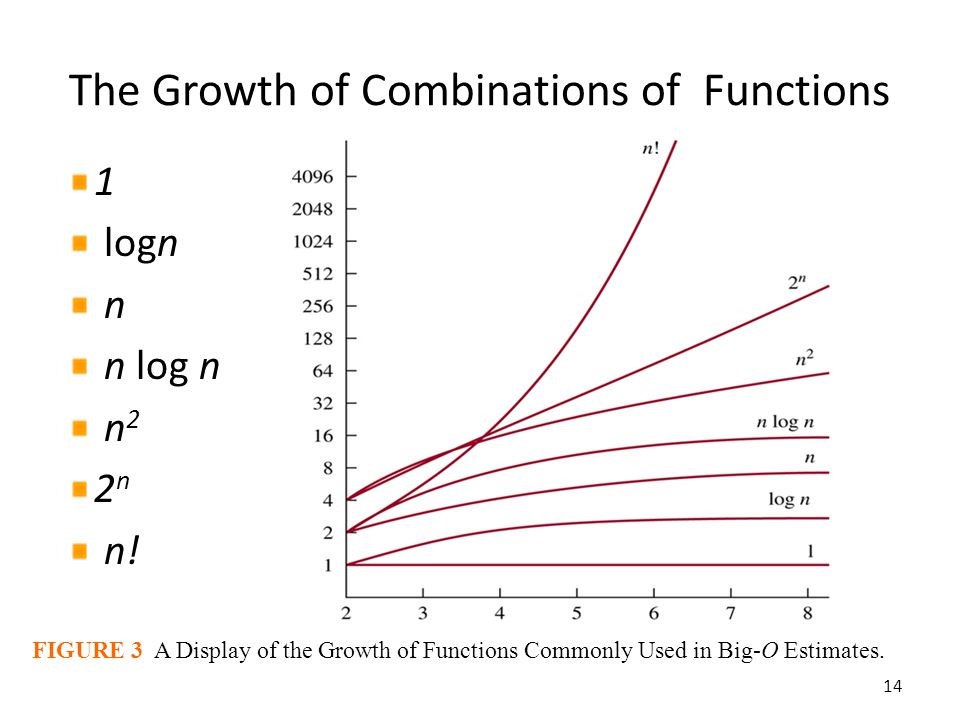
Function Call

Allocations

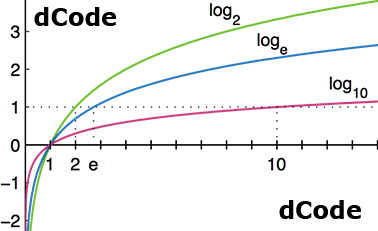
<<028 BigO-cheat-sheet-1.pdf>>

<https://www.bigocheatsheet.com/>





n



n^n>n!>2^n>n^2>nlogn>n>logn>1

Log2n>log10n