# Big O Notation

"Is a mathematical notation used to describe the performance or complexity of an algorithm in relation to the time taken or the memory used for task"

## Big O order of time complexity

**O(1)** Describes an algorithm that always takes the same time to perform the task

e.g an algorithm deciding if a number is positive or negative.

**O(N)** Describes an algorithm where the time increases linearly in relation to the number of items (N)

e.g linear Search.

**O($N^2$)** Describes an algorithm where the time to perform the task will increase linearly to the square of number of items (N)

e.g bubble sort, insertion sort.

**O($2^N$)** Describes an algorithm where the time to perform the tasks double every time the algorithm uses an extra item of data.

e.g calculation of fibonacci numbers using reursion.

**O (Log N)** Describes an algorithm where the time increases logarithmically

e.g binary search.

## Big O order of Space Complexity

**O(1)** Describes an algorithm that always uses the same space to perform the task

e.g any algorithm that just uses variables, for example

d = a + b + c

**O (N)** Describes an algorithm where the space to perform the task increases linearly to the number of items.

e.g any algorithm that uses array.

**(c)** Compare the performance of the algorithms for a binary search and a linear search using Big O notation for order of time complexity.

Linear search is O(n) and Binary Search is O(log n)

(time to search increases linearly in relation to the number of items in the list for a linear search and logarithmically in binary search.

Time to search increases more rapidly for a linear search as you increase number of items in a list and time to search increases less rapidly for a binary search

[3] 3