**Search Algorithms**

[I. Introduction. 2](#_Toc80297663)

[II. Contents. 2](#_Toc80297664)

[1. Liner search. (*Sequential search)* 2](#_Toc80297665)

[2. Binary search. (*Interval search)* 3](#_Toc80297666)

# **I. Introduction.**

* Searching algorithms are designed to check for an element or retrieve an element from any data structure where it is stored.
* Based on the type of search operation, these algorithms are generally classified into two categories:
  + ***Sequential search***.
  + ***Interval search***.
* Depend on the case of the data to use the right one of the two algorithms above.
* To be easier for illustration, the object that contains a individual that we are going to search in this topic is the a consecutive sequence of integers (array of integers)
* There are also different methods to search for a element like *Hashing* or using *Search Trees* which also will be researched deeper in different topics.

# **II. Contents.**

1. **Liner search. (*Sequential search)***

* This is the most simple method to searching for an elements.
* The array is tranversed through every single element while making a comparison between the current element in the array with the element we are looking for.
* Time Complexity: O(n).
* Example:

Array A = {**84**, **65**, **148**, **1**, **49**, **26**}.

* Search for the key = 148:



Example 1 - 1

* Search for key = 0:



Example 1 - 2

* Advantage: Easy to implement.
* Disadvantage: Take very long time.

1. **Binary search. (*Interval search)***

* This algorithm is only able to be used on sorted structures.
* The main action of this algorithm is the division half of the current’s total of elements by eliminating all the elements before or after the middle element depend on what value of the element is looking for and the middle element is considering in the current elements.
* Time Complexity: O(log n).
* Example:

Array A = {**84**, **65**, **148**, **1**, **49**, **26**}.

* Search for the key = 148:



Example 2 - 1

* Search for the key = 55:



Example 2 - 2