# **Linear search vs binary search**

|  |  |  |
| --- | --- | --- |
| **Basis forcomparison** | **Linear search** | **Binary search** |
| Time complexity | O(n) | O(log 2 n) |
| Best case time | First element O(1) | Center element O(1) |
| Prerequisite for an array | No required | Array must be in sorted order |
| Worst case for n number of elements | N comparisons are required | Can conclude after only log2n comparisons |
| Can be implemented on | Array and linked list | Cannot be directly implemented on linked list |
| Insert operation | Easily inserted at the end of list | Require processing to insert at its proper place to maintain a sorted list. |
| Algorithm type | Iterative in nature | Divide and conquer in nature |
| Usefulness | Easy to use and no need for any ordered elements | Somehow tricky algorithm and elements must be arranged in order |
| Lines of code | Less | More |
| Comparison type | Requires equality comparisons | Requires an *ordering* comparison |
| Access to data | Requires sequential access (it means a linear search can stream data of arbitrary size) | Requires random access to the data |
| Run Timing | Search is fast when the amount of data is small | Binary search is efficient for larger array |

# Interpolation Search vs Binary Search

|  |  |  |
| --- | --- | --- |
| **Basis forcomparison** | **Interpolation search** | **Binary search** |
| Time complexity | O(loglogn)  O(n) (worst case) | O(log 2 n) |
|  | works better than Binary Search for a sorted and uniformly distributed array |  |

# Why is Binary Search preferred over Ternary Search?

From the first look, it seems the ternary search does less number of comparisons as it makes Log3n recursive calls, but binary search makes Log2n recursive calls. Let us take a closer look.

The following is recursive formula for counting comparisons in worst case of Binary Search.

T(n) = T(n/2) + 2, T(1) = 1

The following is recursive formula for counting comparisons in worst case of Ternary Search.

T(n) = T(n/3) + 4, T(1) = 1

In binary search, there are 2Log2n + 1 comparisons in worst case. In ternary search, there are 4Log3n + 1 comparisons in worst case.

Time Complexity for Binary search = 2clog2n + O(1)

Time Complexity for Ternary search = 4clog3n + O(1)

Therefore, the comparison of Ternary and Binary Searches boils down the comparison of expressions 2Log3n and Log2n . The value of 2Log3n can be written as (2 / Log23) \* Log2n . Since the value of (2 / Log23) is more than one, Ternary Search does more comparisons than Binary Search in worst case.

# **References**

<https://www.geeksforgeeks.org/searching-algorithms/>