Tutorials ▾ References ▾ Exercises ▾ Sign In

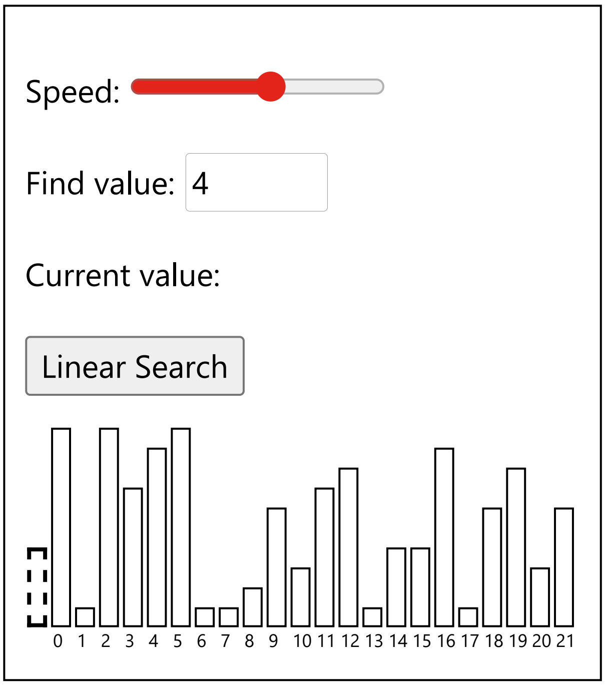
SS JAVASCRIPT SQL PYTHON JAVA PHP HOW TO W3.CSS C C-

DSA Linear Search

❮ Previous Next ❯

Linear Search

The Linear Search algorithm searches through an array and returns the index of the value it searches for.



Speed:

Find value: 4

Current value:

Linear Search

This algorithm is very simple and easy to understand and implement.

If the array is already sorted, it is better to use the much faster Binary Search algorithm that we will explore on the next page.

A big difference between *sorting* algorithms and *searching* algorithms is that sorting algorithms modify the array, but searching algorithms leave the array unchanged.

How it works:

1. Go through the array value by value from the start.
2. Compare each value to check if it is equal to the value we are looking for.
3. If the value is found, return the index of that value.
4. If the end of the array is reached and the value is not found, return -1 to indicate that the value was not found.

Manual Run Through

Let's try to do the searching manually, just to get an even better understanding of how Linear Search works before actually implementing it in a programming language. We will search for value 11.

Step 1: We start with an array of random values.

```
[ 12, 8, 9, 11, 5, 11]
```

Step 2: We look at the first value in the array, is it equal to 11?

```
[ 12, 8, 9, 11, 5, 11]
```

Step 3: We move on to the next value at index 1, and compare it to 11 to see if it is equal.

Step 4: We check the next value at index 2.

```
[ 12, 8, 9, 11, 5, 11]
```

Step 5: We move on to the next value at index 3. Is it equal to 11?

```
[ 12, 8, 9, 11, 5, 11]
```

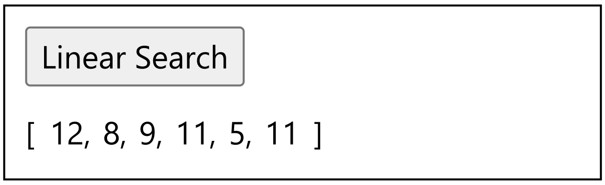
We have found it!

Value 11 is found at index 3.

Returning index position 3.

Linear Search is finished.

Run the simulation below to see the steps above animated:



[12, 8, 9, 11, 5, 11]

Manual Run Through: What Happened?

This algorithm is really straight forward.

Every value is checked from the start of the array to see if the value is equal to 11, the value we are trying to find.

When the value is found, the searching is stopped, and the index where the value is found is returned.

If the array is searched through without finding the value, -1 is returned.

1. An array with values to search through.
2. A target value to search for.
3. A loop that goes through the array from start to end.
4. An if-statement that compares the current value with the target value, and returns the current index if the target value is found.
5. After the loop, return -1, because at this point we know the target value has not been found.

The resulting code for Linear Search looks like this:

Example

```
def linearSearch(arr, targetVal):
    for i in range(len(arr)):
        if arr[i] == targetVal:
            return i
    return -1

arr = [3, 7, 2, 9, 5]
targetVal = 9

result = linearSearch(arr, targetVal)

if result != -1:
    print("Value",targetVal,"found at index",result)
else:
    print("Value",targetVal,"not found")
```

Try it Yourself »

Linear Search Time Complexity

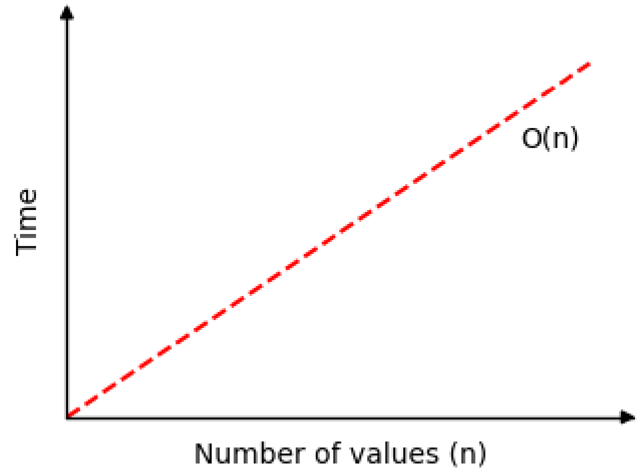
For a general explanation of what time complexity is, visit this page.

But if Linear Search runs through the whole array of n values, without finding the target value, n compares are needed.

This means that time complexity for Linear Search is

$$O(n)$$

If we draw how much time Linear Search needs to find a value in an array of n values, we get this graph:



Run the simulation below for different number of values in an array, and see how many compares are needed for Linear Search to find a value in an array of n values:

Set values: 300

● Random

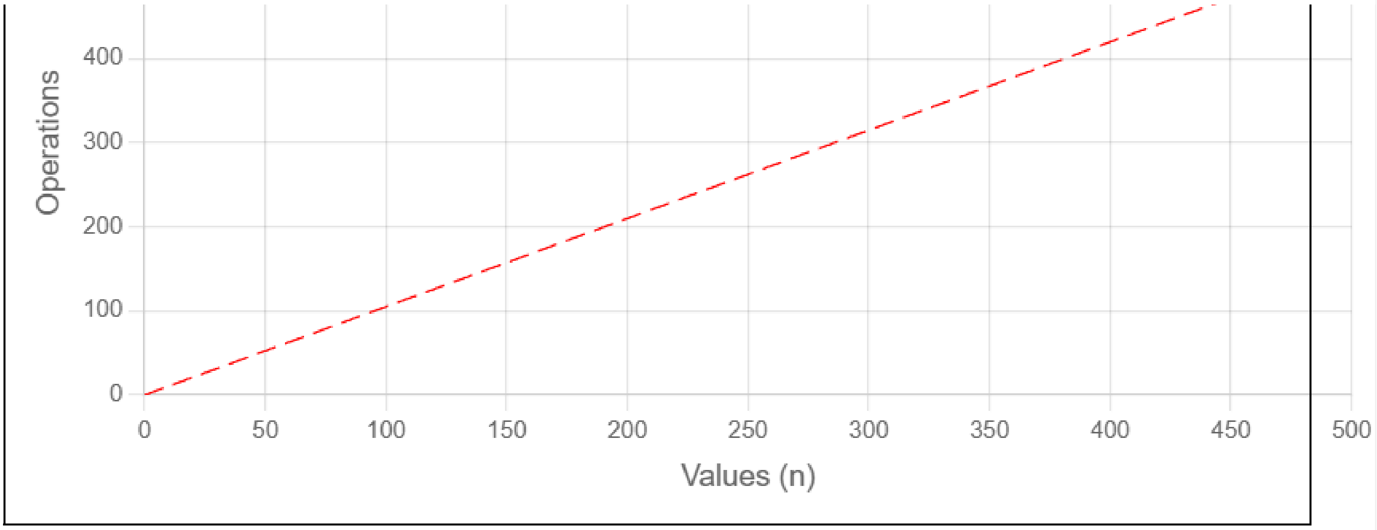
○ Descending

○ Ascending

Operations: 0

○ 10 Random

Run Clear



Choosing "Random", "Descending" or "Ascending" in the simulation above has no effect on how fast Linear Search is.

DSA Exercises

Test Yourself With Exercises

Exercise:

Complete the code.

```
def linearSearch(arr, targetVal):
    for i in range(len(arr)):
        if arr[i] == targetVal:

    return -1
```

Start the Exercise

❮ Previous Sign in to track progress Next ❯

schools PLUS SPACES

GET CERTIFIED FOR TEACHERS

FOR BUSINESS CONTACT US

Top Tutorials

HTML Tutorial
CSS Tutorial
JavaScript Tutorial
How To Tutorial
SQL Tutorial
Python Tutorial
W3.CSS Tutorial
Bootstrap Tutorial
PHP Tutorial
Java Tutorial
C++ Tutorial
jQuery Tutorial

Top References

HTML Reference
CSS Reference
JavaScript Reference
SQL Reference
Python Reference
W3.CSS Reference
Bootstrap Reference
PHP Reference
HTML Colors
Java Reference
AngularJS Reference
jQuery Reference

Top Examples

HTML Examples
CSS Examples
JavaScript Examples
How To Examples
SQL Examples
Python Examples
W3.CSS Examples
Bootstrap Examples
PHP Examples
Java Examples
XML Examples
jQuery Examples

Get Certified

HTML Certificate
CSS Certificate
JavaScript Certificate
Front End Certificate
SQL Certificate
Python Certificate
PHP Certificate
jQuery Certificate
Java Certificate
C++ Certificate
C# Certificate
XML Certificate