**Sorting Algorithm**

A sorting algorithm is used to arrange elements of an array/list in a specific order



# Bubble Sort

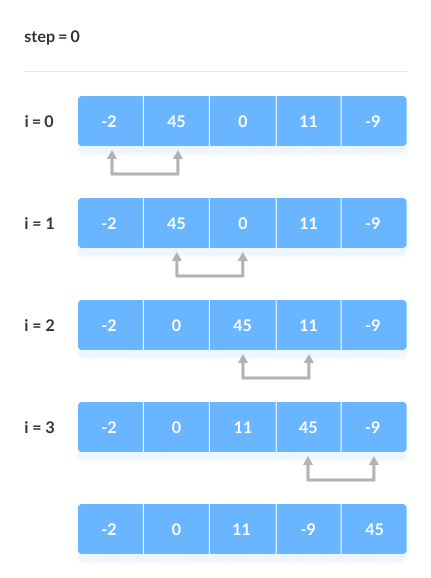
# ****Bubble sort**** is [a sorting algorithm](https://www.programiz.com/dsa/sorting-algorithm) that compares two adjacent elements and swaps them until they are in the intended order.

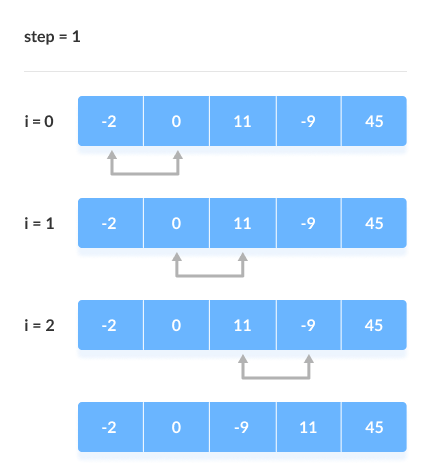
## Working of Bubble Sort

Suppose we are trying to sort the elements in****ascending order**.**

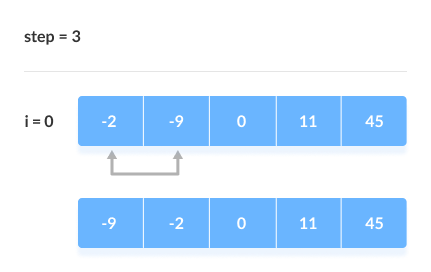
****1. First Iteration (Compare and Swap)****

1. Starting from the first index, compare the first and the second elements.
2. If the first element is greater than the second element, they are swapped.
3. Now, compare the second and the third elements. Swap them if they are not in order.
4. The above process goes on until the last element.



2 Iteration

**3 Iteration**



Example:

# Bubble sort in Python

def bubbleSort(array):

# loop to access each array element

for i in range(len(array)):

# loop to compare array elements

for j in range(0, len(array) - i - 1):

# compare two adjacent elements

# change > to < to sort in descending order

if array[j] > array[j + 1]:

# swapping elements if elements

# are not in the intended order

temp = array[j]

array[j] = array[j+1]

array[j+1] = temp

data = [-2, 45, 0, 11, -9]

bubbleSort(data)

print('Sorted Array in Ascending Order:')

print(data)

## **Searching Algorithm**

A searching algorithm is a method or technique used to find a particular item or value in a collection of data. The collection of data could be an array, a list, a database, or any other data structure. The goal is to determine whether a specific element exists within the collection and, if so, to locate its position.

1. Linear searching

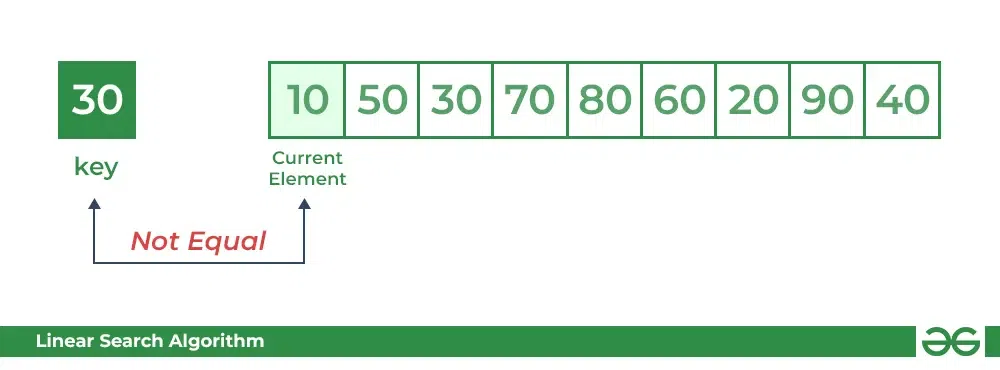
2. Binary searching

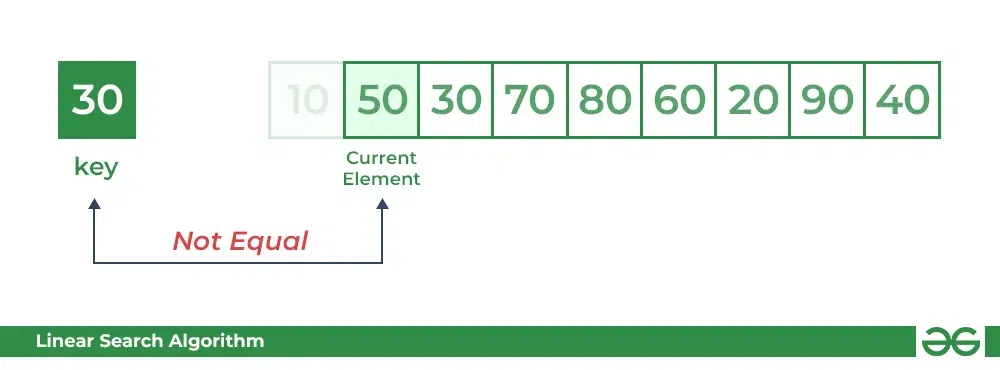
**1. Linear Searching**

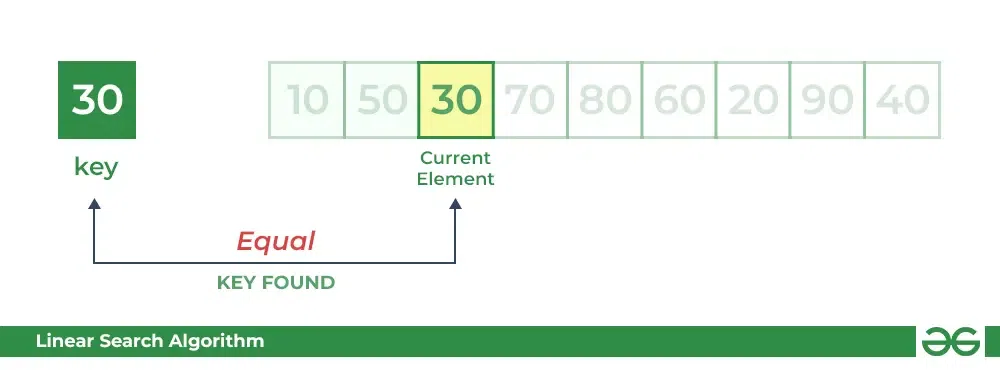
Linear search is a method of finding elements within a list. It is also called a sequential search.  It is the simplest searching algorithm because it searches the desired element in a sequential manner.It compares each and every element with the value that we are searching for. If both are matched, the element is found, and the algorithm returns the key's index position.

**For example:**Consider the array **arr[] = {10, 50, 30, 70, 80, 20, 90, 40}** and **key** = 30

***Step 1:****Start from the first element (index 0) and compare* ***key*** *with each element (arr[i]).*

  
*Comparing key with next element arr[1]. SInce not equal, the iterator moves to the next element as a potential match.*

*****Step 2:******Now when comparing arr[2] with key, the value matches. So the Linear Search Algorithm will yield a successful message and return the index of the element when key is found (here 2).*



**Algorithm**LinearSearch(list,key)

1. for each item in the list
2. if item==value
3. return its index position
4. return -1

**Example:1**

1. def linear\_Search(list1, n, key):
2. # Searching list1 sequentially
3. for i in range(0, n):
4. if (list1[i] == key):
5. return i
6. return -1
7. # Example list
8. list1 = [1, 3, 5, 4, 7, 9]
9. key = 7
11. n = len(list1)
12. res = linear\_Search(list1, n, key)
13. if res == -1:
14. print("Element not found")
15. else:
16. print("Element found at index:", res)

**2.Binary Searching**