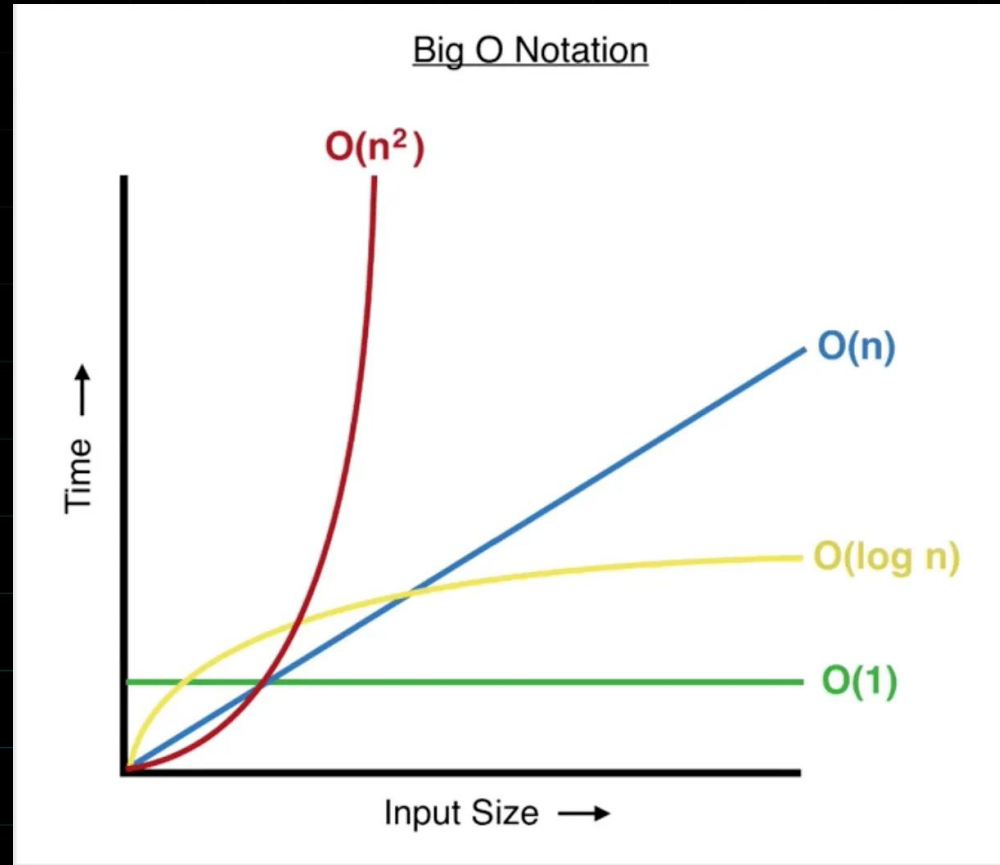


# Big O; Searching, Sorting



# BIG O - Time Space Complexity

## Time Complexity



# BIG O - Time Space Complexity

## Time Complexity

```
● ● ● Problem 1.py

data_array = [1,2,3,4,5]
# Problem:
# How many items are there in the sorted array
no_of_items = 0
for item in data_array:
    no_of_items = no_of_items + 1

print('Number of items: ', no_of_items)
```

# BIG O - Time Space Complexity

## Time Complexity

```
Problem 1.py

data_array = [1,2,3,4,5,6,7,8,9,10]
# Problem:
# How many items are there in the sorted array
no_of_items = 0
for item in data_array:
    no_of_items = no_of_items + 1

print('Number of items: ', no_of_items)
```

# BIG O - Time Space Complexity

## Time Complexity

```
● ● ● Problem 1.py

data_array = [1,2,3,4,5]
# Problem:
# How many items are there in the sorted array
no_of_items = data_array[-1]
|
print('Number of items: ', no_of_items)
```

# BIG O - Time Space Complexity

## Time Complexity

```
● ● ● Problem 1.py

data_array = [1,2,3,4,5,6,7,8,9,10]
# Problem:
# How many items are there in the sorted array
no_of_items = data_array[-1]
|
print('Number of items: ', no_of_items)
```

# BIG O - Time Space Complexity

Time Complexity

$O(1) \leftarrow O(2), O(100), O(1000)$

$O(n) \leftarrow O(2n), O(10n), O(n/2), O(2n + 100),$

$O(\log n) \leftarrow O(3\log n), O(4\log n + 3n + 1)$

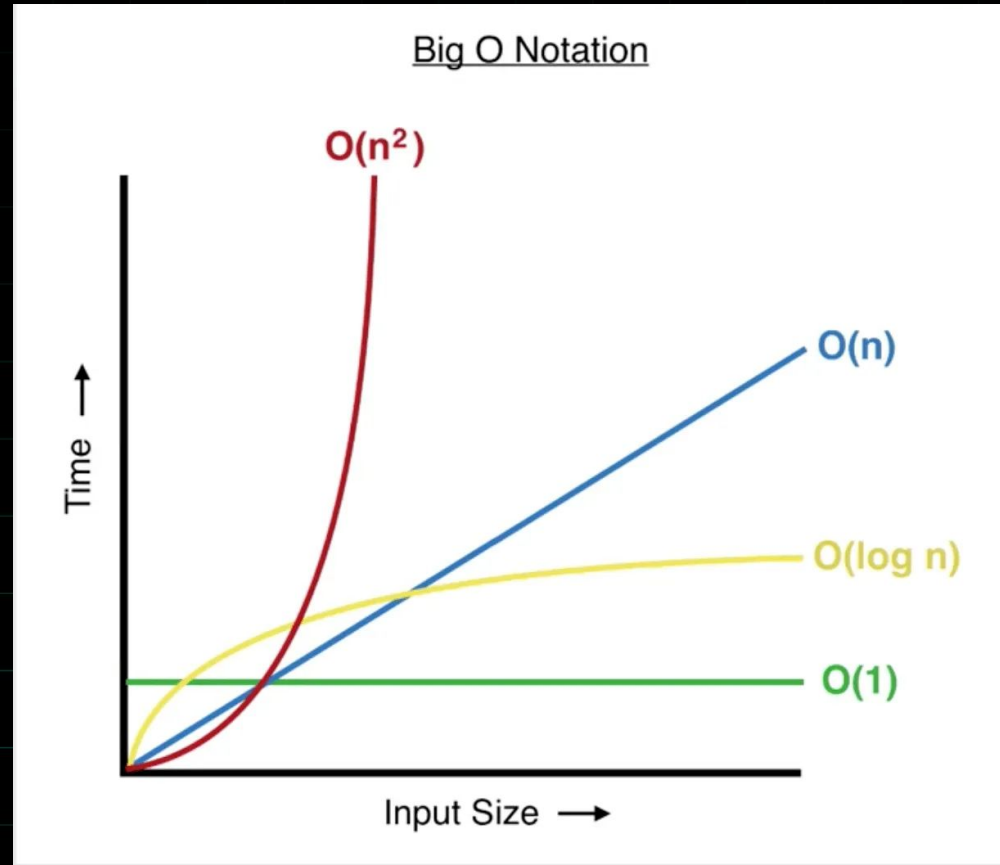
$O(n\log n)$  - Usually in sorting

$O(n^2)$  - For every element of input perform  $n \times n$  instructions

$O(n^3)$  - For every element of input perform  $n \times n \times n$  instructions

# BIG O - Time Space Complexity

## Time Complexity





# BIG O - Time Space Complexity

## Space Complexity

```
Problem 1.py

numbers = [1,10,4,11] # Input
# calculate the sum of the integers
result = 0
for num in numbers:
    result = result + num
print(result)
```

# BIG O - Time Space Complexity

## Space Complexity

```
● ● ● Problem 1.py  
  
numbers = [1,10,4,11,11,20,8,2] # Input  
# calculate the sum of the integers  
result = 0  
for num in numbers:  
    result = result + num  
print(result)
```

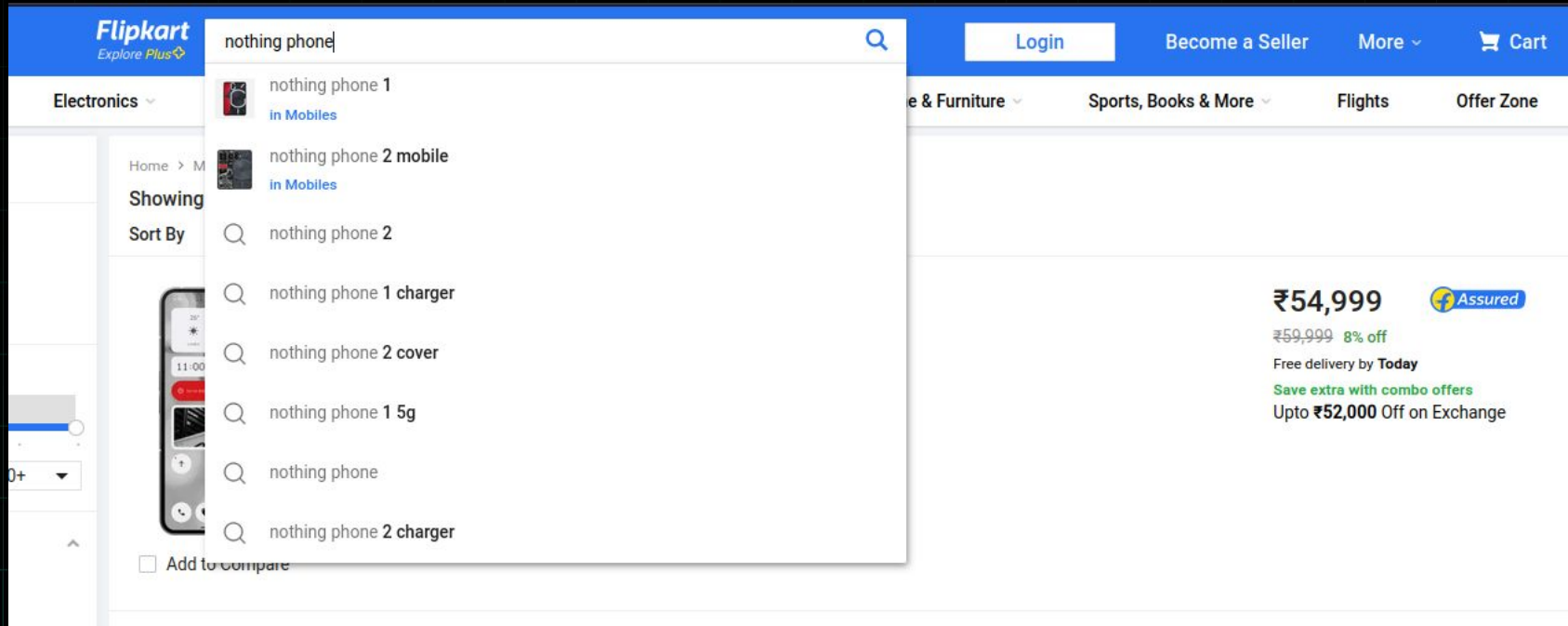
# BIG O - Time Space Complexity

## Space Complexity

```
Untitled-1

num = 10
# create an array containing 10 elements
newList = []
for i in range(num):
    newList.append(i)
print(newList)
```

# Searching & Sorting



# Searching & Sorting

Flipkart  
Explore Plus

nothing phone


Login Become a Seller More Cart

Electronics TVs & Appliances Men Women Baby & Kids Home & Furniture Sports, Books & More Flights Offer Zone

Home > Mobiles & Accessories > Mobiles

Showing 1 – 10 of 10 results for "nothing phone"

Sort By Relevance Popularity **Price -- Low to High** Price -- High to Low Newest First



**Nothing Phone (1) (Black, 128 GB)**

4.3 ★ 77,940 Ratings & 9,307 Reviews

- 8 GB RAM | 128 GB ROM
- 16.64 cm (6.55 inch) Full HD+ Display
- 50MP + 50MP | 16MP Front Camera
- 4500 mAh Lithium-ion Battery
- Qualcomm Snapdragon 778G+ Processor
- Meet the Glyph Interface. A New Way to Communicate
- 1 Billion Colours, True-to-Life Full HD Flexible OLED Display with HDR10+ for Richer Colour and Deeper Contrasts.
- 1 Year Warranty

☐ Add to Compare

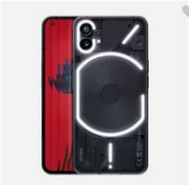
**₹28,999**

₹37,999 23% off

Free delivery

Save extra with combo offers

Upto ₹26,500 Off on Exchange



**Nothing Phone (1) (Black, 256 GB)**

4.3 ★ 77,940 Ratings & 9,307 Reviews

- 8 GB RAM | 256 GB ROM
- 16.64 cm (6.55 inch) Full HD+ Display
- 50MP + 50MP | 16MP Front Camera
- 4500 mAh Lithium-ion Battery
- Qualcomm Snapdragon 778G+ Processor
- Meet the Glyph Interface. A New Way to Communicate
- 1 Billion Colours, True-to-Life Full HD Flexible OLED Display with HDR10+ for Richer Colour and Deeper Contrasts.
- 1 Year Warranty

☐ Add to Compare

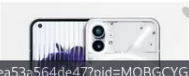
**₹29,999**

₹39,999 25% off

Free delivery

Save extra with combo offers

Upto ₹26,500 Off on Exchange



**Nothing Phone (1) (White, 256 GB)**

4.3 ★ 77,940 Ratings & 9,307 Reviews

**₹30,499**

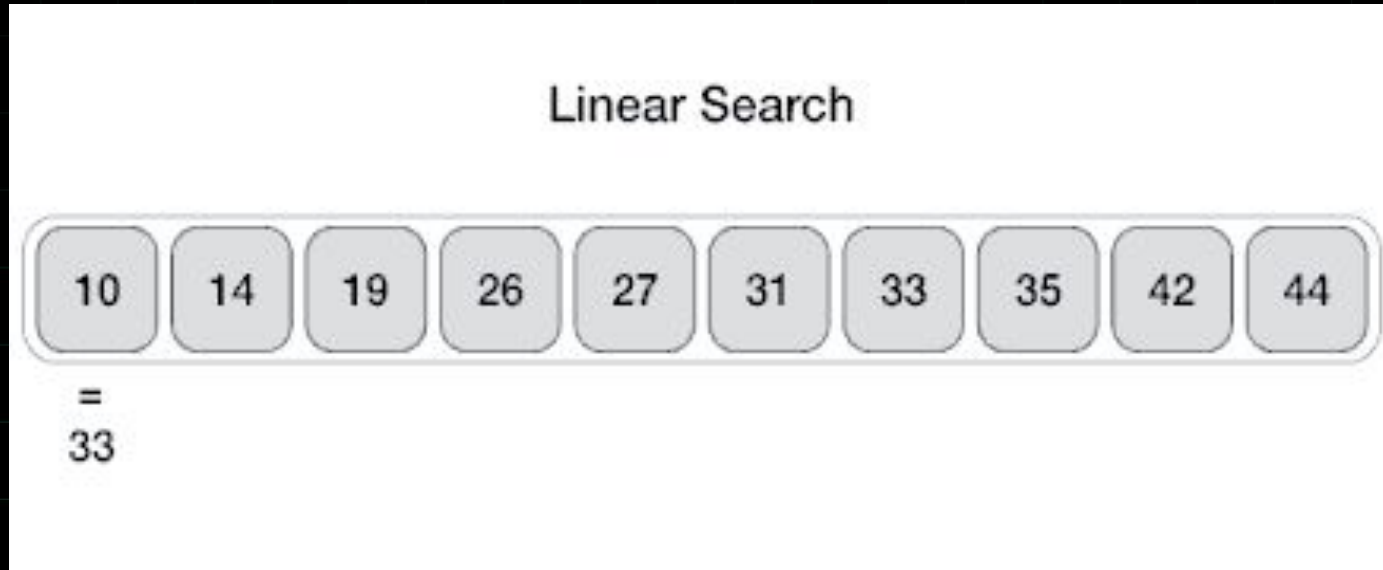
₹39,999 23% off

Free delivery

Save extra with combo offers

Upto ₹26,500 Off on Exchange

# Linear Search



# Linear Search

```
Untitled-1

def linear_search(arr, target):
    for i in range(len(arr)):
        if arr[i] == target:
            return i
    return -1

# Example usage
arr = [10, 15, 20, 5, 30]
target = 30

result = linear_search(arr, target)
if result != -1:
    print("Element found at index", result)
else:
    print("Element not found")
```

# Searching - Binary Search

## Binary Search

Searching for...

41

1	2	7	8	22	28	41	58	67	71	94
0	1	2	3	4	5	6	7	8	9	10



# Searching - Binary Search

```
Untitled-1

def binary_search(arr, x):
    left = 0
    right = len(arr) - 1

    while left <= right:
        mid = (left + right) // 2

        if arr[mid] == x:
            return mid

        if arr[mid] < x:
            left = mid + 1
        else:
            right = mid - 1

    return -1

# Example usage
arr = [2, 4, 6, 8, 10, 12]
x = 10

result = binary_search(arr, x)

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

# Searching - Linear vs Binary Search

Binary search

steps: 0



Sequential search

steps: 0



www.penjee.com

# Sorting - Bubble Sort



# Sorting - Bubble Sort

```
Untitled-1

def bubble_sort(arr):
    n = len(arr)

    for i in range(n):
        for j in range(0, n-i-1):
            if arr[j] > arr[j+1]:
                arr[j], arr[j+1] = arr[j+1], arr[j]

# Example usage
arr = [64, 34, 25, 12, 22, 11, 90]

bubble_sort(arr)

print("Sorted array:")
for i in range(len(arr)):
    print(arr[i])
```

# Sorting - Quick Sort

Unsorted Array



# Sorting - Quick Sort

```
Untitled-1

def partition(arr, low, high):
    pivot = arr[high]
    i = low - 1

    for j in range(low, high):
        if arr[j] <= pivot:
            i += 1
            arr[i], arr[j] = arr[j], arr[i]

    arr[i+1], arr[high] = arr[high], arr[i+1]
    return i+1

def quicksort(arr, low, high):
    if low < high:
        pi = partition(arr, low, high)
        quicksort(arr, low, pi-1)
        quicksort(arr, pi+1, high)

# Example usage
arr = [10, 7, 8, 9, 1, 5]
n = len(arr)
quicksort(arr, 0, n-1)
print(arr)
```

# Continuous Practical Assignment I

- **Task Automation using Python**

# FAQs



# Thank you!

[douglas.cst@rub.edu.bt](mailto:douglas.cst@rub.edu.bt)