

90% Refund

Courses ▾

Placement ▾

Data Science ▾

GATE ▾

Practice ▾

🔍

⚙️

🔔

📄

I

Dash

All

Articles

Videos

Quiz

← Back to Home

Articles (7)

Analysis of Algorithms (Background) ✓
Last Updated: 2025-03-07

Big O Notation, Omega Notation, Theta Notation ✓
Last Updated: 2023-03-25

Introduction to Asymptotic Notation ✓
Last Updated: 2024-03-20

Worst, Average and Best Case Time Complexities ✓
Last Updated: 2025-03-03

Analysis of Common Loop ✓
Last Updated: 2024-07-31

Analysis of Recursion ✓
Last Updated: 2025-03-17

Space Complexity ✓
Last Updated: 2022-10-20

Articles Read

0 of 7 Complete. (0%)

Progress may take upto 2 hours to reflect.

⏪ Prev

Next ⏩

Worst, Average and Best Case Time Complexities

It is important to analyze an algorithm after writing it to find it's efficiency in terms of time and space in order to improve it if possible.

When it comes to analyzing algorithms, the asymptotic analysis seems to be the best way possible to do so. This is because asymptotic analysis analyzes algorithms in terms of the input size. It checks how are the time and space growing in terms of the input size.

In this post, we will take an example of Linear Search and analyze it using Asymptotic analysis.

We can have three cases to analyze an algorithm:

1. Worst Case
2. Average Case
3. Best Case

Below is the algorithm for performing linear search:

Python

📄

▶

```
1 # Searching an element in a list/array in python
2 # can be simply done using \'in\' operator
3 # Example:
4 # if x in arr:
5 # print arr.index(x)
6
7 # If you want to implement Linear Search in python
8
9 # Linearly search x in arr[]
10 # If x is present then return its location
11 # else return -1
12
13 def search(arr, x):
14
15     for i in range(len(arr)):
```

https://www.geeksforgeeks.org/batch/dsa-python-self-paced-april/track/analysis-of-algorithms-basics-python/article/NjM5Ng%3D%3D

1/3

Dash

All

Articles

Videos

Quiz

← Back to Home

Articles (7)

Analysis of Algorithms (Background)

✓

Last Updated: 2025-03-07

Big O Notation, Omega Notation, Theta Notation

✓

Last Updated: 2023-03-25

Introduction to Asymptotic Notation

✓

Last Updated: 2024-03-20

Worst, Average and Best Case Time Complexities

✓

Last Updated: 2025-03-03

Analysis of Common Loop

✓

Last Updated: 2024-07-31

Analysis of Recursion

✓

Last Updated: 2025-03-17

Space Complexity

✓

Last Updated: 2022-10-20

Articles Read

0 of 7 Complete. (0%)

Progress may take upto 2 hours to reflect.

```
16
17     if arr[i] == x:
18         return i
19
20     return -1
```

Worst Case Analysis (Usually Done): In the worst case analysis, we calculate upper bound on running time of an algorithm. We must know the case that causes the maximum number of operations to be executed. For Linear Search, the worst case happens when the element to be searched (x in the above code) is not present in the array. When x is not present, the search() functions compares it with all the elements of arr[] one by one. Therefore, the worst case time complexity of linear search would be O(N), where N is the number of elements in the array.

Average Case Analysis (Sometimes done): In average case analysis, we take all possible inputs and calculate computing time for all of the inputs. Sum all the calculated values and divide the sum by total number of inputs. We must know (or predict) distribution of cases. For the linear search problem, let us assume that all cases are uniformly distributed (including the case of x not being present in array). So we sum all the cases and divide the sum by (N+1). Following is the value of average case time complexity..

Average Case Time=

$$\frac{\sum_{i=1}^{n+1} \theta(i)}{(n+1)}$$
$$= \frac{\theta((n+1)*(n+2)/2)}{(n+1)}$$
$$= \theta(n)$$

Best Case Analysis (Bogus): In the best case analysis, we calculate lower bound on running time of an algorithm. We must know the case that causes minimum number of operations to be executed. In the linear search problem, the best case occurs when x is present at the first location. The number of operations in the best case is constant (not dependent on N). So time complexity in the best case would be Ω(1)

Important Points:

- Most of the times, we do the worst case analysis to analyze algorithms. In the worst analysis, we guarantee an upper bound on the running time of an algorithm which is a good piece of information.
- The average case analysis is not easy to do in most of the practical cases and it is rarely done. In the average case analysis, we must know (or predict) the mathematical distribution of all possible inputs.
- The Best Case analysis is bogus. Guaranteeing a lower bound on an algorithm doesn't provide any information as in the worst case, an algorithm may take years to run.

https://www.geeksforgeeks.org/batch/dsa-python-self-paced-april/track/analysis-of-algorithms-basics-python/article/NjM5Ng%3D%3D

2/3

Mark as Read



Dash



All



Articles



Videos



Quiz

[Back to Home](#)

Articles (7)

Analysis of Algorithms (Background)



Last Updated: 2025-03-07

Big O Notation, Omega Notation, Theta Notation



Last Updated: 2023-03-25

Introduction to Asymptotic Notation



Last Updated: 2024-03-20

Worst, Average and Best Case Time Complexities



Last Updated: 2025-03-03

Analysis of Common Loop



Last Updated: 2024-07-31

Analysis of Recursion



Last Updated: 2025-03-17

Space Complexity



Last Updated: 2022-10-20

[Report An Issue](#)

If you are facing any issue on this page. Please let us know.



Articles Read

0 of 7 Complete. (0%)

Progress may take upto 2 hours to reflect.