Explaining

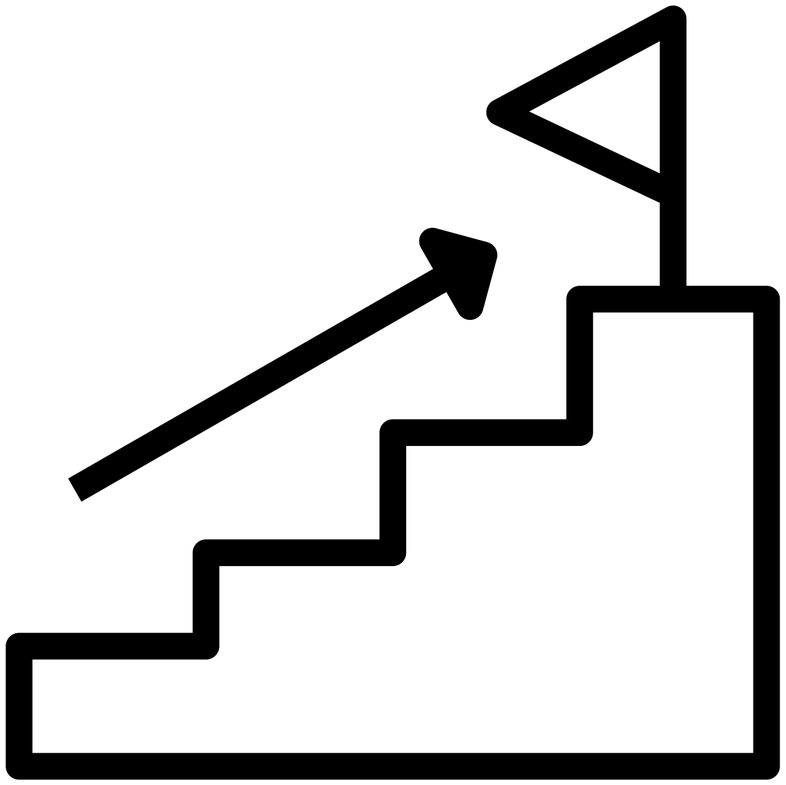
Complexity Analysis & Big O Notation

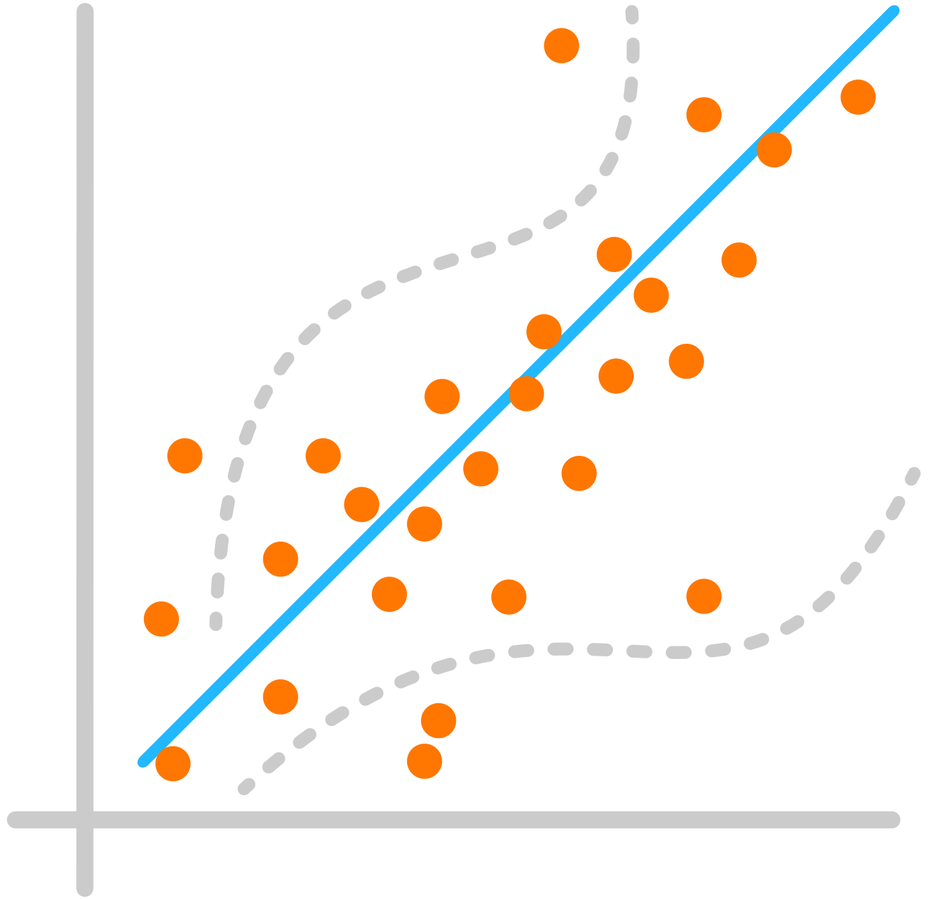
To DevOps Engineers

Complexity Analysis

@Sandip Das

Complexity analysis is the study of how the performance of tasks and algorithms scales with the size of the input or the workload.

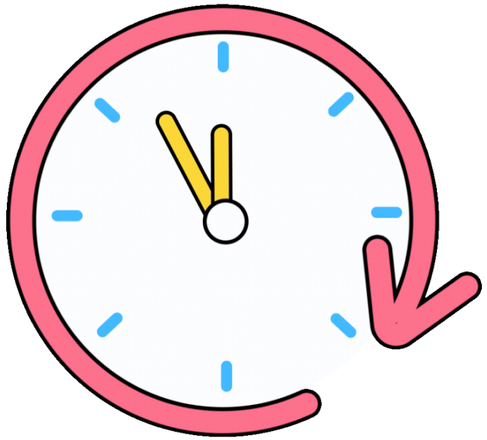




@Sandip Das

Complexity Analysis

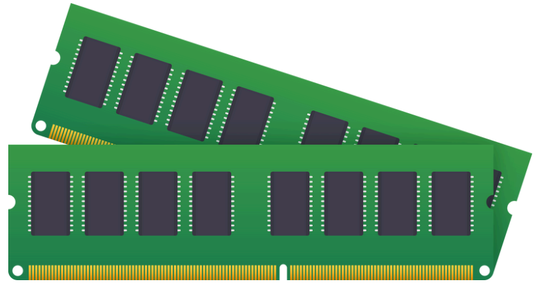
It helps you determine the efficiency of an algorithm in terms of time (time complexity) and space (space complexity)



Time Complexity

- Measures how the runtime of an algorithm grows as the input size increases.
- Important for understanding how fast an algorithm processes data.

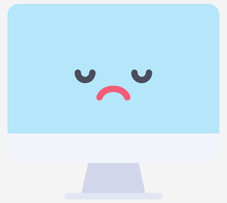
Express Via Big O Notation



Space Complexity

- Measures how the memory usage of an algorithm grows as the input size increases.
- Important for understanding the memory efficiency of an algorithm.

Big O Notation



@Sandip Das

Big O notation helps you understand how the time (like processing time) or resources (like memory) required by an algorithm scale with the size of the input.

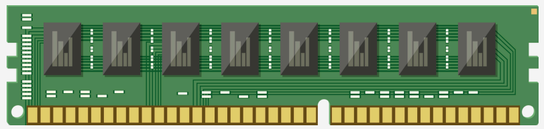
e.g. time and resources to process 1 GB log file is bigger than 1 MB logs file

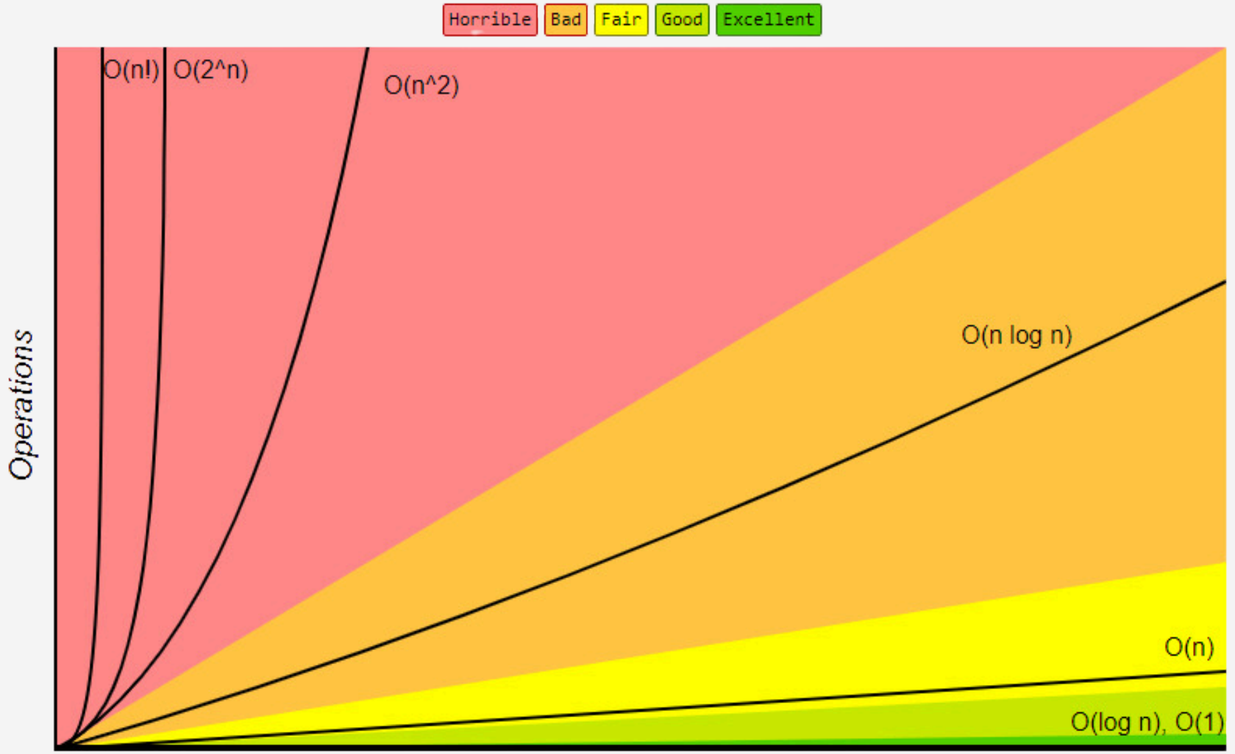
1 MB

1 MB logs processing takes Less Time and Memory

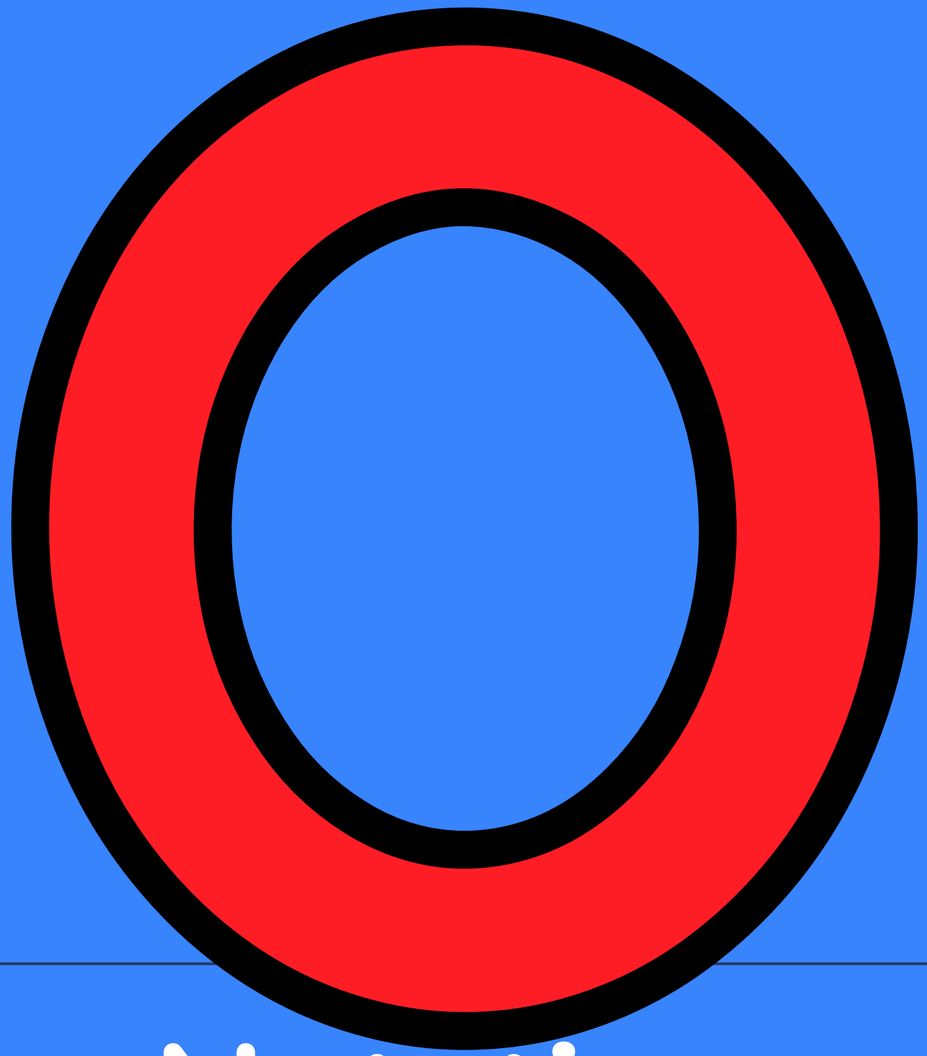
1 GB

1 GB Logs Processing takes more time and memory





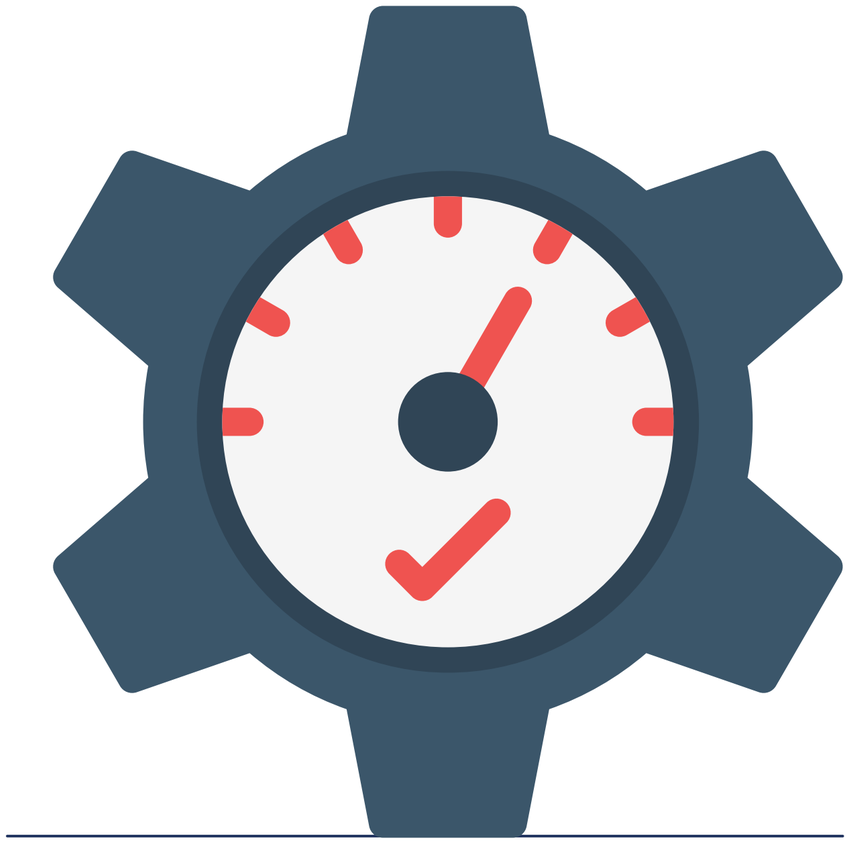
Common Big



Notations

@Sandip Das

O(1) - Constant Time

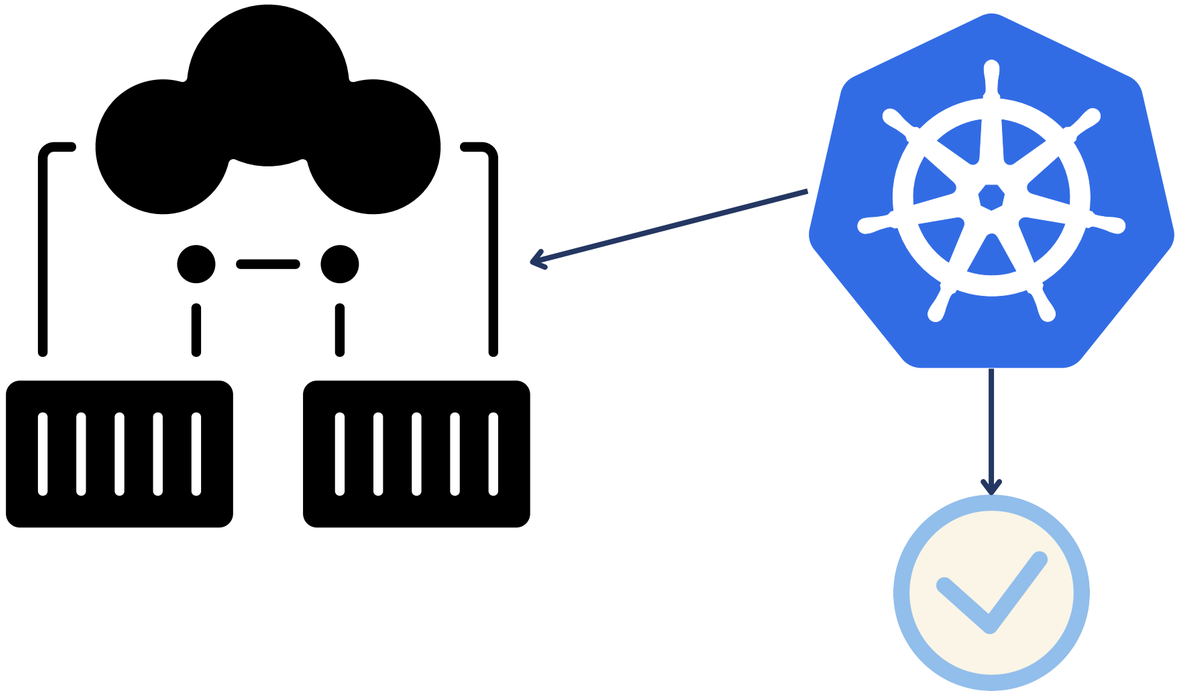


What doesn it mean ?

Execution time remains constant regardless of input size.

Example

Checking the status of a specific service in Kubernetes. The time it takes doesn't depend on the number of services running. It's always the same.




@Sandip Das

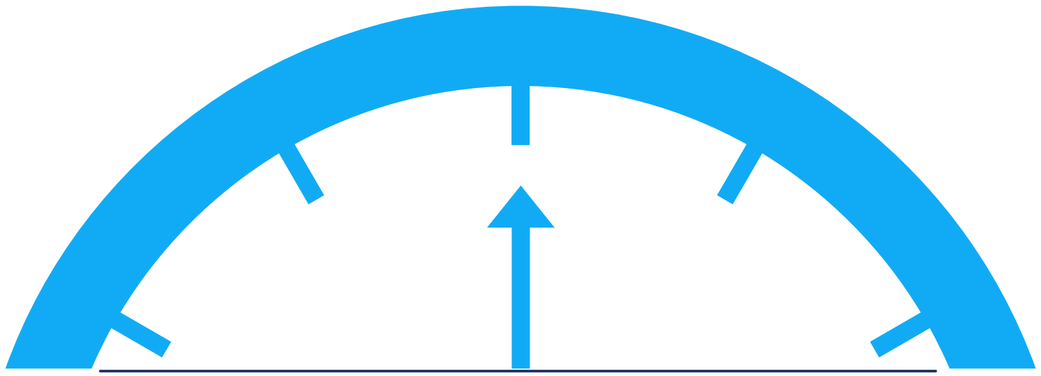
O(log n) - Logarithmic Time

What doesn it mean ?

If you can divide the search space in half each time, you quickly zero in on the entry. The time grows slowly as the log file grows.

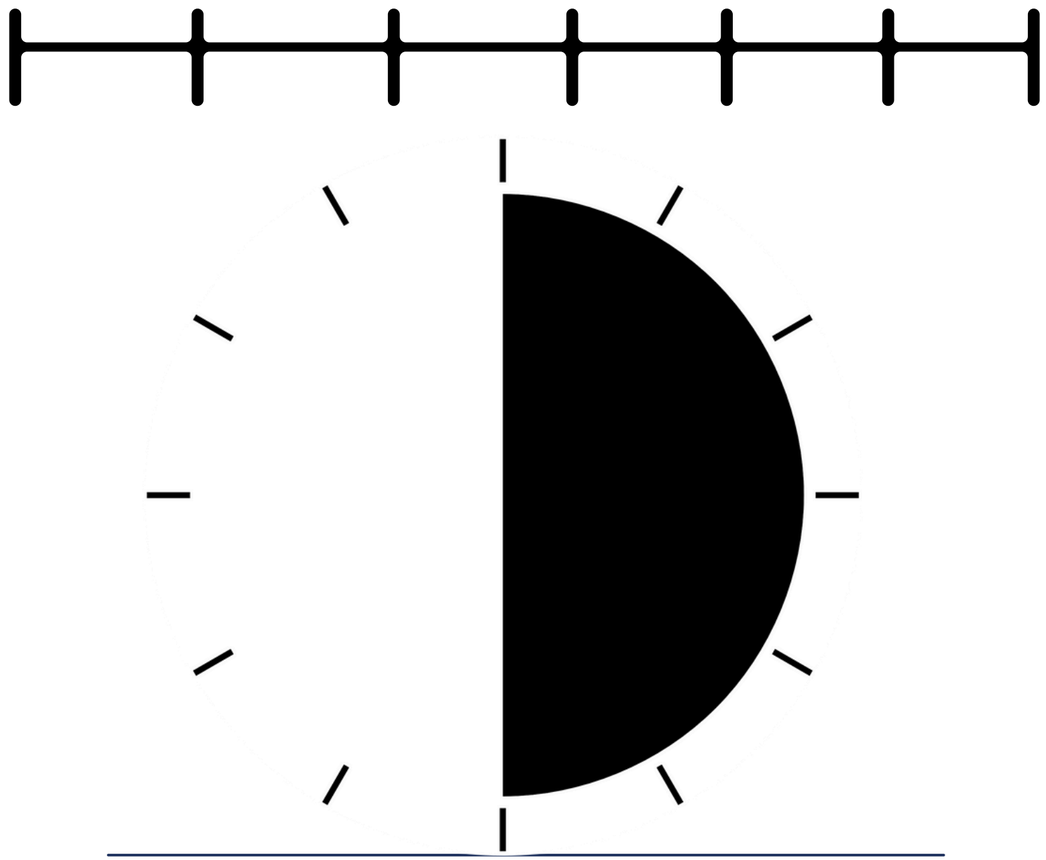
Example

Searching for a log entry in a sorted log file using binary search



@Sandip Das

O(n) - Linear Time

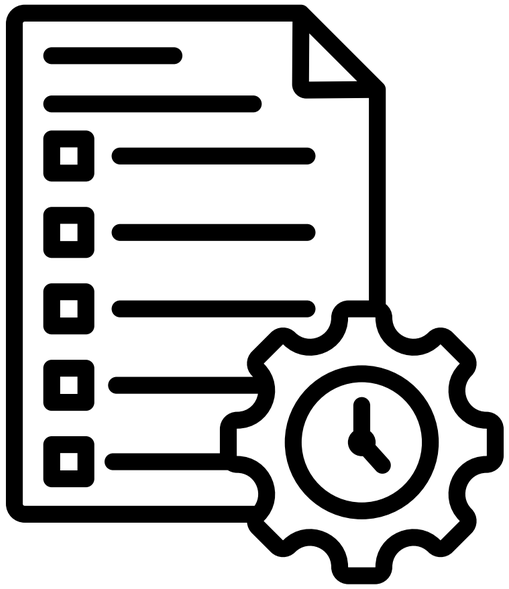


What doesn it mean ?

Execution time grows linearly with the input size

Example

Parsing through a log file line by line.
The time it takes grows directly with the number of lines in the log file. If you have twice as many lines, it takes twice as long




@Sandip Das

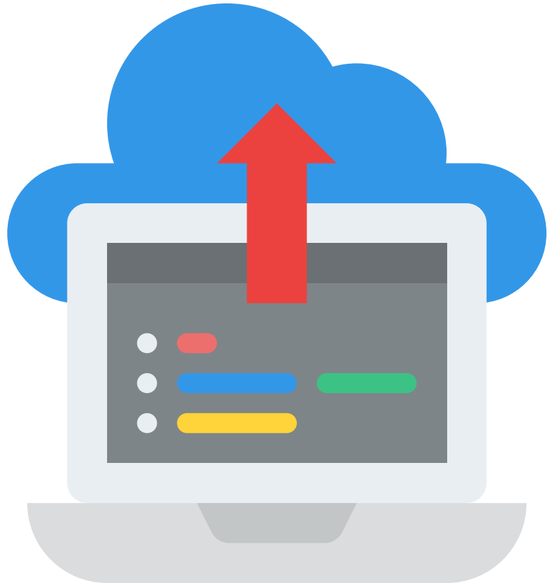
O(n log n) - Linearithmic Time

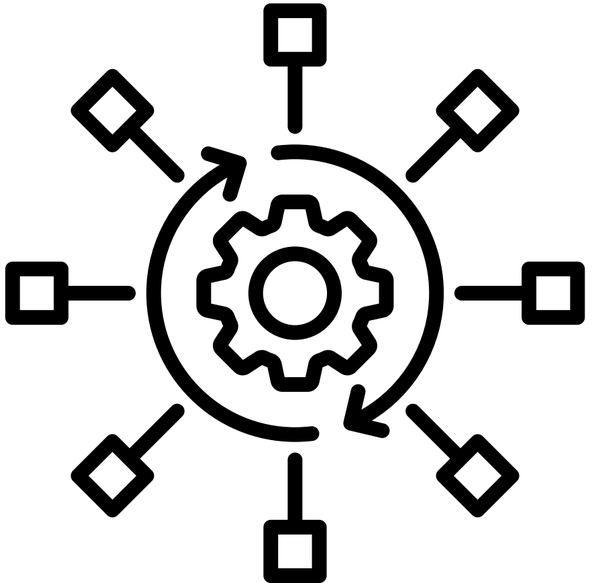
What doesn it mean ?

Execution time grows in proportion to grows linearly and logarithmically(n log n)

Example

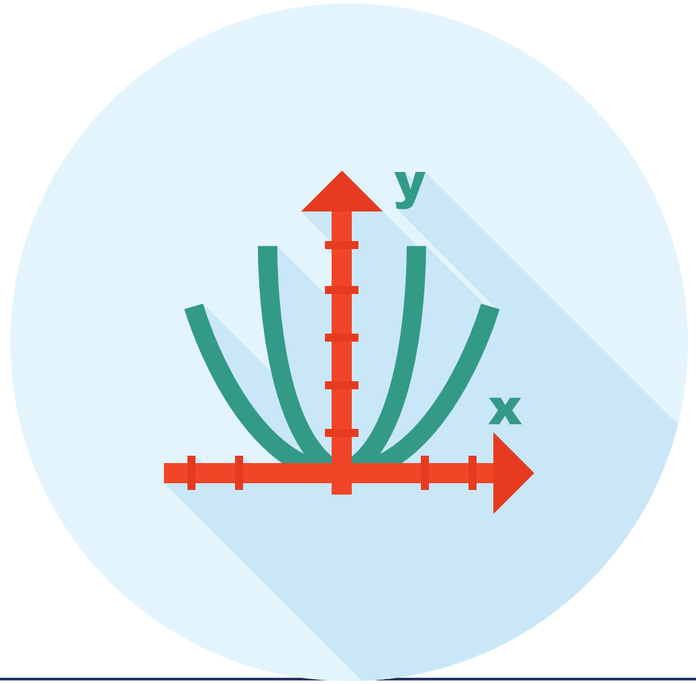
Sorting a list of deployment timestamps.
Efficient sorting algorithms, like merge sort, handle this. They are faster than simple sorts but still slow down as the list gets bigger






@Sandip Das

O(n^2) - Quadratic Time



What doesn it mean ?

Execution time grows quadratically with the input size.

Example

Comparing each log entry with every other entry to find duplicates. The time it takes grows much faster as the number of log entries increases. With double the entries, it takes four times as long.


```
[2024-06-17 10:15:23] ERROR: Unable to connect to database. Connection timeout.
  at DatabaseConnector.connect(DatabaseConnector.java:47)
  at UserService.getUserData(UserService.java:25)
  at MainApp.main(MainApp.java:14)

[2024-06-17 10:15:24] ERROR: Unable to connect to database. Connection timeout.
  at DatabaseConnector.connect(DatabaseConnector.java:47)
  at UserService.getUserData(UserService.java:25)
  at MainApp.main(MainApp.java:14)
```

Thank You

@LearnTechWithSandip

SUBSCRIBE

For staying till the end

Follow on: Sandip Das