***Time Complexity And Space Complexity***

***Time Complexity:***

***O(N) --> Linear time complexity*** (which means run time will be directly proportional to N(input size) as my input increasing the time also increasing)

Run Time directly proportional to the input size

***O(1) --> Constant time complexity*** (No matter if input size is keep on increasing also the time taken to execute the progam will be same)

Run Time directly proportional to 1

example:

--> If I run a algorithm with 100 inputs the time taken to execute the program will be 2 secs even I run the algorithm with 10000 inputs time taken will be same 2 secs

***O(N^2) --> Quadratic Time Complexity*** (If the input size increase two times the time complexity will be increased to 4 times)

Run Time directly proportional to N^2 (N power 2)

***O(log n) --> Logarithmic time Complexity***

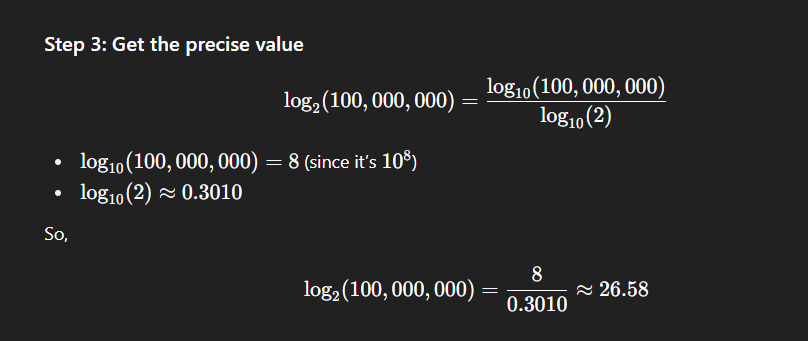
examples:

If the input size is 100 million data

***O(N) ->*** it will be doing 100 million operation

***O(1) ->*** It will be doing only fixed operation

***O(logn)***: it will take 26 operation for 100 million



***O(N^2)*** : input size is 100 million(10^8) now time complexity takes (10^16)

Ranking:

O(1) -> 1

O(log n) -> 2

O(N) -> 3

O(N^2) -> 4

***Python Data Structure Details:***

| **Data Structure** | **Access** | **Search** | **Insert** | **Delete** | **Notes** |
| --- | --- | --- | --- | --- | --- |
| **Array (static)** | O(1) | O(N) | O(N) | O(N) | Fixed size, contiguous memory |
| **List (dynamic array)** | O(1) | O(N) | O(N)\* | O(N) | Amortized O(1) append, resizing may cost O(N) |
| **Tuple** | O(1) | O(N) | – | – | Immutable list |
| **Dict (HashMap)** | O(1) | O(1) | O(1) | O(1) | Worst case O(N) due to collisions |
| **Set (HashSet)** | O(1) | O(1) | O(1) | O(1) | Similar to dict (keys only) |
| **Deque** | O(1) | O(N) | O(1) (ends) | O(1) (ends) | Best for queues & stacks |
| **Heap (Min-Heap)** | O(1) (min) | O(N) | O(log N) | O(log N) | Priority queue |
| **Linked List** | O(N) | O(N) | O(1) (front) | O(1) (front) | Not built-in, custom |