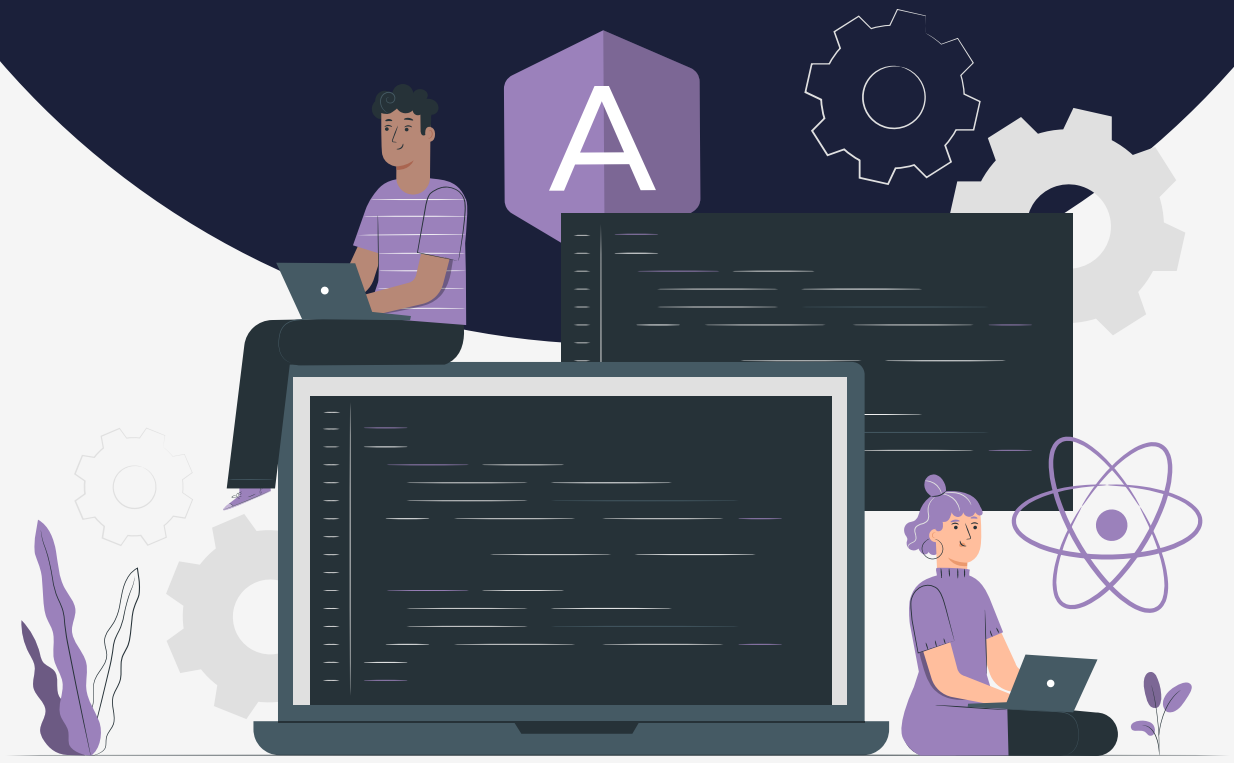
Lesson:

Problems on Time Complexity



Pre-Requisites

- Time and Space Complexity

List of Concepts Involved

- Problems on Time Complexity

Topic 1: Time Complexity Problems

Problem 1:
Calculate the time complexity for the following code snippet.

```
int val = 0;
for(int i = 1; i ⩽ N; i += i){
    val++;
}
```

Explanation:
We will calculate the total number of iterations in the above loop to calculate the time complexity. Total number of iterations will be:
The values of i in the loop:
$1, 2, 4, 8, 16, \ldots 2^k \leqslant N$
$k \sim \log N$
Hence the time complexity will be the total number of iterations that will be k giving us a time complexity of $O(\log N)$.

Problem 2:
Calculate the time complexity for the following code snippet.

```
int val = 0;
for(int i = 1; i ⩽ N; i *= 2){
    for(int j = 1; j ⩽ i; j++) {
        val++;
    }
}
```

Explanation:
Here in this case we calculate the number of iterations in the given nested loops, to calculate this we just calculate the number of times j has iterated.
Let there be k times the ith loop is iterated, then we know that
$2^k \leqslant N$ ... **equation (i)**
For the second loop, the jth pointer will move,
$1 + 2 + 4 + 8 + \ldots + 2^{k-1}$ times then
The total sum for the above Geometric progression will be 2^{k-1}. ... **equation (ii)**
Here from equation (i), we get that $k = \log N$,

So the total time complexity from equation (ii) is sum of iterations will be 2^(k) where k = logN giving a worst case time complexity of $O(2^{(\log N)}) \sim O(N)$.

Problem 3:
Calculate the time complexity for the following code snippet.

```
int val = 0;
for(int i = 1; i ⩽ N; i *= 2){
    for(int j = N; j > i; j--) {
        val++;
    }
}
```

Explanation:
Here we calculate the number of iterations j will take, let's say the number of iterations in i^{th} loop be k, then here $2^k \leqslant N$, the value of k will come out to be **k = logN**.

The number of iterations in jth loop will be
$(N-1) + (N-2) + (N-4) + \ldots + (N-2^{k-1})$
$= k*N - (1+2+4+8+\ldots+2^{k-1})$
$= k*N - (2^k - 1)$

Putting **k = logN**, we get
=> **(NlogN - N)** iterations

Final Time Complexity: **O(N logN)**

Problem 4:
Calculate the time complexity for the following code snippet.

```
int val = 0;
for(int i = 2; i ⩽ N; i *= i){
    val++;
}
```

Explanation:
To calculate the time complexity for the following code snippet, we will calculate the total number of iterations. Let us first analyze the values of **i** in the above loop,
$2, 4, 16, 256, \ldots$
$2, 2^2, 2^4, 2^8, 2^{16}, \ldots$

Let the total above terms in **i** be **k**,
Then values of **i** become
$2, 2^2, 2^4, 2^8, 2^{16}, \ldots, 2^k$

here $2^k < n$, $k \sim \log n$,
also we note that k is also getting incremented in powers of 2, let the total number of iterations be t then $2, 4, 8, 16, \ldots 2^t \leqslant k$
hence **t = logk**

overall time complexity will be the total number of iterations ie, **t = logk,**
as we know **k = logn, t = log(logn)**
hence the overall time complexity becomes **o(log(logn))**.

Upcoming Class Teasers

- Bubble sort