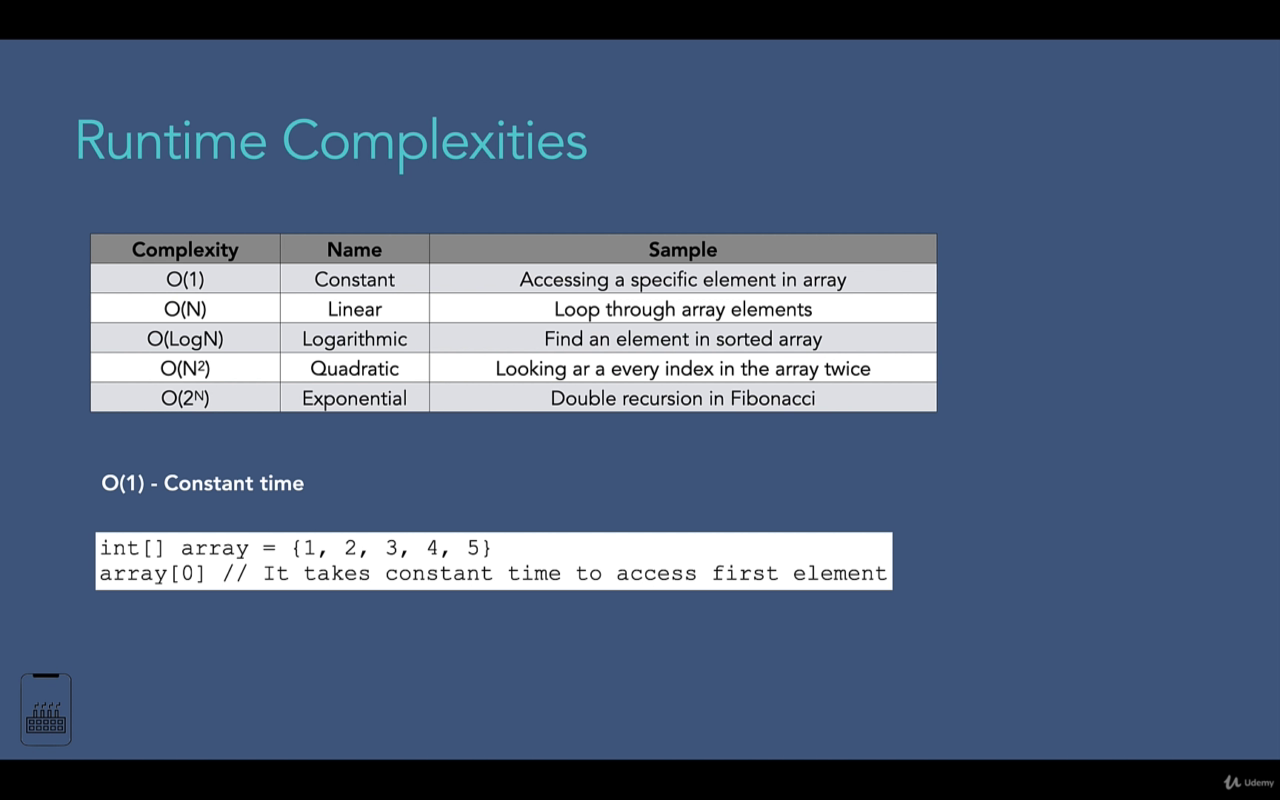
**Big O Notation**

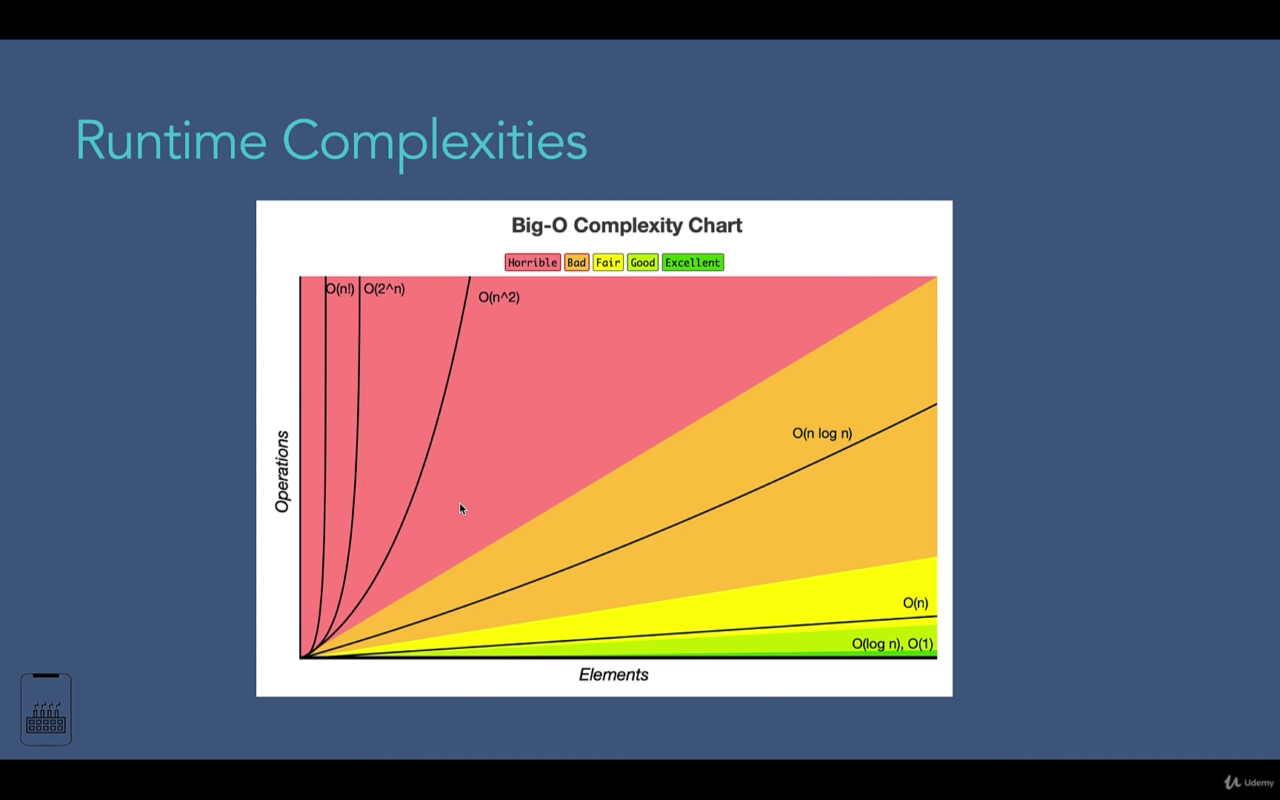
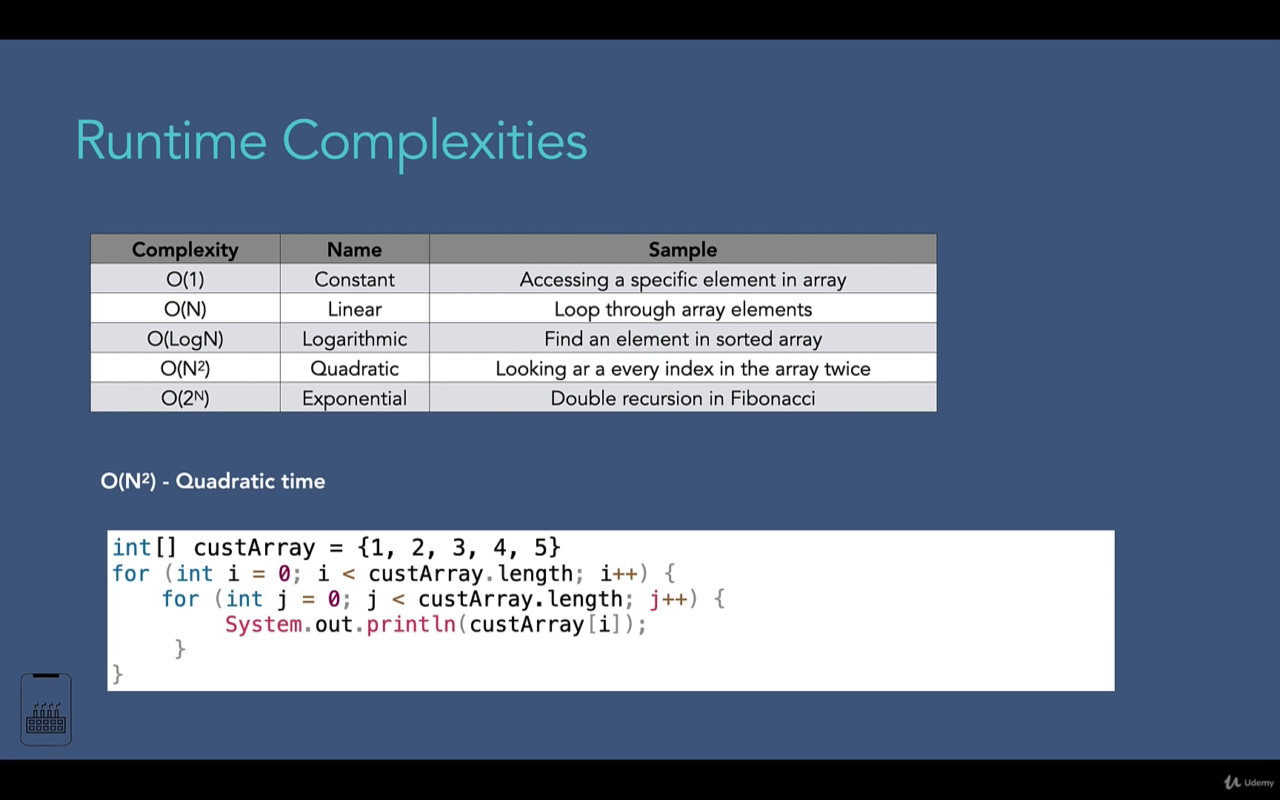
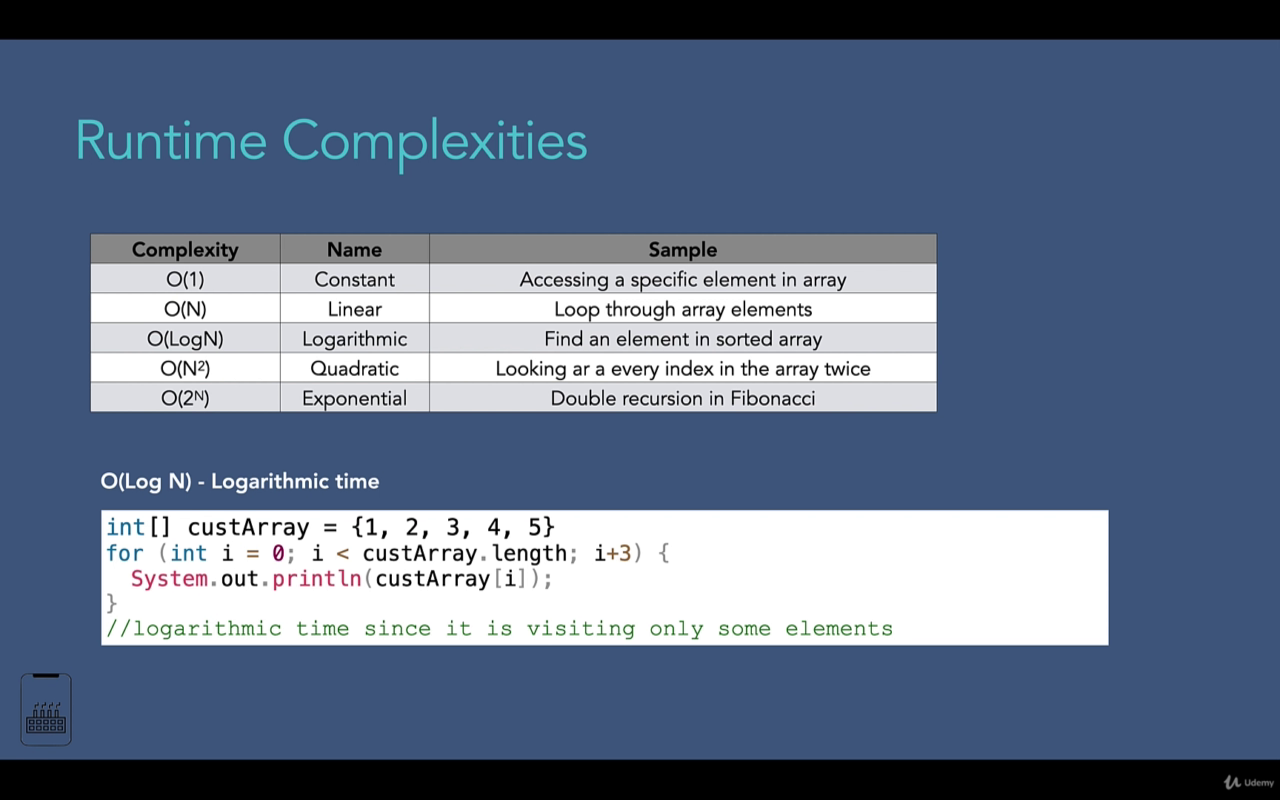
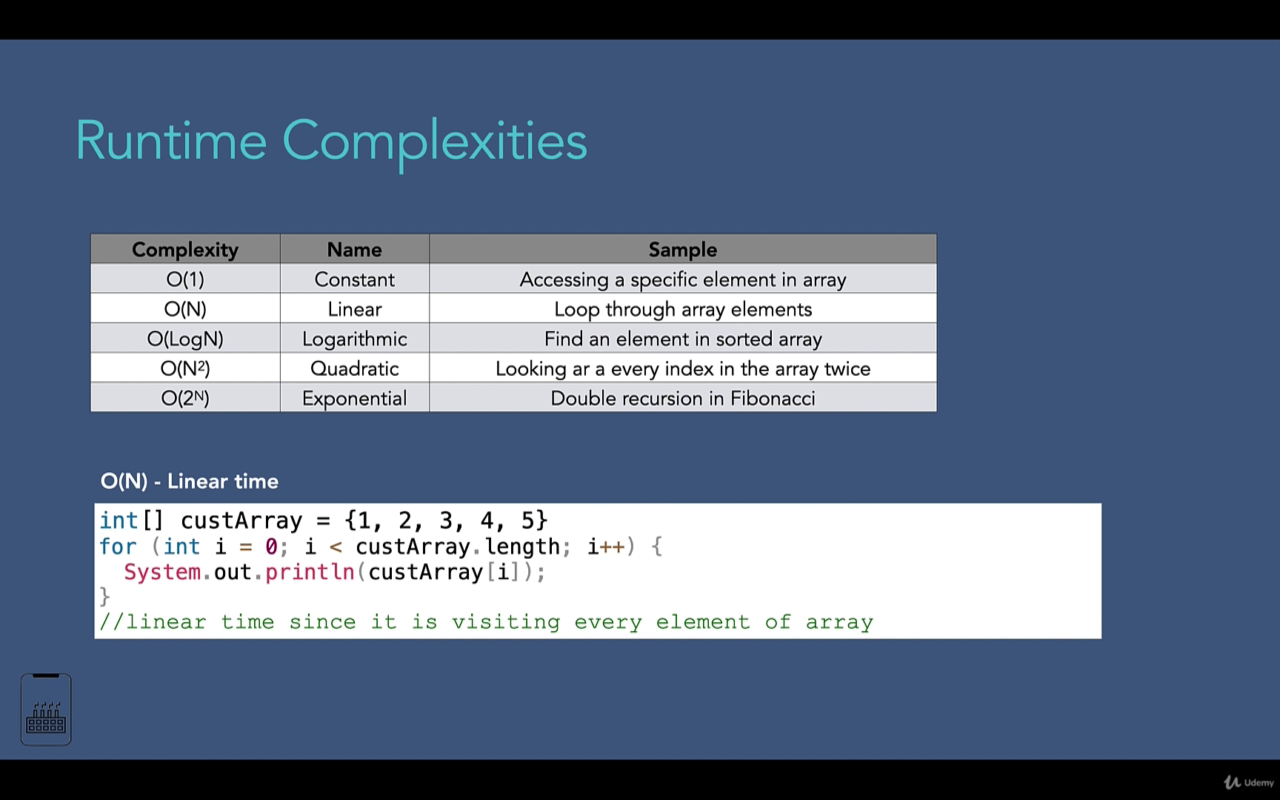
A language and metric used to describe the efficiency of algorithms. How does time and space taken to run a program increase as the number of inputs increase.

* **Time Complexity:** shows how the runtime of a function increases as the size of input increases.

**Algorithms** perform differently based on conditions. Three case scenarios in measuring performance of any given algorithm

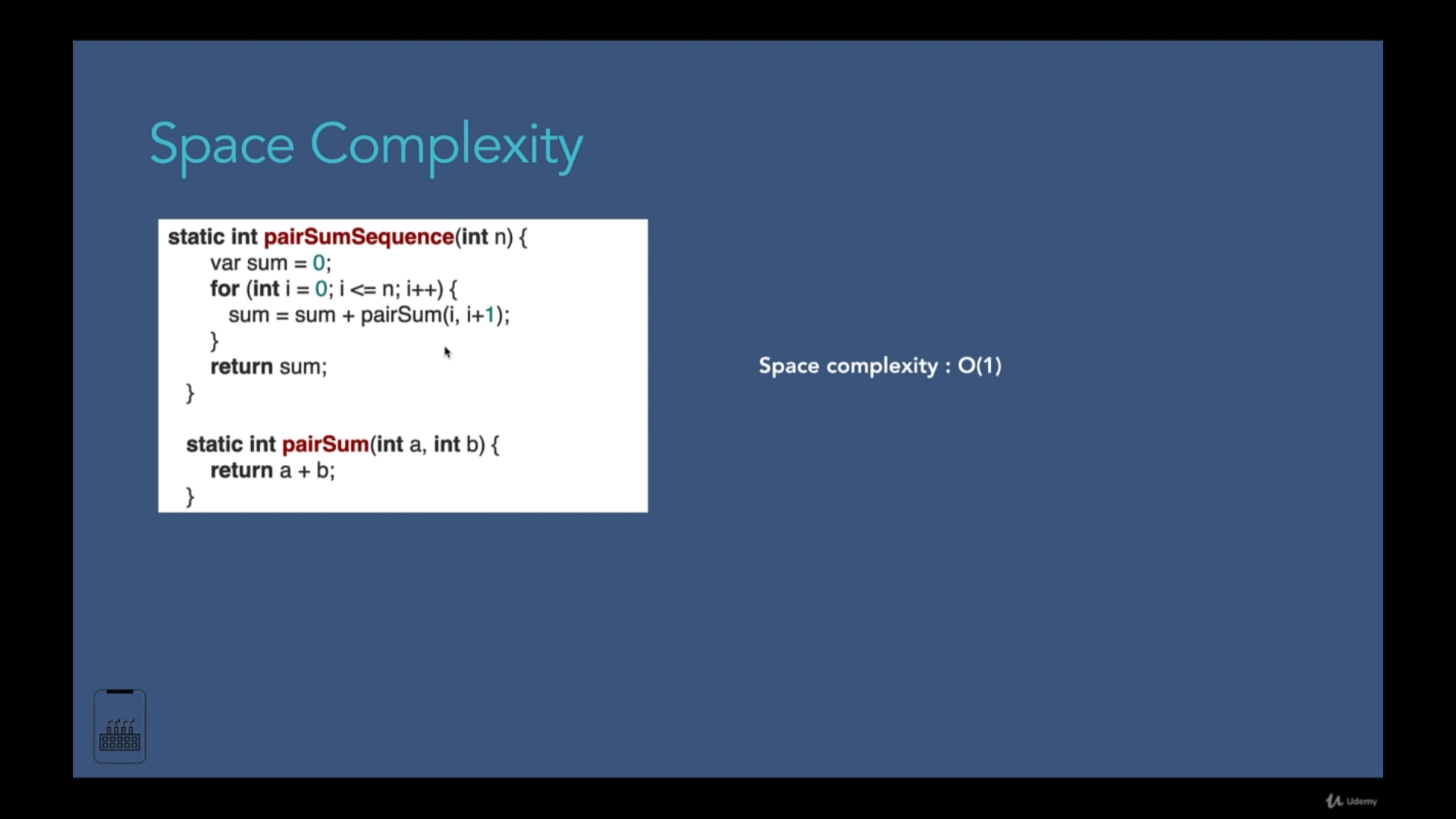
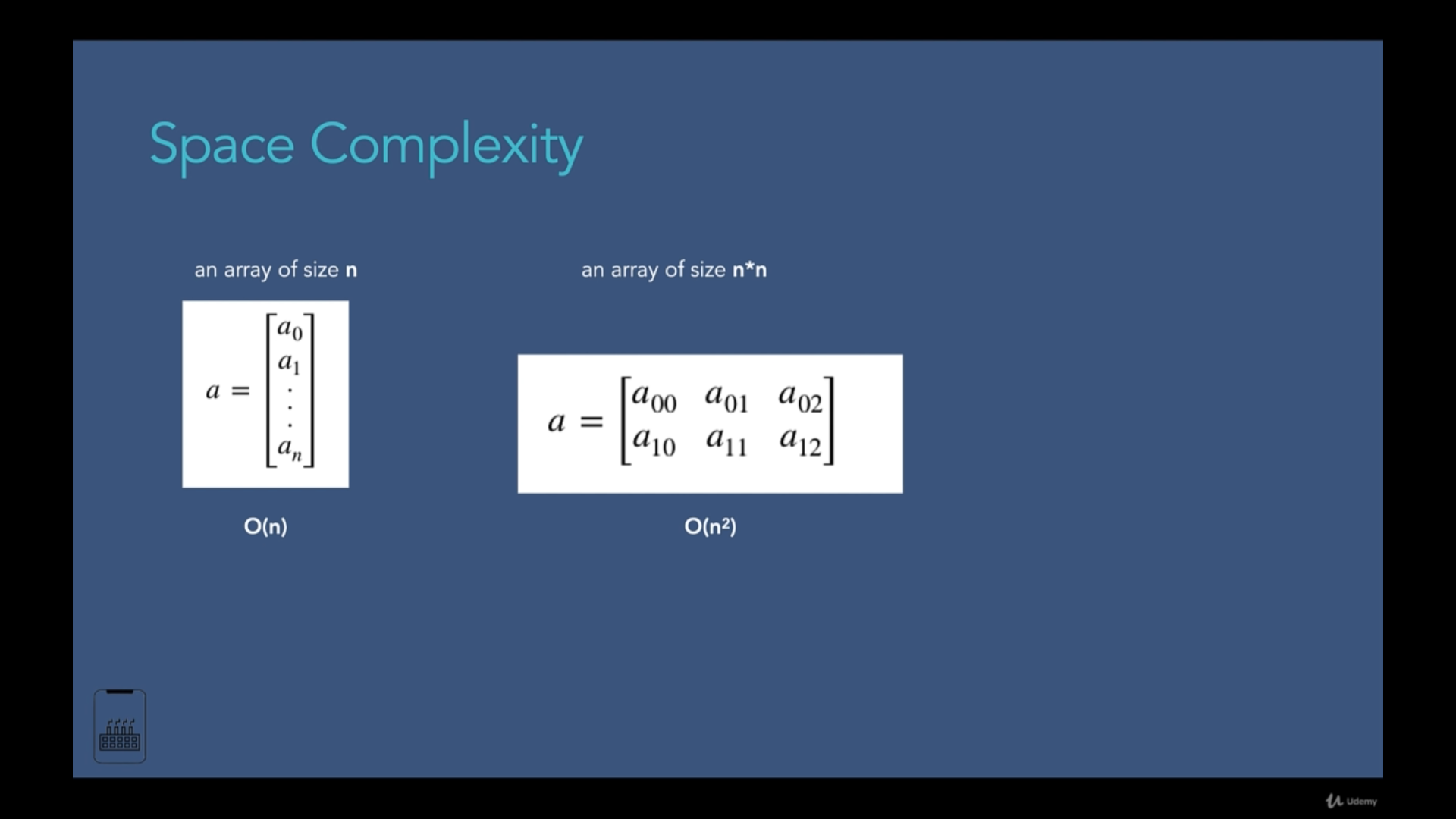
1. Best Case
2. Average Case
3. Worst Case





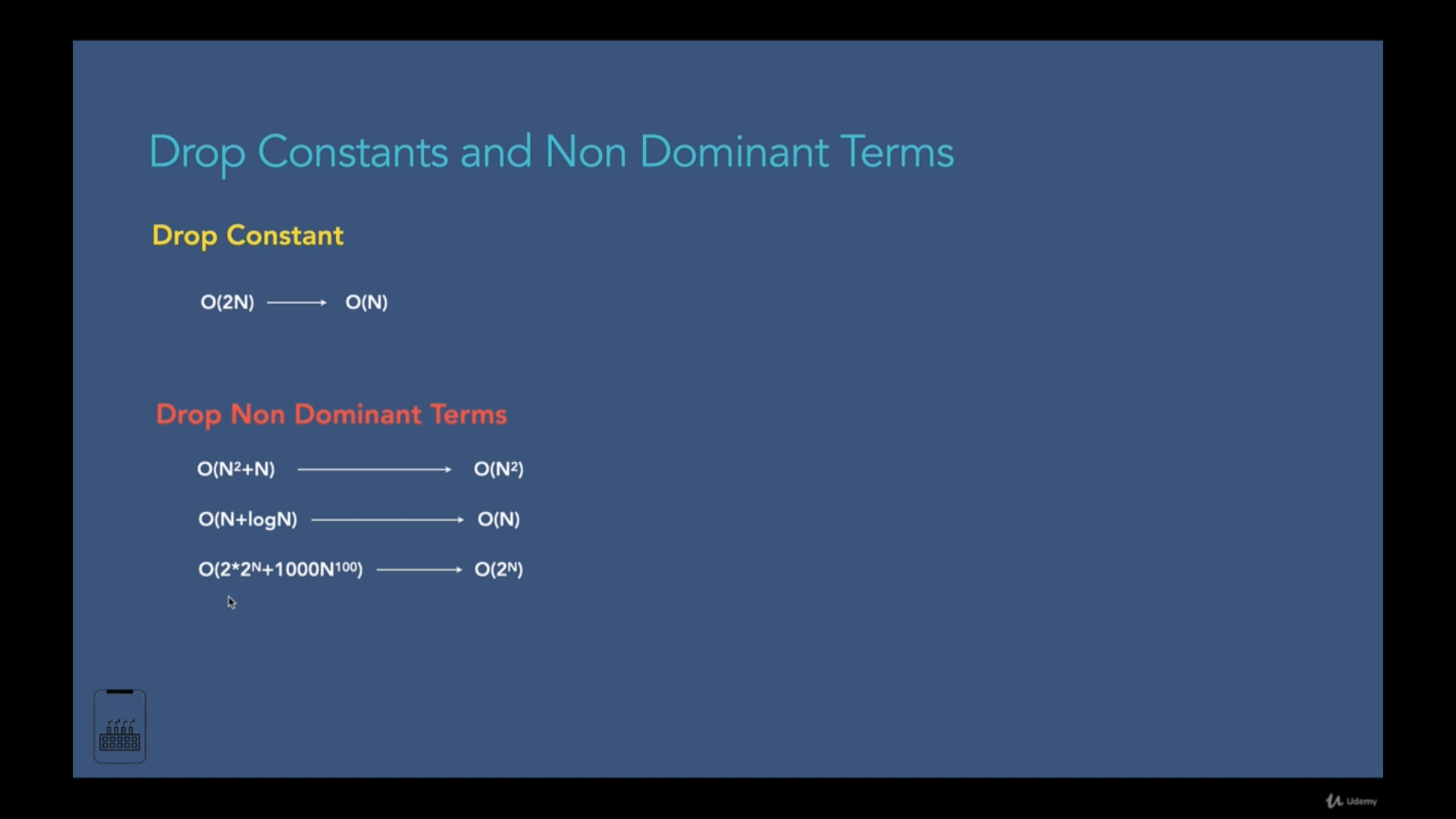
**Time Complexity**

Refers to the amount of memory an algorithm needs at worst case. How does space grows, as the number inputs grows?



**Drop Constants and Non Dominant Terms**

* Drop constant in big O notation because it has no effect. These constant would mean external factors like computer capability, and we are only interested in algorithm only.
* Non Dominant Terms, we pick the bigger terms. Ex O(n+log(n)) - > **O(n)** because, O(log(n) performs better than O(n) and we are always looking at the **worst case.**



**Why**

