TIME & SPACE COMPLEXITY

LESSON 13

CODE FIRST GIRLS

NANODEGREE → ENGINEERING MODULE

AGENDA

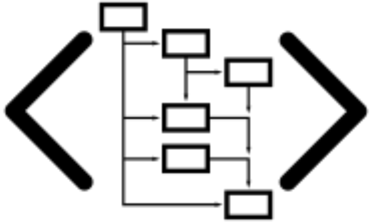
01 Complexity Analysis

02 Big O Notation

03 Practice and coding

COMPLEXITY ANALYSIS

INTRODUCTION



COMPLEXITY ANALYSIS

*The process of determining how **efficient** an algorithm is. Complexity analysis usually involves finding both the **time** complexity and the **space** complexity of an algorithm."*

- Complexity analysis is effectively used to determine how 'good' an algorithm is and whether it's "better" than another one.

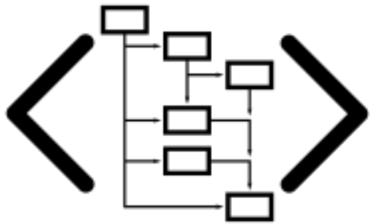
TIME SPACE COMPLEXITY

- **Time complexity** - a measure of how fast an algorithm runs.
- **Space complexity** - a measure of how much auxiliary memory an algorithm takes up.

- Time and space complexities are **central concepts** in the field of algorithms and in coding interviews.
- Time and space complexity is expressed using **“Big O notation”**.

BIG O NOTATION

INTRODUCTION

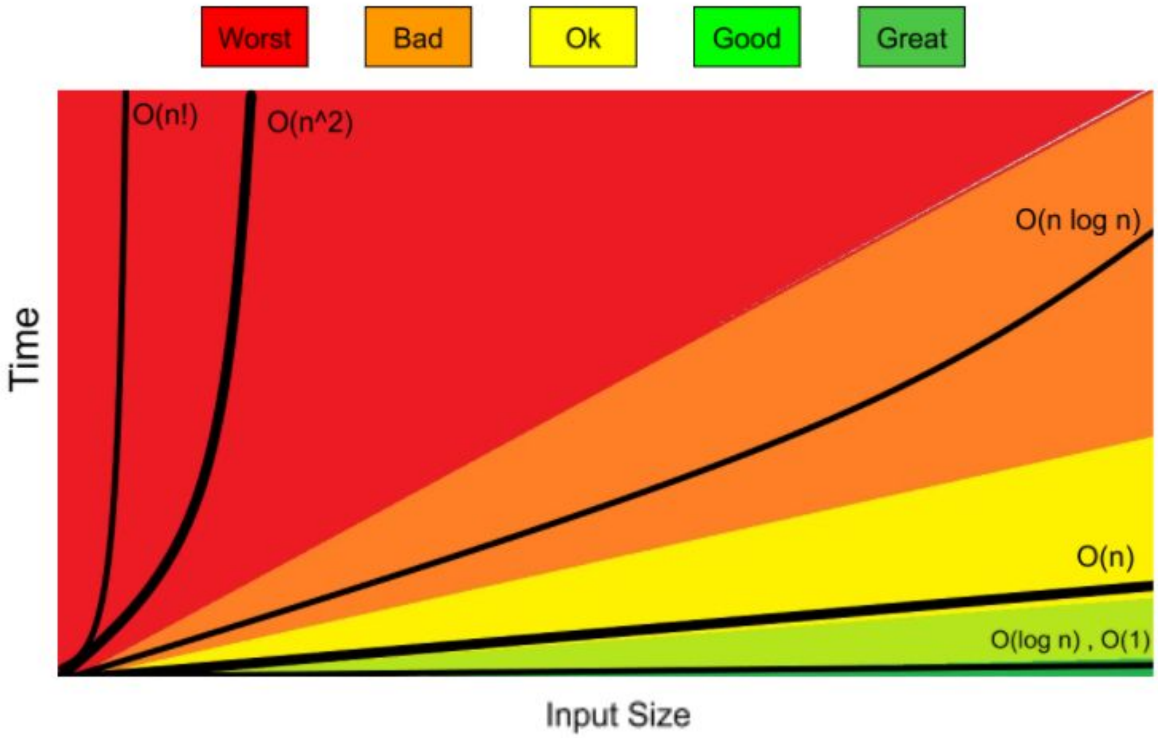


BIG O NOTATION

- The notation is used to describe the time complexity and space complexity of algorithms.
- Variables used in **Big O notation** denote the sizes of inputs to algorithms.
- For example **$O(n + m)$** might be the time complexity of an algorithm that traverses through an array of length **n** and through a string of length **m**.

- **Constant:** $O(1)$
- **Logarithmic:** $O(log(n))$
- **Linear:** $O(n)$
- **Log-linear:** $O(nlog(n))$
- **Quadratic:** $O(n^2)$
- **Cubic:** $O(n^3)$
- **Exponential:** $O(2^n)$
- **Factorial:** $O(n!)$

BIG O NOTATION



DEMO & EXERCISES

ALGORITHMS
EXERCISES & PRACTICE

THANK YOU!