Basic Algorithmic Analysis, Asymptotic Analysis of Upper and Average Complexity Bounds

Instructor: Kayode Oladapo Education: Ph.D in Computer Science
Email: oladapoka@mcu.edu.ng Website: https://sites.google.com/view/kayodeabiodunoladapo

KNOWLEDGE
INTEGRITY
SERVICE
McPHERSON UNIVERSITY

2023

Intended Learning Outcomes

- Understand runtime and space analysis or complexity of algorithms
- Know the different types of analysis
- Understand the typical complexities of an algorithm
- Know how to approximate the time taken by an algorithm

Analysis of Algorithm

Analysis of an algorithm is the same thing as estimating the efficiency of the algorithm. There are two fundamental parameters based on which we can analyze the algorithm and they are Space and Time Complexity.
There is also the concept in Time Complexity of estimating the running time of an algorithm and we have the Best-case, Average-case and Worst-case

SPACE COMPLEXITY ↔ Analysis of Algorithm ↔ Time Complexity

The space complexity can be understood as the amount of space required by an algorithm to run to completion.

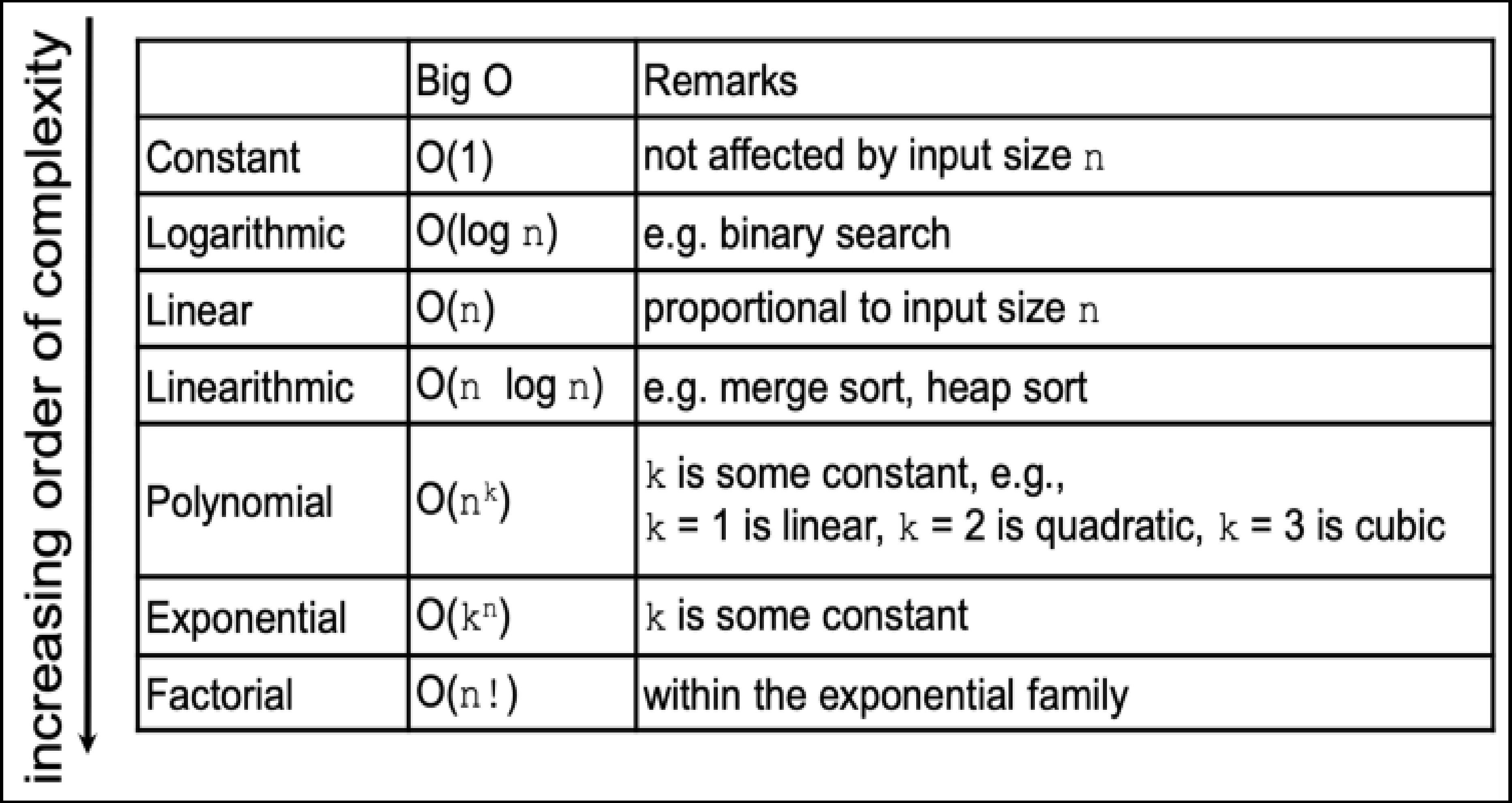
The analysis is a process of estimating the efficiency of an algorithm and that is, trying to know how good or how bad an algorithm could be.
There are two fundamental parameters based on which we can analyze the algorithm and they are Space and Time Complexity.

Time complexity is a function of input size **n** that refers to the amount of time needed by an algorithm to run to completion.

Types of Time Complexity Analysis

- Worst-Case Time Complexity
- Average Case Time Complexity
- Best Case Time Complexity

Complexity of Algorithms



increasing order of complexity

	Big O	Remarks
Constant	$O(1)$	not affected by input size n
Logarithmic	$O(\log n)$	e.g. binary search
Linear	$O(n)$	proportional to input size n
Linearithmic	$O(n \log n)$	e.g. merge sort, heap sort
Polynomial	$O(n^k)$	k is some constant, e.g., $k = 1$ is linear, $k = 2$ is quadratic, $k = 3$ is cubic
Exponential	$O(k^n)$	k is some constant
Factorial	$O(n!)$	within the exponential family

How to Approximate the time taken by the Algorithm (With Examples)

- **Iterative Algorithm:** In the iterative approach, the function repeatedly runs until the condition is met or it fails. It involves the looping construct.
- **Recursive Algorithm:** In the recursive approach, the function calls itself until the condition is met. It integrates the branching structure.

Conclusion

Analysis of algorithms helps us to determine how good or how bad they are in terms of speed or time taken and memory or space utilized. Designing good programs is dependent on how good or how bad the algorithm is and the analysis helps us to determine the efficiency of such algorithms.

Thank You, See you next week