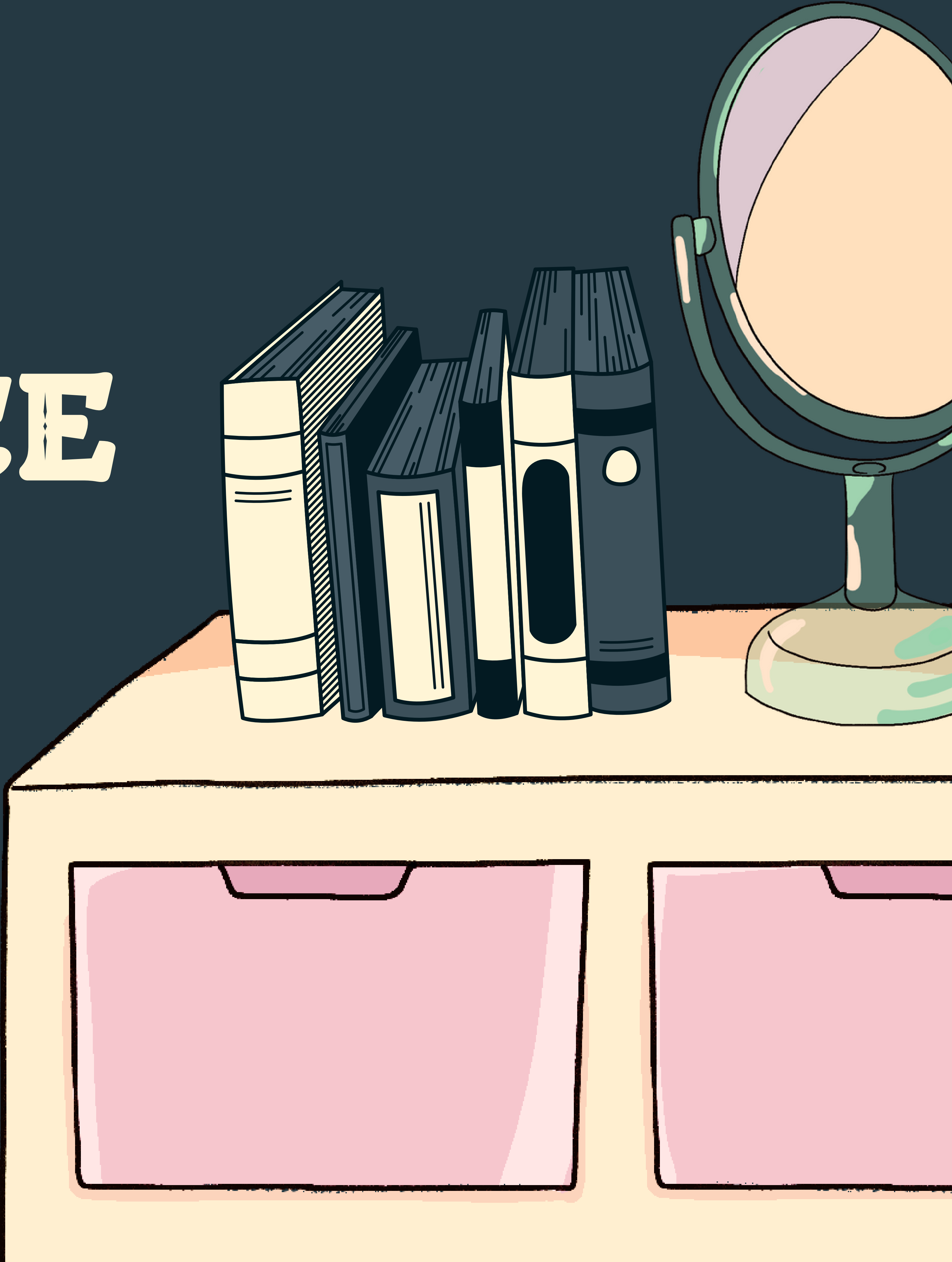
DAY-1
DSA IN JAVA

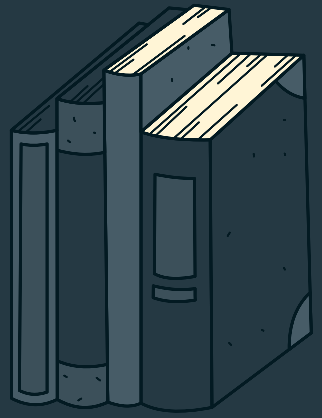
@programming_with_jS

TIME AND SPACE COMPLEXITY



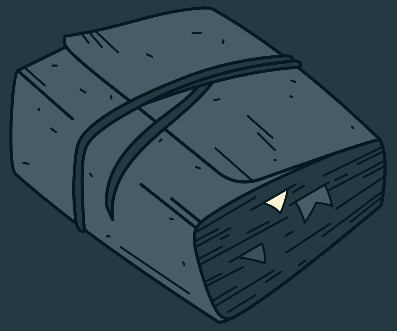
@programming_with_jS

WHAT IS TIME COMPLEXITY?

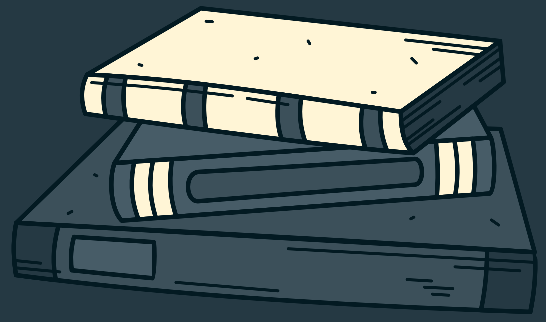


INTRODUCTION

WHY IT IS NECESSARY TO UNDERSTAND TIME COMPLEXITY



iUNDERSTANDING THE CONCEPT



QUESTIONS

@programming_with_jS

INTRODUCTION:

Algorithms are more efficient than others. We would prefer to chose an efficient algorithm, so it would be nice to have metrics for comparing algorithm efficiency. There are two main complexity measures of the efficiency of an algorithm:

1. Time complexity is a function describing the amount of time an algorithm takes in terms of the amount of input to the algorithm. In layman's terms, We can say time complexity is sum of number of times each statements gets executed.

2. Space complexity is a function describing the amount of memory (space) a n algorithm takes in terms of the amount of input to the algorithm. When we say "this algorithm takes constant extra space," because the amount of extra memory needed doesn't vary with the number of items processed.

TIME COMPLEXITY:WHY

Time complexity of an algorithm quantifies the amount of time taken by an algorithm to run as a function of the length of the input.

Types of notations

1. O-notation: It is used to denote asymptotic upper bound. For a given function g(n), we denote it by O(g(n)). Pronounced as “big-oh of g of n”. It is also known as worst case time complexity as it denotes the upper bound in which the algorithm terminates.
2. Ω-notation: It is used to denote asymptotic lower bound. For a given function g(n), we denote it by Ω(g(n)). Pronounced as “big-omega of g of n”. It is also known as best case time complexity as it denotes the lower bound in which the algorithm terminates.
3. ϴ-notation: It is used to denote the average time of a program.

@programming_with_jS

Comparison of functions on the basis of time complexity

It follows the following order in case of time complexity:

O(nn) > O(n!) > O(n3) > O(n2) > O(n.log(n)) > O(n.log(log(n))) > O(n) > O(sqrt(n)) > O(log(n)) > O(1)

Note: Reverse is the order for better performance of a code with corresponding time complexity, i.e. a program with less time complexity is more efficient.

@programming_with_jS

SPACE COMPLEXITY:

Space complexity of an algorithm quantifies the amount of time taken by a program to run as a function of length of the input. It is directly proportional to the largest memory your program acquires at any instance during run time.
For example: int consumes 4 bytes of memory.

@programming_with_jS