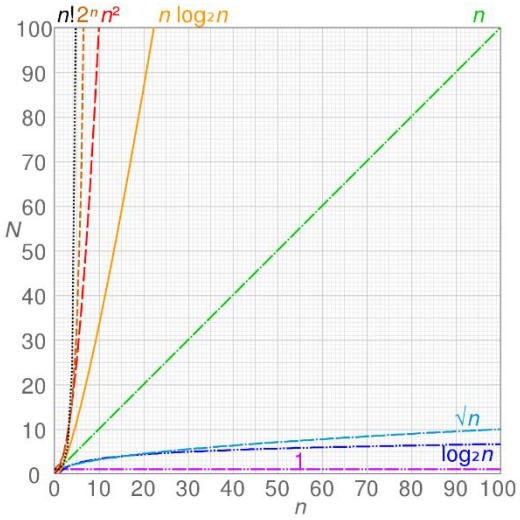
Section 2: Arrays & Big O Notation

Big O Notation

- **Time complexity** is the number of steps that it takes to run the algorithm
- Memory complexity is the amount of memory that it takes to run the algorithm
- Uses the worst-case scenario
- **Constant** - O(1)
- **Logarithmic** - O(log(n))
- **Linear** - O(n)
- **n log-star n** - O(n*log(n))
- **Quadratic** - O(n^2)



Arrays in Java

GitHub

Arrays in Memory

- One contiguous block in memory, they are not scattered
- Static
- Every element occupies the same amount of space in memory
- If an array starts at memory address **X**, and the size of each element in the array is **Y**, we can calculate the memory address of the **i^{th}** element by using the following expression (assuming you know the starting address): **X + i * Y**
- If we know the index of the element in the array, the time to retrieve the element will be the same no matter where it is in the array
- **NOTE:** useful for retrieving elements if we know the index, the time is the same for each element

Big O Values for Array Operations

Retrieve an Element from an Array

1. Multiply the size of the element by its index
2. Get the start address of the array
3. Add the start address to the result of the multiplication

Number of Elements	Steps to Retrieve
1	3
1000	3
100000	3
1000000	3
1000000000	3

Big O Value

- Time complexity is **constant** because the steps to retrieve an element is always three steps no matter the size of the array
- **O(1)**

Operation	Time Complexity
Retrieve with index	O(1) – Constant time
Retrieve without index	O(n) – Linear time
Add an element to a full array	O(n)
Add an element to the end of an array (has space)	O(1)
Insert or update an element at a specific index	O(1)
Delete an element by setting it to null	O(1)
Delete an element by shifting elements	O(n)

Operations

- In the operations if we must loop over the array to perform the operation, that's going to have a **linear** time complexity
- If we can calculate the index, instead of looping, then that will have a **constant** time complexity