Java Collections Framework

&

Time Complexity Of Operations

ArrayList

Use for fast random access and is mainly for storing and accessing data sequentially.

Great for scenarios where you frequently need to access elements by index and the order of elements is important.

Time Complexity

- get: O(1)
- add: O(1)
- remove: O(n)

LinkedList

Use for frequent insertions and deletions in the middle of the list.

Also useful when the order of elements is important.

Time Complexity

- get: O(n)
- add: O(1)
- remove: O(1)
- contains: O(n)

HashSet

Use when you need to store a unique set of elements and perform fast lookups for element existence.

It's great for checking for duplicates.

Time Complexity

- add: O(1)
- remove: O(1)
- contains: O(1)

TreeSet

Use when you need to store a sorted set of unique elements.

Time Complexity

- add: O(log n)
- remove: O(log n)
- contains: O(log n)

HashMap

Use when you need to store key-value pairs and quickly look up values based on keys.

Time Complexity

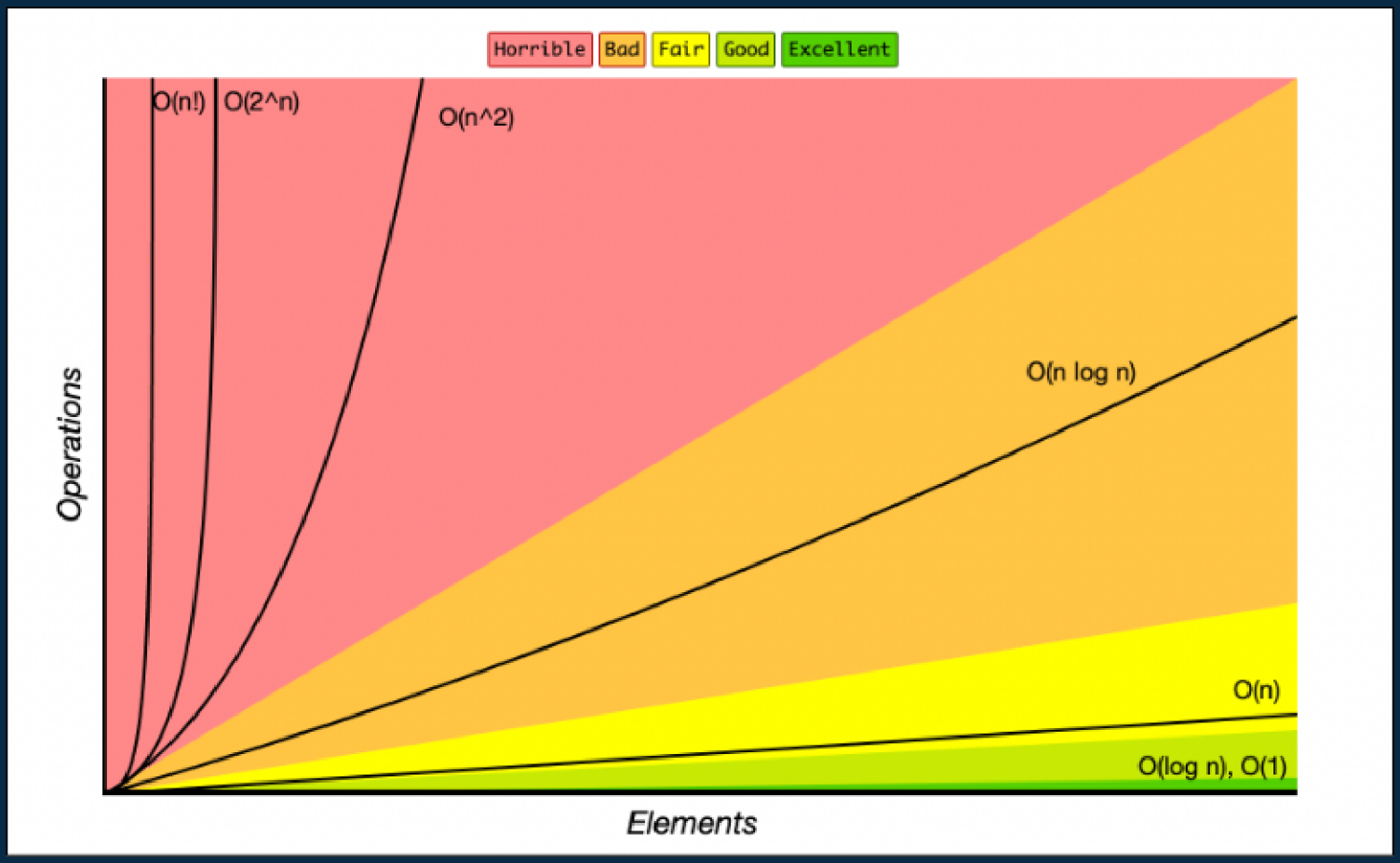
- put: O(1)
- remove: O(1)
- contains: O(1)

TreeMap

Use when you need to store key-value pairs in a sorted order based on the keys.

Time Complexity

- put: O(log n)
- remove: O(log n)
- contains: O(log n)



Big O Cheat Sheet – Time Complexity Chart

For more content like this Follow me on

LinkedIn