Learn by Examples

Mukesh Joshi

Table of Contents

Collection

Space Complexity

```
Process of determining a formula for prediction of how much memory space will be
required for the successdull ececution of the algorithm.
The memory space we generally consider is the space of primary memory.
```

Time Complexity

```
Process of determining a formula for total time required towards execution of the
algorithm.
This calculation will be independent of implementation details and programming
language.
```

O(1)/Constant Complexity

```
Constant.  This means irrelevant of the size of the data set the algorithm will always
take a constant time.
1 item takes 1 second, 10 items takes 1 second, 100 items takes 1 second. It always
takes the same amount of time.
```

O(log n)/Logarithmic Complexity:

```
Not as good as constant, but still pretty good.  The time taken increases with the
size of the data set,
but not proportionately so. This means the algorithm takes longer per item on smaller
datasets relative to larger ones.
1 item takes 1 second, 10 items takes 2 seconds, 100 items takes 3 seconds. If your
dataset has 10 items,
each item causes 0.2 seconds latency. If your dataset has 100, it only takes 0.03
seconds extra per item.
This makes log n algorithms very scalable.
```

O(n)/Linear Complexity:

```
The larger the data set, the time taken grows proportionately. 1 item takes 1 second,
10 items takes 10 seconds, 100 items takes 100 seconds.
```

O(n log n):

```
A nice combination of the previous two.  Normally there’s 2 parts to the sort, the
first loop is O(n),
the second is O(log n), combining to form O(n log n) 1 item takes 2 seconds, 10 items
takes 12 seconds,
100 items takes 103 seconds.
```

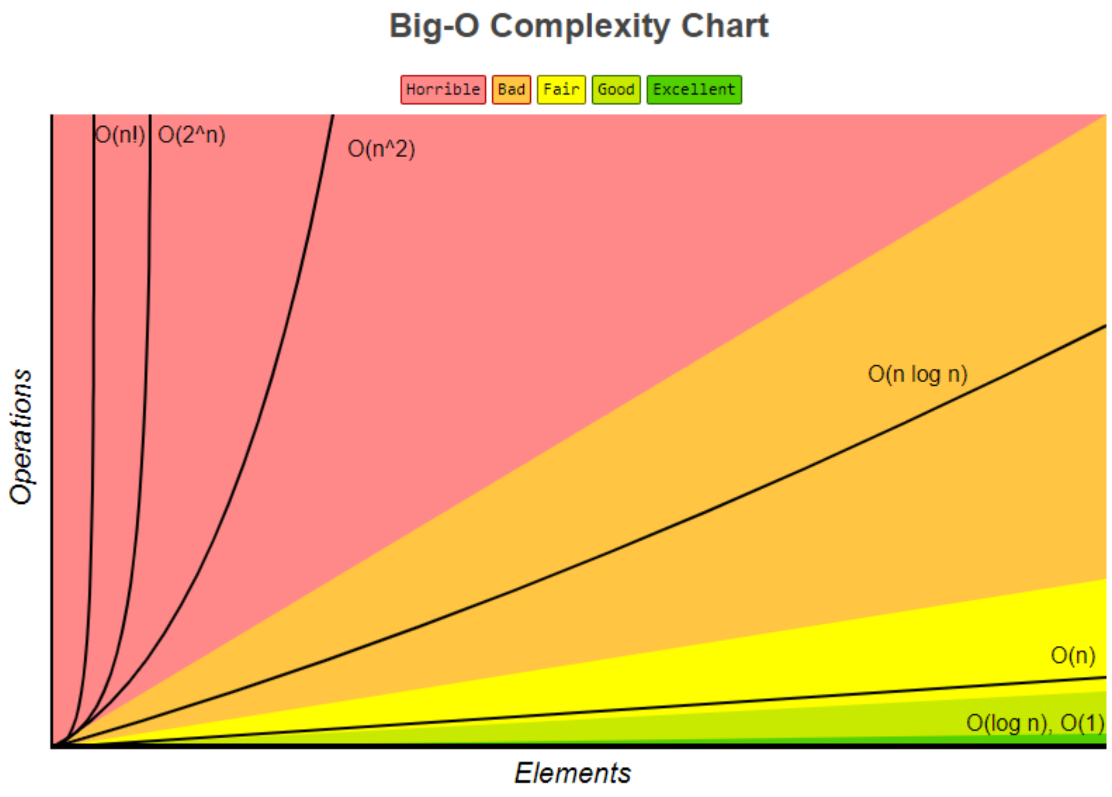

Figure 1. Complexity graph

Table 1. Collection complexity table:

Collection	Time complexity				Space complexity	Remark
	Add /offer /push /pull	Remove /poll /pop	Get /peek	Contains	All	
Array*	O(1)	O(n)	O(1)	O(n)	-	-
Vector	O(1)	O(n)	O(n)	O(n)	-	-
ArrayList	O(n)	O(n)	O(1)	O(n)	-	-
LinkedList	O(1)	O(1)	O(n)	O(n)	-	-
PriorityQueue	-	-	-	-	-	-
HashSet	O(1)	O(1)	O(1)	O(1)	-	-
LinkedHashSet	-	-	-	-	-	-
TreeSet	-	-	-	-	-	-
HashMap*	O(1)	O(1)	O(1)	O(1)	-	-
LinkedHashMap*	-	-	-	-	-	-
TreeMap*	O(log (n))	O(log (n))	O(log (n))	O(log (n))	-	-
Hashtable*	-	-	-	-	-	-

Reference:

- https://docs.oracle.com/javase/8/docs/technotes/guides/collections/overview.html
- https://adrianmejia.com/data-structures-time-complexity-for-beginners-arrays-hashmaps-linked-lists-stacks-queues-tutorial/#DoublyLinkedList.add