Big O Notation

Elshad Karimov

July 1, 2021

1 Analogy and Time Complexity

Big O is the language and metric we use to describe the efficiency of algorithms.

Type: $O(N)$, $O(n^2)$ and $o(2^N)$.

2 Big O, Big Θ and Big Ω

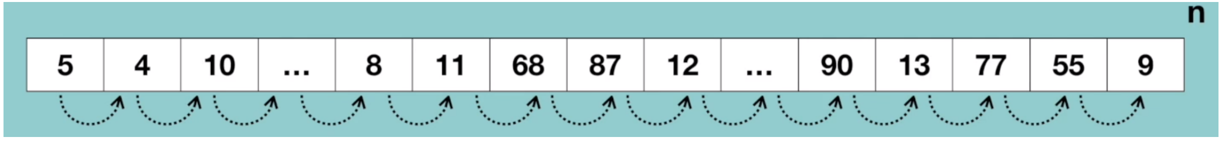
Algorithm run time notations. Just as the car performance evaluation.

- City traffic -20 liters / 100 km. (Worse case)
- Highway - 10 liters / 100 km. (Best case)
- Mixed condition - 15liters / 100 km. (Average case)

Big O, Big Θ and Big Ω

- **Big O**: It is a complexity that is going to be less or equal to the worst case.
- **Big Ω**: It is a complexity that is going to be at least more than the best case.
- **Big Θ**: It is a complexity that is within bounds of the worst and the best case.

Figure 1: Number Finding Algorithm



- **Big** O - $O(N)$
- **Big** Ω - $\Omega(1)$
- **Big** θ - $\theta(n/2)$

3 Algorithm Time Complexity Examples

Table 1: Algorithm run time complexities

Complexity	Name	Sample
$O(1)$	Constant	Accessing a specific element in array
$O(N)$	Linear	Loop through array elements
$O(LogN)$	Logarithmic	Find an element in sorted array
$O(N^2)$	Quadratic	Looking ar a every index in the array twice
$O(2^N$	Exponential	Double recursion in Fibonacci

$O(1)$**-Constant time**

```
# It takes constant time to access first element
array = [1,2,3,4,5]
array[0]
```

$O(N)$**-Linear time**

```
# Linear time since it is visiting every element of array
array = [1,2,3,4,5]
for element in array:
  print(element)
```

$O(logN)$-Logarithmic time

```
# Logarithmic time since it is visiting only some elements
for index in range(0, len(array),3):
  print(array[index])
```

```
# Binary search
search 9 within [1,5,8,9,11,13,15,19,21]
compare 9 to 11 then smaller
search 9 within [1,5,8,9]
compare 9 to 8 then bigger
search 9 within [9]
compare 9 to 9
return
```

$O(N^2)$-Quadratic time

```
array = [1,2,3,45]
for x in array:
  for y in array:
    print(x,y)
```

$O(2^N)$-Exponential time

```
def fibonacci(n)
  if n <= 1:
    return n
  return fibonacci(n-1)+fibonacci(n-2)
```

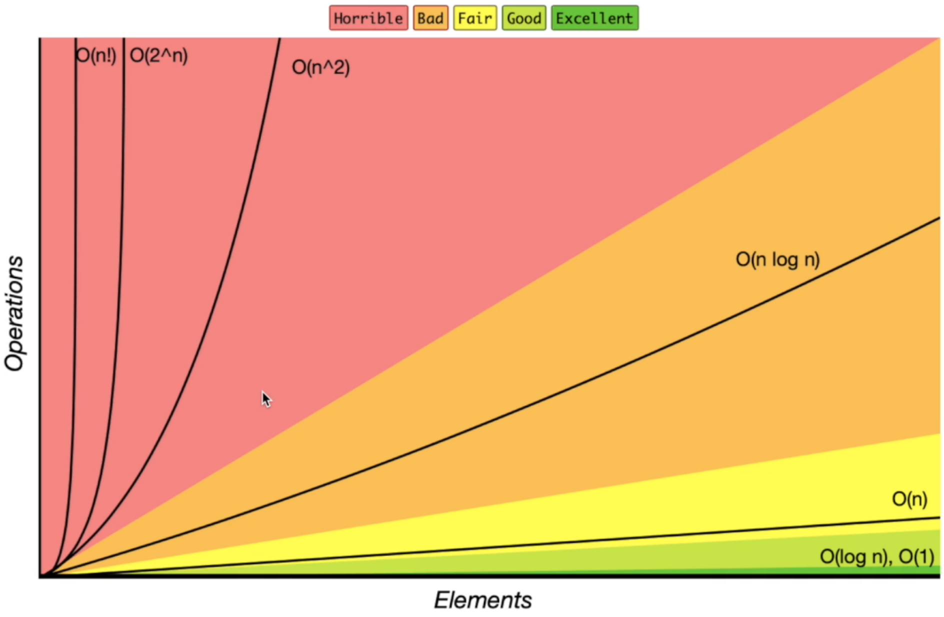


Figure 2: Big-O Complexity Chart

4 Space Complexity