I. Big-O Notation

- a simplified analysis of an algorithm's efficiency

1) Complexity in terms of input size, N.
2) Machine-independent.
3) Basic computer steps.
4) Time and space.

II. Types of measurement

- Worst-case
- Best-case
- Average-case

III. General Rules

- Ignore constants.

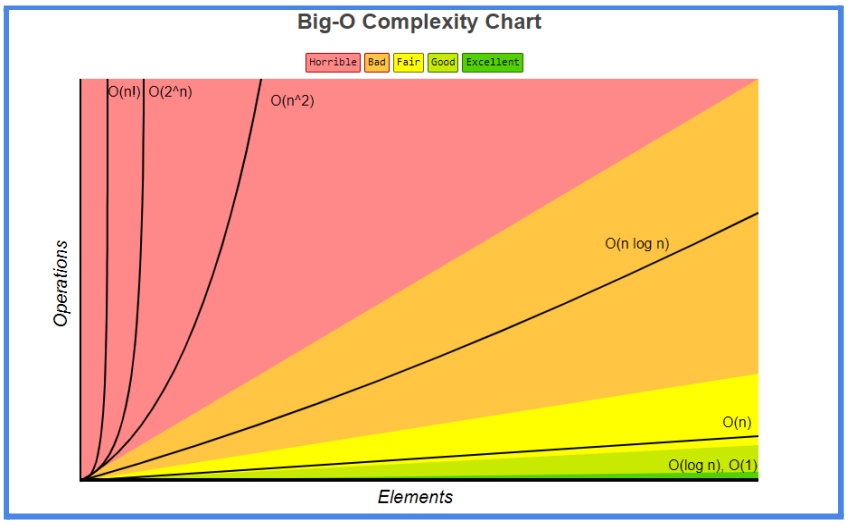
$$5n \rightarrow O(n)$$

- Certain terms "domainate" others.

$$O(1) < O(\log n) < O(n) < O(n \log n) < O(n^2) < O(2^n) < O(n!)$$

i.e. , ignore low-order terms.

IV. Big- O Complexity Charts (https://www.bigocheatsheet.com/)



V. Constant Time

```
x = 5 + ( 15 * 20 );
```

∴ Time = O(1)

→ Independent of input size, n

```
x = 5 + ( 15 * 20 );
y = 15 - 2;
print x + y ;
```

∴ Time = O(1) + O(1) + O(1) = O(1)

IV. Linear Time

```
for x in range( 0, n ):
  print x;  //O(1)
```

∴ Time = n * O(1) = O(n)

```
y = 5 + ( 15 * 20 ); //O(1)
for x in range( 0, n ): //O(n)
   print x;
```

∴ Time = O(n) + O(1) = O(n)

IV. Quadratic Time

```
for x in range( 0, n ): //O(n)
   for y in range( 0, n ): //O(n)
        print x * y;
```

∴ Time = O(n) + O(n) = $O(n^2)$