SCHOOL OF
ENGINEERING
BANGKOK UNIVERSITY

Big O notation and Recursive programming

AIE 311 : Data structure and Algorithm

Big O notation

SCHOOL OF
ENGINEERING
BANGKOK UNIVERSITY

- What is Big O notation?
  - In short : Program performance check.
  - Performance check : Plotting graph, with higher Y-axis will lead into bad performance.
- Why?
  - Mostly for performance check.
- How?
  - Follow the instructions due to coding method.

Big O notation

SCHOOL OF ENGINEERING
BANGKOK UNIVERSITY

Notation	Type of function
$O(1)$	Constant function (ค่าคงที่)
$O(\log n)$	Logarithm function (ฟังก์ชันลอการิทึม)
$O(n)$	Linear function (ฟังก์ชันเชิงเส้น)
$O(n \log n)$	Linearithmic function (ฟังก์ชันลอการิทึมเชิงเส้น)
$O(n^2)$	Quadratic function (ฟังก์ชันกำลังสอง)
$O(n^3)$	Cubic fuction (ฟังก์ชันกำลังสาม)
$O(2^n)$	Exponential function (ฟังก์ชันเอ็กซ์โพเนนเชียล)
$O(n!)$	Factoral function (Factorial function)

Ref: https://kongruksiam.medium.com/data-structure-algorithm-ep-5-big-o-notation-8a412589d62b

Big O notation

- **ตารางเปรียบเทียบประสิทธิภาพ** (n = จำนวนข้อมูล)

ชื่อฟังก์ชั่น	สัญลักษณ์	n = 1	n = 2	n = 4	n = 8	n = 16
Constant	O(1)	1	1	1	1	1
Logarithmic	O(log n)	1	1	2	3	4
Linear	O(n)	1	2	4	8	16
Linearithmic	O(n log n)	1	2	8	24	64
Quadratic	$O(n^2)$	1	4	16	64	256
Cubic	$O(n^3)$	1	8	64	512	4,096
Exponential	$O(2^n)$	2	4	16	256	65,536
Factorial	O(n!)	1	2	24	40,320	20,922,789,888,000

Ref: https://kongruksiam.medium.com/data-structure-algorithm-ep-5-big-o-notation-8a412589d62b

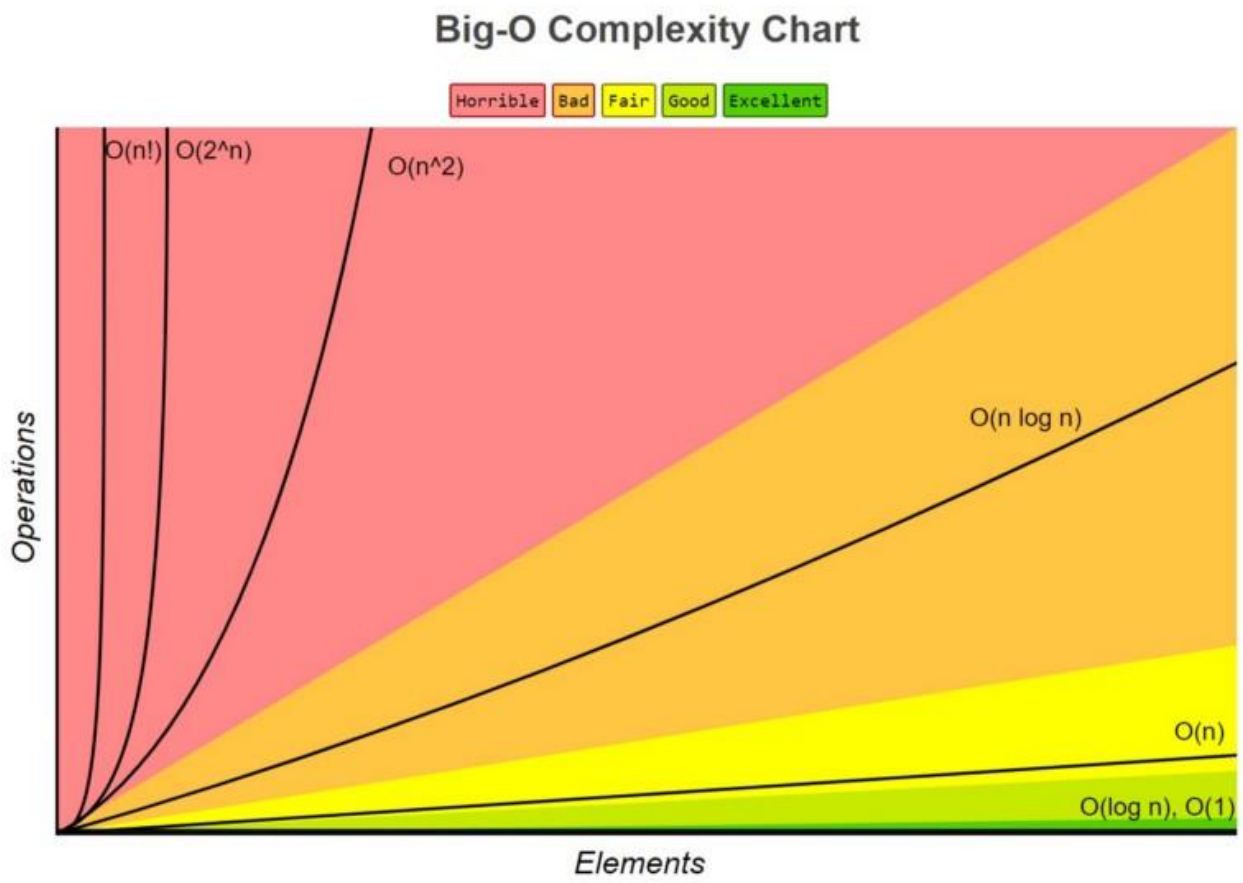
Big O notation

- **ตารางเปรียบเทียบประสิทธิภาพ** (n = จำนวนข้อมูล)

สัญลักษณ์	ชื่อฟังก์ชั่น	ประสิทธิภาพ
O(1)	Constant	ดีที่สุด
O(log n)	Logarithmic	
O(n)	Linear	
O(n log n)	Linearithmic	
$O(n^2)$	Quadratic	
$O(n^3)$	Cubic	
$O(2^n)$	Exponential	
O(n!)	Factorial	แย่ที่สุด

Ref: https://kongruksiam.medium.com/data-structure-algorithm-ep-5-big-o-notation-8a412589d62b

Big O notation (Graph)



Performance graph of Big O notation

Big O notation (Good performance)

SCHOOL OF
ENGINEERING
BANGKOK UNIVERSITY

- O(1) : constant
  - One step done
- O(log n) : logarithmic
  - Reduce loop length by half after done for one round
- O(n) : linear
  - Loop equal to input e.g., 5 inputs loop for 5 steps

Big O notation (Mediocre performance)

SCHOOL OF
ENGINEERING
BANGKOK UNIVERSITY

- O(n log n) : linearithmic
  - Loop n round but with log n inside loop. (Average performance)
- O(n^2) : quadratic
  - Loop will longer 4 times when receive twice input. (Below average)
- O(2^n) : exponential
  - Exponential even less input but performance still bad.
- O(n!) : factorial
  - The worst of Big O performance.

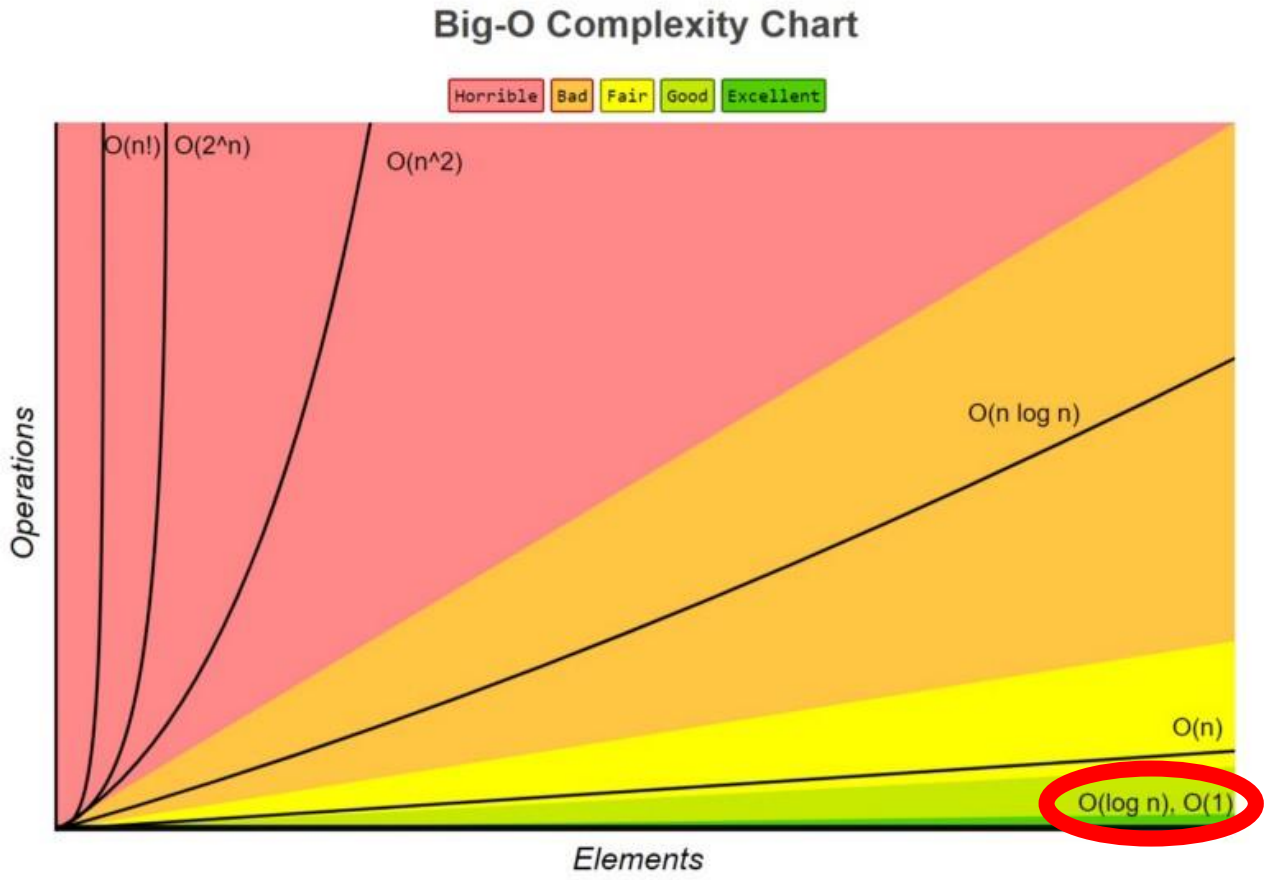
Big O notation (Constant)


```python
def evenOrOdd(num):
    if (num % 2 == 0):
        print("Even")
    else:
        print("Odd")

evenOrOdd(0)
evenOrOdd(7)
```

O(1) : constant (The best)

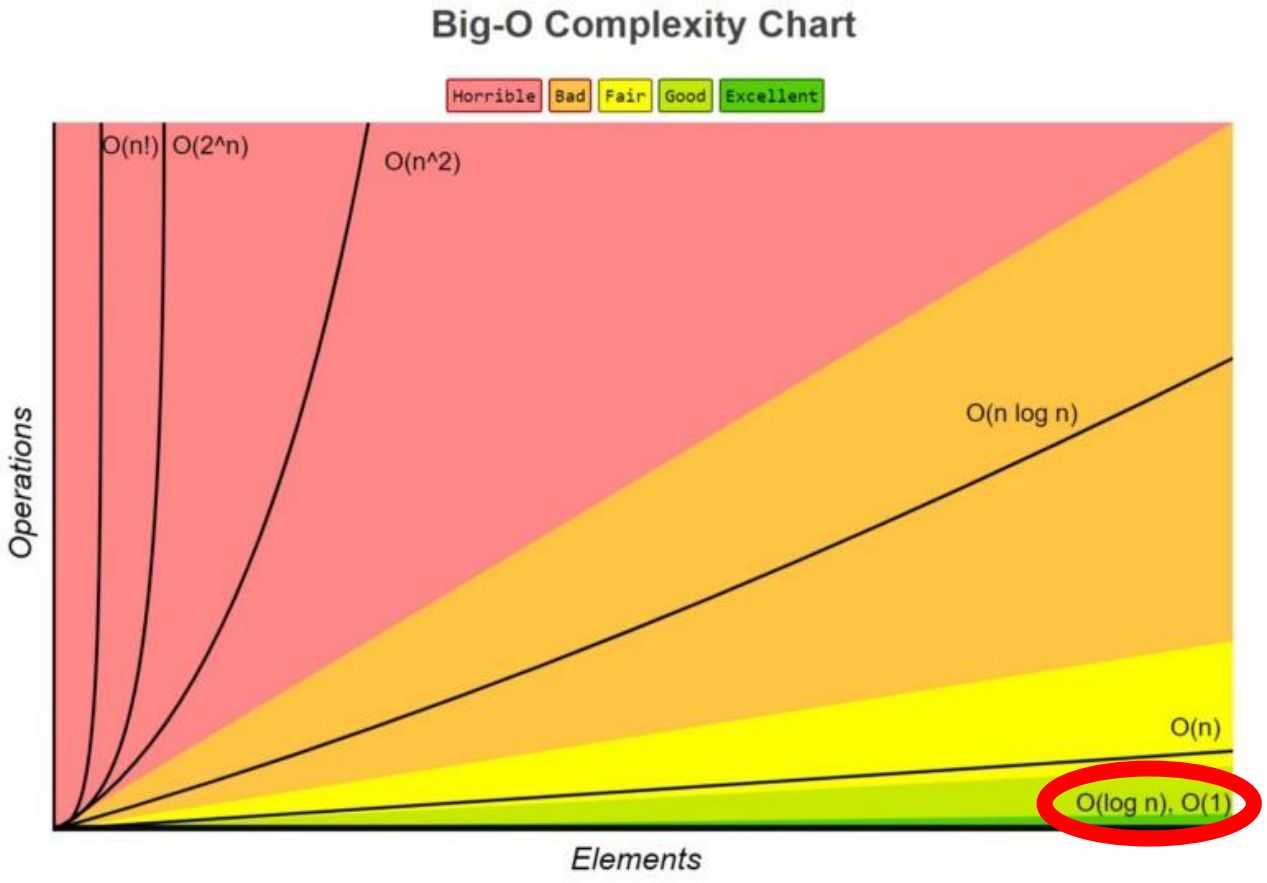
- One step done



Big O notation (Logarithmic)


```python
def binarySearch(arr, value, first, last):
    if last >= first:
        mid = (first + last) // 2
        if arr[mid] == value:
            return mid
        elif arr[mid] > value:
            return binarySearch(arr, value, first, mid - 1)
        else:
            return binarySearch(arr, value, mid + 1, last)
    else:
        return -1

numArray = [1, 2, 7, 10, 22, 31]
number = 31
result = binarySearch(numArray, number, 0, len(numArray) - 1)
print(result)
```



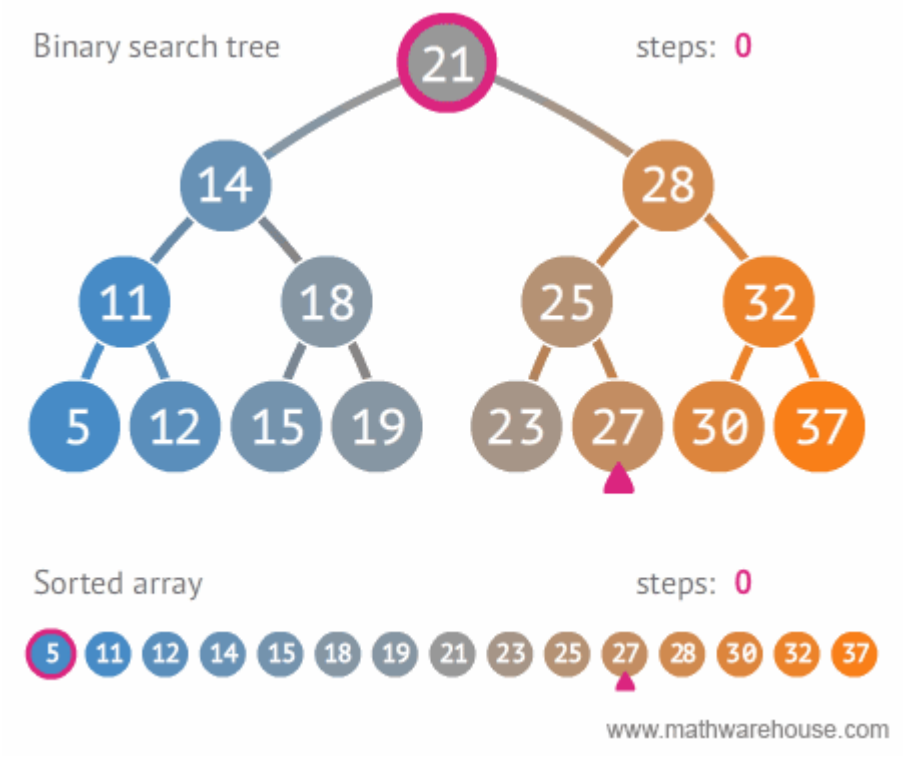
O(log n) : logarithmic (The best)

- Reduce loop length by half after done for one round.

Big O notation (Logarithmic)

SCHOOL OF ENGINEERING BANGKOK UNIVERSITY

- Implementation (Binary Search)



O(log n) : logarithmic (The best)

- Reduce loop length by half after done for one round.

Ref: https://www.mathwarehouse.com/programming/gifs/binary-search-tree.php

Big O notation (Linear)

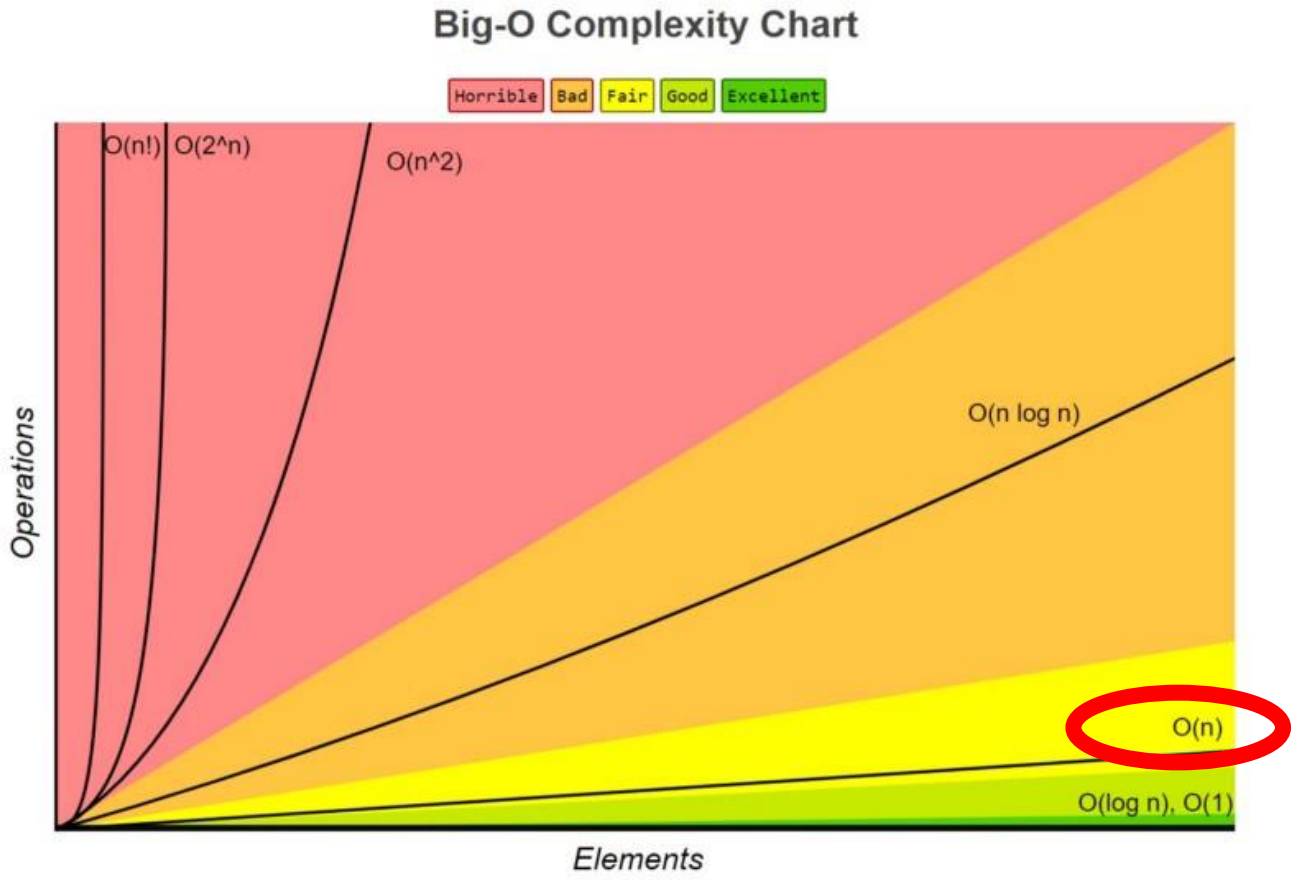

```python
def searchNumber(arr, value):
 for i in range(len(arr)):
   if(arr[i] == value):
     return i

numArray = [7, 16, 2, 0, 5, 1, 30]
number = 30

result = searchNumber(numArray, number)
print(result)
```

O(n) : linear (Good)

- Loop equal to input e.g., 5 inputs loop for 5 steps.

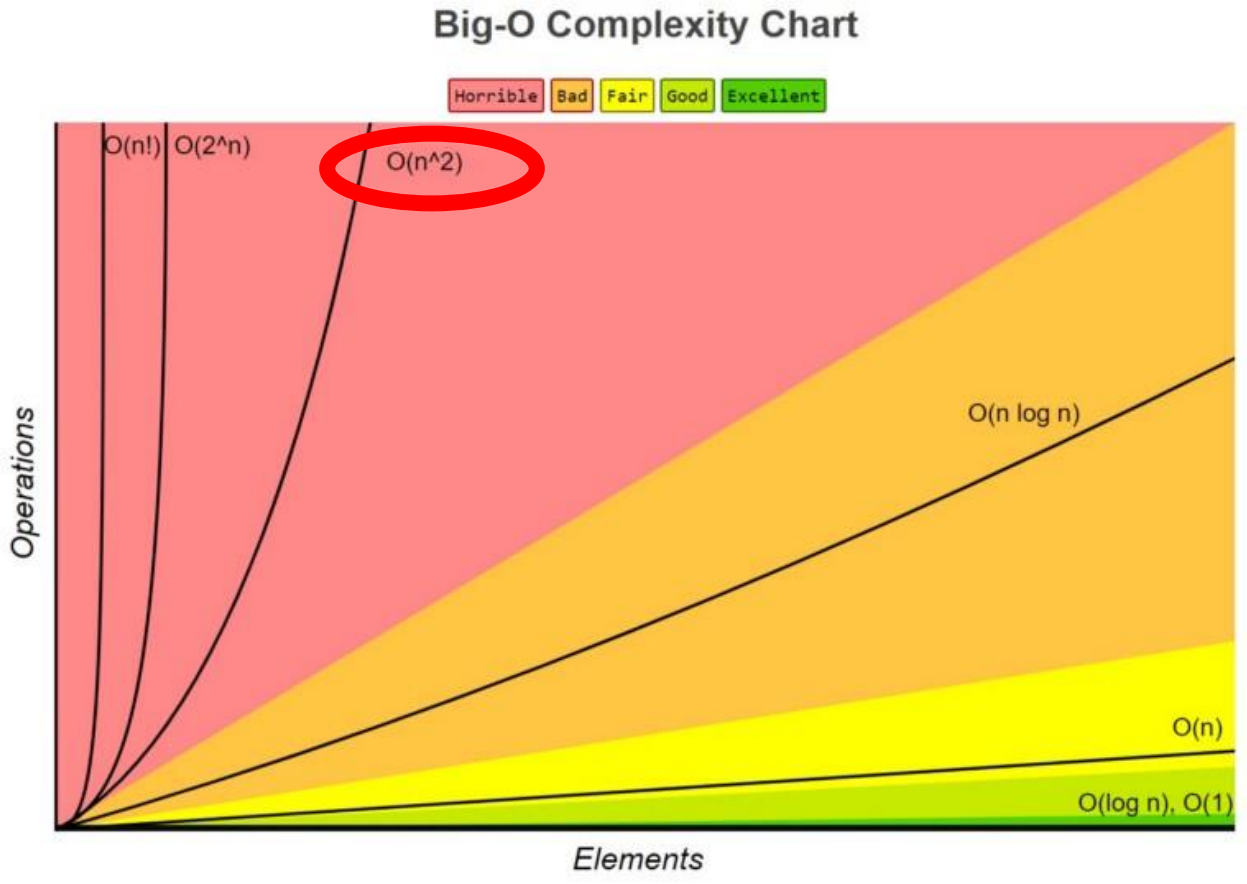


Big O notation (Quadratic)


```python
def duplicateCheck(arr):
 for i in range(len(arr)):
   a = arr[i]
   for j in range(i+1, len(arr)):
     b = arr[j]
     print(a, b)
     if(a == b):
       return "duplicated"
 return "not duplicate"

numArray = [1,3,5,9]

result = duplicateCheck(numArray)
print(result)
```



O(n^2) : quadratic (Below average)

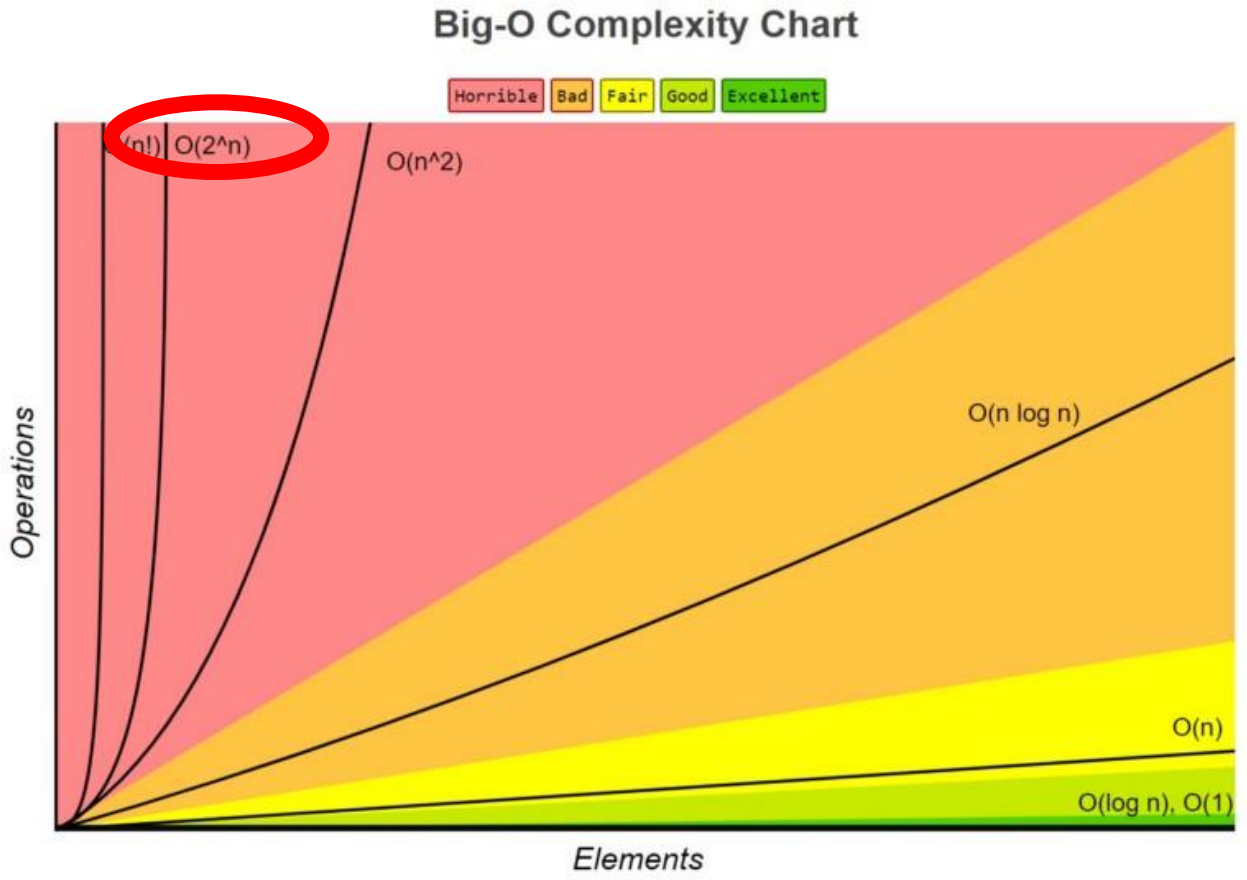
- Loop will longer 4 times when receive twice input.

Big O notation (Exponential)


```python
def fibonacci(n):
    if n==1:
        return 0
    elif n==2:
        return 1
    else:
        return (fibonacci(n-1)+fibonacci(n-2))
print(fibonacci(5))
print(fibonacci(21))
```

O(2^n) : exponential (Bad)

- Exponential even less input but performance still bad.



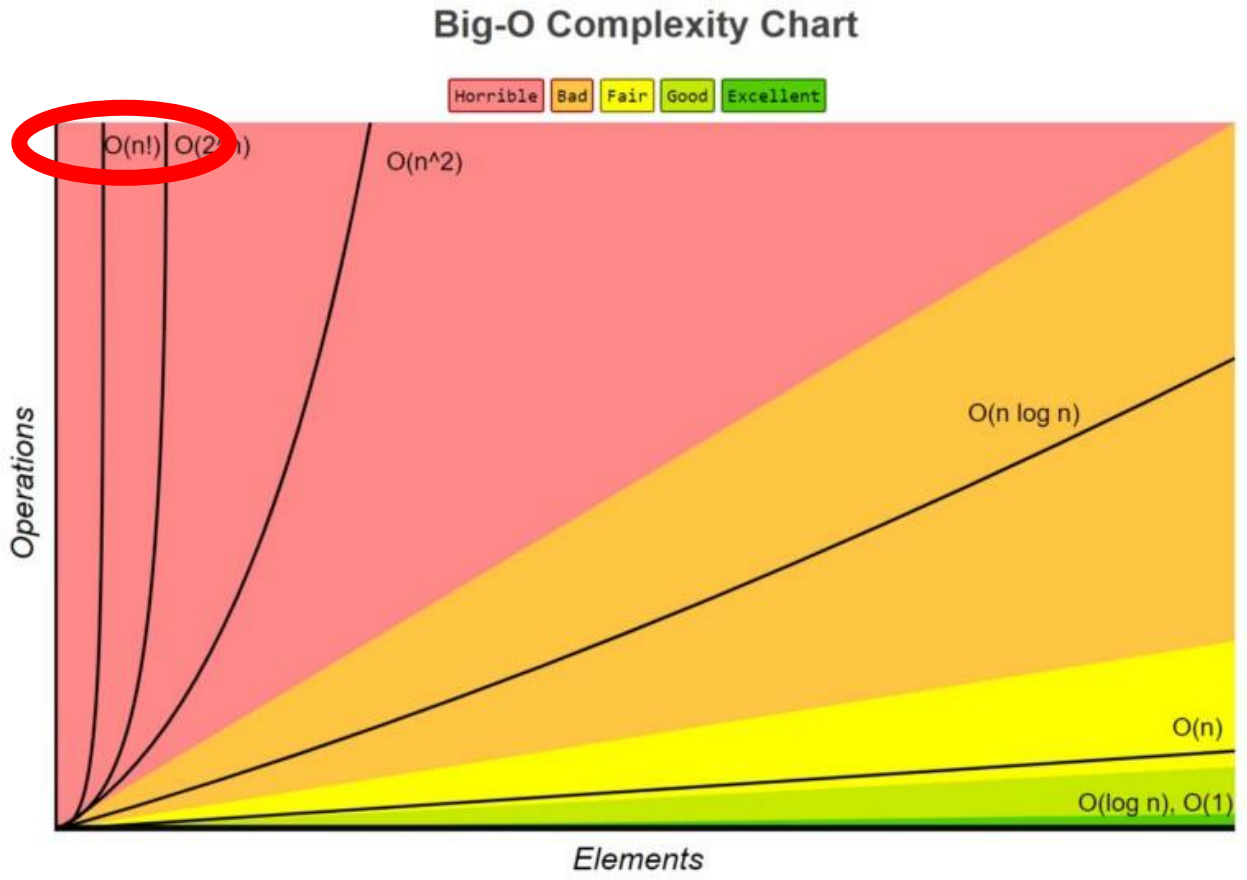
Big O notation (Factorial)


```python
def factorial(num):
 if(num == 1):
   return 1
 for i in range(0, num):
   return num * factorial(num-1)

print(factorial(9))
```

O(n!) : factorial (The worst)

- The worst of Big O performance.



Big O notation

- **ตารางเปรียบเทียบประสิทธิภาพ** (n = จำนวนข้อมูล)

ชื่อฟังก์ชั่น	สัญลักษณ์	n = 1	n = 2	n = 4	n = 8	n = 16
Constant	O(1)	1	1	1	1	1
Logarithmic	O(log n)	1	1	2	3	4
Linear	O(n)	1	2	4	8	16
Linearithmic	O(n log n)	1	2	8	24	64
Quadratic	$O(n^2)$	1	4	16	64	256
Cubic	$O(n^3)$	1	8	64	512	4,096
Exponential	$O(2^n)$	2	4	16	256	65,536
Factorial	O(n!)	1	2	24	40,320	20,922,789,888,000

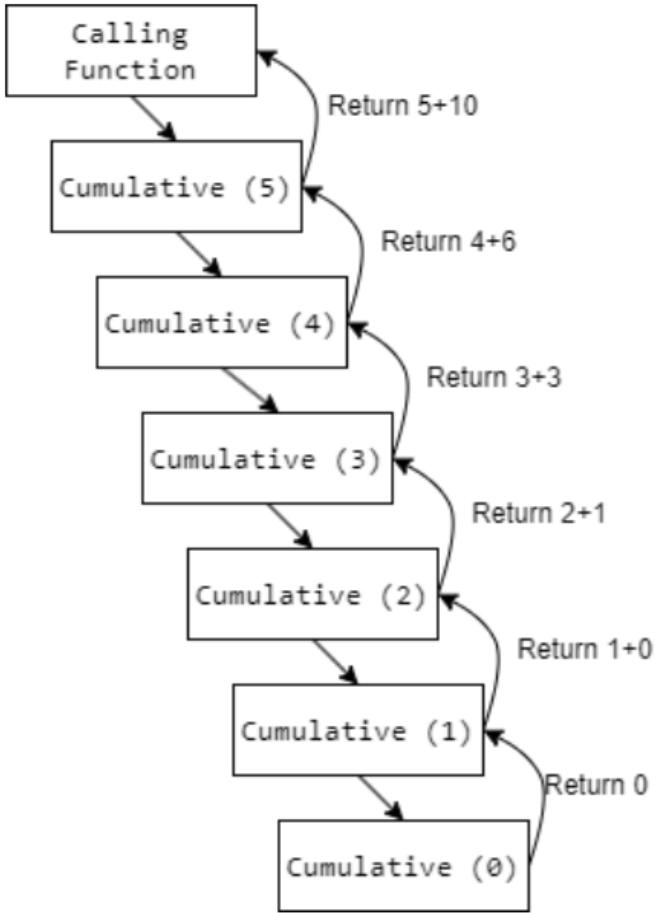
Ref: https://kongruksiam.medium.com/data-structure-algorithm-ep-5-big-o-notation-8a412589d62b

Recursive programming

SCHOOL OF
ENGINEERING
BANGKOK UNIVERSITY

- What is recursive programming?
  - The method to repeat created function to called itself.
- How?
  - Inside the function it will call itself function
    - e.g., Function A will call Function A inside
- Why?
  - Recursive programming will declare parametres inside the function and after completed. The parametres will be automatically removed.
- Less case of use (Specific used)
- Consume lots of memory

Recursive programming (Cumulative)

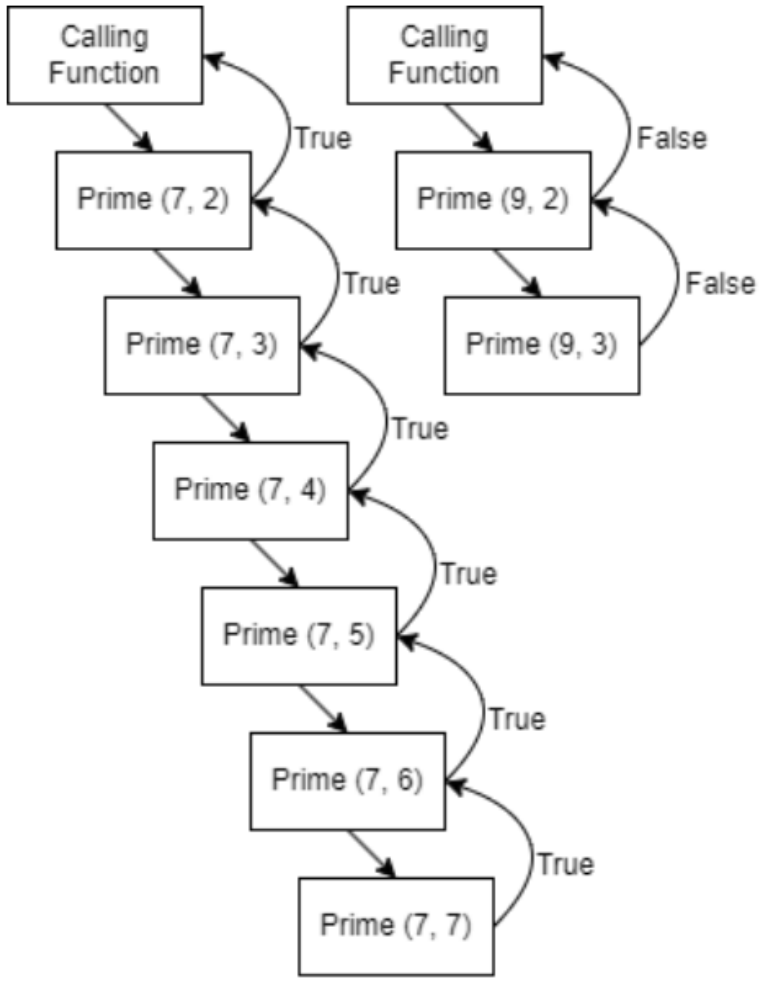



```python
def Cumulative(Input):
    if Input == 0:
        return 0
    else:
        return Input + Cumulative(Input - 1)
print(Cumulative(5))
```

Cumulative is a function

- Inside Cumulative will call another Cumulative.
  - In this case next Cumulative will call as
    - Cumulative (n - 1)
    - After input equal to 0 or (n - 1) it will return input value (n - 1)
      - Except (0 - 1) : This will return 0 instead.

Recursive programming (Prime number)




```
def Prime(Input,Count):
    Status = True
    if Count == Input:
        return True
    else:
        if Input % Count == 0:
            return False
        else:
            Count += 1
            Status = (Prime (Input,Count) and Status)
            return Status
def Recursive(Input):
    if Prime(Input,2) == True:
        return " is prime number"
    else:
        return " is not prime number"
# Input
Input = 7
print(str(Input) + Recursive(Input))
Input = 9
print(str(Input) + Recursive(Input))
```

Prime number check function.

- Input call Recursive function.
- Recursive function call Prime function.
- Prime function call Prime function.
- Until reach its input or not prime number.

References

- [1] freeCodeCamp.org, “All you need to know about ‘big O notation’ to crack your next coding interview,” freeCodeCamp.org, https://www.freecodecamp.org/news/all-you-need-to-know-about-big-o-notation-to-crack-your-next-coding-interview-9d575e7eec4/ (accessed Aug. 20, 2023).
- [2] Developer, “Big O คืออะไร ? - Borntodev Digital Academy,” BorntoDev เขียนโปรแกรม ขั้นเทพ !, https://www.borntodev.com/2020/03/21/big-o-%E0%B8%84%E0%B8%B7%E0%B8%AD%E0%B8%AD%E0%B8%B0%E0%B9%84%E0%B8%A3/ (accessed Aug. 20, 2023).
- [3] M. Goodrich, R. Tamassia and M. Goldwasser, Data Structures and Algorithms in Python. 2013.
- [4] D. Suthaputchakun, Lecture Note CE311: Data Structure and Algorithm. pp. 12 – 69
- [5] MathWarehouse, https://www.mathwarehouse.com/programming/gifs/binary-search-tree.php
- [6] Data Structure & Algorithm(EP.5) — Big-O Notation,

  https://kongruksiam.medium.com/data-structure-algorithm-ep-5-big-o-notation-8a412589d62b