# Design And Analysis of Algorithms

## Lab 2

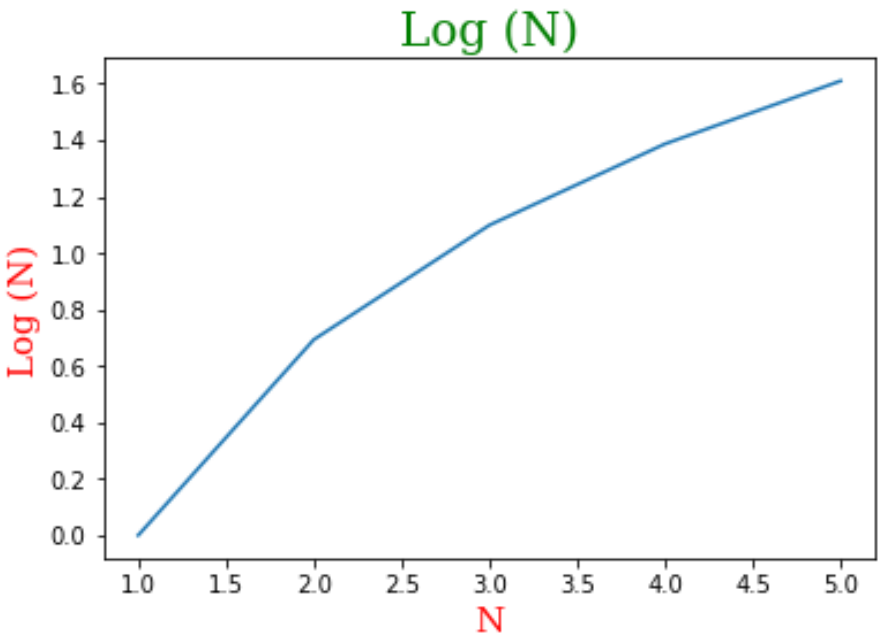
**S Abhishek**

**AM.EN.U4CSE19147**

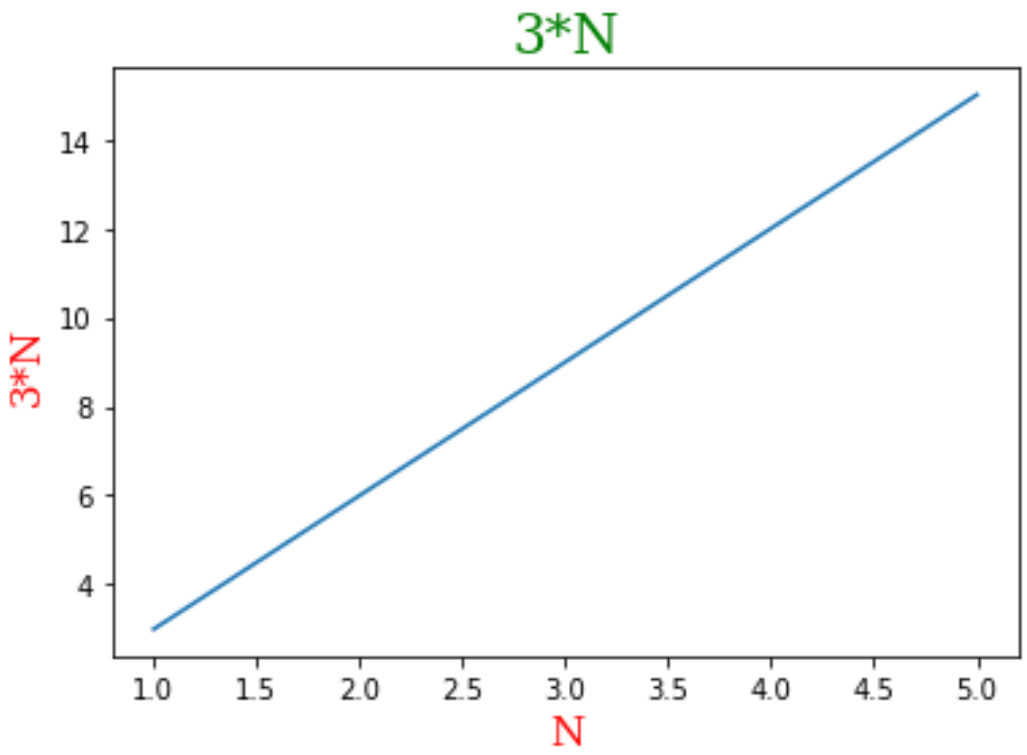
Collab Link

## Plot n

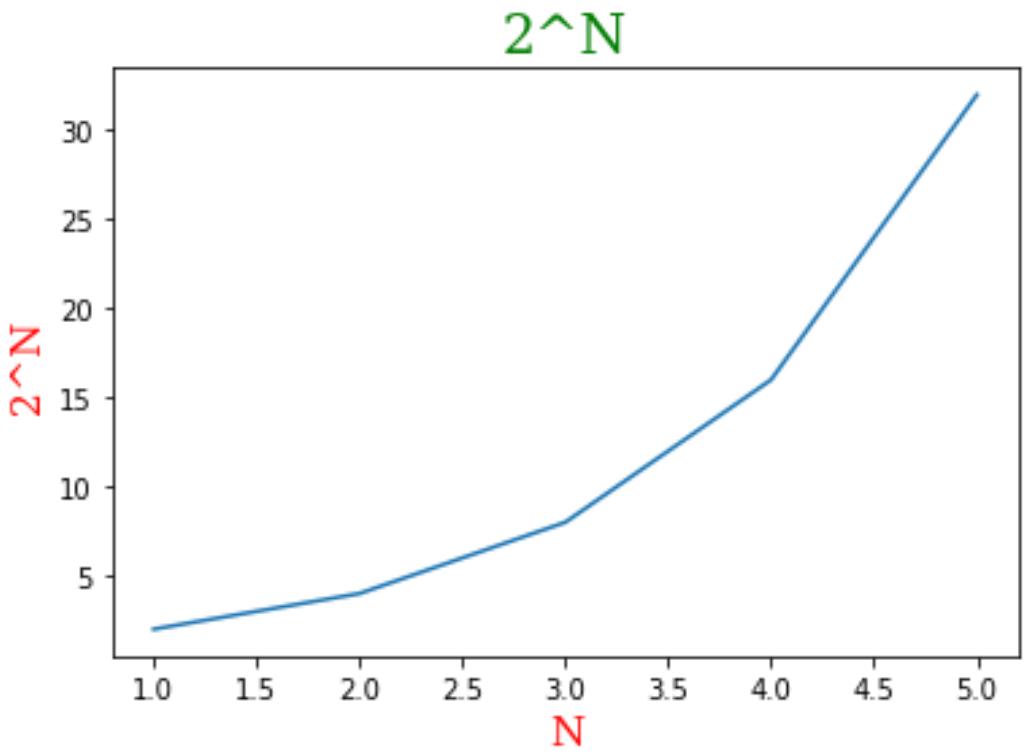
```
# Log (N)

import matplotlib.pyplot as plt
import numpy as np
import math

x = []
y1 = []

for i in range(1,6):
  x.append(i)
  y1.append(math.log(i))

plt.plot(x,y1)

f = {'family':'serif','color':'red','size':15}
T = {'family':'serif','color':'Green','size':20}

plt.title("Log (N)", fontdict = T)
plt.xlabel("N", fontdict = f)
plt.ylabel("Log (N)", fontdict = f)

plt.show()
```



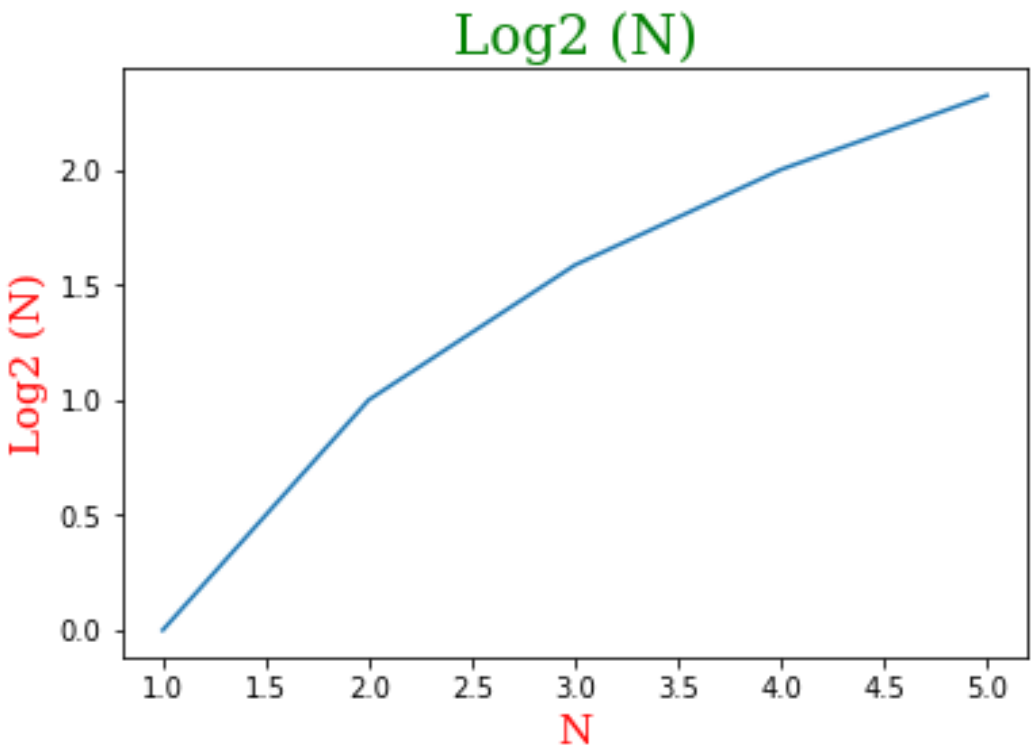

```
# 3*N

import matplotlib.pyplot as plt
import numpy as np

y2 = []

for i in range(1,6):
  y2.append(3*i)

plt.plot(x,y2)

f = {'family':'serif','color':'red','size':15}
T = {'family':'serif','color':'Green','size':20}

plt.title("3*N", fontdict = T)
plt.xlabel("N", fontdict = f)
plt.ylabel("3*N", fontdict = f)

plt.show()
```



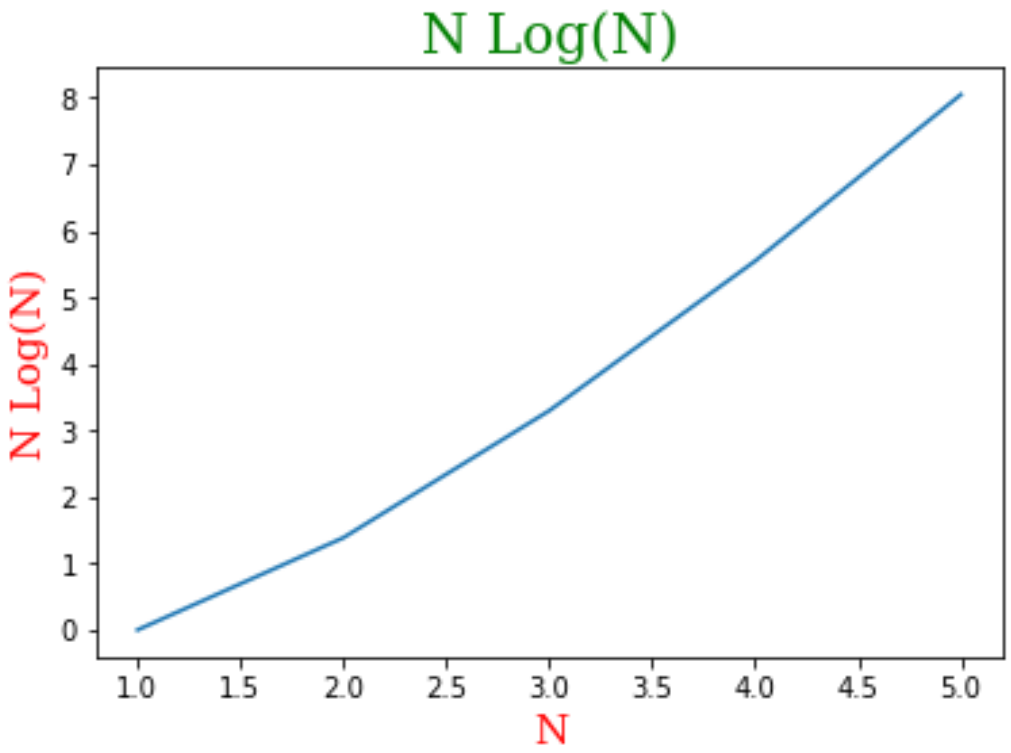

```
# 2^N

import matplotlib.pyplot as plt
import numpy as np

y3 = []

for i in range(1,6):
  y3.append(2**i)

plt.plot(x,y3)

f = {'family':'serif','color':'red','size':15}
T = {'family':'serif','color':'Green','size':20}

plt.title("2^N", fontdict = T)
plt.xlabel("N", fontdict = f)
plt.ylabel("2^N", fontdict = f)

plt.show()
```



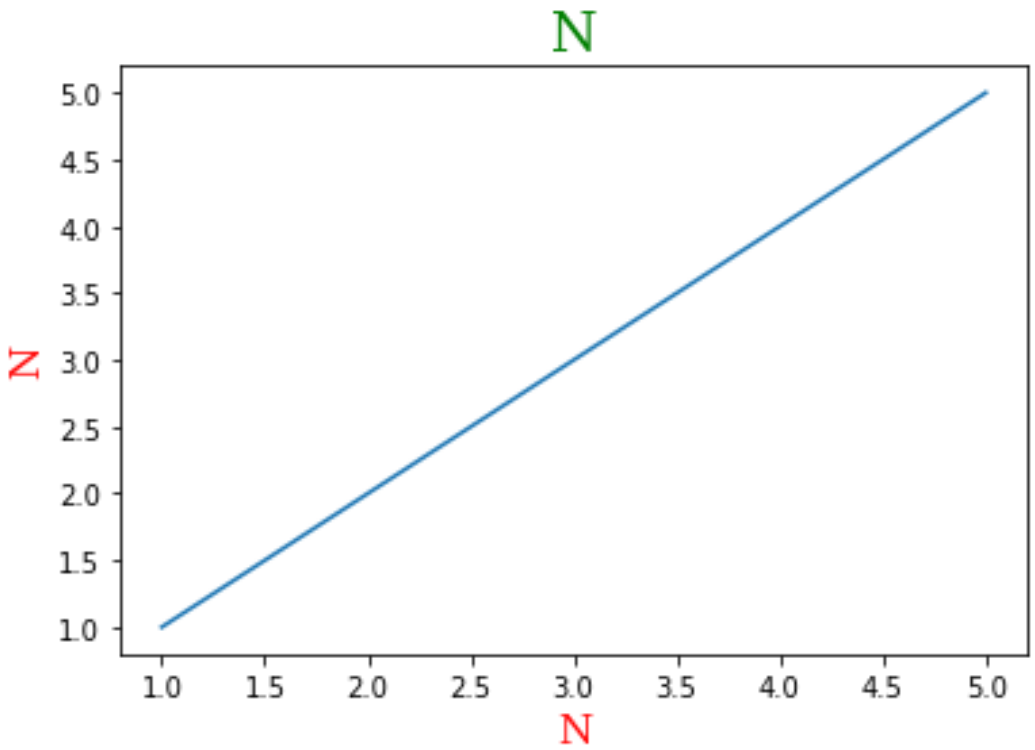

```
# Log2 (N)

import matplotlib.pyplot as plt
import numpy as np
import math

y4 = []

for i in range(1,6):
  y4.append(math.log2(i))

plt.plot(x,y4)

f = {'family':'serif','color':'red','size':15}
T = {'family':'serif','color':'Green','size':20}

plt.title("Log2 (N)", fontdict = T)
plt.xlabel("N", fontdict = f)
plt.ylabel("Log2 (N)", fontdict = f)

plt.show()
```



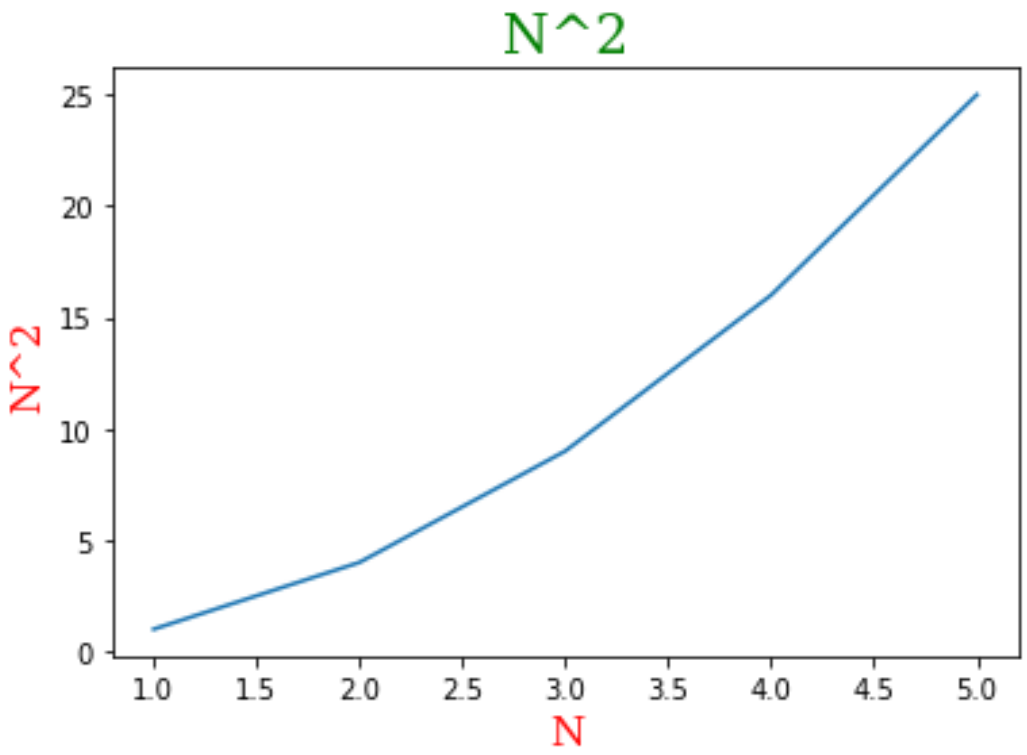

```
# N Log(N)

import matplotlib.pyplot as plt
import numpy as np
import math

y5 = []

for i in range(1,6):
  y5.append(i*(math.log(i)))

plt.plot(x,y5)

f = {'family':'serif','color':'red','size':15}
T = {'family':'serif','color':'Green','size':20}

plt.title("N Log(N)", fontdict = T)
plt.xlabel("N", fontdict = f)
plt.ylabel("N Log(N)", fontdict = f)

plt.show()
```



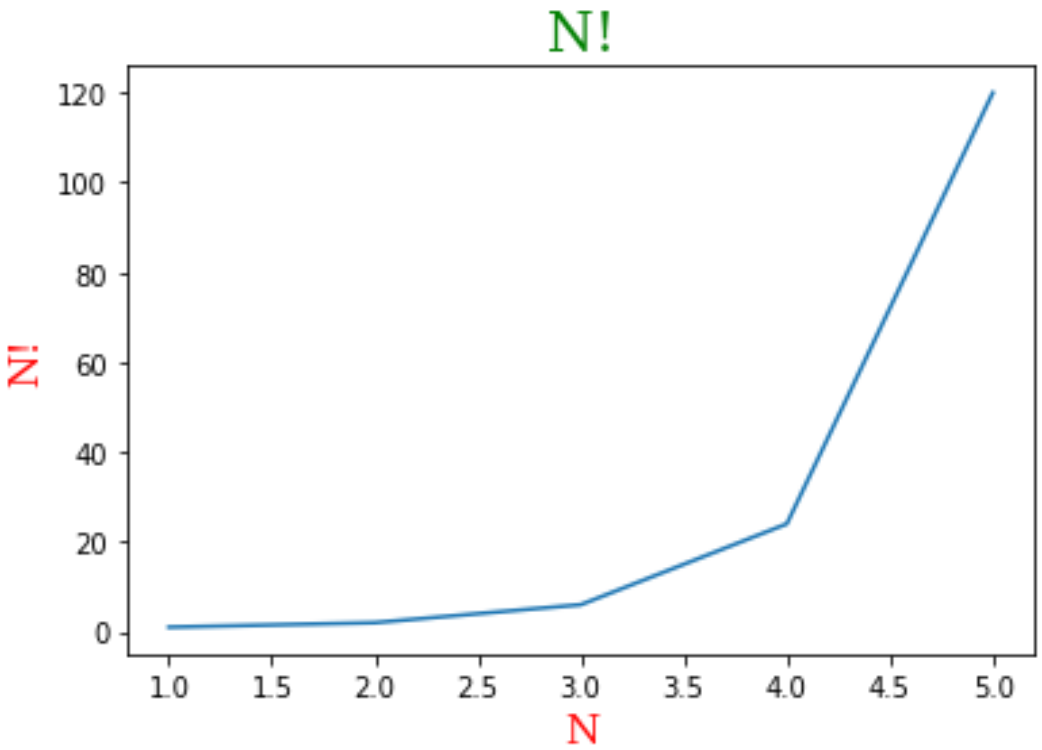

```
# N

import matplotlib.pyplot as plt
import numpy as np

y6 = []

for i in range(1,6):
  y6.append(i)

plt.plot(x,y6)

f = {'family':'serif','color':'red','size':15}
T = {'family':'serif','color':'Green','size':20}

plt.title("N", fontdict = T)
plt.xlabel("N", fontdict = f)
plt.ylabel("N", fontdict = f)

plt.show()
```



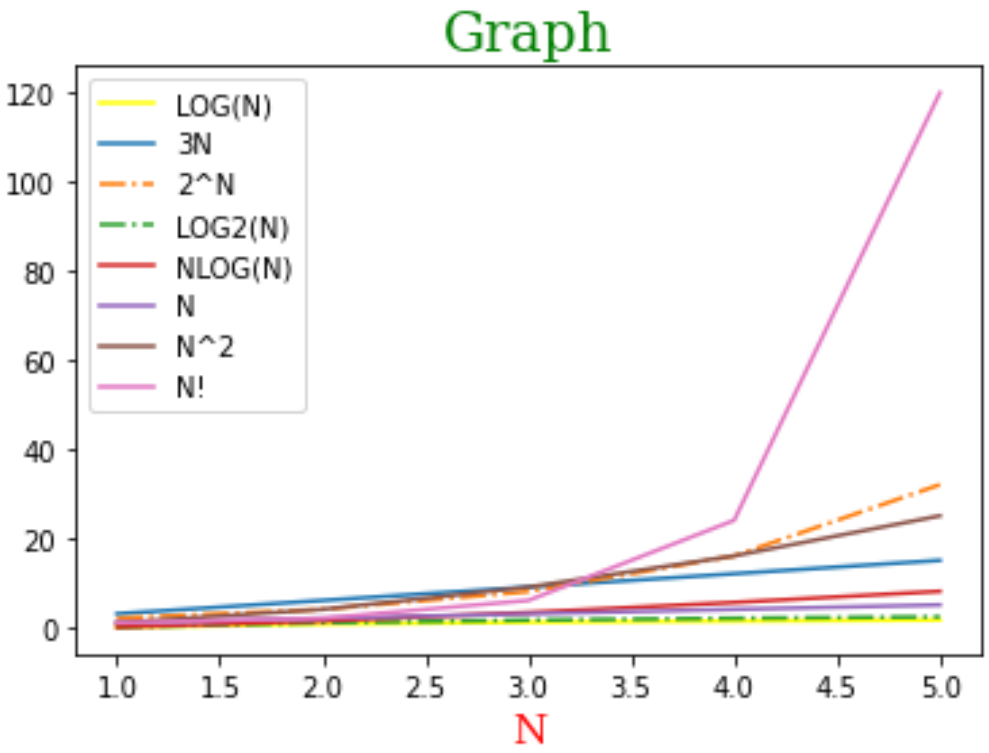

```
# N^2

import matplotlib.pyplot as plt
import numpy as np

y7 = []

for i in range(1,6):
  y7.append(i**2)

plt.plot(x,y7)

f = {'family':'serif','color':'red','size':15}
T = {'family':'serif','color':'Green','size':20}

plt.title("N^2", fontdict = T)
plt.xlabel("N", fontdict = f)
plt.ylabel("N^2", fontdict = f)

plt.show()
```




```
# N!

import matplotlib.pyplot as plt
import numpy as np
import math

y8 = []

for i in range(1,6):
  y8.append(math.factorial(i))

plt.plot(x,y8)

f = {'family':'serif','color':'red','size':15}
T = {'family':'serif','color':'Green','size':20}

plt.title("N!", fontdict = T)
plt.xlabel("N", fontdict = f)
plt.ylabel("N!", fontdict = f)

plt.show()
```

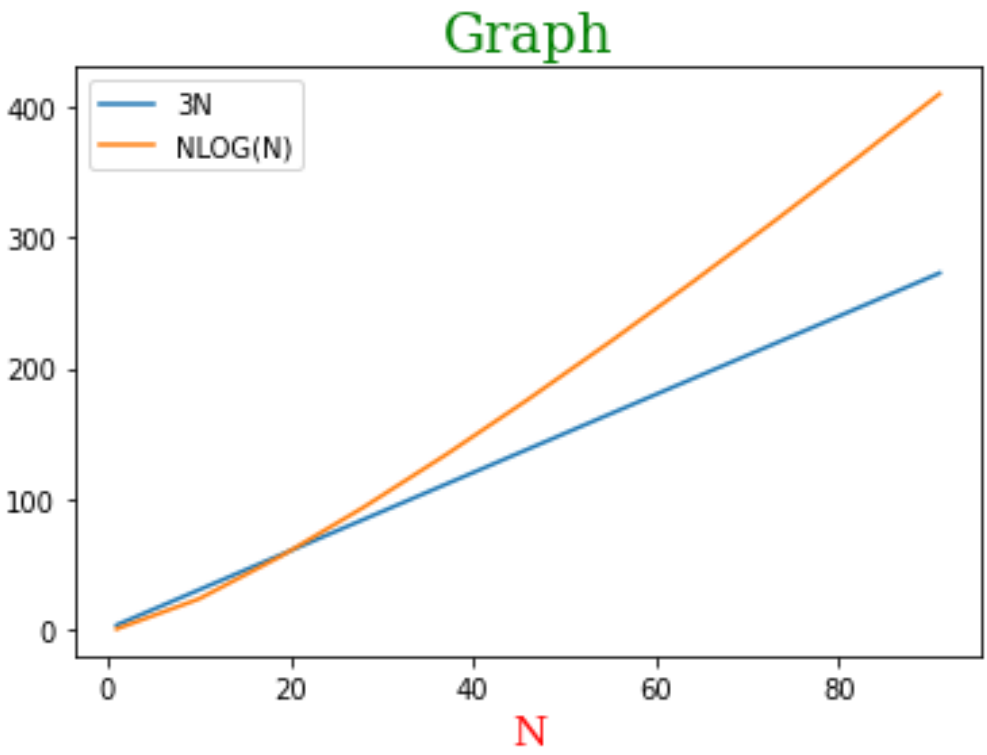



```
plt.plot(x,y1,label = "LOG(N)",color = "yellow")
plt.plot(x,y2,label = "3N")
plt.plot(x,y3,label = "2^N", linestyle="-.")
plt.plot(x,y4,label = "LOG2(N)", linestyle="-.")
plt.plot(x,y5,label = "NLOG(N)")
plt.plot(x,y6,label = "N")
plt.plot(x,y7,label = "N^2")
plt.plot(x,y8,label = "N!")

f = {'family':'serif','color':'red','size':15}
T = {'family':'serif','color':'Green','size':20}

plt.title("Graph", fontdict = T)
plt.xlabel("N", fontdict = f)

plt.legend()
plt.show()
```




```
# 3*N

import matplotlib.pyplot as plt
import numpy as np

y2 = []
x = []

for i in range(1,100,9):
  y2.append(3*i)
  x.append(i)

# N Log(N)

import matplotlib.pyplot as plt
import numpy as np
import math

y5 = []

for i in range(1,100,9):
  y5.append(i*(math.log(i)))

plt.plot(x,y2,label = "3N")
plt.plot(x,y5,label = "NLOG(N)")
```

```
f = {'family':'serif','color':'red','size':15}
T = {'family':'serif','color':'Green','size':20}

plt.title("Graph", fontdict = T)
plt.xlabel("N", fontdict = f)

plt.legend()
plt.show()
```




## Increasing Order Of Growth Rate

Log(N) -> Log2(N) -> N -> 3N -> N Log(N) -> N^2 -> 2^N -> N!

## Print Hello

```
# 3n Times

n = int(input("Enter the N : "))

print("\nHello will be printed {} times because the 3*{} is
{}".format(3*n,n,3*n))

print("\nHello"*3*n)
```

```
Enter the N : 3

Hello will be printed 9 times because the 3*3 is 9

Hello
Hello
Hello
Hello
Hello
Hello
Hello
Hello
Hello
```

```
# Log(N)

import math

n = int(input("Enter the N : "))

print("\nHello will be printed {} times because the log({}) is
{}".format(int(math.log10(n)),n,int(math.log10(n))))

print("\nHello"*int(math.log2(n)))
```

```
Enter the N : 20

Hello will be printed 1 times because the log(20) is 1

Hello
Hello
Hello
Hello
```

```
# 3n/4 times

n = int(input("Enter the N : "))

print("\nHello will be printed {} times because the 3N/4 is
{}".format(round((3*n)/4),(3*n)/4))

print("\nHello"*round((3*n)/4))
```

```
Enter the N : 10

Hello will be printed 8 times because the 3N/4 is 7.5

Hello
Hello
Hello
```

```
Hello
Hello
Hello
Hello
Hello
```

```
# N^2 Times

n = int(input("Enter the N : "))

print("\nHello will be printed {} times because the {}^2 is
{}".format(n**2,n,n**2))

print("\nHello"*n**2)
```

```
Enter the N : 3

Hello will be printed 9 times because the 3^2 is 9

Hello
Hello
Hello
Hello
Hello
Hello
Hello
Hello
Hello
```

```
# N(N+1)/2 Times

n = int(input("Enter the N : "))

p = int(round(n*(n+1))/2)

print("\nHello will be printed {} times because the N(N+1)/2 is
{}".format(p,p))

print("\nHello"*p)
```

```
Enter the N : 3

Hello will be printed 6 times because the N(N+1)/2 is 6

Hello
Hello
Hello
Hello
Hello
Hello
```

```
# N*LOG(N) Times

import math

n = int(input("Enter the N : "))

p = int(n*math.log(n))

print("\nHello will be printed {} times because the N*LOG(N) is
{}".format(p,p))

print("\nHello"*p)
```

```
Enter the N : 5

Hello will be printed 8 times because the N*LOG(N) is 8

Hello
Hello
Hello
Hello
Hello
Hello
Hello
Hello
```

## Unique Or Not

```
# Elements in a list are distinct or not with O(n^2) complexity

n = int(input("Enter the Number of Elements in the List : "))
print()
2
arr = []

for i in range(n):
  arr.append(int(input("Enter the Element {} : ".format(i+1))))

print("\nList :",arr)

f = 0

for i in range(n-1):
  for j in range(i+1,n):
    if arr[j] == arr[i]:
      f = 1

if f == 1:
```

```
  print("\nNot Distinct :(")
else:
  print("\nDistinct :)")
```

```
Enter the Number of Elements in the List : 6

Enter the Element 1 : 4
Enter the Element 2 : 5
Enter the Element 3 : 1
Enter the Element 4 : 3
Enter the Element 5 : 2
Enter the Element 6 : 4

List : [4, 5, 1, 3, 2, 4]

Not Distinct :(
```

```
# Elements in a list are distinct or not with O(nlogn) complexity

def partition(array, start, end):
    pivot = array[start]
    low = start + 1
    high = end

    while True:

        while low <= high and array[high] >= pivot:
            high = high - 1

        while low <= high and array[low] <= pivot:
            low = low + 1

        if low <= high:
            array[low], array[high] = array[high], array[low]
        else:
            break

    array[start], array[high] = array[high], array[start]

    return high


def quick_sort(array, start, end):
    if start >= end:
        return

    p = partition(array, start, end)
    quick_sort(array, start, p-1)
    quick_sort(array, p+1, end)
```

```
def search(arr):

  f = 0

  for i in range(len(arr)-1):
      if arr[i] == arr[i+1]:
         f = 1

  if f == 1:
    print("\nNot Distinct :(")
  else:
    print("\nDistinct :)")


if __name__ == "__main__":

  n = int(input("Enter the Number of Elements in the List : "))
  print()

  arr = []

  for i in range(n):
    arr.append(int(input("Enter the Element {} : ".format(i+1))))

  print("\nList :",arr)

  quick_sort(arr, 0, len(arr) - 1)

  search(arr)
```

```
Enter the Number of Elements in the List : 5

Enter the Element 1 : 7
Enter the Element 2 : 4
Enter the Element 3 : 8
Enter the Element 4 : 1
Enter the Element 5 : 4

List : [7, 4, 8, 1, 4]

Not Distinct :(
```

## Ternary Search

```
# Ternary Search
```

```
def ternary_search(left_index, right_index, search_key, list_vals):

    if right_index >= left_index:

        mid_index1 = left_index + (right_index -left_index)//3
        mid_index2 = right_index - (right_index - left_index)//3

        if list_vals[mid_index1] == search_key:
            return mid_index1

        if (list_vals[mid_index2] == search_key):
            return mid_index2

        if (search_key < list_vals[mid_index1]):
            return ternary_search(left_index,mid_index1-1, search_key,
list_vals)

        elif (search_key > list_vals[mid_index2]):
            return ternary_search(mid_index2+1, right_index, search_key,
list_vals)

        else:
            return ternary_search(mid_index1+1, mid_index2-1, search_key,
list_vals)

    return -1

if __name__ == "__main__":

  n = int(input("Enter the Number of Elements in the List : "))
  print()

  arr = []

  for i in range(n):
    arr.append(int(input("Enter the Element {} : ".format(i+1))))

  print("\nList :",arr)

  m = int(input("\nEnter the Element to be found : "))

  if ternary_search(0, n-1, m, arr):
    print("\nThe Element {} is found at the position
{}".format(m,ternary_search(0, n, m, arr)))
  else:
    print("\nElement {} is not found :(")
```

```
Enter the Number of Elements in the List : 5
```

```
Enter the Element 1 : 1
Enter the Element 2 : 6
Enter the Element 3 : 3
Enter the Element 4 : 9
Enter the Element 5 : 4

List : [1, 6, 3, 9, 4]

Enter the Element to be found : 4

The Element 4 is found at the position 4
```

**Time Complexity of Ternary Search is O(log3 n)**

- Ternary search should be faster than the Binary search since log2(N) >= log3(N) but this is not the case.
- When we calculate the Time complexity of any algorithm, we generally ignore the constants.
- But the constants in Ternary search are relatively larger than Binary search.
- Number of Comparision in each iteration of Binary Search = 2
- Number of Comparision in each iteration of Ternary Search = 4
- Due to this, the Ternary search is slower.