```
import random as rn
import matplotlib.pyplot as plt
import time
import numpy as np
from scipy import optimize
```

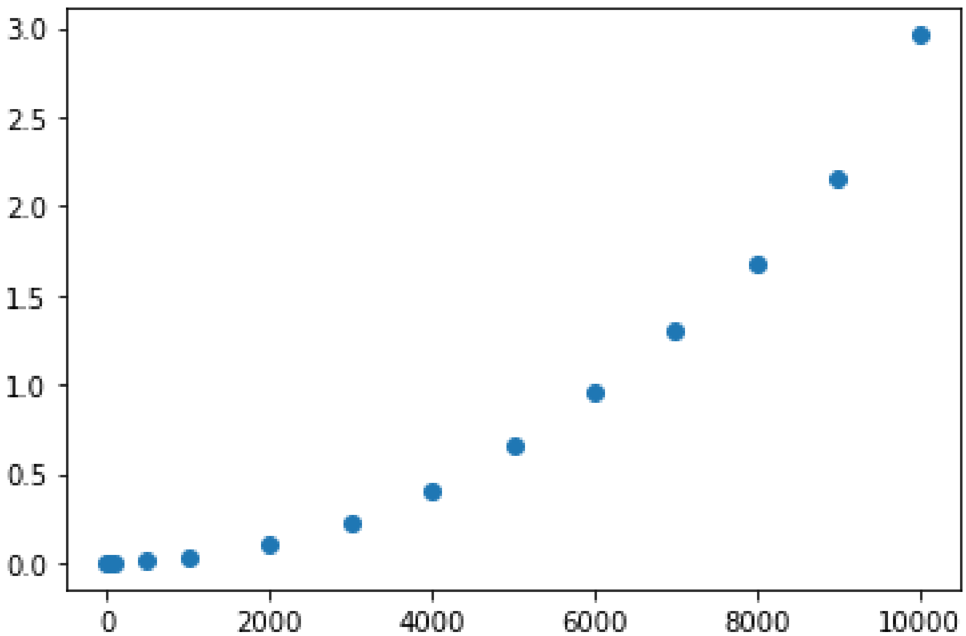
Bubble Sort

```
def bubbleSort(arr):
    for i in range(len(arr)):
        for j in range(0, len(arr)-i-1):
            if arr[j] > arr[j+1] :
                temp=arr[j]
                arr[j]=arr[j+1]
                arr[j+1]=temp



runTimeBubble=[]

size=[5,10,100,500,1000,2000,3000,4000,5000,6000,7000,8000,9000,10000]

arr=[]
for i in range (5):
    arr.append(rn.randint(1,1000000))

start=time.time()
bubbleSort(arr)
stop=time.time()

runTimeBubble.append(stop-start)

arr=[]
for i in range (10):
    arr.append(rn.randint(1,1000000))

start=time.time()
bubbleSort(arr)
stop=time.time()

runTimeBubble.append(stop-start)


arr=[]
for i in range (100):
    arr.append(rn.randint(1,1000000))

start=time.time()
bubbleSort(arr)
```

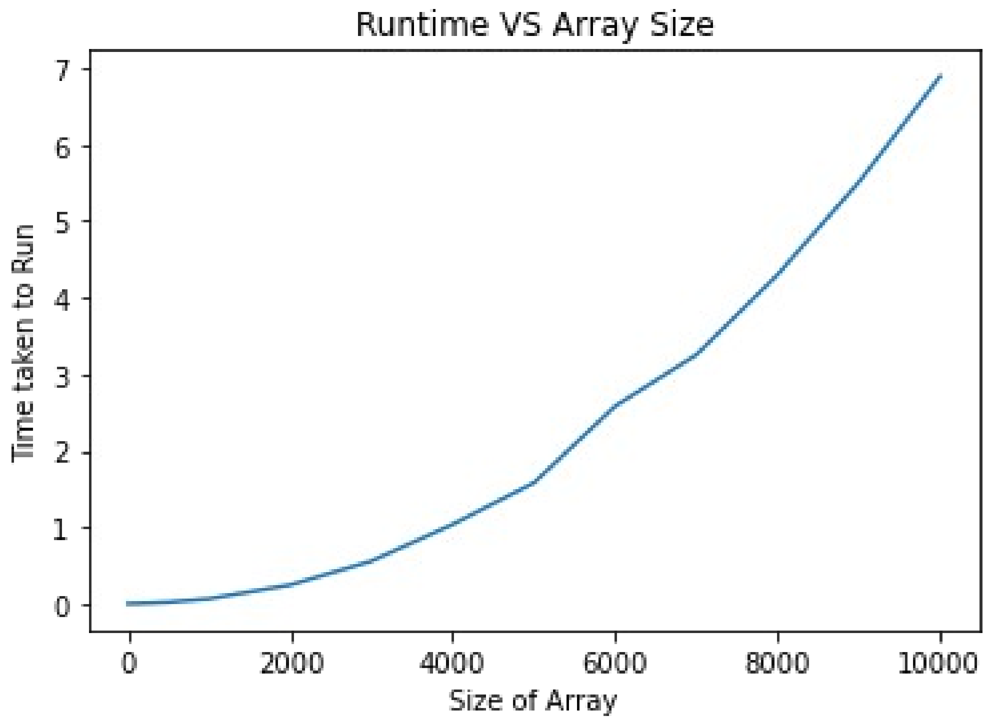
```python
stop=time.time()

runTimeBubble.append(stop-start)

arr=[]
for i in range (500):
    arr.append(rn.randint(1,1000000))

start=time.time()
bubbleSort(arr)
stop=time.time()

runTimeBubble.append(stop-start)

arr=[]
for i in range (1000):
    arr.append(rn.randint(1,1000000))

start=time.time()
bubbleSort(arr)
stop=time.time()

runTimeBubble.append(stop-start)

arr=[]
for i in range (2000):
    arr.append(rn.randint(1,1000000))

start=time.time()
bubbleSort(arr)
stop=time.time()

runTimeBubble.append(stop-start)

arr=[]
for i in range (3000):
    arr.append(rn.randint(1,1000000))

start=time.time()
bubbleSort(arr)
stop=time.time()

runTimeBubble.append(stop-start)

arr=[]
for i in range (4000):
    arr.append(rn.randint(1,1000000))

start=time.time()
```

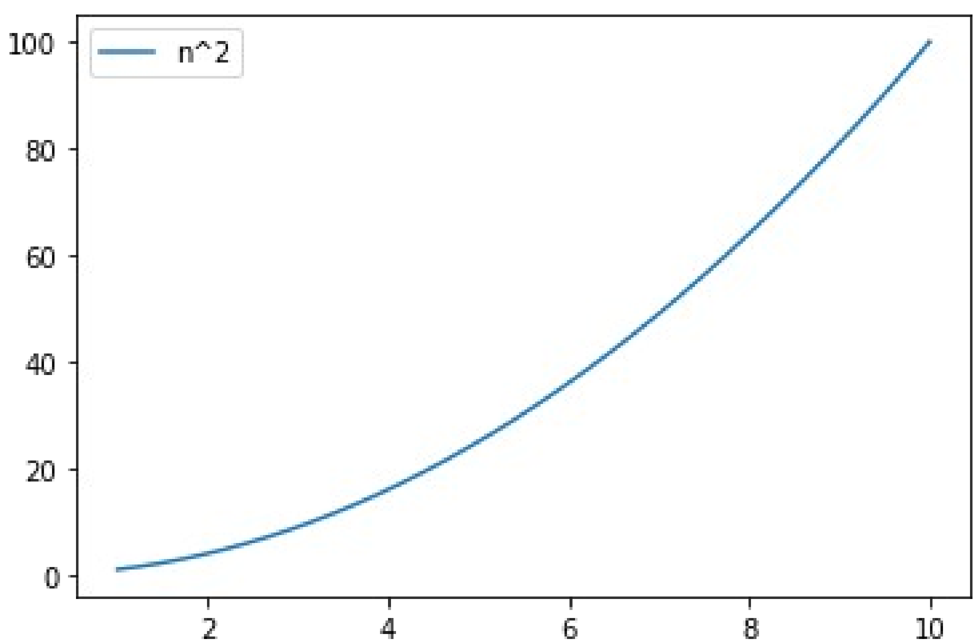
```python
bubbleSort(arr)
stop=time.time()

runTimeBubble.append(stop-start)

arr=[]
for i in range (5000):
    arr.append(rn.randint(1,1000000))

start=time.time()
bubbleSort(arr)
stop=time.time()

runTimeBubble.append(stop-start)

arr=[]
for i in range (6000):
    arr.append(rn.randint(1,1000000))

start=time.time()
bubbleSort(arr)
stop=time.time()

runTimeBubble.append(stop-start)

arr=[]
for i in range (7000):
    arr.append(rn.randint(1,1000000))

start=time.time()
bubbleSort(arr)
stop=time.time()

runTimeBubble.append(stop-start)

arr=[]
for i in range (8000):
    arr.append(rn.randint(1,1000000))

start=time.time()
bubbleSort(arr)
stop=time.time()

runTimeBubble.append(stop-start)

arr=[]
for i in range (9000):
    arr.append(rn.randint(1,1000000))
```

```
start=time.time()
bubbleSort(arr)
stop=time.time()

runTimeBubble.append(stop-start)

arr=[]
for i in range (10000):
    arr.append(rn.randint(1,1000000))

start=time.time()
bubbleSort(arr)
stop=time.time()

runTimeBubble.append(stop-start)

plt.scatter(size,runTime)
plt.show()
```



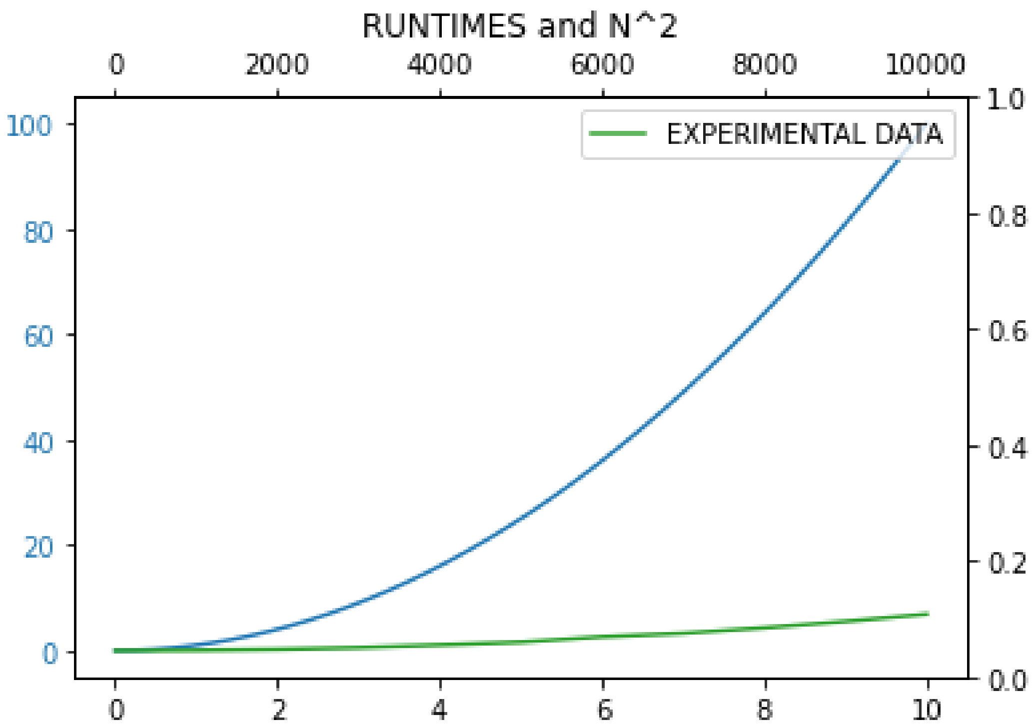

```
x =np.linspace(1,10,1000000)
y=x**2
plt.plot(size,runTimeBubble)
plt.title("Runtime VS Array Size")
plt.xlabel("Size of Array")
plt.ylabel("Time taken to Run")
plt.figure()
plt.plot(x,y,label='n^2')

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






```
t = np.arange(0.01, 10.0, 0.001)
data1 = t**2
fig, ax1 = plt.subplots()
color = 'tab:blue'
ax1.plot(t, data1, color = color,label= 'CURVE FIT DATA')
ax1.tick_params(axis ='y', labelcolor = color)
ax2 = ax1.twinx()
ax2 = ax1.twiny()
```

```
color = 'tab:green'
ax2.plot(size,runTimeBubble, color = color,label='EXPERIMENTAL DATA')
ax2.tick_params(axis ='y', labelcolor = color)
plt.title('RUNTIMES and N^2')
plt.legend()
plt.show()
```




```
#print(runTime)
```

Insertion Sort

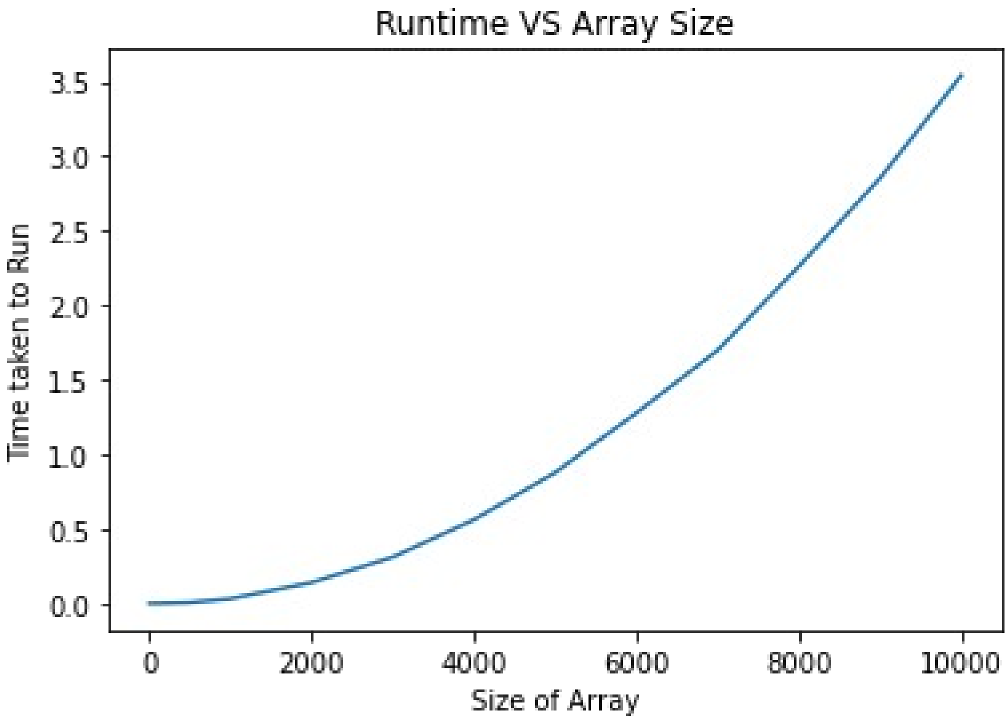
```
def insertionSort(arr):
    for i in range(1,len(arr)):
        key=arr[i]
        j=i-1
        while key<arr[j] and j>=0:
            arr[j+1]=arr[j]
            j-=1
        arr[j+1]=key
    return

runTimeInsertion=[]

arr=[]
for i in range (5):
    arr.append(rn.randint(1,1000000))
```

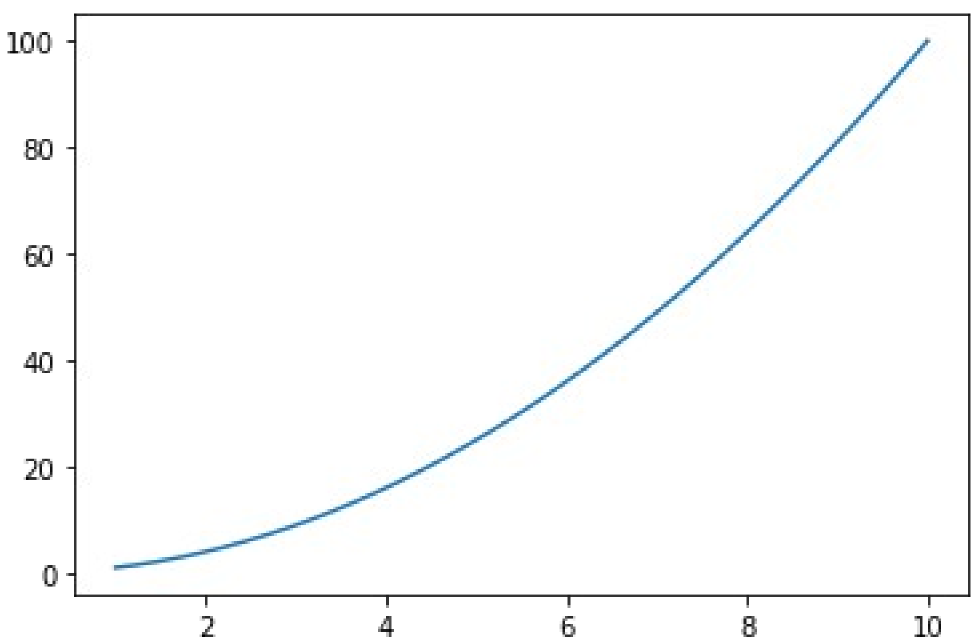
```python
start=time.time()
insertionSort(arr)
stop=time.time()

runTimeInsertion.append(stop-start)

arr=[]
for i in range (10):
    arr.append(rn.randint(1,1000000))

start=time.time()
insertionSort(arr)
stop=time.time()

runTimeInsertion.append(stop-start)


arr=[]
for i in range (100):
    arr.append(rn.randint(1,1000000))

start=time.time()
insertionSort(arr)
stop=time.time()

runTimeInsertion.append(stop-start)

arr=[]
for i in range (500):
    arr.append(rn.randint(1,1000000))

start=time.time()
insertionSort(arr)
stop=time.time()

runTimeInsertion.append(stop-start)

arr=[]
for i in range (1000):
    arr.append(rn.randint(1,1000000))

start=time.time()
insertionSort(arr)
stop=time.time()

runTimeInsertion.append(stop-start)

arr=[]
for i in range (2000):
```

```
    arr.append(rn.randint(1,1000000))

start=time.time()
insertionSort(arr)
stop=time.time()

runTimeInsertion.append(stop-start)

arr=[]
for i in range (3000):
    arr.append(rn.randint(1,1000000))

start=time.time()
insertionSort(arr)
stop=time.time()

runTimeInsertion.append(stop-start)

arr=[]
for i in range (4000):
    arr.append(rn.randint(1,1000000))

start=time.time()
insertionSort(arr)
stop=time.time()

runTimeInsertion.append(stop-start)

arr=[]
for i in range (5000):
    arr.append(rn.randint(1,1000000))

start=time.time()
insertionSort(arr)
stop=time.time()

runTimeInsertion.append(stop-start)

arr=[]
for i in range (6000):
    arr.append(rn.randint(1,1000000))

start=time.time()
insertionSort(arr)
stop=time.time()

runTimeInsertion.append(stop-start)

arr=[]
```

```python
for i in range (7000):
    arr.append(rn.randint(1,1000000))

start=time.time()
insertionSort(arr)
stop=time.time()

runTimeInsertion.append(stop-start)

arr=[]
for i in range (8000):
    arr.append(rn.randint(1,1000000))

start=time.time()
insertionSort(arr)
stop=time.time()

runTimeInsertion.append(stop-start)

arr=[]
for i in range (9000):
    arr.append(rn.randint(1,1000000))

start=time.time()
insertionSort(arr)
stop=time.time()

runTimeInsertion.append(stop-start)

arr=[]
for i in range (10000):
    arr.append(rn.randint(1,1000000))

start=time.time()
insertionSort(arr)
stop=time.time()

runTimeInsertion.append(stop-start)



x =np.linspace(1,10,1000000)
y=x**2
plt.plot(size,runTimeInsertion)
plt.title("Runtime VS Array Size")
plt.xlabel("Size of Array")
plt.ylabel("Time taken to Run")
plt.figure()
```

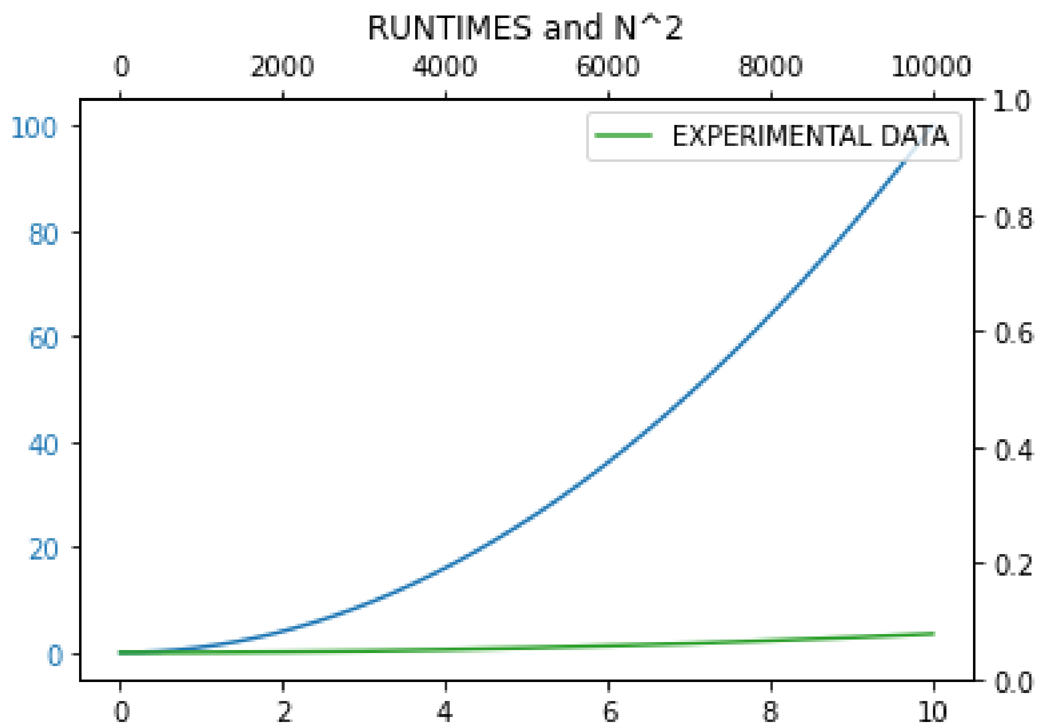
```
plt.plot(x,y,label='n^2')
plt.show()
```






```
t = np.arange(0.01, 10.0, 0.001)
data1 = t**2
fig, ax1 = plt.subplots()
color = 'tab:blue'
ax1.plot(t, data1, color = color,label= 'CURVE FIT DATA')
```

```
ax1.tick_params(axis ='y', labelcolor = color)
ax2 = ax1.twinx()
ax2 = ax1.twiny()
color = 'tab:green'
ax2.plot(size,runTimeInsertion, color = color,label='EXPERIMENTAL
DATA')
ax2.tick_params(axis ='y', labelcolor = color)
plt.title('RUNTIMES and N^2')
plt.legend()
plt.show()
```



Selection Sort

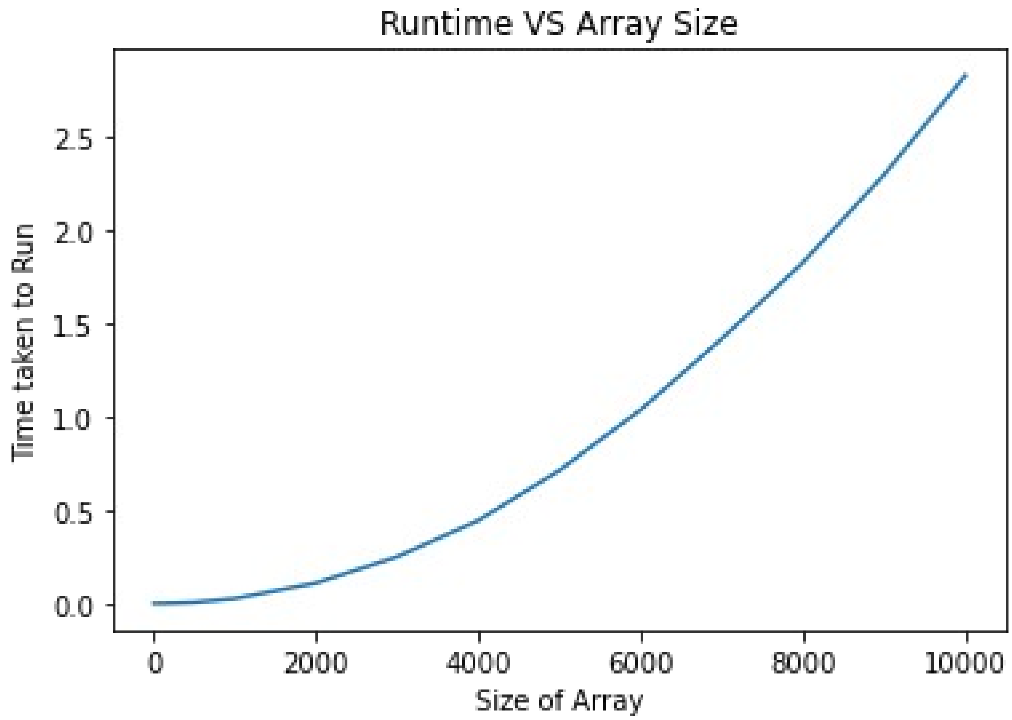
```
def selectionSort(arr):
    for i in range(0,len(arr)-1):
        min_index=i
        for j in range(i,len(arr)):
            if arr[min_index]>arr[j]:
                min_index=j
        temp=arr[min_index]
        arr[min_index]=arr[i]
        arr[i]=temp
    return

runTimeSelection=[]

arr=[]
```

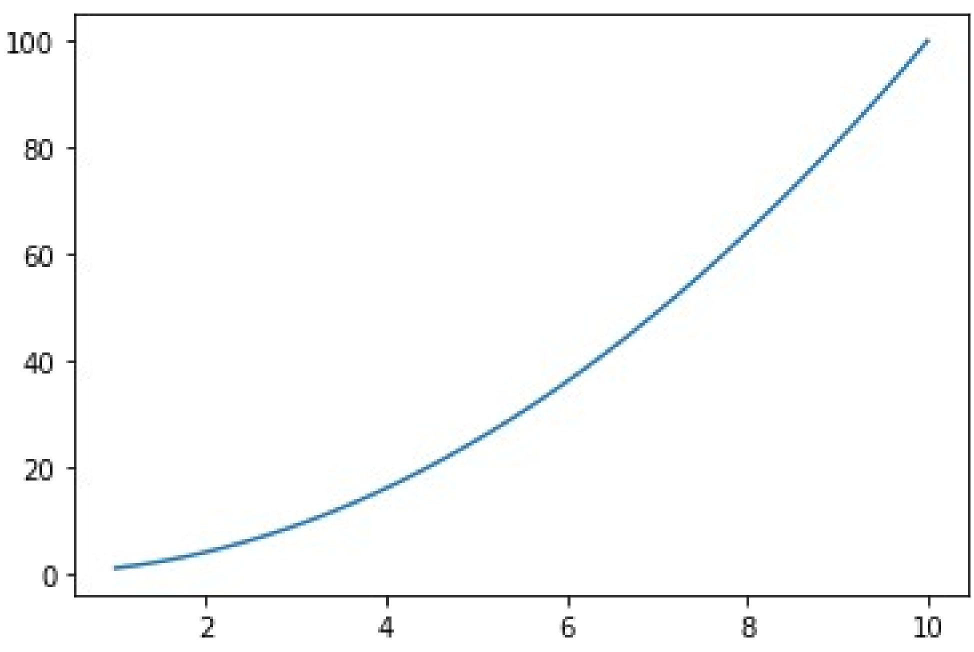
```python
for i in range (5):
    arr.append(rn.randint(1,1000000))

start=time.time()
selectionSort(arr)
stop=time.time()

runTimeSelection.append(stop-start)

arr=[]
for i in range (10):
    arr.append(rn.randint(1,1000000))

start=time.time()
selectionSort(arr)
stop=time.time()

runTimeSelection.append(stop-start)


arr=[]
for i in range (100):
    arr.append(rn.randint(1,1000000))

start=time.time()
selectionSort(arr)
stop=time.time()

runTimeSelection.append(stop-start)

arr=[]
for i in range (500):
    arr.append(rn.randint(1,1000000))

start=time.time()
selectionSort(arr)
stop=time.time()

runTimeSelection.append(stop-start)

arr=[]
for i in range (1000):
    arr.append(rn.randint(1,1000000))

start=time.time()
selectionSort(arr)
stop=time.time()

runTimeSelection.append(stop-start)
```

```
arr=[]
for i in range (2000):
    arr.append(rn.randint(1,1000000))

start=time.time()
selectionSort(arr)
stop=time.time()

runTimeSelection.append(stop-start)

arr=[]
for i in range (3000):
    arr.append(rn.randint(1,1000000))

start=time.time()
selectionSort(arr)
stop=time.time()

runTimeSelection.append(stop-start)

arr=[]
for i in range (4000):
    arr.append(rn.randint(1,1000000))

start=time.time()
selectionSort(arr)
stop=time.time()

runTimeSelection.append(stop-start)

arr=[]
for i in range (5000):
    arr.append(rn.randint(1,1000000))

start=time.time()
selectionSort(arr)
stop=time.time()

runTimeSelection.append(stop-start)

arr=[]
for i in range (6000):
    arr.append(rn.randint(1,1000000))

start=time.time()
selectionSort(arr)
stop=time.time()
```

```
runTimeSelection.append(stop-start)

arr=[]
for i in range (7000):
    arr.append(rn.randint(1,1000000))

start=time.time()
selectionSort(arr)
stop=time.time()

runTimeSelection.append(stop-start)

arr=[]
for i in range (8000):
    arr.append(rn.randint(1,1000000))

start=time.time()
selectionSort(arr)
stop=time.time()

runTimeSelection.append(stop-start)

arr=[]
for i in range (9000):
    arr.append(rn.randint(1,1000000))

start=time.time()
selectionSort(arr)
stop=time.time()

runTimeSelection.append(stop-start)

arr=[]
for i in range (10000):
    arr.append(rn.randint(1,1000000))

start=time.time()
selectionSort(arr)
stop=time.time()

runTimeSelection.append(stop-start)

x=np.linspace(1,10,1000000)
y=x**2
plt.plot(size,runTimeSelection)
plt.title("Runtime VS Array Size")
plt.xlabel("Size of Array")
plt.ylabel("Time taken to Run")
plt.figure()
```

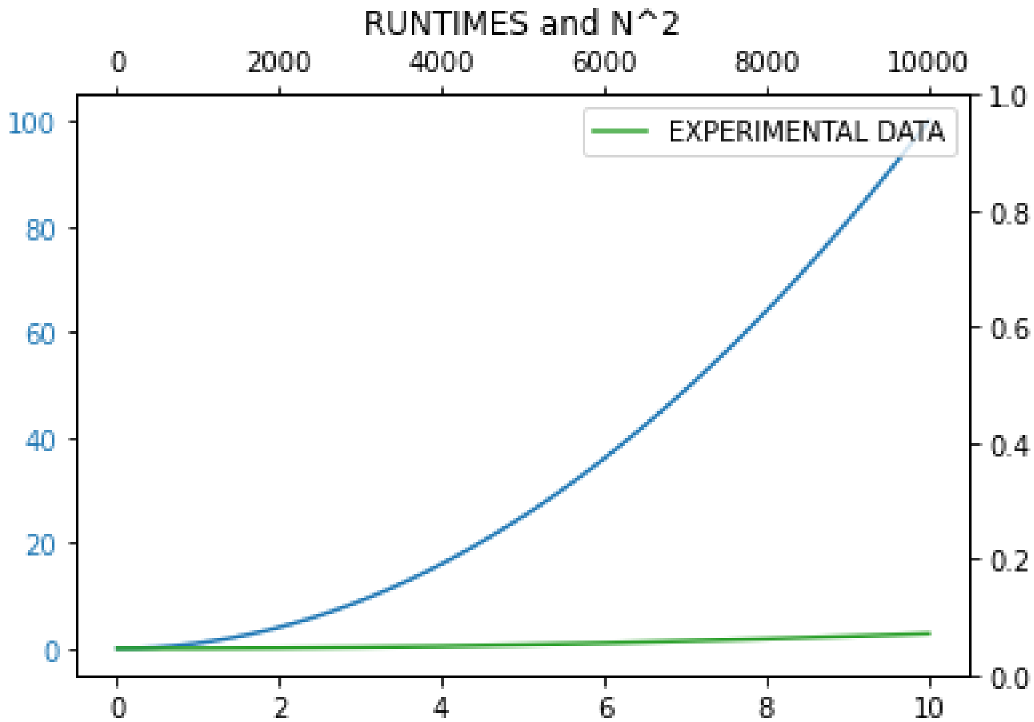
```
plt.plot(x,y,label='n^2')
plt.show()
```





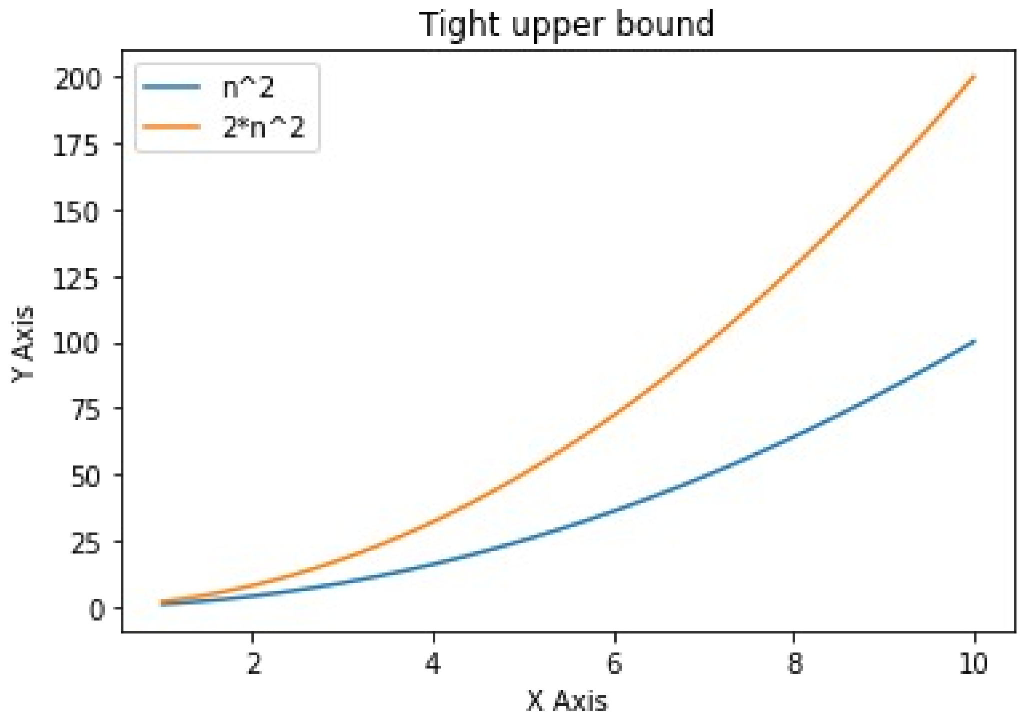

```
t = np.arange(0.01, 10.0, 0.001)
data1 = t**2
fig, ax1 = plt.subplots()
color = 'tab:blue'
ax1.plot(t, data1, color = color,label= 'CURVE FIT DATA')
```

```
ax1.tick_params(axis ='y', labelcolor = color)
ax2 = ax1.twinx()
ax2 = ax1.twiny()
color = 'tab:green'
ax2.plot(size,runTimeSelection, color = color,label='EXPERIMENTAL
DATA')
ax2.tick_params(axis ='y', labelcolor = color)
plt.title('RUNTIMES and N^2')
plt.legend()
plt.show()
```

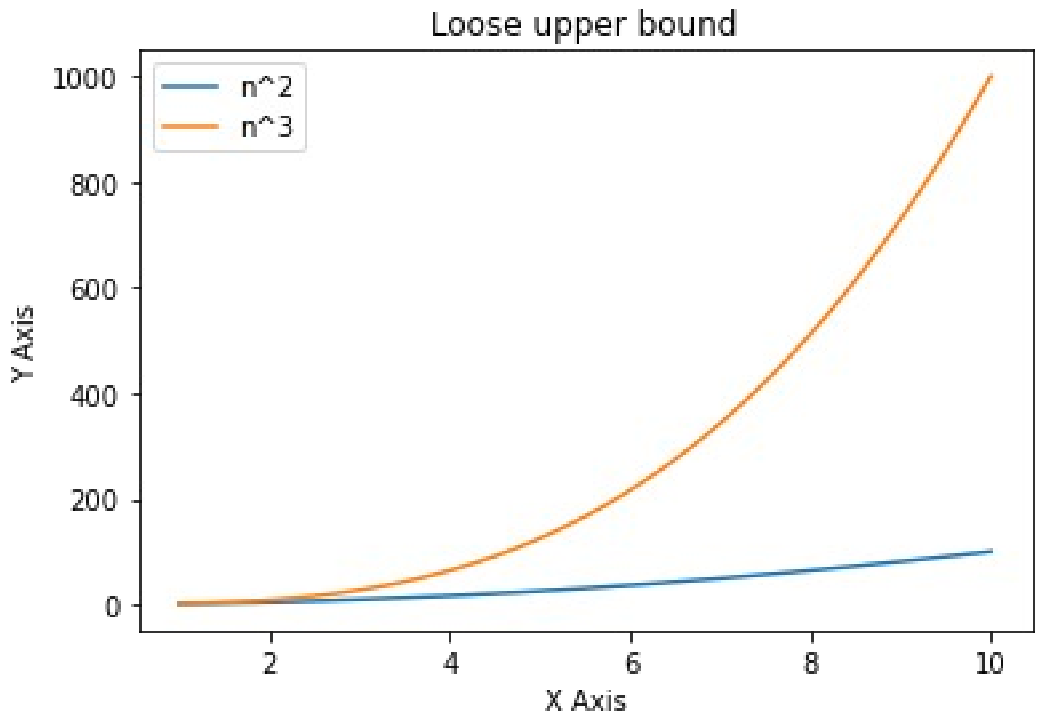


Tight and loose bounds for bubble, selection and quick sorts

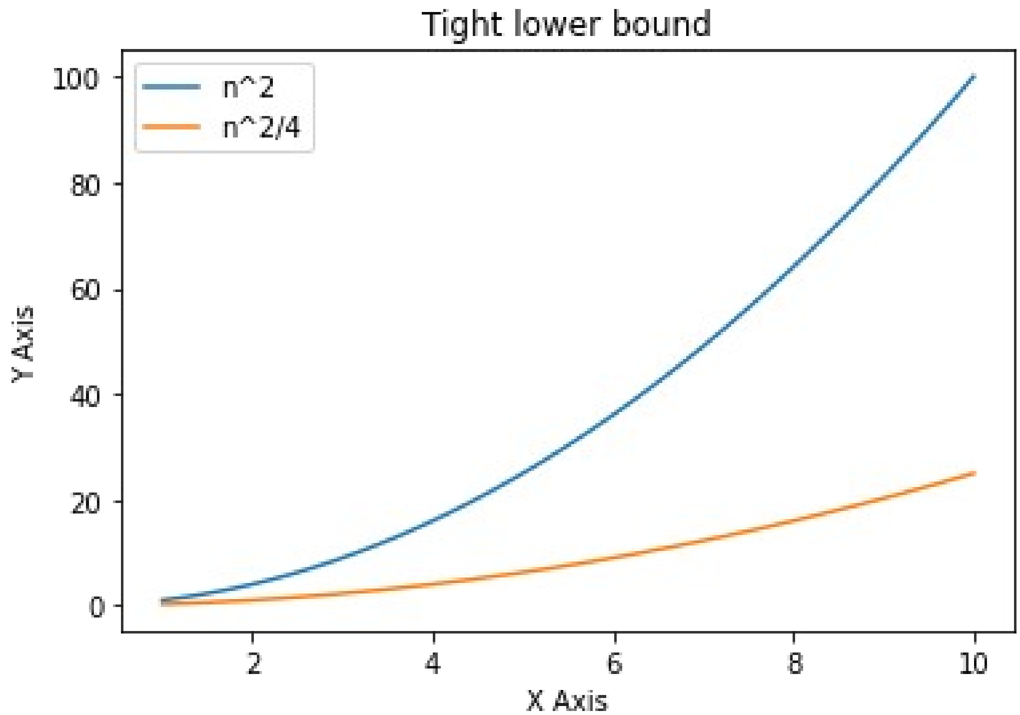
```
x =np.linspace(1,10,1000000)
y =x**2
y1=2*y
plt.plot(x,y,label='n^2')
plt.plot(x,y1,label='2*n^2')
plt.title("Tight upper bound")
plt.xlabel('X Axis')
plt.ylabel('Y Axis')
plt.legend()
plt.show()
```



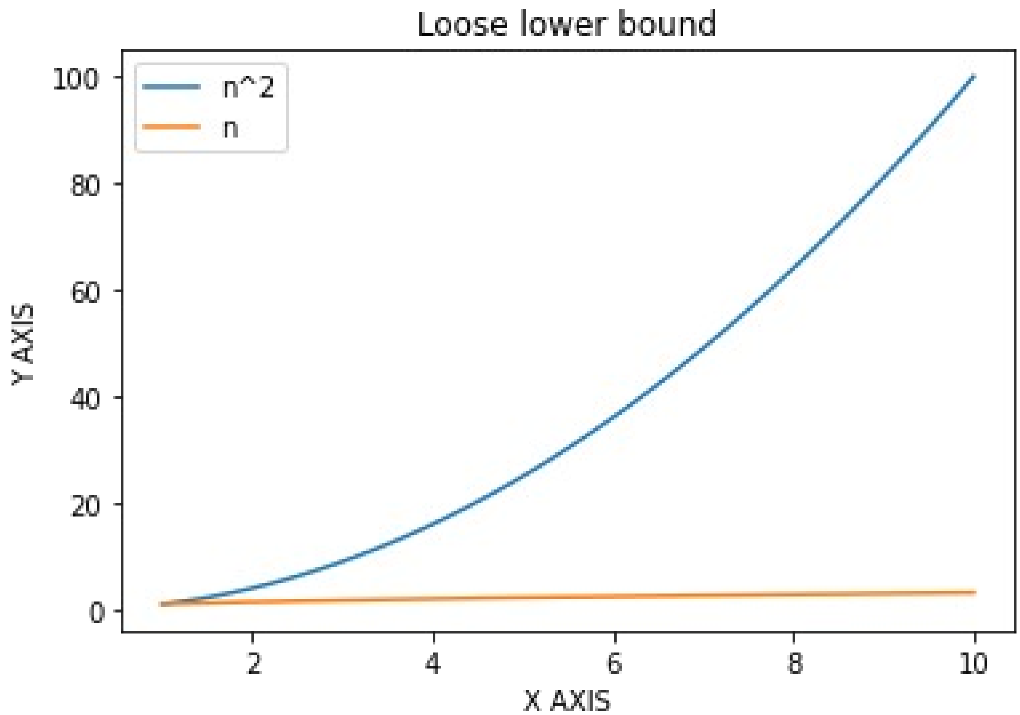

```
x =np.linspace(1,10,1000000)
y =x**2
y1=x**3
plt.plot(x,y,label='n^2')
plt.plot(x,y1,label='n^3')
plt.title("Loose upper bound")
plt.xlabel('X Axis')
plt.ylabel('Y Axis')
plt.legend()
plt.show()
```




```
x =np.linspace(1,10,1000000)
a=7/8
y =x**2
y1=x**2/4
plt.plot(x,y,label='n^2')
plt.plot(x,y1,label='n^2/4')
plt.title("Tight lower bound")
plt.xlabel('X Axis')
plt.ylabel('Y Axis')
plt.legend()
plt.show()
```

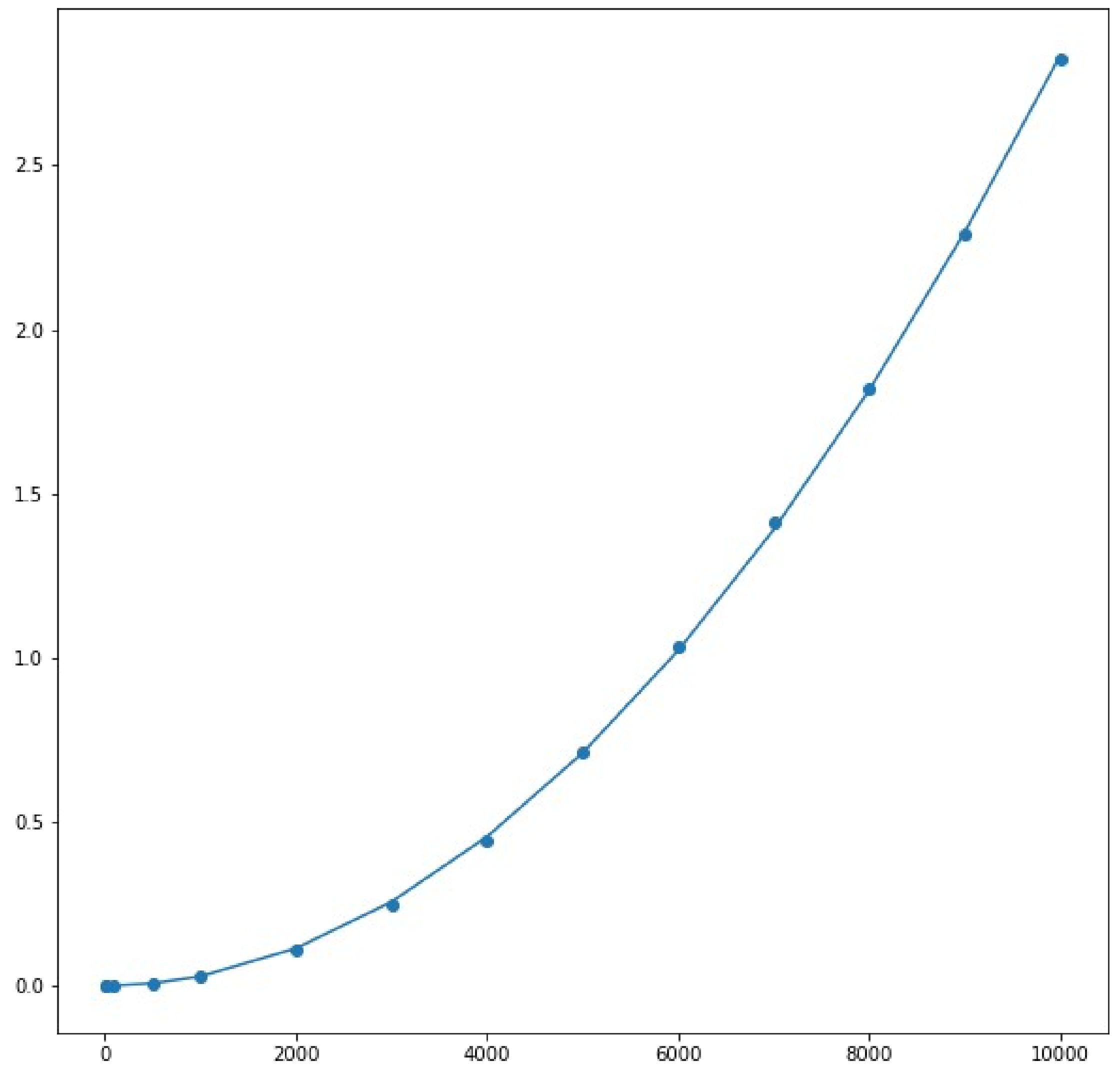



```
x =np.linspace(1,10,1000000)
a=1/2
y =x**2
y1=x**a
plt.plot(x,y,label='n^2')
plt.plot(x,y1,label='n')
plt.title("Loose lower bound")
plt.xlabel('X AXIS')
plt.ylabel('Y AXIS')
plt.legend()
plt.show()
```




```
def func(x,a,b):
    arr=[]
    for i in range(len(x)):
        arr.append(a*x[i]**2+b)
    return arr

params,
params_covariance=optimize.curve_fit(func,size,runTimeSelection)
fig , ax = plt.subplots(figsize=(10,10))
ax.scatter(size,runTimeSelection)
ax.plot(size,func(size,params[0],params[1]))
```

```
[<matplotlib.lines.Line2D at 0x13d1da3e220>]
```



Merge Sort

```
def merge(a,b,arr):
    i=0
    j=0
    k=0

    while (i<len(a) and j<len(b)):
```

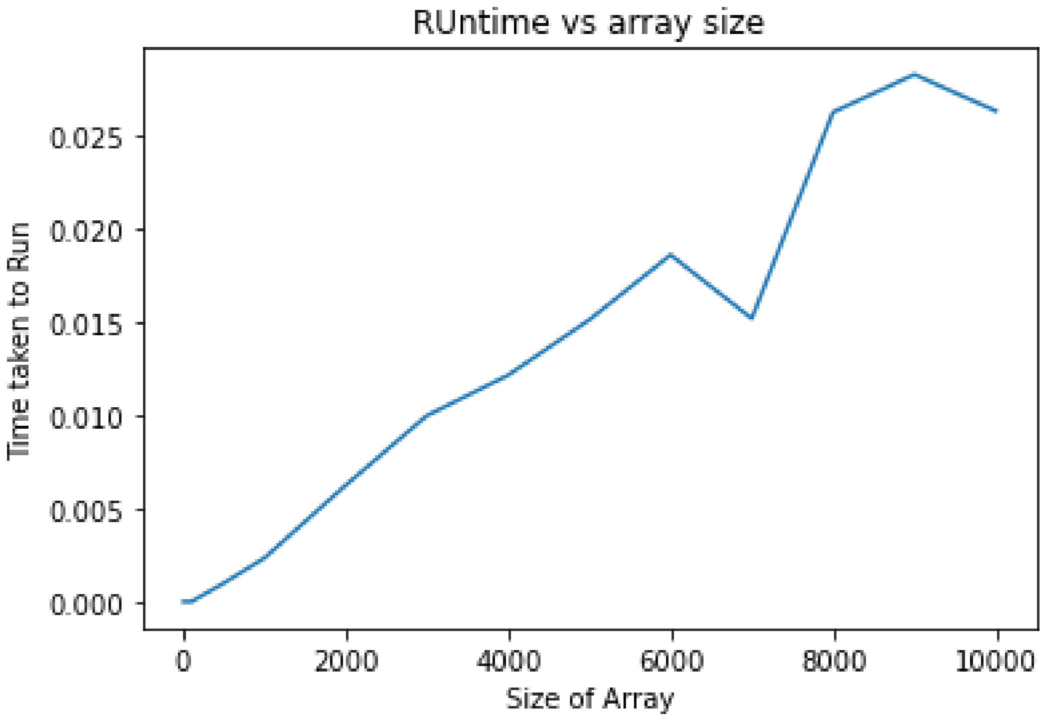
```
        if (a[i]<=b[j]):
            arr[k]=a[i]
            i+=1
        else:
            arr[k]=b[j]
            j+=1
        k+=1

    while (i<len(a)):
        arr[k]=a[i]
        i+=1
        k+=1

    while (j<len(b)):
        arr[k]=b[j]
        j+=1
        k+=1

def mergeSort(arr):
    if len(arr)<=1:
        return

    mid=len(arr)//2
    low=arr[:mid]
    high=arr[mid:]

    mergeSort(low)
    mergeSort(high)
    merge(low,high,arr)

runTimeMerge=[]

arr=[]
for i in range (5):
    arr.append(rn.randint(1,1000000))

start=time.time()
mergeSort(arr)
stop=time.time()

runTimeMerge.append(stop-start)

arr=[]
for i in range (10):
    arr.append(rn.randint(1,1000000))

start=time.time()
mergeSort(arr)
stop=time.time()
```

```
runTimeMerge.append(stop-start)


arr=[]
for i in range (100):
    arr.append(rn.randint(1,1000000))

start=time.time()
mergeSort(arr)
stop=time.time()

runTimeMerge.append(stop-start)

arr=[]
for i in range (500):
    arr.append(rn.randint(1,1000000))

start=time.time()
mergeSort(arr)
stop=time.time()

runTimeMerge.append(stop-start)

arr=[]
for i in range (1000):
    arr.append(rn.randint(1,1000000))

start=time.time()
mergeSort(arr)
stop=time.time()

runTimeMerge.append(stop-start)

arr=[]
for i in range (2000):
    arr.append(rn.randint(1,1000000))

start=time.time()
mergeSort(arr)
stop=time.time()

runTimeMerge.append(stop-start)

arr=[]
for i in range (3000):
    arr.append(rn.randint(1,1000000))

start=time.time()
mergeSort(arr)
```

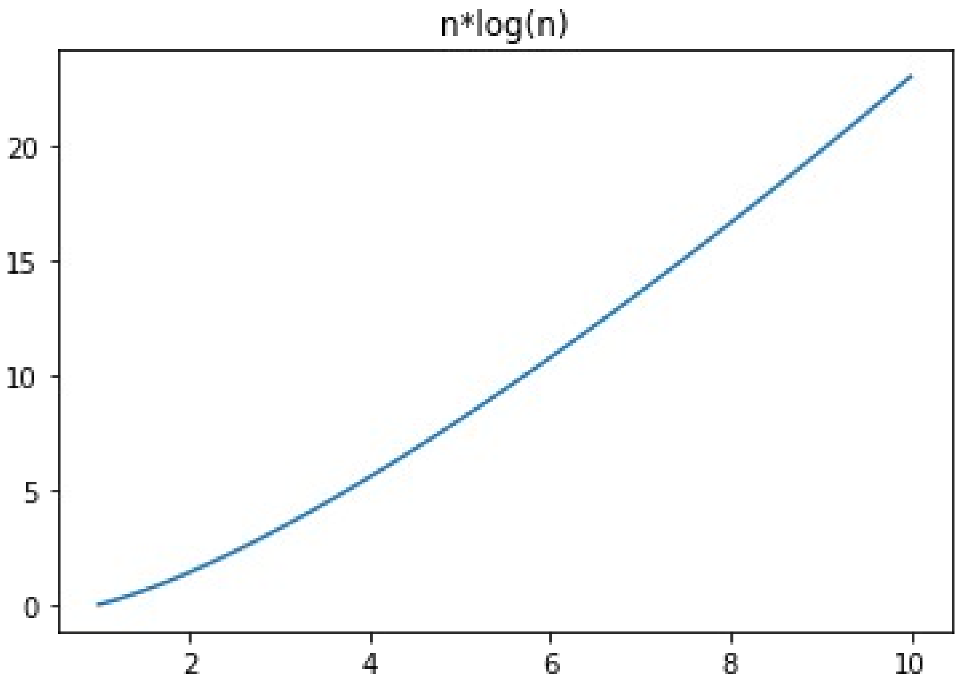
```
stop=time.time()

runTimeMerge.append(stop-start)

arr=[]
for i in range (4000):
    arr.append(rn.randint(1,1000000))

start=time.time()
mergeSort(arr)
stop=time.time()

runTimeMerge.append(stop-start)

arr=[]
for i in range (5000):
    arr.append(rn.randint(1,1000000))

start=time.time()
mergeSort(arr)
stop=time.time()

runTimeMerge.append(stop-start)

arr=[]
for i in range (6000):
    arr.append(rn.randint(1,1000000))

start=time.time()
mergeSort(arr)
stop=time.time()

runTimeMerge.append(stop-start)

arr=[]
for i in range (7000):
    arr.append(rn.randint(1,1000000))

start=time.time()
mergeSort(arr)
stop=time.time()

runTimeMerge.append(stop-start)

arr=[]
for i in range (8000):
    arr.append(rn.randint(1,1000000))

start=time.time()
```

```
mergeSort(arr)
stop=time.time()

runTimeMerge.append(stop-start)

arr=[]
for i in range (9000):
    arr.append(rn.randint(1,1000000))

start=time.time()
mergeSort(arr)
stop=time.time()

runTimeMerge.append(stop-start)

arr=[]
for i in range (10000):
    arr.append(rn.randint(1,1000000))

start=time.time()
mergeSort(arr)
stop=time.time()

runTimeMerge.append(stop-start)

plt.plot(size,runTimeMerge)
plt.xlabel("Size of Array")
plt.ylabel("Time taken to Run")
plt.title("RUntime vs array size")
```

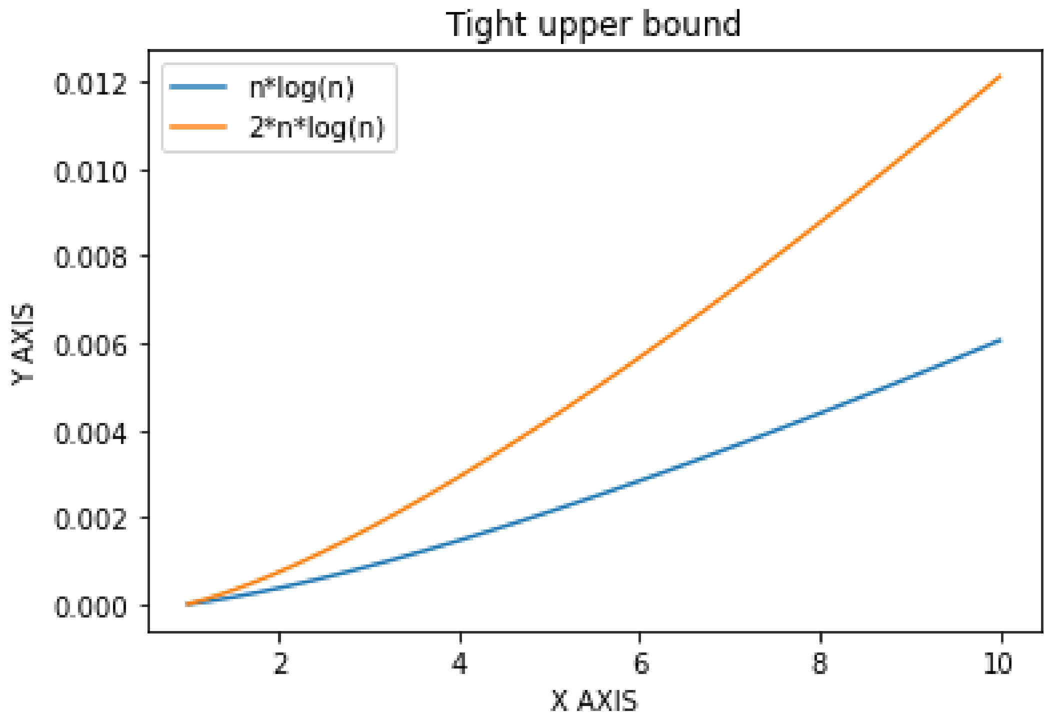
```
Text(0.5, 1.0, 'RUntime vs array size')
```



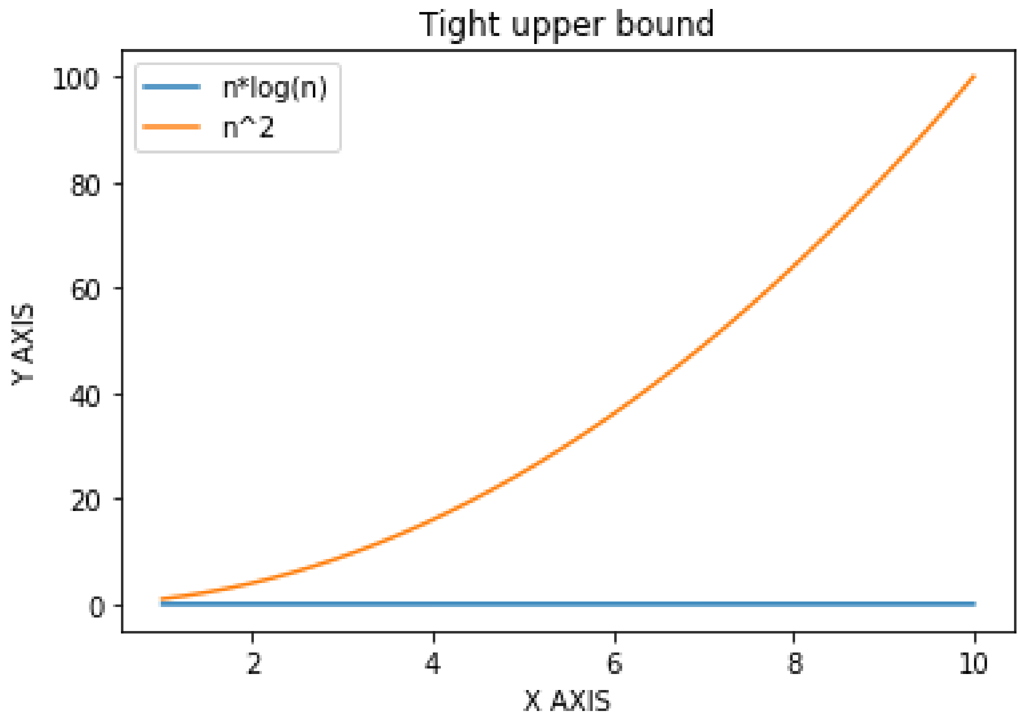

```
x =np.linspace(1,10,1000000)
y =x*np.log(x)
y1=y*2
y2=x**2
plt.figure()
plt.title(" n*log(n) ")
plt.plot(x,y)
plt.show()
```



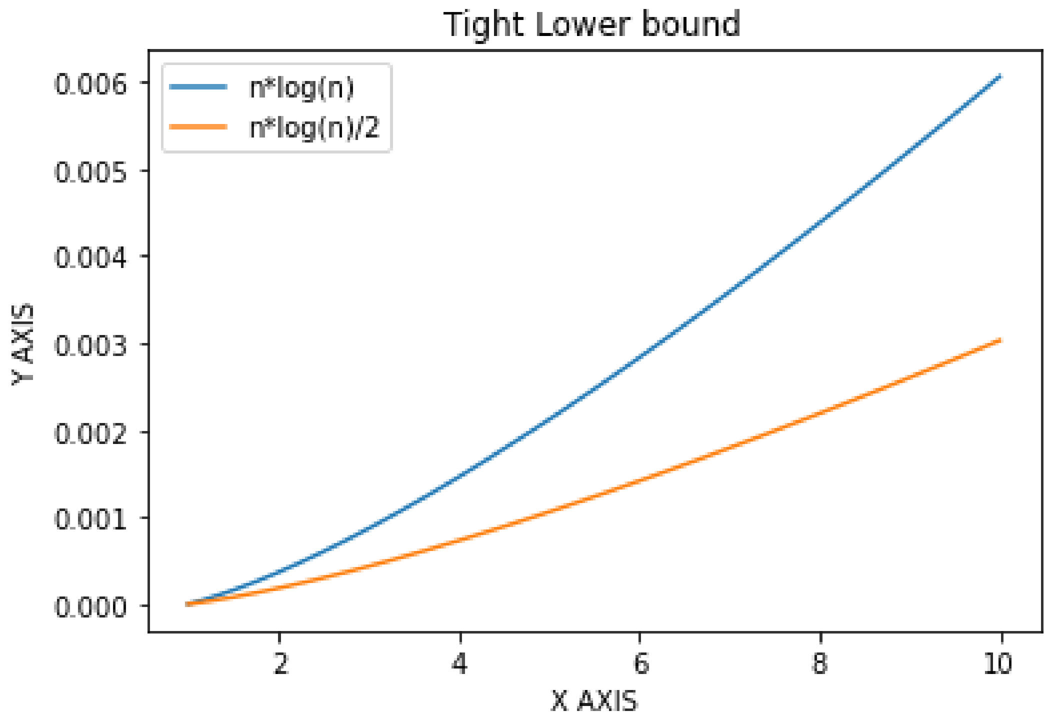

```
x =np.linspace(1,10,1000000)
y =x*np.log(x)/3800
y1=2*y
plt.plot(x,y,label='n*log(n)')
plt.plot(x,y1,label='2*n*log(n)')
plt.title("Tight upper bound")
plt.xlabel('X AXIS')
plt.ylabel('Y AXIS')
plt.legend()
plt.show()
```



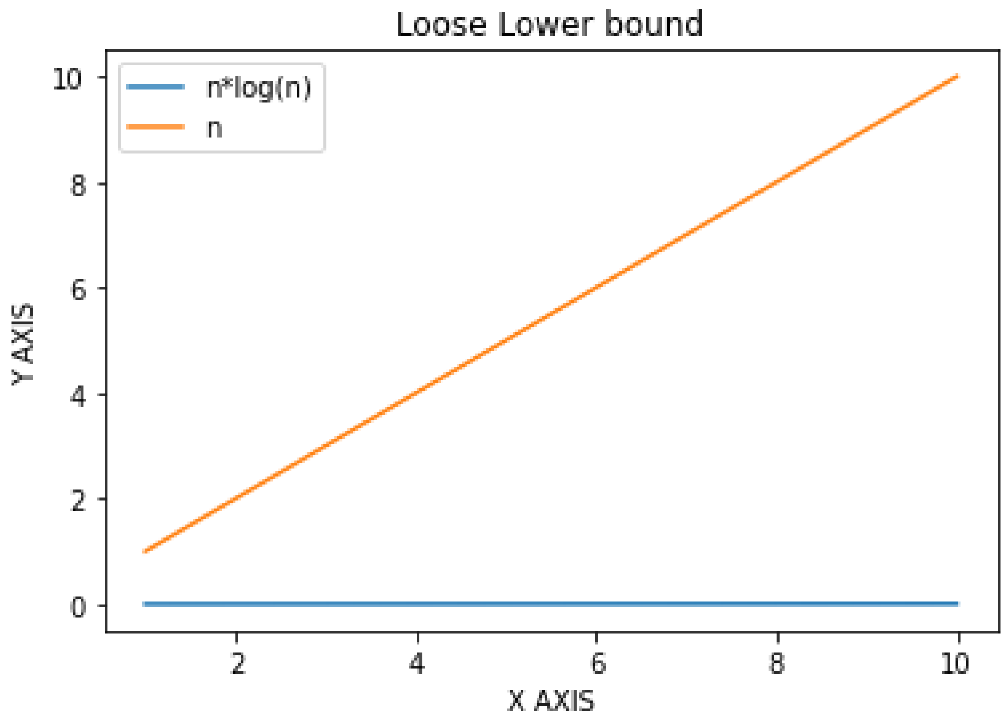

```
x =np.linspace(1,10,1000000)
y =x*np.log(x)/3800
y1=x**2
plt.plot(x,y,label='n*log(n)')
plt.plot(x,y1,label='n^2')
plt.title("Tight upper bound")
plt.xlabel('X AXIS')
plt.ylabel('Y AXIS')
plt.legend()
plt.show()
```



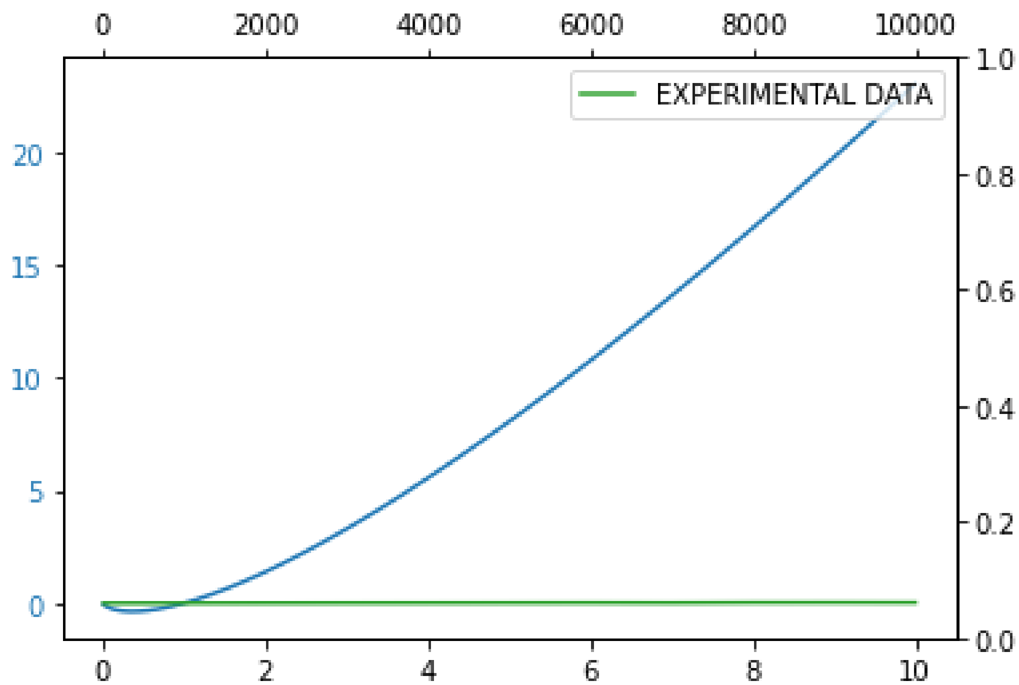

```
x =np.linspace(1,10,1000000)
y =x*np.log(x)/3800
y1=y/2
plt.plot(x,y,label='n*log(n)')
plt.plot(x,y1,label='n*log(n)/2')
plt.title("Tight Lower bound")
plt.xlabel('X AXIS')
plt.ylabel('Y AXIS')
plt.legend()
plt.show()
```



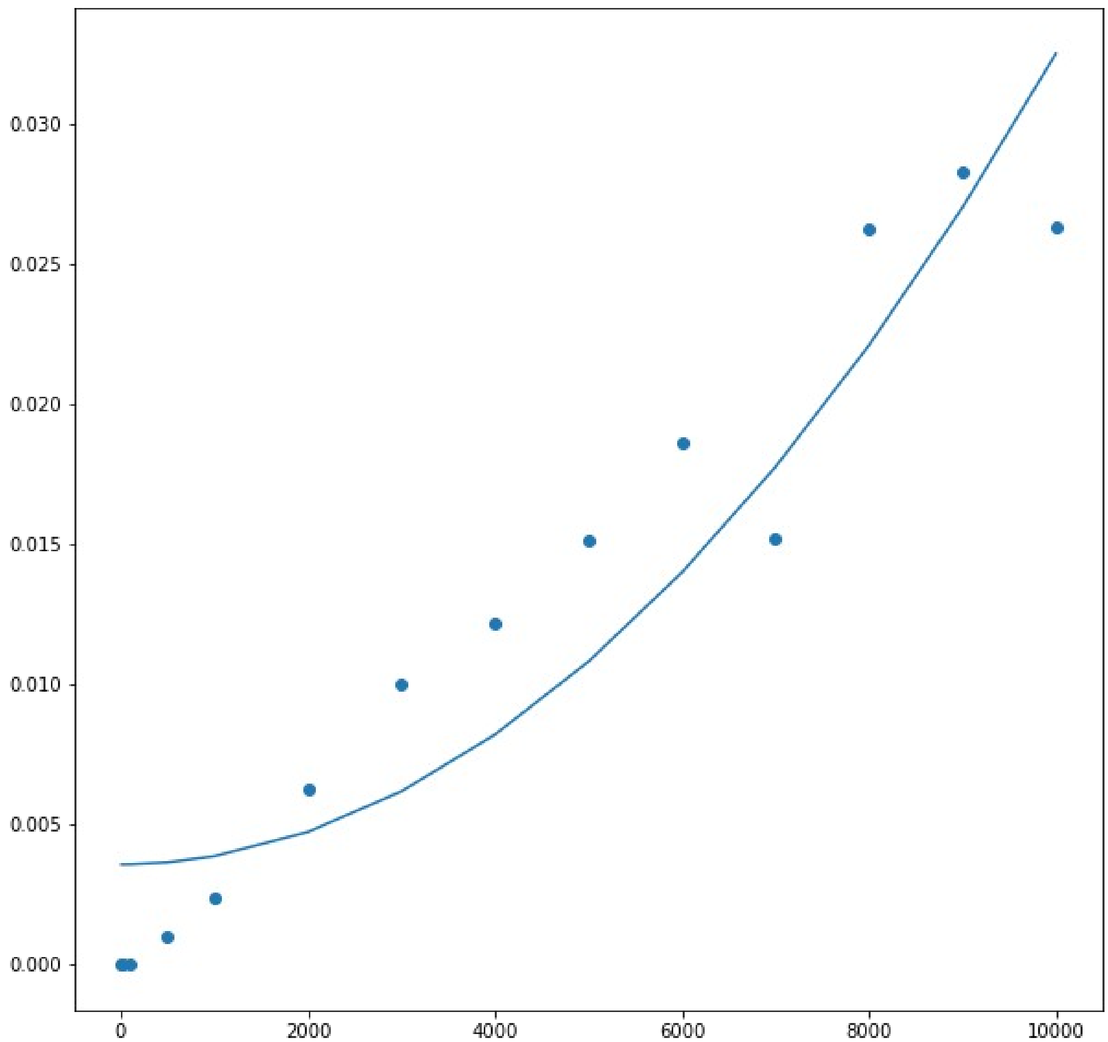

```
x =np.linspace(1,10,1000000)
y =x*np.log(x)/3800
y1=x
plt.plot(x,y,label='n*log(n)')
plt.plot(x,y1,label='n')
plt.title("Loose Lower bound")
plt.xlabel('X AXIS')
plt.ylabel('Y AXIS')
plt.legend()
plt.show()
```




```
t = np.arange(0.01, 10.0, 0.001)
data1 = t*np.log(t)
fig, ax1 = plt.subplots()
color = 'tab:blue'
ax1.plot(t, data1, color = color,label= 'CURVE FIT DATA')
ax1.tick_params(axis ='y', labelcolor = color)
ax2 = ax1.twinx()
ax2 = ax1.twiny()
color = 'tab:green'
ax2.plot(size,runTimeMerge, color = color,label='EXPERIMENTAL DATA')
ax2.tick_params(axis ='y', labelcolor = color)
plt.legend()
plt.show()
```



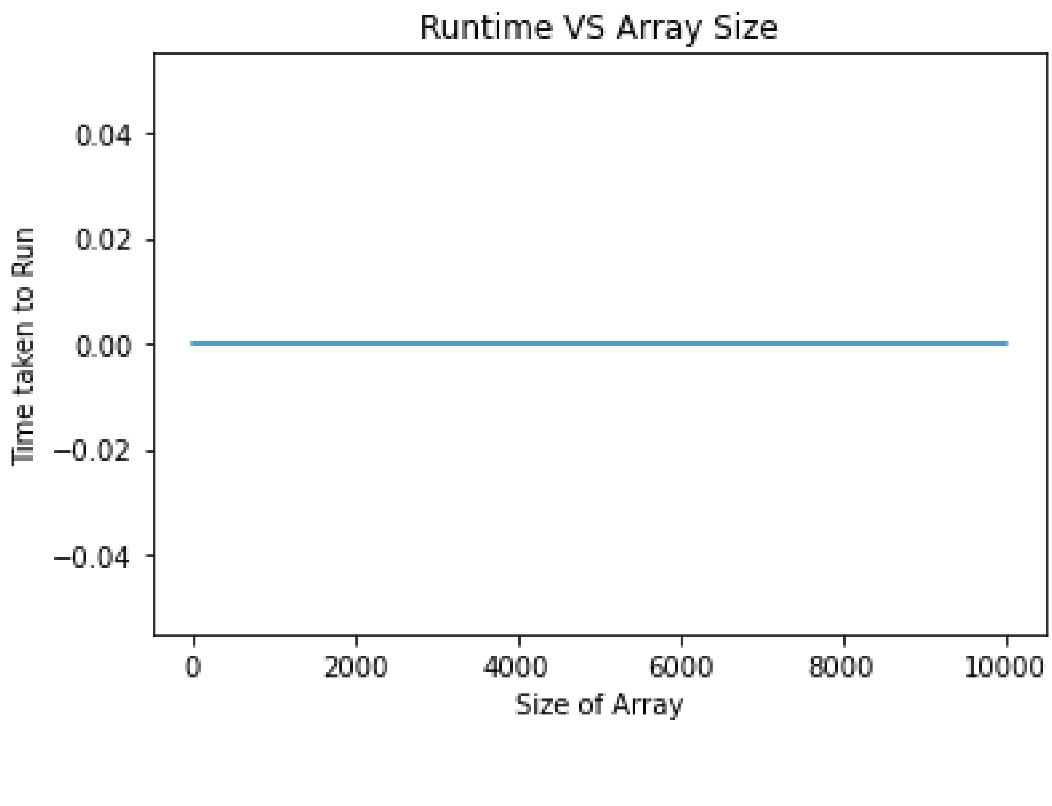

```
params,params_covariance=optimize.curve_fit(func,size,runTimeMerge)
fig,ax = plt.subplots(figsize=(10,10))
ax.scatter(size,runTimeMerge)
ax.plot(size,func(size,params[0],params[1]))
plt.show()
```

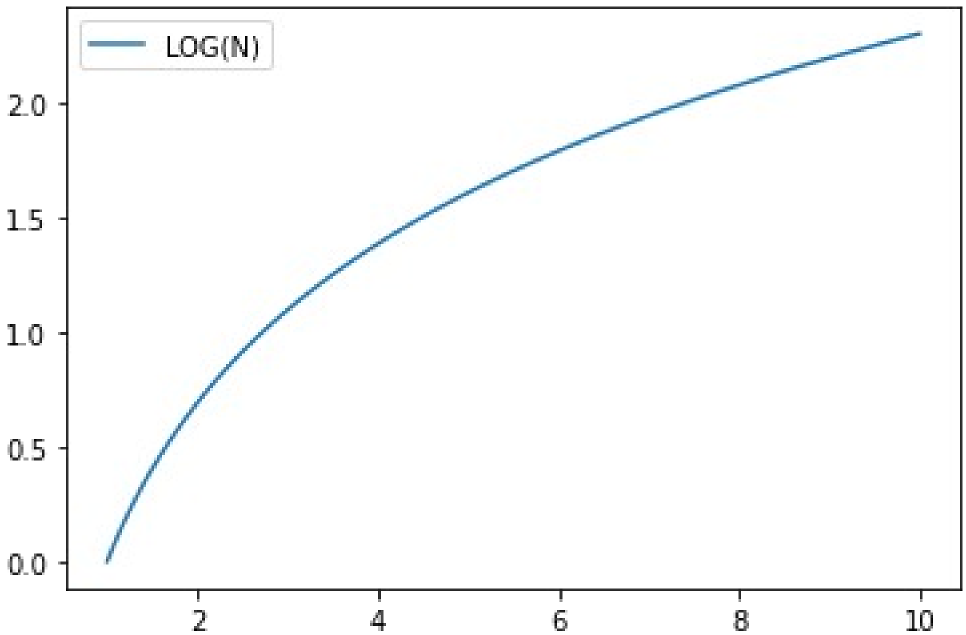


Binary search

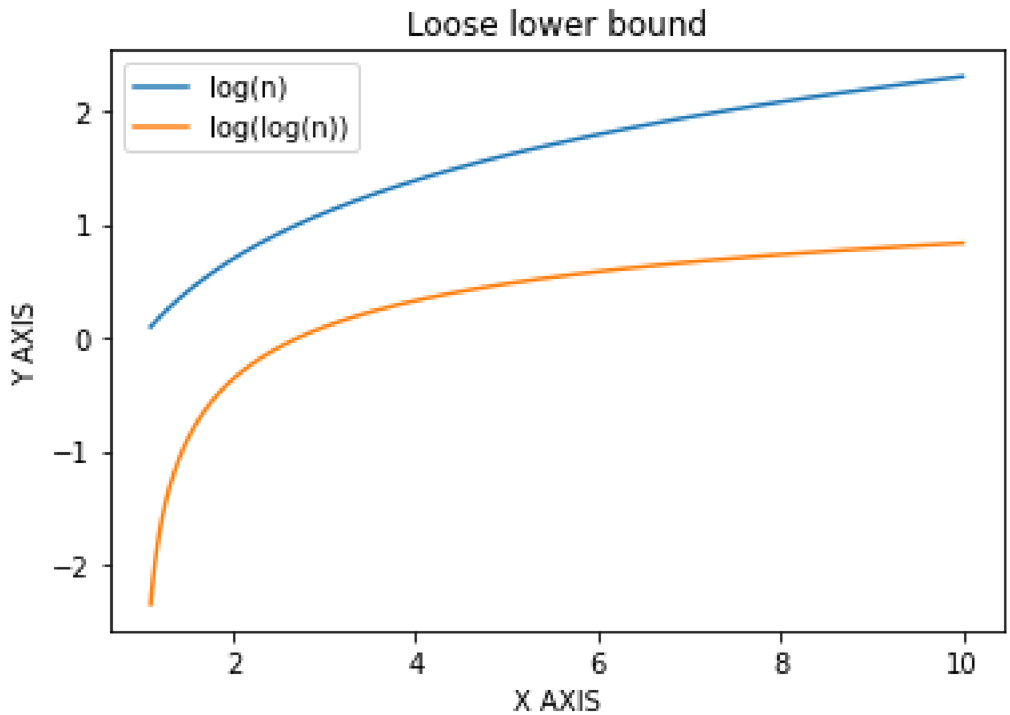
```
def BinarySearch(arr, min, max, val):
    if min<=max:
        mid=(max+min)//2
        if arr[mid]==val:
            return mid
        elif arr[mid]>val:
            return BinarySearch(arr,min,mid-1,val)
        else:
            return BinarySearch(arr,mid+1,max,val)

    else:
        return -1

N=[10,100,1000,2000,3000,4000,5000,6000,7000,8000,9000,10000]
T3=[]
for i in range(len(N)):
```

```
    B=[]
    for j in range(N[i]):
        r =rn.randint(1,100000000)
        B.append(r)
        x1=rn.randint(1,1000000)

    start=time.time()
    result=BinarySearch(B, 0, len(B)-1,x1)
    end=time.time()
    t1=end-start
    T3.append(t1)

x =np.linspace(1,10,1000000)
y=np.log(x)
plt.plot(N,T3)
plt.title("Runtime VS Array Size")
plt.xlabel("Size of Array")
plt.ylabel("Time taken to Run")
plt.figure()
plt.plot(x,y,label='LOG(N)')
plt.legend()
plt.show()
```

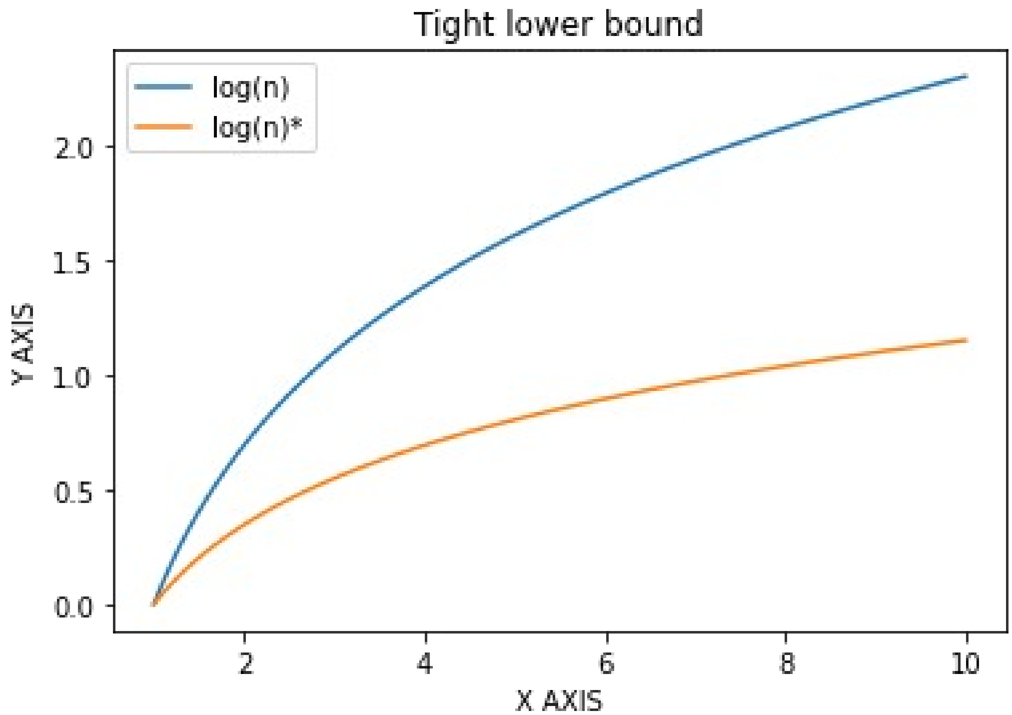




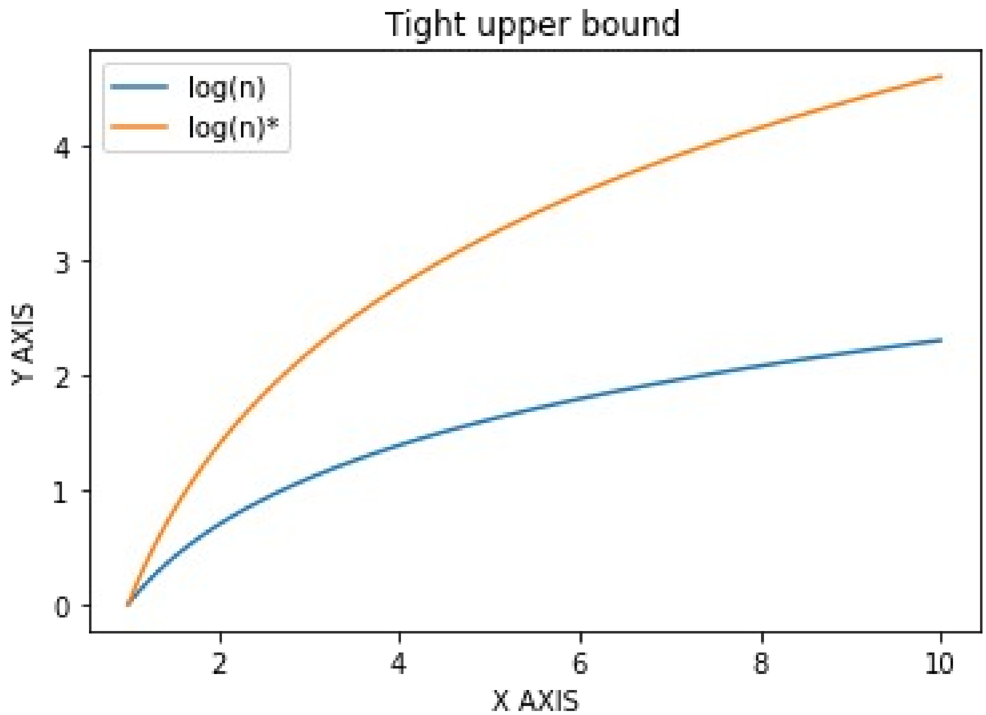

```
x =np.linspace(1.1,10,1000000)
a=1/2
y =np.log(x)
y1=np.log(y)
plt.plot(x,y,label='log(n)')
plt.plot(x,y1,label='log(log(n))')
plt.title("Loose lower bound")
plt.xlabel('X AXIS')
plt.ylabel('Y AXIS')
plt.legend()
plt.show()
```



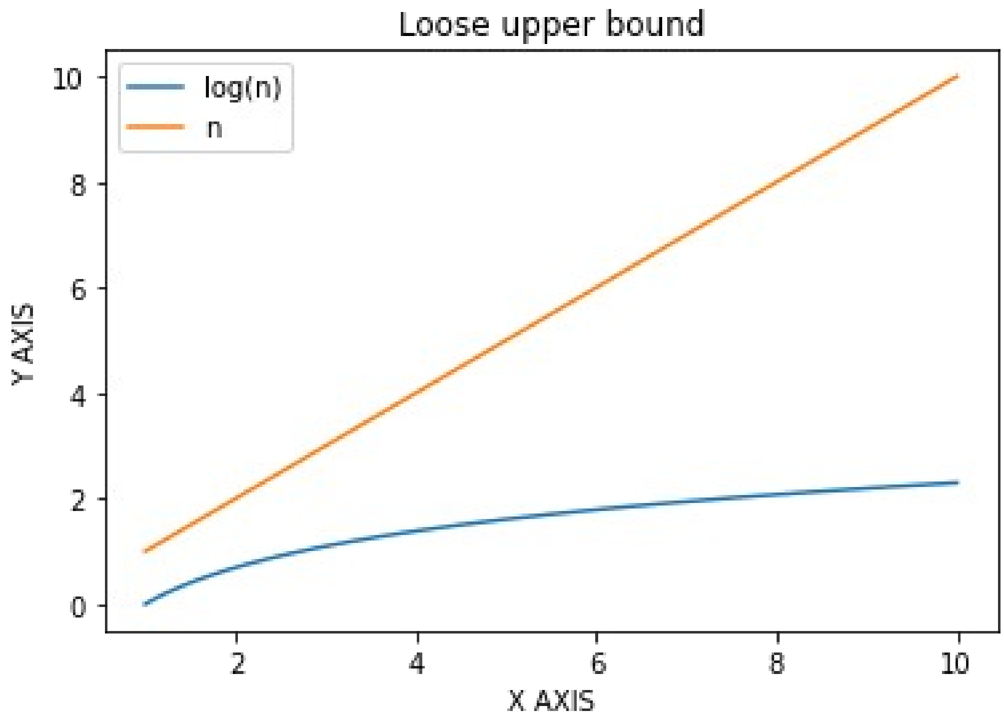

```
x =np.linspace(1,10,1000000)
a=1/2
y =np.log(x)
y1=y/2
plt.plot(x,y,label='log(n)')
plt.plot(x,y1,label='log(n)*')
plt.title("Tight lower bound")
plt.xlabel('X AXIS')
plt.ylabel('Y AXIS')
plt.legend()
plt.show()
```



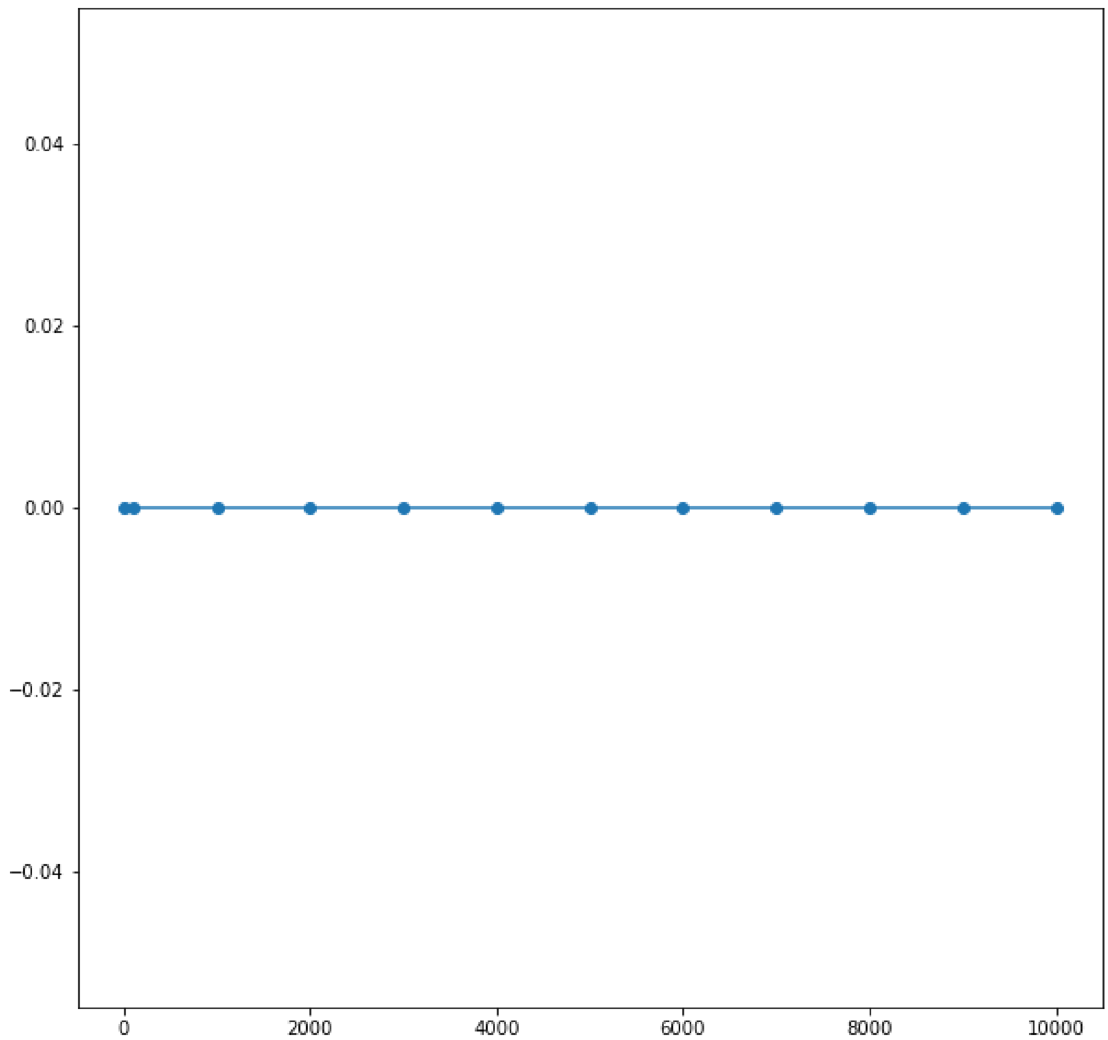

```
x =np.linspace(1,10,1000000)
a=1/2
y =np.log(x)
y1=y*2
plt.plot(x,y,label='log(n)')
plt.plot(x,y1,label='log(n)*')
plt.title("Tight upper bound")
plt.xlabel('X AXIS')
plt.ylabel('Y AXIS')
plt.legend()
plt.show()
```




```
x =np.linspace(1,10,1000000)
a=1/2
y =np.log(x)
y1=x
plt.plot(x,y,label='log(n)')
plt.plot(x,y1,label='n')
plt.title("Loose upper bound")
plt.xlabel('X AXIS')
plt.ylabel('Y AXIS')
plt.legend()
plt.show()
```




```
from scipy import optimize
def func(x,a,b):
    return a*np.log(x)+b
params, params_covariance=optimize.curve_fit(func,N,T3)
fig , ax = plt.subplots(figsize=(10,10))
ax.scatter(N,T3)
ax.plot(N,func(N,params[0],params[1]))
plt.show()
```



Suitable boundary functions for algorithms

Bubble Sort,Insertion Sort, Selection Sort: n^2 Merge Sort:n*log(n) Binary Search:log(n)