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

Exercise 2

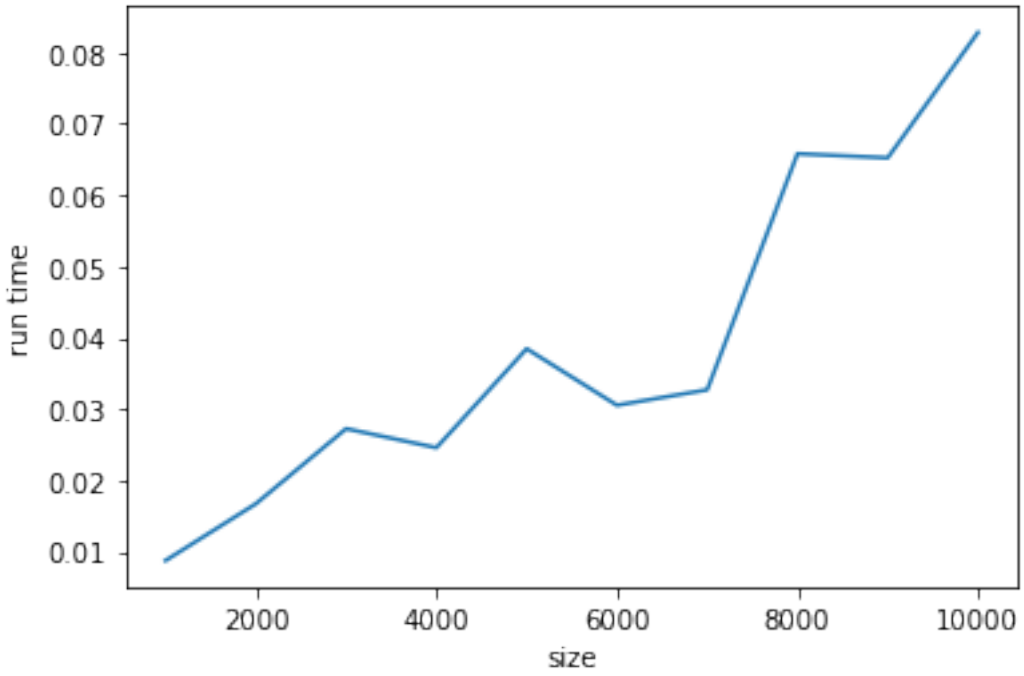
Case-1 (a=b):

```
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):
    #print(arr)
    if len(arr)<=1:
        return

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

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

runTime=[]

arr=[]
n=1000
```

```
for i in range(n):
    arr.append(rn.randint(1,100000))

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

runTime.append(stop-start)

arr=[]
n=2000

for i in range(n):
    arr.append(rn.randint(1,100000))

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

runTime.append(stop-start)

arr=[]
n=3000

for i in range(n):
    arr.append(rn.randint(1,100000))

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

runTime.append(stop-start)

arr=[]
n=4000

for i in range(n):
    arr.append(rn.randint(1,100000))

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

runTime.append(stop-start)

arr=[]
n=5000

for i in range(n):
```

```
    arr.append(rn.randint(1,100000))

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

runTime.append(stop-start)

arr=[]
n=6000

for i in range(n):
    arr.append(rn.randint(1,100000))

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

runTime.append(stop-start)

arr=[]
n=7000

for i in range(n):
    arr.append(rn.randint(1,100000))

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

runTime.append(stop-start)

arr=[]
n=8000

for i in range(n):
    arr.append(rn.randint(1,100000))

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

runTime.append(stop-start)

arr=[]
n=9000

for i in range(n):
    arr.append(rn.randint(1,100000))
```

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

runTime.append(stop-start)

arr=[]
n=10000

for i in range(n):
    arr.append(rn.randint(1,100000))

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

runTime.append(stop-start)

size=[1000,2000,3000,4000,5000,6000,7000,8000,9000,10000]

import math

plt.plot(size,runTime)
plt.xlabel('size')
plt.ylabel('run time')

plt.show()
```



Here since a=b=2, we can write the recurrence relation using masters theorem as T(n)= 2*T(n/2)+cn. Now, since f(n)= cn, it is of the form theta(n). Therefore from masters theorem we can say that T(n) can be given by T(n)=Theta(n*logn)

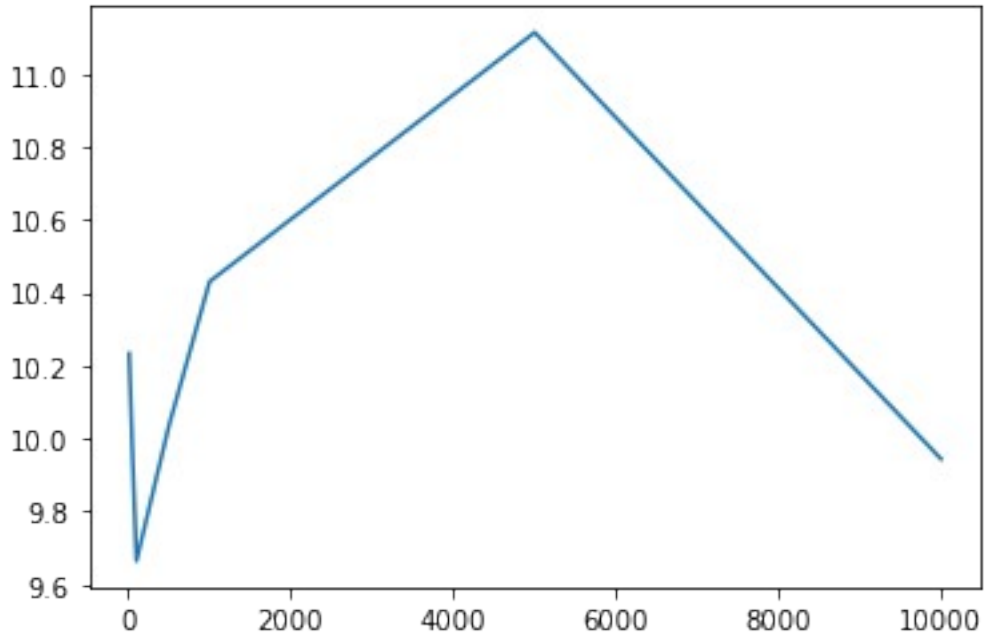
Case-2 (a=3,b=2):

In this case since a=3 and b=2 we get the recurrence relation as 3T(n/2)+n^2. Here f(n) is of the order n^2 and hence f(n)=W(n^(log_2(3)+E). Implies T(n)=O(n^2)

Exercise-1

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



size=[10,50,100,500,1000,5000,10000]
runTimeBubble=[]
for j in range (len(size)):
    arr=[]
    for i in range (n):
        arr.append(rn.randint(1,100000))

    start=time.time()
    bubbleSort(arr)
    stop=time.time()
    runTimeBubble.append(stop-start)

plt.plot(size,runTimeBubble)
plt.show()
```



Here T(n)=T(n-1)+cn.

let, n=log_2(m) and G(m)=T(log_2(m))

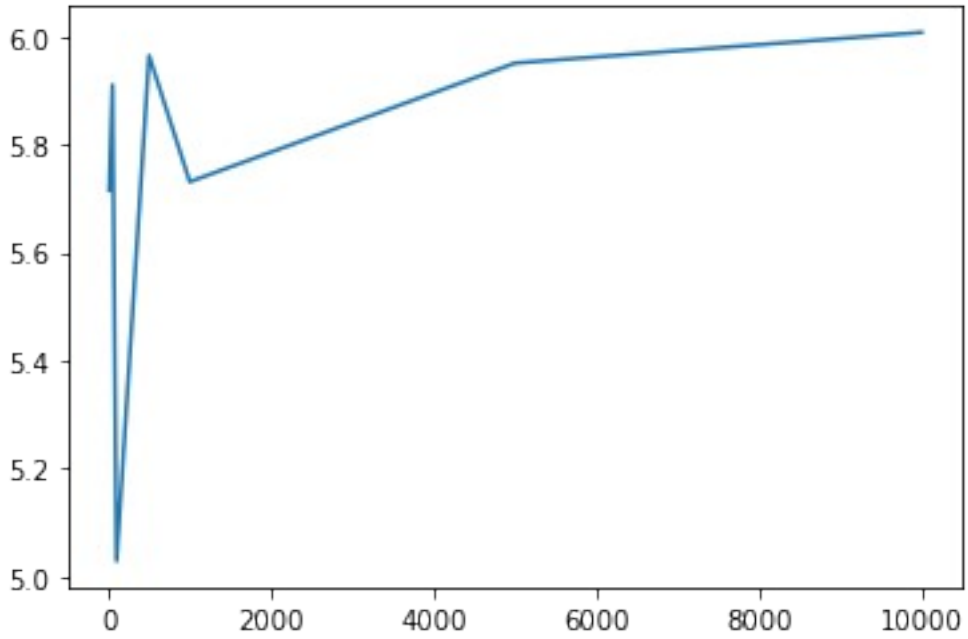
Implies S(m)=S(m/2)+log(m)

as f(n) is of the form Theta(logm), T(n)=O(n^2)

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

size=[10,50,100,500,1000,5000,10000]
runTimeInsertion=[]
for j in range (len(size)):
    arr=[]
    for i in range (n):
        arr.append(rn.randint(1,100000))

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

```
    stop=time.time()
    runTimeInsertion.append(stop-start)

plt.plot(size,runTimeInsertion)
plt.show()
```



Here T(n)=T(n-1)+cn.

let, n=log_2(m) and G(m)=T(log_2(m))

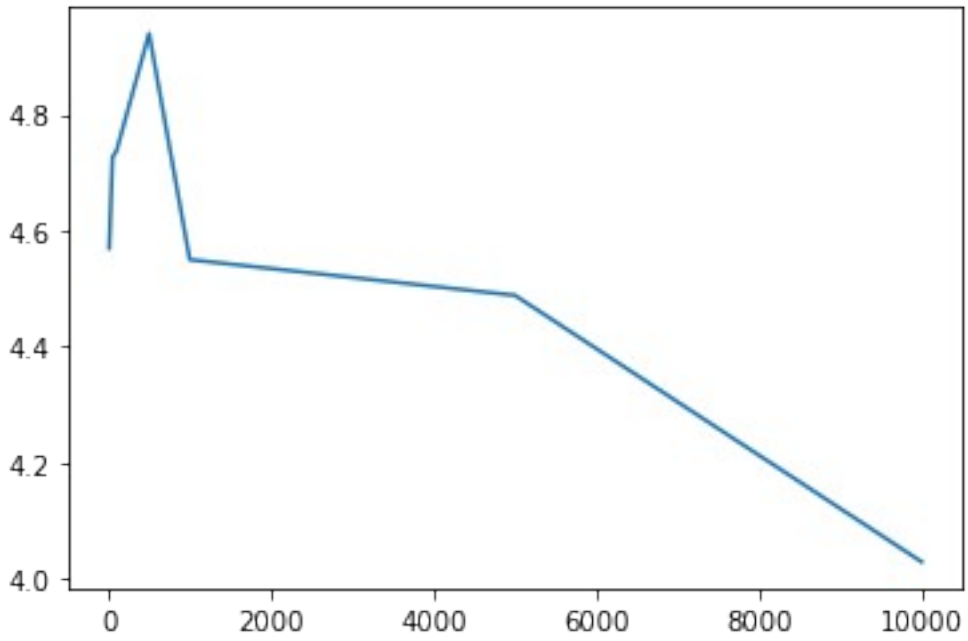
Implies S(m)=S(m/2)+log(m)

as f(n) is of the form Theta(logm), T(n)=O(n^2)

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

size=[10,50,100,500,1000,5000,10000]
runTimeSelection=[]
for j in range (len(size)):
```

```
    arr=[]
    for i in range (n):
        arr.append(rn.randint(1,100000))

    start=time.time()
    selectionSort(arr)
    stop=time.time()
    runTimeSelection.append(stop-start)

plt.plot(size,runTimeSelection)
plt.show()
```



Here T(n)=T(n-1)+cn.

let, n=log_2(m) and G(m)=T(log_2(m))

Implies S(m)=S(m/2)+log(m)

as f(n) is of the form Theta(logm), T(n)=O(n^2)