ASSIGNMENT 1

Rohith M

3122 21 5001 085

Insertion Sort

In [49]:

```
import random
def insertionsort(array):
    l1=time.time()
    for step in range(1, len(array)):
        key = array[step]
        j = step - 1

        # Compare key with each element on the left of it until an element smaller
        # For descending order, change key<array[j] to key>array[j].
        while j >= 0 and key < array[j]:
            array[j + 1] = array[j]
            j = j - 1

        # Place key at after the element just smaller than it.
        array[j + 1] = key

    l2=time.time()

    return (l2-l1)*1000
```

Creation of array of random numbers

In [51]:

```
randomlist10=[]
for i in range(0,10):
    n = random.randint(1,10000)
    randomlist10.append(n)

randomlist100=[]
for i in range(0,100):
    n = random.randint(1,10000)
    randomlist100.append(n)

randomlist1000=[]
for i in range(0,1000):
    n = random.randint(1,10000)
    randomlist1000.append(n)

randomlist10k=[]
for i in range(0,10000):
    n = random.randint(1,10000)
    randomlist10k.append(n)
randomlist100k=[]
for i in range(0,100000):
    n = random.randint(1,10000)
    randomlist100k.append(n)
```

In [52]:

```
#performing sorting

times=[]
times.append(insertionsort(randomlist10))
times.append(insertionsort(randomlist100))
times.append(insertionsort(randomlist1000))
times.append(insertionsort(randomlist10k))
times.append(insertionsort(randomlist100k))
```

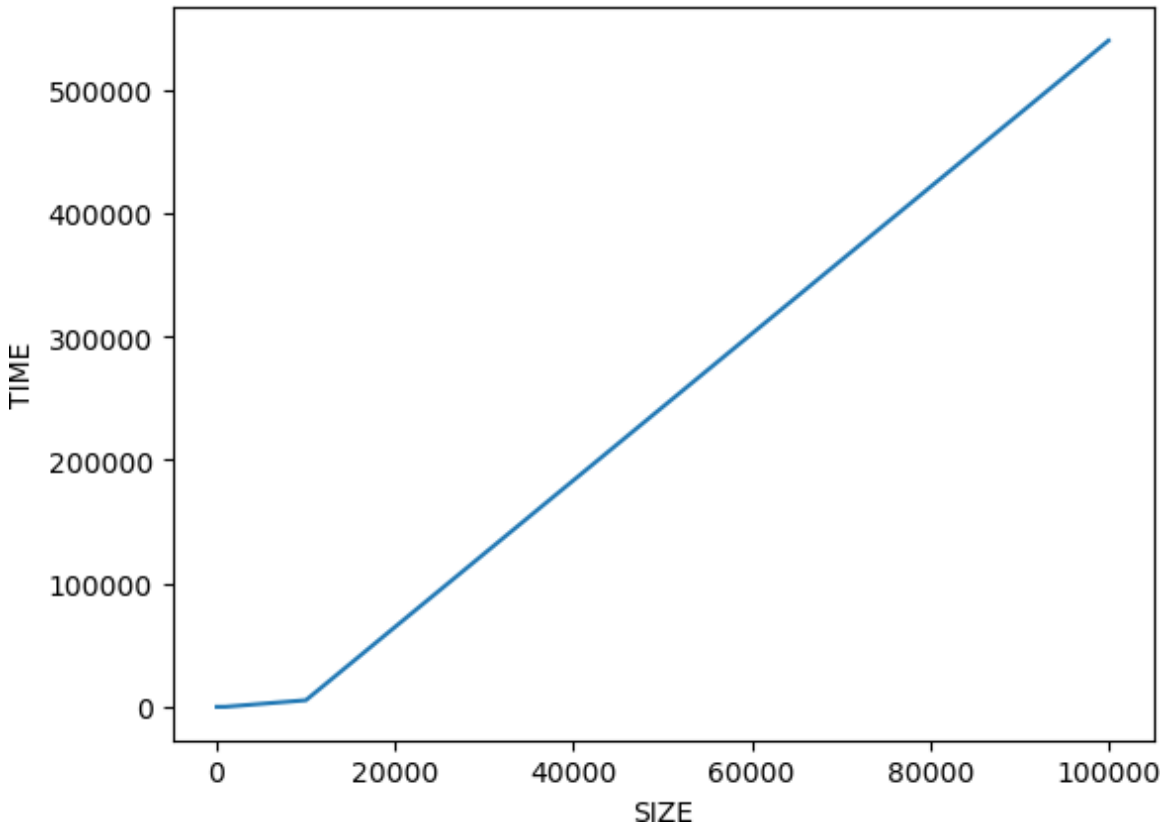
Plotting the graph

In [55]:

```
x=[10,100,1000,10000,100000]
import numpy as np
x=np.array(x)
import matplotlib.pyplot as plt
plt.plot(x,times)
plt.xlabel("SIZE")
plt.ylabel("TIME")
```

Out[55]:

```
Text(0, 0.5, 'TIME')
```



Shell Sort

In [72]:

```
def shellsort(array):
    l1=time.time()
    n=len(array)
    interval = n // 2
    while interval > 0:
        for i in range(interval, n):
            temp = array[i]
            j = i
            while j >= interval and array[j - interval] > temp:
                array[j] = array[j - interval]
                j -= interval
            array[j] = temp
        interval //= 2
    l2=time.time()

    return (l2-l1)*1000
```

Creation of array of random numbers

In [73]:

```
randomlist10=[]
for i in range(0,10):
    n = random.randint(1,10000)
    randomlist10.append(n)

randomlist100=[]
for i in range(0,100):
    n = random.randint(1,10000)
    randomlist100.append(n)

randomlist1000=[]
for i in range(0,1000):
    n = random.randint(1,10000)
    randomlist1000.append(n)

randomlist10k=[]
for i in range(0,10000):
    n = random.randint(1,10000)
    randomlist10k.append(n)
randomlist100k=[]
for i in range(0,100000):
    n = random.randint(1,10000)
    randomlist100k.append(n)
```

In [74]:

```
#performing sorting

times=[]

times.append(shellsort(randomlist10))
times.append(shellsort(randomlist100))
times.append(shellsort(randomlist1000))
times.append(shellsort(randomlist10k))
times.append(shellsort(randomlist100k))
```

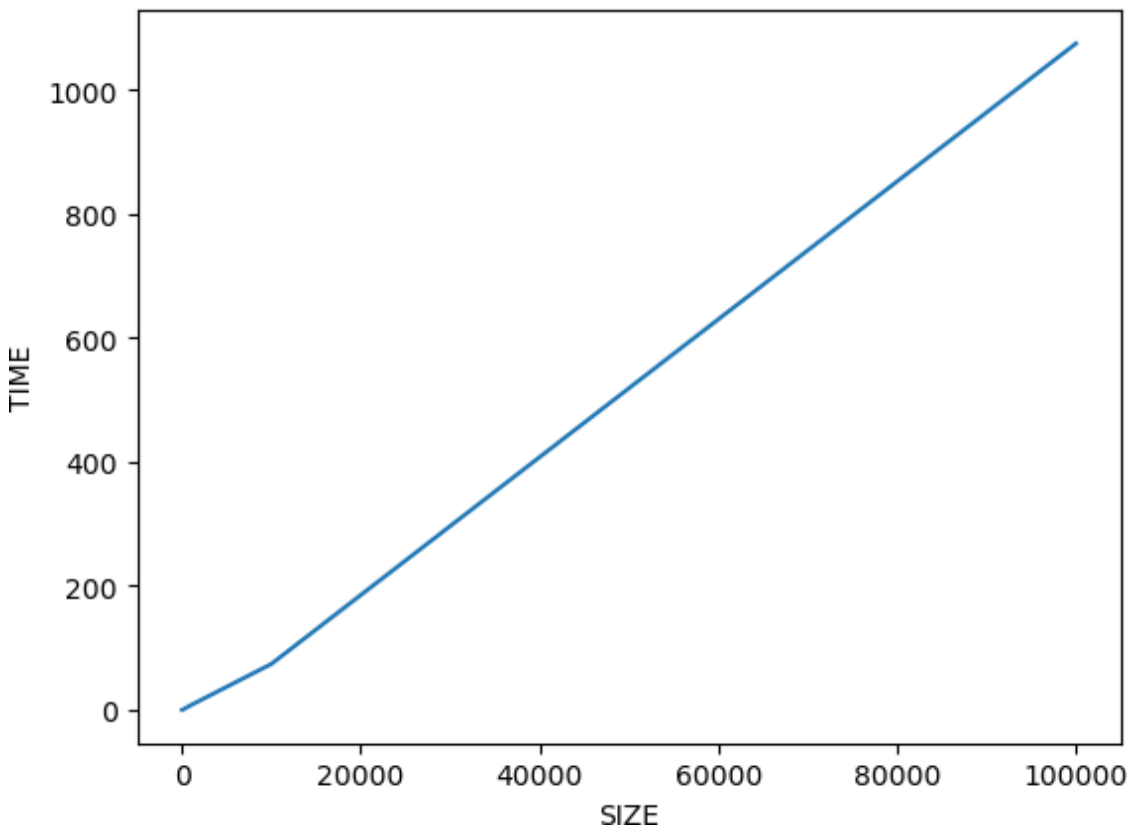
Plotting the graph

In [75]:

```
x=[10,100,1000,10000,100000]
import numpy as np
x=np.array(x)
import matplotlib.pyplot as plt
plt.plot(x,times)
plt.xlabel("SIZE")
plt.ylabel("TIME")
```

Out[75]:

```
Text(0, 0.5, 'TIME')
```



Radix Sort

In [86]:

```
def countingSort(array, place):
    size = len(array)
    output = [0] * size
    count = [0] * 10
    for i in range(0, size):
        index = array[i] // place
        count[index % 10] += 1


    for i in range(1, 10):
        count[i] += count[i - 1]


    i = size - 1
    while i >= 0:
        index = array[i] // place
        output[count[index % 10] - 1] = array[i]
        count[index % 10] -= 1
        i -= 1

    for i in range(0, size):
        array[i] = output[i]



def radixsort(array):
    l1=time.time()
    max_element = max(array)


    place = 1
    while max_element // place > 0:
        countingSort(array, place)
        place *= 10
    l2=time.time()

    return (l2-l1)*1000
```

Creation of array of random numbers

In [87]:

```
randomlist10=[]
for i in range(0,10):
    n = random.randint(1,10000)
    randomlist10.append(n)

randomlist100=[]
for i in range(0,100):
    n = random.randint(1,10000)
    randomlist100.append(n)

randomlist1000=[]
for i in range(0,1000):
    n = random.randint(1,10000)
    randomlist1000.append(n)

randomlist10k=[]
for i in range(0,10000):
    n = random.randint(1,10000)
    randomlist10k.append(n)
randomlist100k=[]
for i in range(0,100000):
    n = random.randint(1,10000)
    randomlist100k.append(n)
```

In [89]:

```
#performing sorting

times=[]
times.append(radixsort(randomlist10))
times.append(radixsort(randomlist100))
times.append(radixsort(randomlist1000))
times.append(radixsort(randomlist10k))
times.append(radixsort(randomlist100k))
```

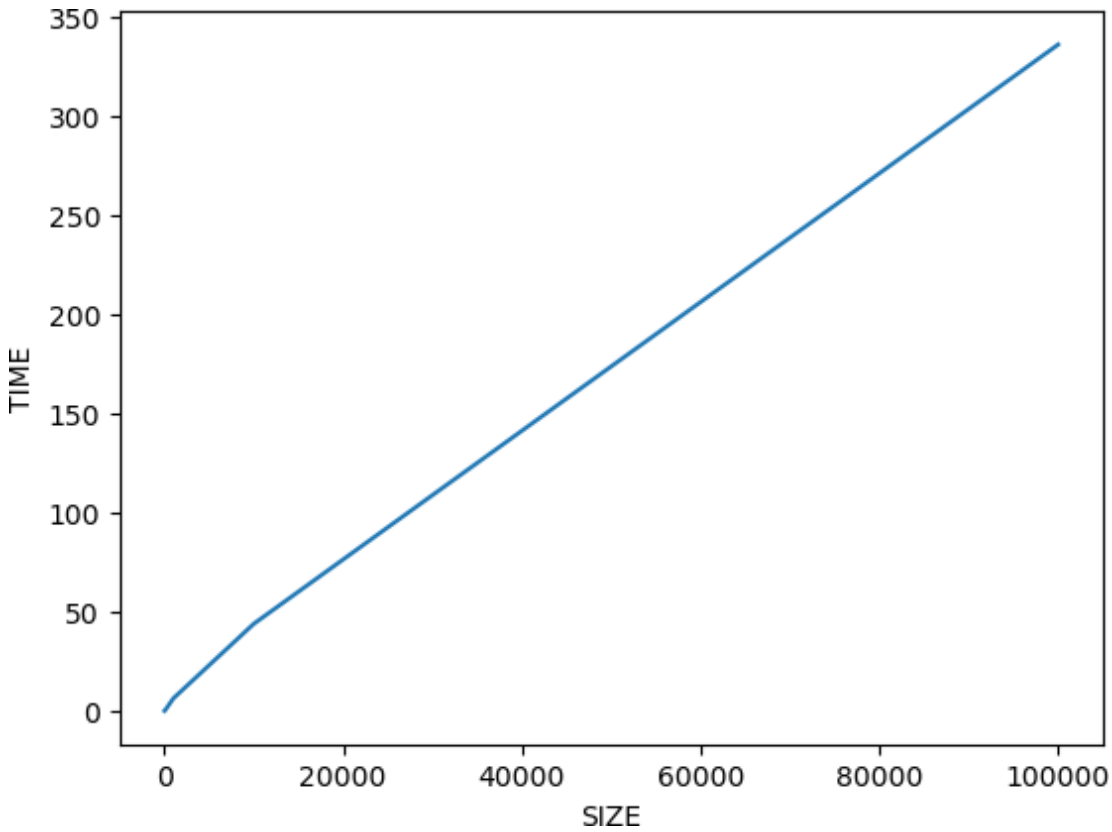
Plotting the graph

In [92]:

```
x=[10,100,1000,10000,100000]
import numpy as np
x=np.array(x)
import matplotlib.pyplot as plt
plt.plot(x,times)
plt.xlabel("SIZE")
plt.ylabel("TIME")
```

Out[92]:

```
Text(0, 0.5, 'TIME')
```



SEARCHING

BINARY SEARCH NON RECURSIVE

In [129]:

```
def binarysearch(arr, x):
    l1=time.time()
    found=0
    left, right = 0, len(arr) - 1
    while left <= right:
        mid = (left + right) // 2
        if arr[mid] == x:
            found=1
            l2=time.time()
            return (l2-l1)*1000
        elif arr[mid] < x:
            left = mid + 1
        else:
            right = mid - 1
```

Creation of array of random numbers

In [114]:

```
randomlist10=[]
for i in range(0,10):
    n = random.randint(1,10000)
    randomlist10.append(n)

randomlist100=[]
for i in range(0,100):
    n = random.randint(1,10000)
    randomlist100.append(n)

randomlist1000=[]
for i in range(0,1000):
    n = random.randint(1,10000)
    randomlist1000.append(n)

randomlist10k=[]
for i in range(0,10000):
    n = random.randint(1,10000)
    randomlist10k.append(n)
randomlist100k=[]
for i in range(0,100000):
    n = random.randint(1,10000)
    randomlist100k.append(n)
```

Sorting using Shell Sort

In [115]:

```
def shellsort(array):
    l1=time.time()
    n=len(array)
    interval = n // 2
    while interval > 0:
        for i in range(interval, n):
            temp = array[i]
            j = i
            while j >= interval and array[j - interval] > temp:
                array[j] = array[j - interval]
                j -= interval
            array[j] = temp
        interval //= 2
    l2=time.time()

    return array

def chooserandombeforesort(array):
    r_in=random.randint(1,len(array))
    return array[r_in]


#Choosing elements before sort

x10=chooserandombeforesort(randomlist10)
x100=chooserandombeforesort(randomlist100)
x1000=chooserandombeforesort(randomlist1000)
x10k=chooserandombeforesort(randomlist10k)
x100k=chooserandombeforesort(randomlist100k)
randomlist10=shellsort(randomlist10)

randomlist100=shellsort(randomlist100)
randomlist1000=shellsort(randomlist1000)
randomlist10k=shellsort(randomlist10k)
randomlist100k=shellsort(randomlist100k)
```

In [116]:

```
#performing searching

times=[]

times.append(binarysearch(randomlist10,x10))
times.append(binarysearch(randomlist100,x100))
times.append(binarysearch(randomlist1000,x1000))
times.append(binarysearch(randomlist10k,x10k))
times.append(binarysearch(randomlist100k,x100k))
times
```

Out[116]:

```
[0.0040531158447265625,
 0.0030994415283203125,
 0.005245208740234375,
 0.0069141387939453125,
 0.009298324584960938]
```

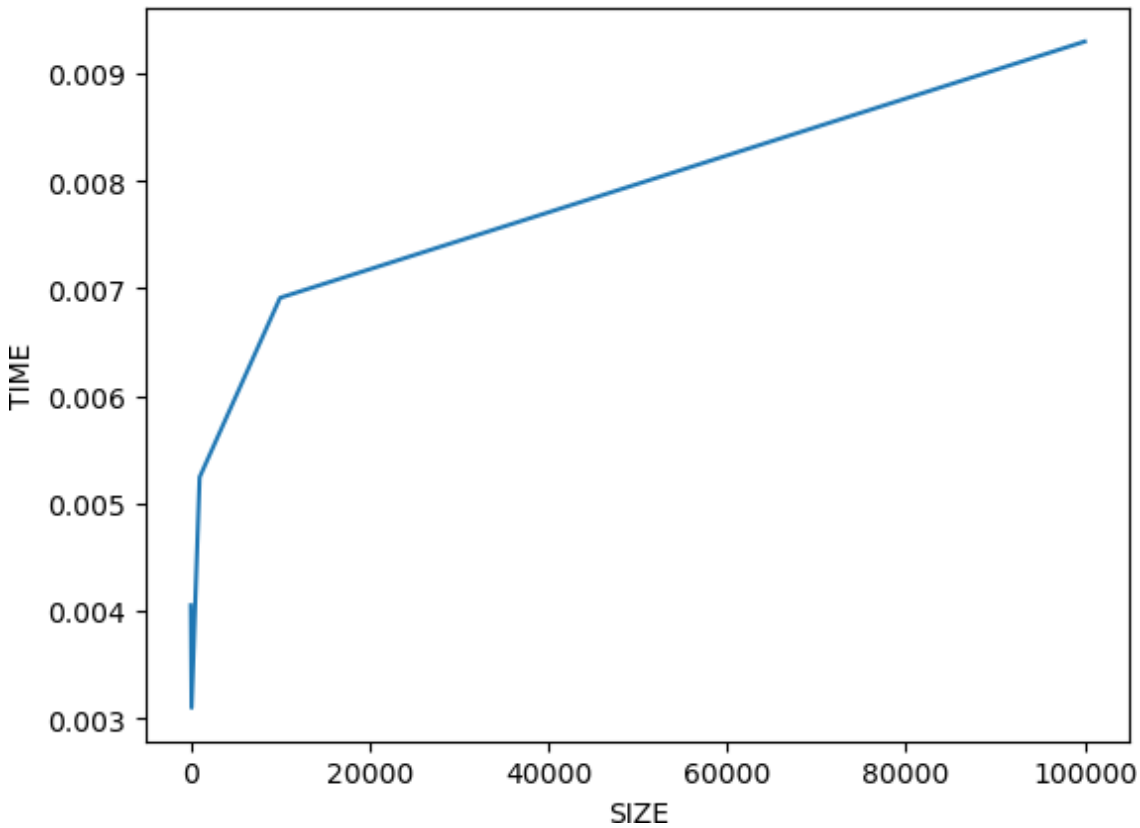
Plotting the graph

In [117]:

```
x=[10,100,1000,10000,100000]
import numpy as np
x=np.array(x)
import matplotlib.pyplot as plt
plt.plot(x,times)
plt.xlabel("SIZE")
plt.ylabel("TIME")
```

Out[117]:

```
Text(0, 0.5, 'TIME')
```



BINARY SEARCH RECURSIVE

In [120]:

```
def binary_search(arr, x, left, right):
    if left > right:
        return -1
    mid = (left + right) // 2
    if arr[mid] == x:
        return mid
    elif arr[mid] < x:
        return binary_search(arr, x, mid + 1, right)
    else:
        return binary_search(arr, x, left, mid - 1)
```

Creation of array of random numbers

In [121]:

```
randomlist10=[]
for i in range(0,10):
    n = random.randint(1,10000)
    randomlist10.append(n)

randomlist100=[]
for i in range(0,100):
    n = random.randint(1,10000)
    randomlist100.append(n)

randomlist1000=[]
for i in range(0,1000):
    n = random.randint(1,10000)
    randomlist1000.append(n)

randomlist10k=[]
for i in range(0,10000):
    n = random.randint(1,10000)
    randomlist10k.append(n)
randomlist100k=[]
for i in range(0,100000):
    n = random.randint(1,10000)
    randomlist100k.append(n)
```

Sorting using Shell Sort

In [122]:

```
def shellsort(array):
    l1=time.time()
    n=len(array)
    interval = n // 2
    while interval > 0:
        for i in range(interval, n):
            temp = array[i]
            j = i
            while j >= interval and array[j - interval] > temp:
                array[j] = array[j - interval]
                j -= interval
            array[j] = temp
        interval //= 2
    l2=time.time()

    return array

def chooserandombeforesort(array):
    r_in=random.randint(1,len(array))
    return array[r_in]


#Choosing elements before sort

x10=chooserandombeforesort(randomlist10)
x100=chooserandombeforesort(randomlist100)
x1000=chooserandombeforesort(randomlist1000)
x10k=chooserandombeforesort(randomlist10k)
x100k=chooserandombeforesort(randomlist100k)
randomlist10=shellsort(randomlist10)

randomlist100=shellsort(randomlist100)
randomlist1000=shellsort(randomlist1000)
randomlist10k=shellsort(randomlist10k)
randomlist100k=shellsort(randomlist100k)
```

In [125]:

```
#performing searching

times1=[]
l1=time.time()
binary_search(randomlist10,x10,0,len(randomlist10))
l2=time.time()
times1.append((l2-l1)*1000)
l1=time.time()
binary_search(randomlist100,x100,0,len(randomlist100))
l2=time.time()
times1.append((l2-l1)*1000)
l1=time.time()
binary_search(randomlist1000,x1000,0,len(randomlist1000))
l2=time.time()
times1.append((l2-l1)*1000)
l1=time.time()
binary_search(randomlist10k,x10k,0,len(randomlist10k))
l2=time.time()
times1.append((l2-l1)*1000)
l1=time.time()
binary_search(randomlist100k,x100k,0,len(randomlist100k))
l2=time.time()
times1.append((l2-l1)*1000)

times1
```

Out[125]:

```
[0.18405914306640625,
 0.15783309936523438,
 0.11396408081054688,
 0.10585784912109375,
 0.1087188720703125]
```

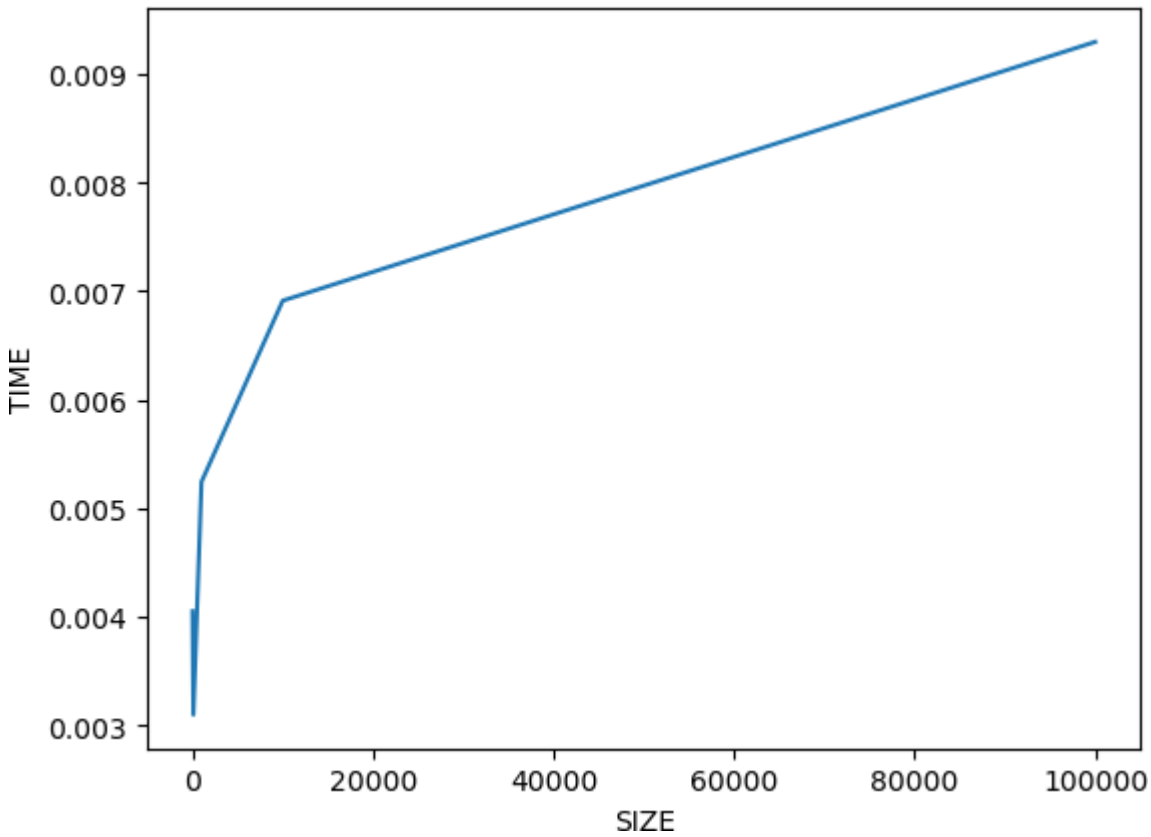
Plotting the graph

In [126]:

```
x=[10,100,1000,10000,100000]
import numpy as np
x=np.array(x)
import matplotlib.pyplot as plt
plt.plot(x,times)
plt.xlabel("SIZE")
plt.ylabel("TIME")
```

Out[126]:

```
Text(0, 0.5, 'TIME')
```



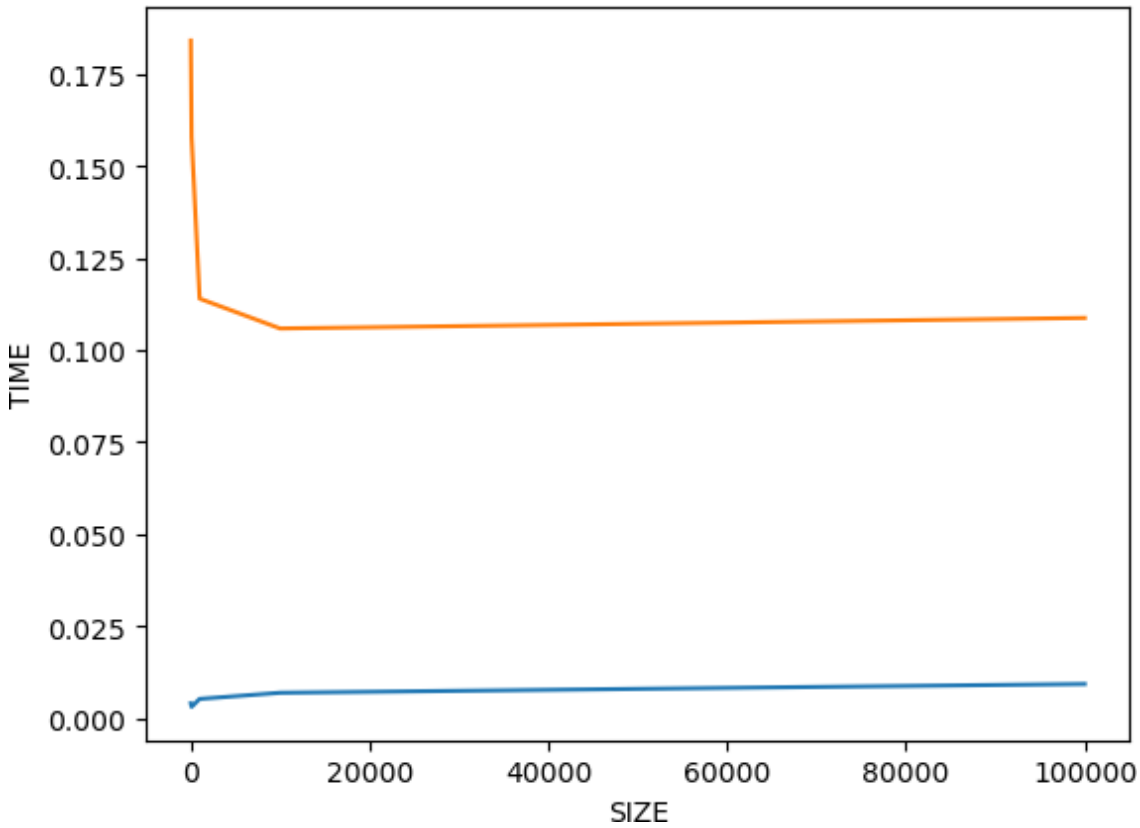
Comparison of Non Recursive and Recursive

In [127]:

```
x=[10,100,1000,10000,100000]
import numpy as np
x=np.array(x)
import matplotlib.pyplot as plt
plt.plot(x,times)
plt.plot(x,times1)
plt.xlabel("SIZE")
plt.ylabel("TIME")
```

Out[127]:

```
Text(0, 0.5, 'TIME')
```



Comparring Non Recurrsive and Recursive for size 1000

In [134]:

```
randomlist1000=[]
for i in range(0,1000):
    n = random.randint(1,10000)
    randomlist1000.append(n)
x1000=chooserandombeforesort(randomlist1000)
randomlist1000=shellsort(randomlist1000)
print(str(binarysearch(randomlist1000,x1000))+"  is the time taken for size 1000 arr
randomlist1000=[]
for i in range(0,1000):
    n = random.randint(1,10000)
    randomlist1000.append(n)
x1000=chooserandombeforesort(randomlist1000)
randomlist1000=shellsort(randomlist1000)
l1=time.time()
binary_search(randomlist1000,x1000,0,len(randomlist1000))
l2=time.time()
print(str((l2-l1)*1000) +" is the time taken to search recursively in size 1000 arra
```

```
0.006198883056640625  is the time taken for size 1000 array to search
non recursively
0.07772445678710938 is the time taken to search recursively in size 10
00 array
```

In [ ]:

```

```