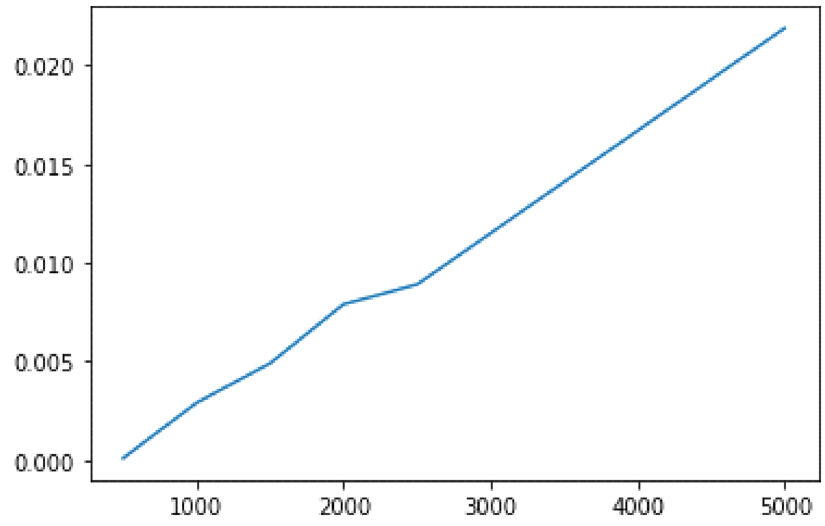
3a. Implement Linear Search compute space and time complexities, plot graph using asymptomatic notations.

```
import matplotlib.pyplot as plt
import numpy as np
import time
start=time.time()
def linear(ar,n,k):
    for i in range(n):
        if ar[i]==k:
            return i
    return-1
arr = [ ]
n = int(input("enter the no of elements"))
for i in range(n):
    arr.append(int(input('enter the element')))
print("Array elements are ", arr)
k=int(input("enter the key elements to search"))
res=linear(arr,len(arr),k)
if res== -1:
    print("element not found")
else:
    print("element %d found at index" %k , res)
end=time.time()
print({end-start})
xpoints=np.array([500,1000,1500,2000,2500,5000])
ypoints=np.array([0.00009,0.0029,0.0049,0.0079,0.0089,0.00219])
plt.plot(xpoints,ypoints)
plt.show()
```

Output:

```
enter the no of elements5
enter the element10
enter the element20
enter the element30
enter the element40
enter the element50
Array elements are  [10, 20, 30, 40, 50]
enter the key elements to search30
element 30 found at index 2

enter the no of elements5
enter the element1
enter the element2
enter the element3
enter the element4
enter the element5
Array elements are  [1, 2, 3, 4, 5]
enter the key elements to search6
element not found
{7.044062852859497}
```



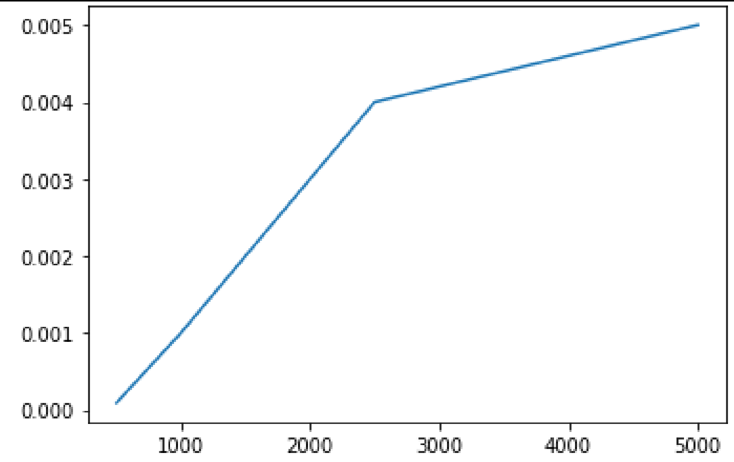
3b. Implement Bubble sorting algorithms compute space and time complexities, plot graph using asymptomatic notations

```
import matplotlib.pyplot as plt
import numpy as np
import time
start = time.time()
def bubble_sort(ar):
    n = len(ar)
    for i in range(n - 1):
        for j in range(0, n - 1 - i):
            if (ar[j] > ar[j + 1]):
                ar[j], ar[j + 1] = ar[j + 1], ar[j]

arr = [ ]
n = int(input("enter the no of elements"))
for i in range(n):
    arr.append(int(input('enter the element')))
print("unsorted array", arr)
bubble_sort(arr)
print("sorted arr:",arr)
end = time.time()
print({end - start})
xpoints = np.array([500,1000,1500,2000,2500,5000])
ypoints = np.array([0.00009,0.0010,0.0020,0.0030,0.0040,0.0050])
plt.plot(xpoints,ypoints)
plt.show()
```

Output:

```
enter the no of elements5
enter the element10
enter the element5
enter the element6
enter the element78
enter the element42
unsorted array [10, 5, 6, 78, 42]
sorted array [5 6 10 42 78 ]
 {16.526478052139282}
```

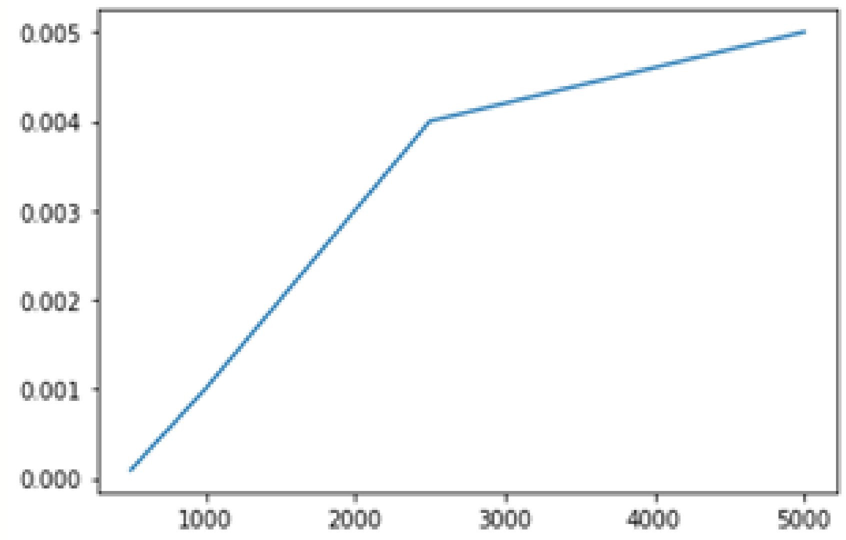


3c. Implement Selection sorting algorithms compute space and time complexities, plot graph using asymptomatic notations

```
import matplotlib.pyplot as plt
import numpy as np
import time
start = time.time()
def selection_sort(arr):
    n=len(arr)
    for i in range(n):
        min=i
        for j in range(i+1,n):
            if arr[min]>arr[j]:
                min=j
        arr[i], arr[min]= arr[min],arr[i]
arr= [ ]
n=int(input("enter the number of element"))
for i in range (n):
    arr.append(int(input("enter the element")))
print("unsorted array",arr)
selection_sort(arr)
print("sorted array:",arr)
end=time.time()
print({end-start})
xpoints = np.array([500,1000,1500,2000,2500,5000])
ypoints = np.array([0.00009,0.0010,0.0020,0.0030,0.0040,0.0050])
plt.plot(xpoints,ypoints)
plt.show()
```

Output:

```
enter the number of element5
enter the element4
enter the element5
enter the element-3
enter the element0
enter the element1
unsorted array [4, 5, -3, 0, 1]
sorted array  [-3, 0, 1, 4, 5]
{12.0227792263031}
```

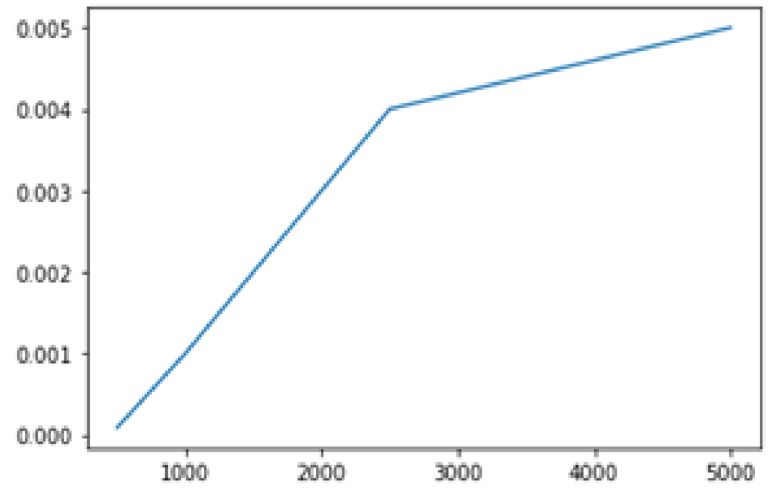


3d. Implement insertion sorting algorithms compute space and time complexities, plot graph using asymptomatic notations

```
import matplotlib.pyplot as plt
import numpy as np
import time
start=time.time()
def insertionSort(a):
    for i in range(1, len(a)):
        key = a[i]
        j = i - 1
        while j >= 0 and key < a[j]:
            a[j + 1] ,a[j]= a[j], a[j+1]
            j = j - 1
arr=[]
n=int(input("enter the number of element"))
for i in range (n):
    arr.append(int(input("enter the element")))
print("unsorted array",arr)
insertionSort(arr)
print ("Sorted array is:",arr)
end=time.time()
print({end-start})
xpoints = np.array([500,1000,1500,2000,2500,5000])
ypoints = np.array([0.00009,0.0010,0.0020,0.0030,0.0040,0.0050])
plt.plot(xpoints,ypoints)
plt.show()
```

Output:

```
enter the number of element 5
enter the element7
enter the element4
enter the element6
enter the element12
enter the element10
unsorted array [7, 4, 6, 12, 10]
sorted array [4, 6, 7, 10, 12]
{9.21582579612732}
```


0.005
0.004
0.003
0.002
0.001
0.000
1000
2000
3000
4000
5000