KATHMANDU UNIVERSITY

DHULIKHEL KAVRE

Subject: COMP 314: Algorithms and Complexity
Lab no: 1

Submitted By:

Name: Ayush Kumar Shah
Roll no: 44
Group: CE 3rd year 2nd sem
Level: UNG

Submitted To:

Dr. Bal Krishna Bal

Date of Submission:
06/05/2018

Lab 1
Searching (Linear and Binary Search)
COMP 314 – Algorithms and Complexity

Chapter 5 – Searching and Sorting from the book “Data Structures and Algorithms Using Python – Rance D. Necaise”

Objective:
Implementation and analysis of Linear and Binary Search Algorithms.

Generate 100000 random but unique numbers and apply linear and binary search algorithms to find a particular element. Calculate the times of the search for the best and worst cases. Show that the search times correspond to O(1) and O(n) for linear search and O(1) and O(log n) for binary search.

1. **Linear search**
1.1. Implementation of the linear search on an unsorted sequence
1.2. Implementation of the linear search on a sorted sequence
1.3. Searching for the smallest value in an unsorted sequence
2. **Binary search**

Generating random numbers and sorting them

```
>>> import random
>>> myrandomnumber = random.sample(range(1000000),100000)
>>>mysortedrandomnumber = sorted(myrandomnumber)
```

Randomly picking any number from the list:

```
>>> randomnumberfromthelist = random.choice(mysortedrandomnumber)
```

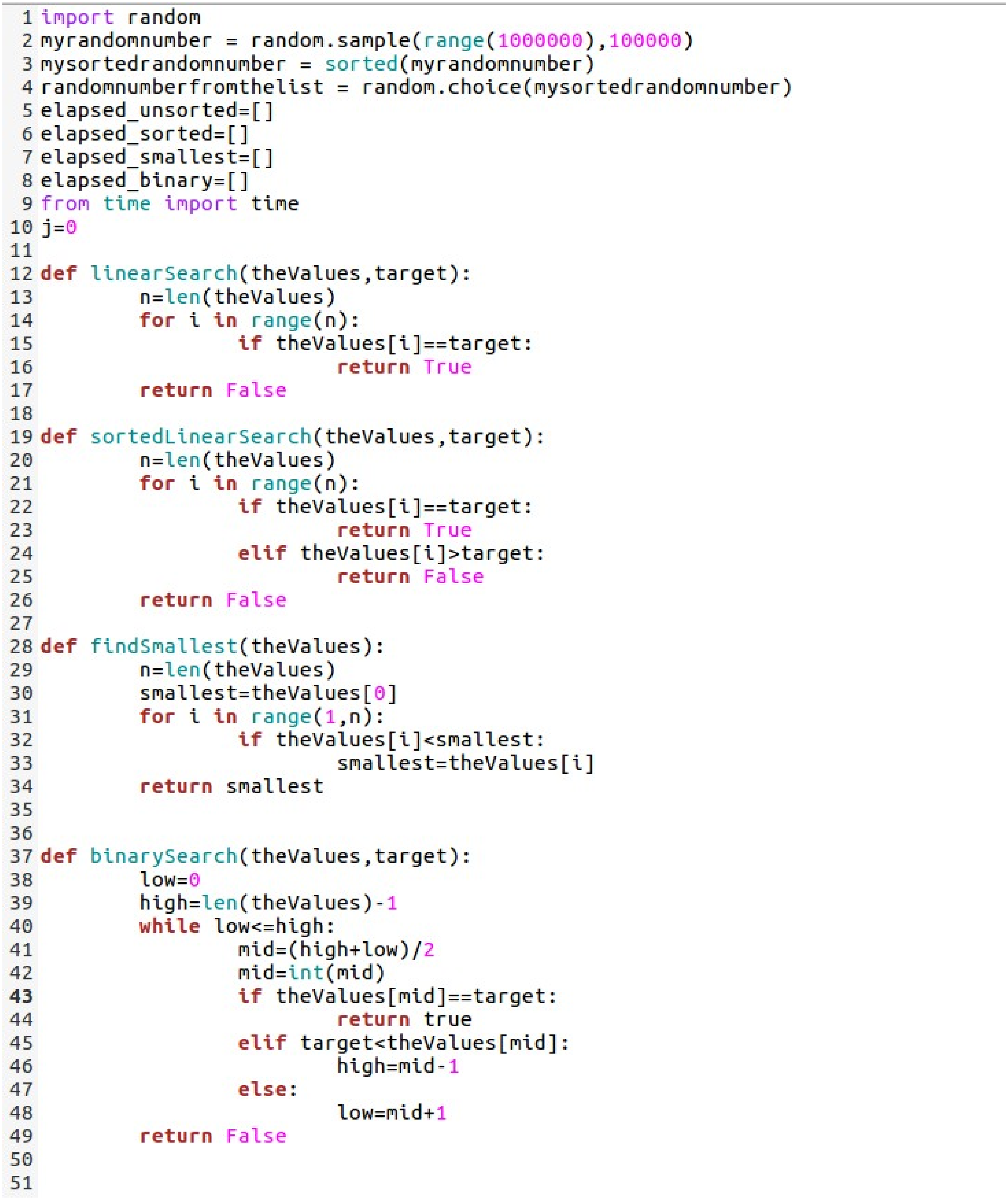
How to record the times of execution of an algorithm

```
>>> from time import time
>>> start time = time( ) # record the starting time
>>> run algorithm
>>> end time = time( ) # record the ending time
>>>elapsed = end time − start time # compute the elapsed time
```

Recording the search times for linear and binary search algorithms for different input sizes in the range 10000 – 100000, step size is 10000.

For the recorded times of linear search and binary search on input step size of 10000, plot graphs for both linear and binary. The graphs should correspond to the algorithmic functions described by the corresponding algorithms.

Python code:



```python
import random
myrandomnumber = random.sample(range(1000000),100000)
mysortedrandomnumber = sorted(myrandomnumber)
randomnumberfromthelist = random.choice(mysortedrandomnumber)
elapsed_unsorted=[]
elapsed_sorted=[]
elapsed_smallest=[]
elapsed_binary=[]
from time import time
j=0

def linearSearch(theValues,target):
        n=len(theValues)
        for i in range(n):
                if theValues[i]==target:
                        return True
        return False

def sortedLinearSearch(theValues,target):
        n=len(theValues)
        for i in range(n):
                if theValues[i]==target:
                        return True
                elif theValues[i]>target:
                        return False
        return False

def findSmallest(theValues):
        n=len(theValues)
        smallest=theValues[0]
        for i in range(1,n):
                if theValues[i]<smallest:
                        smallest=theValues[i]
        return smallest


def binarySearch(theValues,target):
        low=0
        high=len(theValues)-1
        while low<=high:
                mid=(high+low)/2
                mid=int(mid)
                if theValues[mid]==target:
                        return true
                elif target<theValues[mid]:
                        high=mid-1
                else:
                        low=mid+1
        return False
```

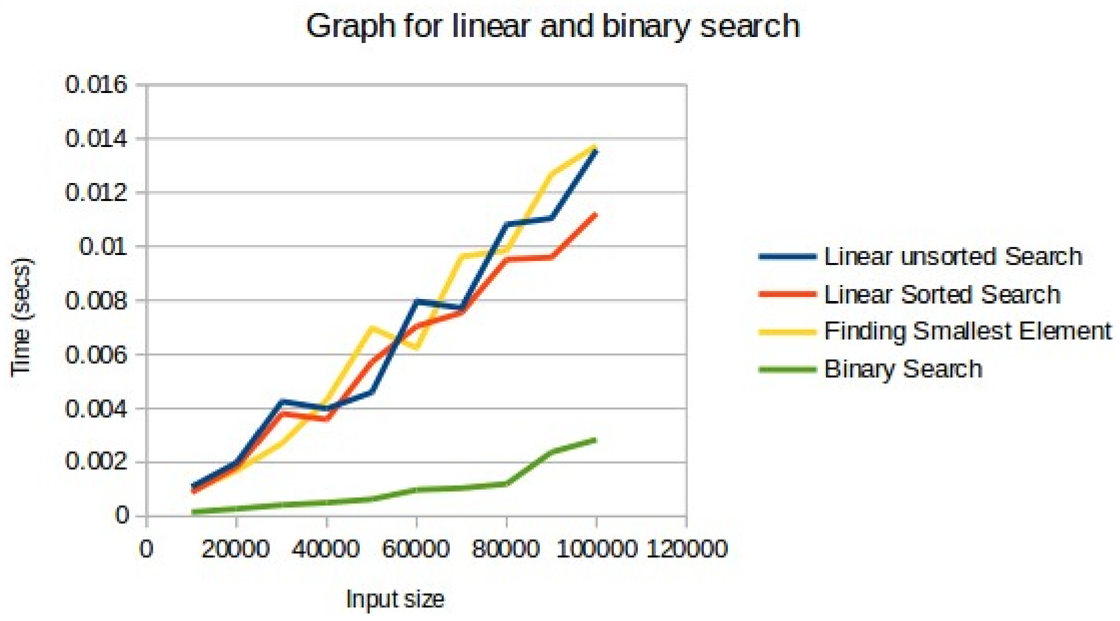
```
for i in range(10000,100001,10000):
        start_time = time( ) # record the starting time

        #run algorithm
        linearSearch(myrandomnumber[0:i],randomnumberfromthelist)
        end_time = time( ) # record the ending time
        elapsed_unsorted.insert(j,end_time-start_time) # compute the elapsed time
        j+=1
print ("\nUnsorted Linear Search times")
for i in range(10):
        print (elapsed_unsorted[i])
print

for i in range(10000,100001,10000):
        start_time = time( ) # record the starting time
        sortedLinearSearch(mysortedrandomnumber[0:i],randomnumberfromthelist)
        end_time = time( ) # record the ending time
        elapsed_sorted.insert(j,end_time-start_time) # compute the elapsed time
        j+=1
print ("\nSorted Linear Search times")
for i in range(10):
        print (elapsed_sorted[i])
print

for i in range(10000,100001,10000):
        start_time = time( ) # record the starting time
        findSmallest(myrandomnumber[0:i])
        end_time = time( ) # record the ending time
        elapsed_smallest.insert(j,end_time-start_time) # compute the elapsed time
        j+=1
print ("\nFinding Smallest element times")
for i in range(10):
        print (elapsed_smallest[i])
print

for i in range(10000,100001,10000):
        start_time = time( ) # record the starting time

        #run algorithm
        binarySearch(myrandomnumber[0:i],randomnumberfromthelist)
        end_time = time( ) # record the ending time
        elapsed_binary.insert(j,end_time-start_time) # compute the elapsed time
        j+=1
print ("\nBinary Search times")
for i in range(10):
        print (elapsed_binary[i])

```

Output:

```
Unsorted Linear Search times
0.0010902881622314453
0.0019898414611816406
0.004260063171386719
0.003995656967163086
0.0045986175537109375
0.007964372634887695
0.007730245590209961
0.010812997817993164
0.011041402816772461
0.013585567474365234

Sorted Linear Search times
0.0008881092071533203
0.0018243789672851562
0.003809690475463867
0.003592967987060547
0.00572657585144043
0.007056474685668945
0.0075604915618896484
0.009517669677734375
0.009590625762939453
0.011223077774047852

Finding Smallest element times
0.0009272098541259766
0.00168609619140625
0.0027093887329101562
0.00432276725769043
0.006988048553466797
0.006250858306884766
0.00962209701538086
0.009847402572631836
0.01268315315246582
0.013735294342041016

Binary Search times
0.00015616416931152344
0.0002765655517578125
0.0004169940948486328
0.0005044937133789062
0.0006313323974609375
0.000978708267211914
0.0010445117950439453
0.001201629638671875
0.002380847930908203
0.002843618392944336
```

Graph:

Input Size	Linear unsorted Search	Linear Sorted Search	Finding Smallest Element	Binary Search
10000	0.0010902882	0.0008881092	0.0009272099	0.0001561642
20000	0.0019898415	0.001824379	0.0016860962	0.0002765656
30000	0.0042600632	0.0038096905	0.0027093887	0.0004169941
40000	0.003995657	0.003592968	0.0043227673	0.0005044937
50000	0.0045986176	0.0057265759	0.0069880486	0.0006313324
60000	0.0079643726	0.0070564747	0.0062508583	0.0009787083
70000	0.0077302456	0.0075604916	0.009622097	0.0010445118
80000	0.0108129978	0.0095176697	0.0098474026	0.0012016296
90000	0.0110414028	0.0095906258	0.0126831532	0.0023808479
100000	0.0135855675	0.0112230778	0.0137352943	0.0028436184



Conclusion:

Hence, from the graph we can see that search times correspond to O(1) and O(n) for linear search and O(1) and O(log n) for binary search.