Algorithms Assignment 1

Saturday, August 17, 2019 2:47 AM

- Assignment 1

Write two search functions linear search and binary search.
Create a random array of size n, perform m searches on the array, where the element to be searched is also determined randomly. Calculate the no. of comparisons your function makes. Vary the value of m from m=1,1000, 5000, 8000, 10000. Plot the graph m vs no. of comparisons for both linear and binary search on the same graph. For binary search also consider the number of comparisons of your sorting algorithm on the array which you use only once. Make 3 search graphs for n=1000, 5000, 10000.

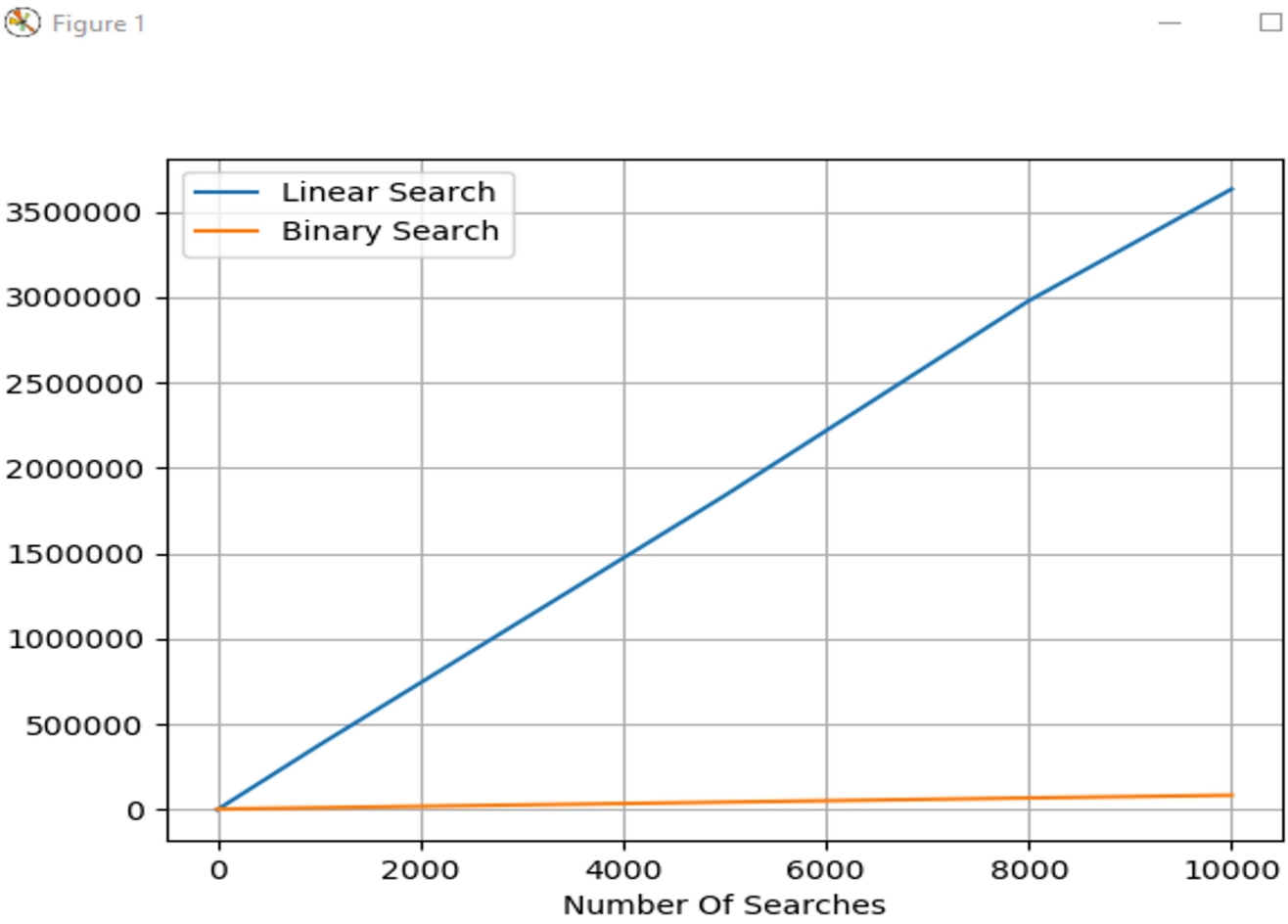
- **Code :**

```
import random
import matplotlib.pyplot as plt
sizesOfArray=[1000,5000,10000]
noofSearches=[1,1000,5000,8000,10000]
def linearsearch(arr, x):
    count=0
    for i in range(len(arr)):
        count+=1
        if arr[i] == x:
            return count
def binarySearch(arr,x, l, r ):
    count=0
    while l <= r:
        mid = l + (r - l)//2;
        if arr[mid] == x:
            count+=1
            return count
        elif arr[mid] < x:
            count+=1
            l = mid + 1
        else:
            count+=1
            r = mid - 1
    return count
def heapify(arr, n, i):
    count=0
    largest = i
    l = 2 * i + 1
    r = 2 * i + 2
    if l < n and arr[i] < arr[l]:
        count+=1
        largest = l
    if r < n and arr[largest] < arr[r]:
        count+=1
        largest = r
    if largest != i:
```

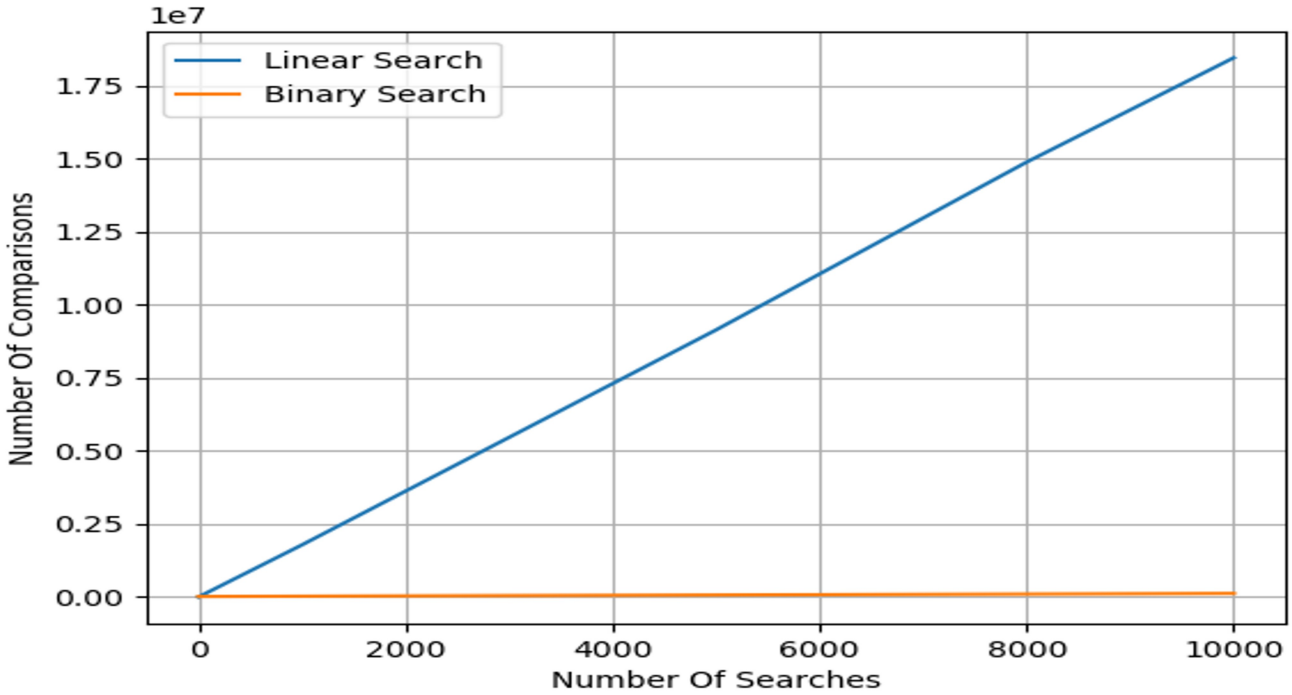
```
        count+=1
        arr[i],arr[largest] = arr[largest],arr[i]
        heapify(arr, n, largest)
    return count
def heapSort(arr):
    count=0
    n = len(arr)
    for i in range(n, -1, -1):
        count=count+heapify(arr, n, i)
    for i in range(n-1, 0, -1):
        arr[i], arr[0] = arr[0], arr[i]
        count=count+heapify(arr, i, 0)
    return count
for i in range(len(sizesOfArray)):
    sizeOfArray=sizesOfArray[i]
    print('For n = '+str(sizeOfArray)+':')
    rangeOfFullArray=list(range(0,sizeOfArray+1))
    randomlyGeneratedArray=[]
    for i in range(sizeOfArray):
        randomlyGeneratedInt=random.choice(rangeOfFullArray)
        randomlyGeneratedArray+=[randomlyGeneratedInt]
    nosOfComparisonsForLinearSearch=[]
    nosOfComparisonsForBinarySearch=[]
    for j in range(len(noofSearches)):
        Searches=noofSearches[j]
        totalNoOfComparisonsOfBinarySearch=0
        if j==0:
            previousRandomArray=randomlyGeneratedArray[:]
            sortedArray=randomlyGeneratedArray
            heapsortComparisons=heapSort(sortedArray)
        totalNoOfComparisonsOfLinearSearch=0
        totalNoOfComparisonsOfBinarySearch+=heapsortComparisons
        for k in range(Searches):
            randomlyGeneratedNumber=random.choice(randomlyGeneratedArray)
            totalNoOfComparisonsOfLinearSearch+=linearsearch(previousRandomArray,randomlyGeneratedNumber)
            totalNoOfComparisonsOfBinarySearch+=binarySearch(sortedArray,randomlyGeneratedNumber,0,len(sortedArray)-1)
        print('For m = '+str(Searches)+' Number Of Comparisons In Linear Search = '+str(totalNoOfComparisonsOfLinearSearch))
        print('For m = '+str(Searches)+' Number Of Comparisons In Binary Search = '+str(totalNoOfComparisonsOfBinarySearch))
        nosOfComparisonsForLinearSearch+=[totalNoOfComparisonsOfLinearSearch]
        nosOfComparisonsForBinarySearch+=[totalNoOfComparisonsOfBinarySearch]
    plt.xlabel('Number Of Searches')
    plt.ylabel('Number Of Comparisons')
    plt.plot(noofSearches,nosOfComparisonsForLinearSearch,label='Linear Search')
    plt.plot(noofSearches,nosOfComparisonsForBinarySearch,label='Binary Search')
    plt.grid()
    plt.legend()
    plt.show()
```

- # Output :

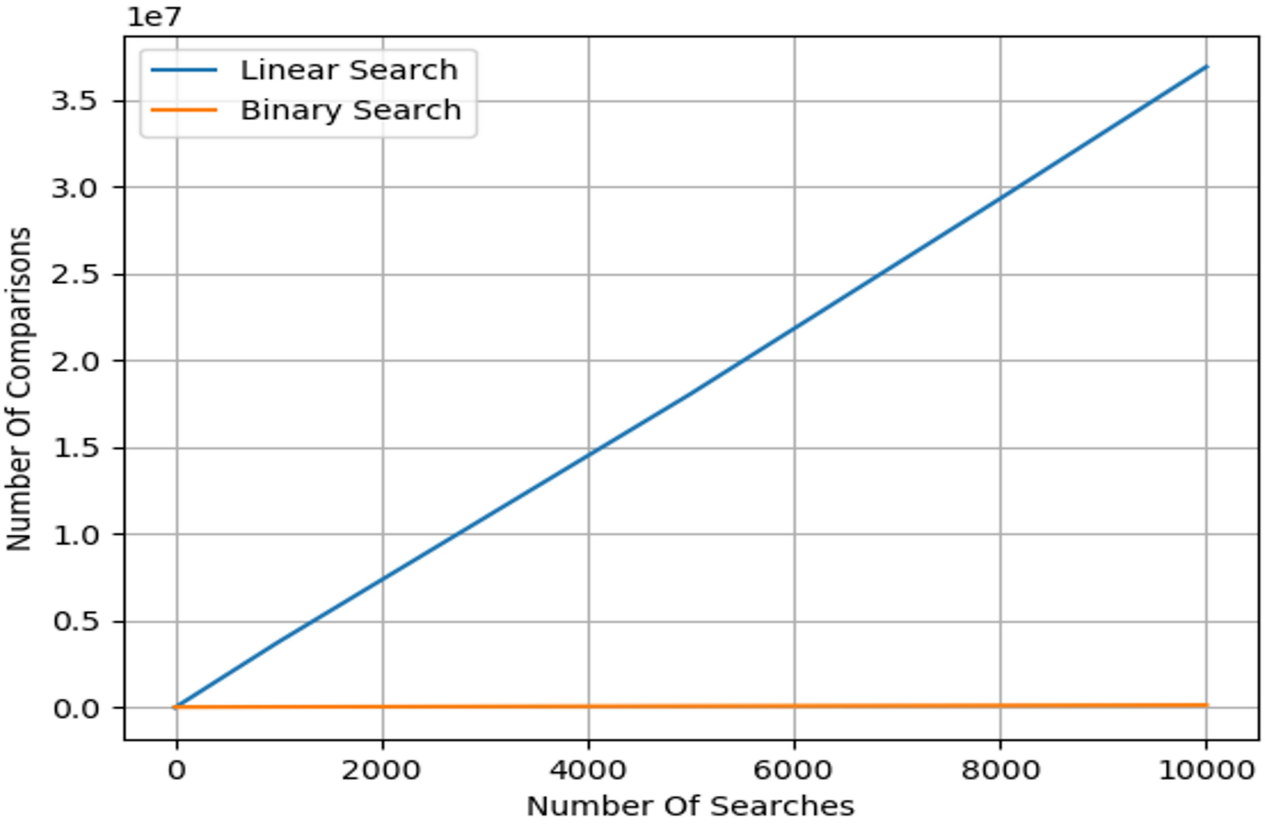
- **For n=1000 :**



- **For n=5000 :**



- **For n=10000 :**


1e7
Linear Search
Binary Search
3.5
3.0
2.5
2.0
1.5
1.0
0.5
0.0
Number Of Comparisons
0
2000
4000
6000
8000
10000
Number Of Searches