Kathmandu University

Department of Computer Science and Engineering

Dhulikhel, Kavre

Algorithms and Complexity (COMP 314) – Lab 1

Submitted To:

Dr. Rajani Chulyadyo

Department of Computer Science and Engineering

Submitted By:

Mani Dumaru (15)

CE-2019 Year/Sem: 3rd Year/2nd Sem

Submission Date: 12th April, 2022

Implementation, Testing and Performance measurement of Linear Search and Binary Search Algorithms.

Implementation

Linear Search Algorithm:

```
3   def linear_search(data, target):
4       for i in range(len(data)):
5           if (data[i] == target):
6               return i
7       return -1
8
```

Binary Search Algorithm:

```
8
9   def binary_search(data, target, l_index, r_index):
10      mid = (l_index + r_index)/2
11      mid = floor(mid)
12
13      try:
14          if (target > data[mid]):
15              l_index = mid + 1
16              return binary_search(data, target, l_index, r_index)
17          elif (target < data[mid]):
18              r_index = mid - 1
19              return binary_search(data, target, l_index, r_index)
20          else:
21              return mid
22      except:
23          return -1
24
```

Testing

Testing of Linear Search Algorithm:

```python
import unittest
from search import linear_search, binary_search

class searchTest(unittest.TestCase):

    def test_linearSearch(self):
        print("linear_search")
        values = [2,4,3,1,0,7]
        self.assertEqual(linear_search(values, 4), 1)
        self.assertEqual(linear_search(values, 0), 4)

    # def test_binarySearch(self):
    #     print("binary_search")
    #     values = [4,8,11,12,17,34,46]
    #     self.assertEqual(binary_search(values, 8, 0, len(values)), 1)
    #     self.assertEqual(binary_search(values, 12, 0, len(values)), 3)


if __name__ == "__main__":
    unittest.main()
```

Output of Testing Linear Search

```
D:\CE-2019\Sem 6\Algorithm\Lab\Lab1>python test.py
linear_search
.
----------------------------------------------------------------------
Ran 1 test in 0.001s

OK
```

Output of Testing Binary Search

```
D:\CE-2019\Sem 6\Algorithm\Lab\Lab1>python test.py
binary_search
.
----------------------------------------------------------------------
Ran 1 test in 0.000s

OK
```

Performance

Code for generating random data and applying Linear and Binary Search algorithm to check the time required.

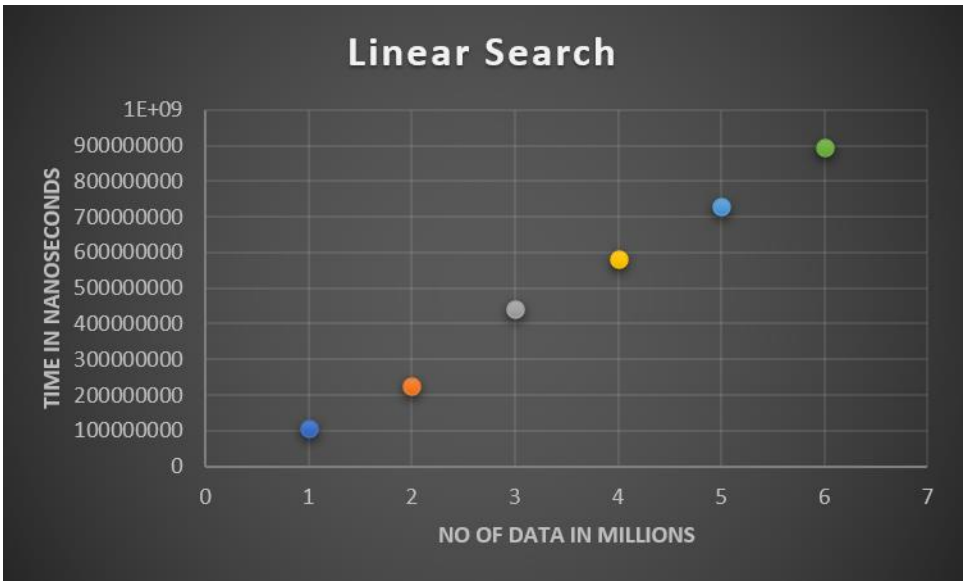
```python
from random import sample
from time import time_ns
from search import linear_search, binary_search

def run(n):
    data = sample(range(1, n*1000), n)
    # data = sorted(data)

    # print(data)
    start_time = time_ns()
    linear_search(data, data[-1])
    # binary_search(data, data[-1], 0, len(data))
    end_time = time_ns()

    total_time = end_time - start_time
    print(f"{n} data = {total_time} nanoseconds")



if __name__ == "__main__":
    n = 6000000
    run(n)
```

This code generates given number of random data and stores it in an array “data”. This array is then passed to both linear_search and binary_search algorithm to search for the very last element of array. Total time for searching is calculated using the time_ns function from time. Time vs Number of data graph is then plotted to observe the curve.

Output for Linear Search:

```
D:\CE-2019\Sem 6\Algorithm\Lab\Lab1>python generate.py
1000000 data = 106719000 nanoseconds
2000000 data = 225404500 nanoseconds
3000000 data = 490689200 nanoseconds
4000000 data = 581435100 nanoseconds
5000000 data = 731051900 nanoseconds
6000000 data = 893647900 nanoseconds
```

Graph for Linear Search:

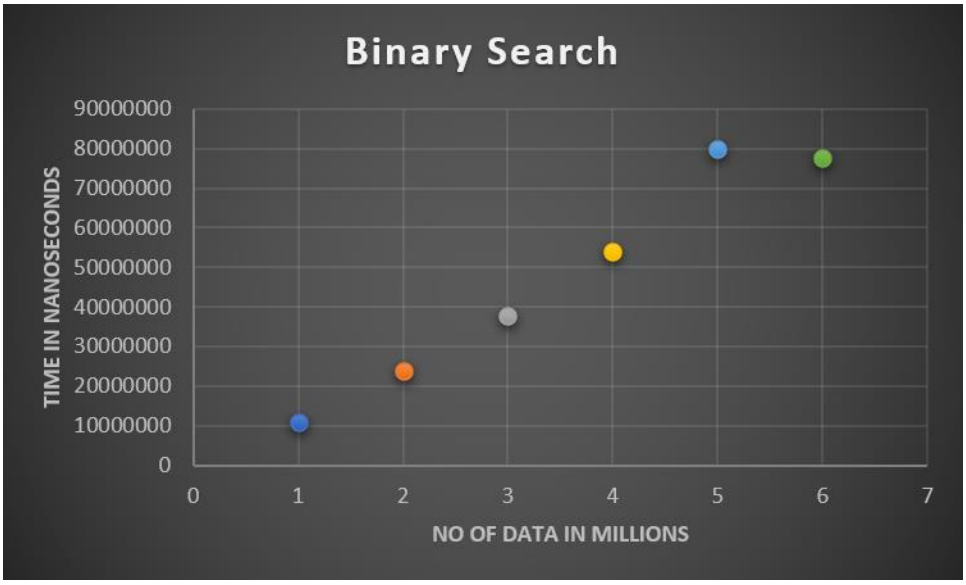


From this graph, we can observe that the linear search algorithm gives a linear path while plotting no. of data vs time graph. This means that the time complexity for linear graph is O(n).

Output for Binary Search:

```
D:\CE-2019\Sem 6\Algorithm\Lab\Lab1>python generate.py
1000000 data = 10969200 nanoseconds

D:\CE-2019\Sem 6\Algorithm\Lab\Lab1>python generate.py
2000000 data = 23935300 nanoseconds

D:\CE-2019\Sem 6\Algorithm\Lab\Lab1>python generate.py
3000000 data = 37897200 nanoseconds

D:\CE-2019\Sem 6\Algorithm\Lab\Lab1>python generate.py
4000000 data = 53858200 nanoseconds

D:\CE-2019\Sem 6\Algorithm\Lab\Lab1>python generate.py
5000000 data = 79784500 nanoseconds

D:\CE-2019\Sem 6\Algorithm\Lab\Lab1>python generate.py
6000000 data = 77782300 nanoseconds

D:\CE-2019\Sem 6\Algorithm\Lab\Lab1>
```

Graph for Binary Search:



From this graph, we can see that the time complexity for binary search is O(log n).

Conclusion:

Hence, algorithm for linear and binary search was implemented and tested if they gave the appropriate results. Each of their performance was also tested based on the time they took to search the provided data from the list of millions of randomly generated data. Time complexity of Linear Search algorithm was found to be O(n) and O(log n) for Binary Search.