Kathmandu University

Department of Computer Science and Engineering

Dhulikhel, Kavre

A Report on '**Lab Work 1'** [COMP 314]

Submitted by:

Chandan Kumar Mahato (31)

III-year, II semester

Submitted to:

Dr. Rajani Chulyadyo

Department of Computer Science and Engineering

Submission Date: Apr 12, 2023

Qlab1

Implementation, testing and performance measurement of linear search and binary search algorithms.

Solution:

Implementing Linear Search, Binary Search Recursive, Binary Search Iterative algorithm using JavaScript (node.js) and writing test using (jest) a JavaScript testing framework. Plotting graph execution-time vs input-size using nodeplotlib.

Source Code:

https://github.com/ChandankMahato/DSA_Lab_6th_Sem

Scripts:

npm start
npm run test
npm run time
npm run graph

1. **Linear Search Implementation:**

```
module.exports = function linearSearch(array, x) {
  for (i in array)
    if (array[i] == x) {
      return i;
    }
  return -1;
};
```

2. **Recursive Binary Search Implementation:**

```
module.exports = function binarySearchRecursive(arr, low, high, x) {
  if (high >= low) {
    mid = Math.floor((high + low) / 2);
    if (arr[mid] == x) {
      return mid;
    } else if (arr[mid] > x) {
      return binarySearchRecursive(arr, low, mid - 1, x);
```

```
    } else return binarySearchRecursive(arr, mid + 1, high, x);
  } else return -1;
};
```

3. Iterative Binary Search Implementation:

```
module.exports = function binarySearchIterative(arr, x) {
  low = 0;
  high = arr.length - 1;
  mid = 0;
  while (low <= high) {
    mid = Math.floor((high + low) / 2);
    if (arr[mid] < x) {
      low = mid + 1;
    } else if (arr[mid] > x) {
      high = mid - 1;
    } else return mid;
  }
  return -1;
};
```

Test Implementation:

```
const binarySearchIterative = require("../../Search/binaryIterative");
const binarySearchRecursive = require("../../Search/binaryRecursive");
const linearSearch = require("../../Search/linearSearch");

describe("Search Algo Test", () => {
  it("should return index of element using linear search", () => {
    const result = linearSearch([2, 4, 1, 3, 7, 0], 1);
    expect(result).toEqual("2");
  });
  it("should return index of element using binary search recursive", () => {
    const result = binarySearchRecursive([2, 3, 4, 10, 40], 0, 5, 10);
    expect(result).toEqual(3);
  });
  it("should return index of element using binary search iterative", () => {
    const result = binarySearchIterative([2, 3, 4, 10, 40], 40);
    expect(result).toEqual(4);
  });
});
```

Performance Implementation:

```
const { performance } = require("perf_hooks");
const binarySearchIterative = require("../Search/binaryIterative");
const binarySearchRecursive = require("../Search/binaryRecursive");
const linearSearch = require("../Search/linearSearch");
const createGraph = require("./draw");

const linearPlotData = [];
const iterativePlotData = [];
const recursivePlotData = [];
let t0, t1, Obj;
function collectData(length) {
  const dataSet = Array.from({ length: length }, () =>
    Math.floor(Math.random() * length)
  );
  dataSet.sort((a, b) => a - b);
  const searchX = length + 1;

  t0 = performance.now();
  linearSearch(dataSet, searchX);
  t1 = performance.now();
  Obj = { x: length, y: (t1 - t0).toFixed(5) };
  linearPlotData.push(Obj);

  t0 = performance.now();
  binarySearchIterative(dataSet, searchX);
  t1 = performance.now();
  Obj = { x: length, y: (t1 - t0).toFixed(5) };
  iterativePlotData.push(Obj);

  t0 = performance.now();
  binarySearchRecursive(dataSet, 0, dataSet.length - 1, searchX);
  t1 = performance.now();
  Obj = { x: length, y: (t1 - t0).toFixed(5) };
  recursivePlotData.push(Obj);
}

for (let i = 1000; i <= 100000; i += 10000) {
  collectData(i);
}

createGraph(linearPlotData, "LinearSearch");
createGraph(recursivePlotData, "Binary Search Recursive");
createGraph(iterativePlotData, "Binary Search Iterative");
```

Graph Implementation:

```
const { plot } = require("nodeplotlib");

module.exports = function (data, type) {
  let xArray = [];
  let yArray = [];
  data.map((obj) => {
    xArray.push(obj.x);
    yArray.push(obj.y);
  });

  const plotData = [
    {
      x: xArray,
      y: yArray,
      type: "scatter",
    },
  ];

  const layout = {
    title: type,
    xaxis: {
      title: "Input Size",
    },
    yaxis: {
      title: "Execution Time",
    },
  };

  plot(plotData, layout);
};
```

Observation:

Linear search and binary search are two algorithms used to search for an element in an array. Linear search is a simple algorithm that iterates through each element of the array until it finds the target element. Although easy to implement, it has a time complexity of O(n), where n is the number of elements in the array.

In contrast, binary search is a more efficient algorithm that divides the array into two halves and searches for the target element in one of the halves. It has a time complexity of O(logn), where n is the number of elements in the array.

However binary search requires the array to be sorted before searching can begin. Linear search is useful when the array is small or unsorted, while binary search is more efficient for large arrays or when the array is sorted.

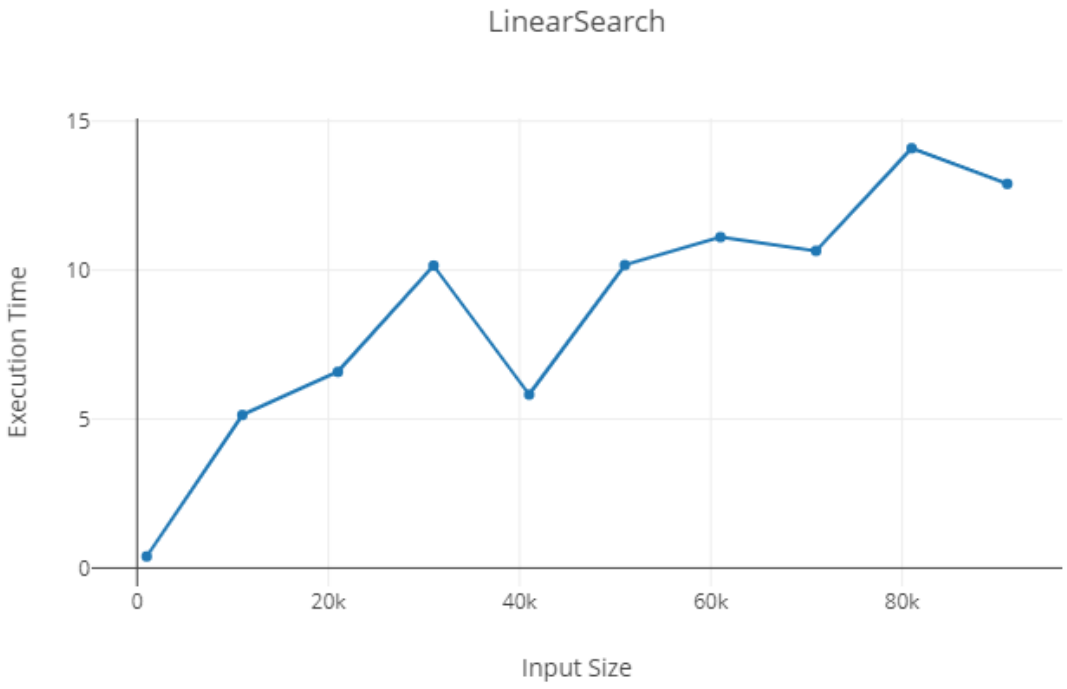
Conclusion:

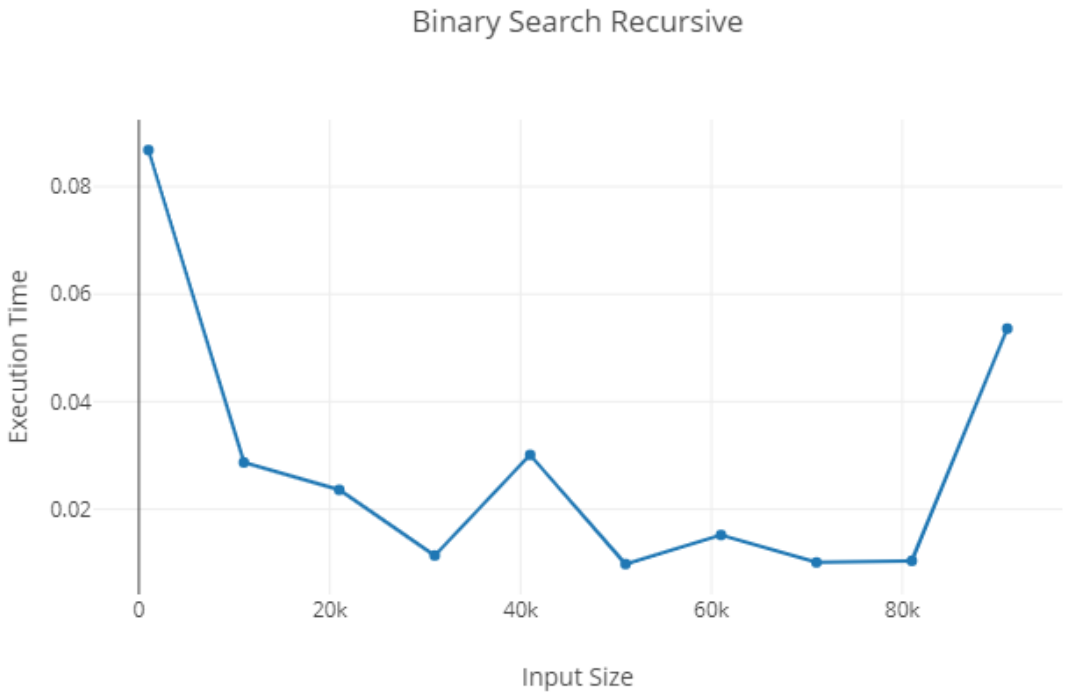
Linear Search and binary Search (recursive, iterative) were implemented in the JavaScript. The written code was benchmarked, a graph was plotted (execution-time vs input-size), and test cases were written using jest.

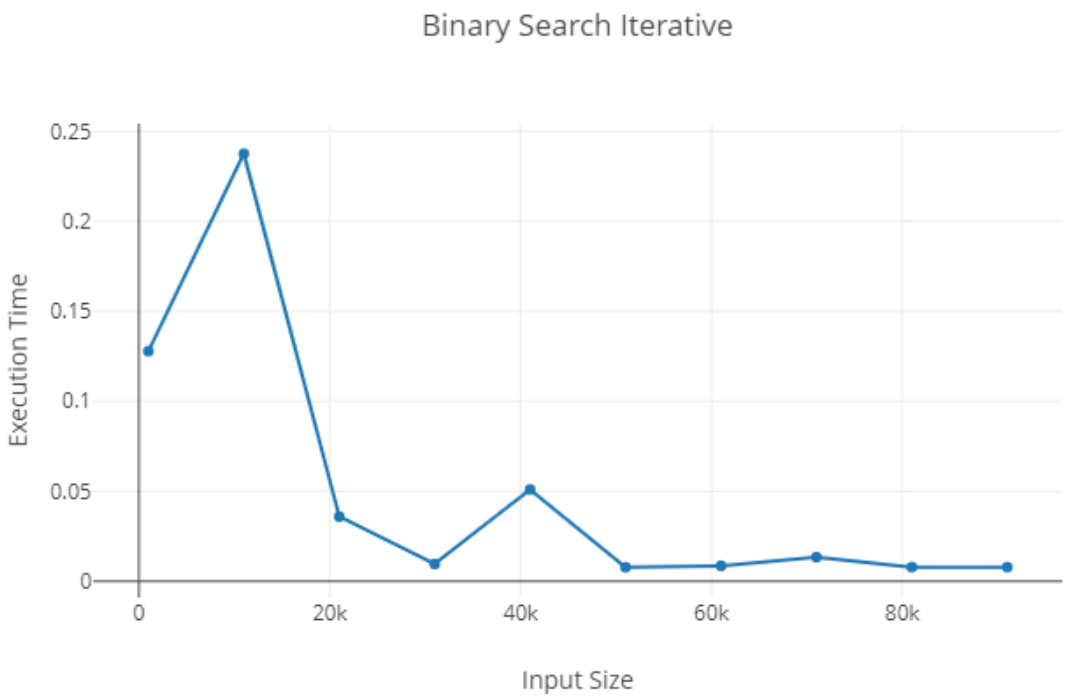
Output and Screenshots:

```
 PASS  tests/unit/search.test.js
  Search Algo Test
    √ should return index of element using linear search (9 ms)
    √ should return index of element using binary search recursive (2 ms)
    √ should return index of element using binary search iterative (3 ms)

--------------------|---------|----------|---------|---------|-------------------
File                | % Stmts | % Branch | % Funcs | % Lines | Uncovered Line #s
--------------------|---------|----------|---------|---------|-------------------
 PASS  tests/unit/search.test.js
  Search Algo Test
    √ should return index of element using linear search (6 ms)
    √ should return index of element using binary search recursive (3 ms)
    √ should return index of element using binary search iterative (3 ms)

--------------------|---------|----------|---------|---------|-------------------
File                | % Stmts | % Branch | % Funcs | % Lines | Uncovered Line #s
--------------------|---------|----------|---------|---------|-------------------
All files           |   84.61 |    83.33 |     100 |   84.61 |
 binaryIterative.js |   83.33 |       75 |     100 |   83.33 | 10,13
 binaryRecursive.js |   88.88 |    83.33 |     100 |   88.88 | 9
 linearSearch.js    |      80 |      100 |     100 |      80 | 6
--------------------|---------|----------|---------|---------|-------------------
Test Suites: 1 passed, 1 total
Tests:       3 passed, 3 total
Snapshots:   0 total
Time:        0.91 s, estimated 3 s
Ran all test suites.

Watch Usage: Press w to show more.
```

```
A:\Chandan Semester Work\6th sem\Algorithm and complexity\DSA_Lab_6th_Sem\Lab1>npm start

> lab1_test_demo@1.0.0 start
> node index.js

binarySearchIterative
Element Found at: 4
binarySearchRecursive
Element Found at: 3
linearSearch
Element Found at: 2

A:\Chandan Semester Work\6th sem\Algorithm and complexity\DSA_Lab_6th_Sem\Lab1>npm run time

> lab1_test_demo@1.0.0 time
> node Performance/time.js

LinearSearch
Element Not Found
Time taken to search: 457.4684000015259 ms

binarySearchIterative
Element Not Found
Time taken to search: 0.31869999691843987 ms

binarySearchRecursive
Element Not Found
Time taken to search: 0.999499998986721 ms


A:\Chandan Semester Work\6th sem\Algorithm and complexity\DSA_Lab_6th_Sem\Lab1>npm run graph

> lab1_test_demo@1.0.0 graph
> node Performance/graphData.js

[Nodeplotlib] App is already up and running
[Nodeplotlib] App is already up and running
[Nest] 12404  - 04/12/2023, 2:08:19 PM     LOG [NestFactory] Starting Nest application...
[Nest] 12404  - 04/12/2023, 2:08:19 PM     LOG [InstanceLoader] ServerModule dependencies initialized +80ms
[Nest] 12404  - 04/12/2023, 2:08:19 PM     LOG [InstanceLoader] ServeStaticModule dependencies initialized +3ms
[Nest] 12404  - 04/12/2023, 2:08:19 PM     LOG [WebSocketsController] PlotsGateway subscribed to the "readplots" message +174ms
[Nest] 12404  - 04/12/2023, 2:08:19 PM     LOG [NestApplication] Nest application successfully started +10ms
[Nodeplotlib] Server running at http://localhost:60311
[Nodeplotlib] client connected
```


LinearSearch
15
10
5
0
Execution Time
0
20k
40k
60k
80k
Input Size


Binary Search Recursive
0.08
0.06
0.04
0.02
Execution Time
0
20k
40k
60k
80k
Input Size


Binary Search Iterative
Execution Time
0.25
0.2
0.15
0.1
0.05
0
0
20k
40k
60k
80k
Input Size