# CSC 236 Practical

## 2020/2021 Session


Adeoti Warith Adetayo

Computer Science

Matric No: 214851


# Analysis of Binary and Tenary Search Algortihms

CSC 236 Practical

# Report on Analysis of Binary and Tenary Searching Algorithms

## Codes

Programming Language: **C#**
Type of Algorithm Implementation: **Recursive**

Structure of program:

- There are Two files
  - ***util.cs***: Containing the important codes necessary for the program (the searching and random number generation routines)
  - ***Program.cs***: The driver code of the program. Contains the following routines
    - ***Main():*** the entry point of the program. Call to RandomNumber Generator and ImplementAlgortihm routines are made here
    - ***ImplementAlgorithm():*** this is the routine that performs the analysis making calls to the rest of the routines present in the code
    - ***Run():*** this routines calls a particular searching algorithm specified by the caller and returns the runtime of the algorithm
    - ***ExportData():*** this routine is used to export all the generated data (runtimes)
    - ***GiveSummary()*:** used to display a brief summary of the runtime of each searching algorithm
    - ***SortArrays():*** used to sort all the randomly generated arrays

***util.cs***

```
using System;

namespace Utility
{
    class Generator
    {
        public static int[] RandomArray( int size, int lRange, int uRange )
        {
            Random rn = new Random();
            int[] array = new int[size];

            for (int i = 0; i < size; i++)
            {
                array[i] = rn.Next(lRange, uRange);
            }

            return array;
        }
    }
    class Search
    {
```

```
        public static bool Binary( int[] array, int x )
        {
            static bool BinarySearch(int[] array, int start, int end, int x)
            {
                int mid = start + (end - start) / 2;

                if (start > end) return false;

                else if (x == array[mid]) return true;

                else if (x < array[mid]) return BinarySearch(array, start, mid - 1, x);

                else return BinarySearch(array, mid + 1, end, x);
            }

            return BinarySearch(array, 0, array.Length - 1, x);
        }

        public static bool Tenary( int [] array, int x)
        {
            static bool TenarySearch(int[] array, int start, int end, int x)
            {
                int div = (end - start) / 3;
                int mid1 = start + div;
                int mid2 = end - div;



                if (start > end) {return false;}

                else if (x == array[mid1] || x == array[mid2]) {return true;}

                else if (x < array[mid1]) {return TenarySearch(array, start, mid1 - 1, x);}

                else if (x < array[mid2]) {return TenarySearch(array, mid1 + 1, mid2 - 1, x);
}

                else {return TenarySearch(array, mid2 + 1, end, x);}
            }

            return TenarySearch(array, 0, array.Length - 1, x);
        }
    }


    class Util
    {
        public static string Repeat(string str, int num)
        {
            string rep = "";
```

```
            for (int i = 0; i < num; i++)
            {
                rep += str;
            }

            return rep;
        }
    }
}
```

Program.cs

```
using System;
using Utility;
using System.Collections.Generic;
using System.Diagnostics;
using System.IO;

namespace Searching
{
    class Program
    {
        static int p = 30;
        static string[] NUMBERING = {"I", "II", "III", "IV"};
        static int step = 0;

        static void Main(string[] args)
        {
            // Searching Algorithm Implementation & Analysis
            Console.WriteLine("\t\t\tBINARY AND TENARY SEARCHING ALGORITHMS ANALYSIS");
            Console.WriteLine("\n\n");

            Console.WriteLine("STEP " + NUMBERING[step]);
            step++;
            Console.WriteLine("Generating Random arrays of sizes 50000, 100000, 250000, 50000
0 and 750000 with values between 1 and 100");
            Console.WriteLine(Util.Repeat("___", p));

            int[] arraySizes = {50000, 100000, 250000, 500000, 750000};
            int[][] randomArrays = new int[arraySizes.Length][];

            for (int i = 0; i < arraySizes.Length; i++)
            {
                Console.WriteLine($"\tGenerating Array: Size = {arraySizes[i]}...");
                randomArrays[i] = Generator.RandomArray(arraySizes[i], 1, 100);
                Console.WriteLine("\tArray Generated!!!\n");
            }

            Console.WriteLine(Util.Repeat("___", p));
```

```
            Console.WriteLine("\n\n ");

            string[] arrayTypes = {"Unsorted","Sorted"};
            int[] searchElement = {70, 120};
            bool sort = true;

            foreach (string arraytype in arrayTypes)
            {
                Console.WriteLine("STEP " + NUMBERING[step]);
                step++;
                Console.WriteLine($"Running the Searching Algorithm on {arraytype} arrays");
                Console.WriteLine(Util.Repeat("___", p));
                foreach(int x in searchElement)
                {
                    Console.WriteLine("\n");

                    string msg = (x == searchElement[0]) ? "Element within the Array" : "Elem
ent outside the Array";

                    Console.WriteLine($"Search Element: {x} ({msg})\n");

                    Console.WriteLine(Util.Repeat("---", p - 5));

                    ImplementAlgorithm(randomArrays, arraySizes, arraytype, x, msg, $"test(se
arch_{x}_in_{arraytype}_Array)");

                    Console.WriteLine(Util.Repeat("---", p - 5));

                }
                Console.WriteLine(Util.Repeat("___", p));

                if (sort)
                {
                    Console.WriteLine("\n\n");
                    SortArrays(randomArrays);
                    sort = false;
                    Console.WriteLine("\n\n");
                }

            }

            Console.WriteLine("\nEND OF PROGRAM!!!");


        }

        static void ImplementAlgorithm(int[][] arrays, int[] arraySizes, string arrayType,  i
nt x, string msg, string exportFileName)
        {
            // Implementing Binary Search Algorithm
```

```
        Console.WriteLine("Implementing Binary Search...");
        double[] binarySearchTime = new double[arrays.Length];

        for (int i = 0; i < arrays.Length; i++)
        {
            binarySearchTime[i] = Run(arrays[i], x, "Binary");
        }

        Console.WriteLine(Util.Repeat("-----", 10));

        // Implementing Tenary Search Algorithm
        Console.WriteLine("Implementing Tenary Search...");
        double[] tenarySearchTime = new double[arrays.Length];

        for (int i = 0; i < arrays.Length; i++)
        {
            tenarySearchTime[i] = Run(arrays[i], x, "Tenary");
        }

        Console.WriteLine("\n");

        double[][] searchTime = {binarySearchTime, tenarySearchTime};

        GiveSummary(searchTime, arraySizes, arrayType, msg);

        ExportData(arraySizes, binarySearchTime, tenarySearchTime, exportFileName);

    }

    static double Run(int[] array, int x, string algorithm)
    {
        string tab = "\t";
        Console.WriteLine(tab + "Implementing Search");
        Console.WriteLine(tab + $"Algorithm: {algorithm}");
        Console.WriteLine(tab + $"Array Size: {array.Length}");
        Console.WriteLine(tab + $"Searching for: {x}");
        Console.WriteLine(tab + "Searching...");

        bool status;

        // start stopwatch
        Stopwatch watch = new Stopwatch();
        watch.Start();
        if (algorithm == "Linear")
            status = Search.Linear(array, x);
        else if (algorithm == "Binary")
            status = Search.Binary(array, x);
        else
            status = Search.Tenary(array, x);
```

```
            watch.Stop();
            double runtime = watch.Elapsed.TotalMilliseconds * 1000000;
            watch.Reset();
            // end stopwatch

            Console.WriteLine(tab + "Searching Completed");
            string msg = (status) ? "Element Found!!!" : "Element Not Found!!!";
            Console.WriteLine(tab + msg);
            Console.WriteLine(tab + $"Runtime: {runtime}ns");

            Console.WriteLine("\n");

            return runtime;

        }

        static void ExportData(int[] arraySizes, double[] binarySearchTime, double[] tenarySe
archTime, string filename)
        {
            StreamWriter writer = new StreamWriter(filename + ".csv");

            string line1 = "," + string.Join(",", arraySizes);
            string line2 = "Binary," + string.Join(",", binarySearchTime);
            string line3 = "Tenary," + string.Join(",", tenarySearchTime);

            writer.WriteLine(line1);
            writer.WriteLine(line2);
            writer.WriteLine(line3 + "\n");
            writer.Close();

            Console.WriteLine("Data exported!!! " + filename + ".csv");
        }

        static void GiveSummary(double[][] searchTime, int[] arraySizes, string arrayType, st
ring msg)
        {
            Console.Write("SUMMARY OF TEST: ");
            Console.WriteLine("\tRuntime(ns) of Binary and Tenary Search on " + arrayType + "
 arrays (for " + msg + ")\n");

            Console.WriteLine("Array Size\tBinary Search Time\tTenary Search Time");

            for (int i = 0; i < arraySizes.Length; i++)
            {
                Console.WriteLine(arraySizes[i] + "\t\t\t" + (int)searchTime[0][i] + "\t\t\t"
 + (int)searchTime[1][i]);
            }
        }

        static void SortArrays(int[][] randomArrays)
```

```
        {
            Console.WriteLine("STEP " + NUMBERING[step]);
            step++;
            Console.WriteLine("Sorting all Arrays");
            Console.WriteLine(Util.Repeat("___", p));
            for (int i = 0; i < randomArrays.Length; i++)
            {
                int[] array = randomArrays[i];
                Array.Sort(array);

                randomArrays[i] = array;
                Console.WriteLine($"Array Size: {array.Length}");
                Console.WriteLine("Array Sorted!!!");
            }
            Console.WriteLine("All Arrays Sorted!!!");
            Console.WriteLine(Util.Repeat("___", p));
        }
    }
}
```

## Output of the program

```
            BINARY AND TENARY SEARCHING ALGORITHMS ANALYSIS



STEP I
Generating Random arrays of sizes 50000, 100000, 250000, 500000 and 750000 with values betwee
n 1 and 100
_____________________________________________________________________________________________
    Generating Array: Size = 50000...
    Array Generated!!!

    Generating Array: Size = 100000...
    Array Generated!!!

    Generating Array: Size = 250000...
    Array Generated!!!

    Generating Array: Size = 500000...
    Array Generated!!!

    Generating Array: Size = 750000...
    Array Generated!!!


_____________________________________________________________________________________________
```

```
STEP II
Running the Searching Algorithm on Unsorted arrays
_____________________________________________________________________________________


Search Element: 70 (Element within the Array)

-------------------------------------------------------------------------
Implementing Binary Search...
    Implementing Search
    Algorithm: Binary
    Array Size: 50000
    Searching for: 70
    Searching...
    Searching Completed
    Element Not Found!!!
    Runtime: 1047900ns


    Implementing Search
    Algorithm: Binary
    Array Size: 100000
    Searching for: 70
    Searching...
    Searching Completed
    Element Not Found!!!
    Runtime: 6500ns


    Implementing Search
    Algorithm: Binary
    Array Size: 250000
    Searching for: 70
    Searching...
    Searching Completed
    Element Found!!!
    Runtime: 2700ns


    Implementing Search
    Algorithm: Binary
    Array Size: 500000
    Searching for: 70
    Searching...
    Searching Completed
    Element Not Found!!!
    Runtime: 7300ns


    Implementing Search
```

```
    Algorithm: Binary
    Array Size: 750000
    Searching for: 70
    Searching...
    Searching Completed
    Element Not Found!!!
    Runtime: 5100ns


--------------------------------------------------
Implementing Tenary Search...
    Implementing Search
    Algorithm: Tenary
    Array Size: 50000
    Searching for: 70
    Searching...
    Searching Completed
    Element Not Found!!!
    Runtime: 1734700ns


    Implementing Search
    Algorithm: Tenary
    Array Size: 100000
    Searching for: 70
    Searching...
    Searching Completed
    Element Not Found!!!
    Runtime: 5500ns


    Implementing Search
    Algorithm: Tenary
    Array Size: 250000
    Searching for: 70
    Searching...
    Searching Completed
    Element Not Found!!!
    Runtime: 8300ns


    Implementing Search
    Algorithm: Tenary
    Array Size: 500000
    Searching for: 70
    Searching...
    Searching Completed
    Element Not Found!!!
    Runtime: 8300ns
```

```
    Implementing Search
    Algorithm: Tenary
    Array Size: 750000
    Searching for: 70
    Searching...
    Searching Completed
    Element Not Found!!!
    Runtime: 7600ns




SUMMARY OF TEST:    Runtime(ns) of Binary and Tenary Search on Unsorted arrays (for Element w
ithin the Array)

Array Size  Binary Search Time  Tenary Search Time
50000           1047900         1734700
100000          6500            5500
250000          2700            8300
500000          7300            8300
750000          5100            7600
Data exported!!! test(search_70_in_Unsorted_Array).csv
--------------------------------------------------------------------------------


Search Element: 120 (Element outside the Array)

--------------------------------------------------------------------------------
Implementing Binary Search...
    Implementing Search
    Algorithm: Binary
    Array Size: 50000
    Searching for: 120
    Searching...
    Searching Completed
    Element Not Found!!!
    Runtime: 6800ns


    Implementing Search
    Algorithm: Binary
    Array Size: 100000
    Searching for: 120
    Searching...
    Searching Completed
    Element Not Found!!!
    Runtime: 4000ns


    Implementing Search
    Algorithm: Binary
```

```
    Array Size: 250000
    Searching for: 120
    Searching...
    Searching Completed
    Element Not Found!!!
    Runtime: 4300ns


    Implementing Search
    Algorithm: Binary
    Array Size: 500000
    Searching for: 120
    Searching...
    Searching Completed
    Element Not Found!!!
    Runtime: 5900ns


    Implementing Search
    Algorithm: Binary
    Array Size: 750000
    Searching for: 120
    Searching...
    Searching Completed
    Element Not Found!!!
    Runtime: 8300ns


--------------------------------------------------
Implementing Tenary Search...
    Implementing Search
    Algorithm: Tenary
    Array Size: 50000
    Searching for: 120
    Searching...
    Searching Completed
    Element Not Found!!!
    Runtime: 5100ns


    Implementing Search
    Algorithm: Tenary
    Array Size: 100000
    Searching for: 120
    Searching...
    Searching Completed
    Element Not Found!!!
    Runtime: 6300ns


    Implementing Search
```

```
    Algorithm: Tenary
    Array Size: 250000
    Searching for: 120
    Searching...
    Searching Completed
    Element Not Found!!!
    Runtime: 5100ns


    Implementing Search
    Algorithm: Tenary
    Array Size: 500000
    Searching for: 120
    Searching...
    Searching Completed
    Element Not Found!!!
    Runtime: 5600ns


    Implementing Search
    Algorithm: Tenary
    Array Size: 750000
    Searching for: 120
    Searching...
    Searching Completed
    Element Not Found!!!
    Runtime: 5600ns




SUMMARY OF TEST:    Runtime(ns) of Binary and Tenary Search on Unsorted arrays (for Element o
utside the Array)

Array Size  Binary Search Time  Tenary Search Time
50000           6800            5100
100000          4000            6300
250000          4300            5100
500000          5900            5600
750000          8300            5600
Data exported!!! test(search_120_in_Unsorted_Array).csv
--------------------------------------------------------------------------------
________________________________________________________________________________



STEP III
Sorting all Arrays
________________________________________________________________________________
Array Size: 50000
Array Sorted!!!
```

```
Array Size: 100000
Array Sorted!!!
Array Size: 250000
Array Sorted!!!
Array Size: 500000
Array Sorted!!!
Array Size: 750000
Array Sorted!!!
All Arrays Sorted!!!
```

---

```
STEP IV
Running the Searching Algorithm on Sorted arrays
```

---

```
Search Element: 70 (Element within the Array)

------------------------------------------------------------------------
Implementing Binary Search...
    Implementing Search
    Algorithm: Binary
    Array Size: 50000
    Searching for: 70
    Searching...
    Searching Completed
    Element Found!!!
    Runtime: 4200ns


    Implementing Search
    Algorithm: Binary
    Array Size: 100000
    Searching for: 70
    Searching...
    Searching Completed
    Element Found!!!
    Runtime: 3100ns


    Implementing Search
    Algorithm: Binary
    Array Size: 250000
    Searching for: 70
    Searching...
    Searching Completed
    Element Found!!!
    Runtime: 3000ns
```

```
    Implementing Search
    Algorithm: Binary
    Array Size: 500000
    Searching for: 70
    Searching...
    Searching Completed
    Element Found!!!
    Runtime: 5000ns


    Implementing Search
    Algorithm: Binary
    Array Size: 750000
    Searching for: 70
    Searching...
    Searching Completed
    Element Found!!!
    Runtime: 5100ns


--------------------------------------------------
Implementing Tenary Search...
    Implementing Search
    Algorithm: Tenary
    Array Size: 50000
    Searching for: 70
    Searching...
    Searching Completed
    Element Found!!!
    Runtime: 3300ns


    Implementing Search
    Algorithm: Tenary
    Array Size: 100000
    Searching for: 70
    Searching...
    Searching Completed
    Element Found!!!
    Runtime: 4300ns


    Implementing Search
    Algorithm: Tenary
    Array Size: 250000
    Searching for: 70
    Searching...
    Searching Completed
    Element Found!!!
    Runtime: 4300ns
```

```
    Implementing Search
    Algorithm: Tenary
    Array Size: 500000
    Searching for: 70
    Searching...
    Searching Completed
    Element Found!!!
    Runtime: 3300ns


    Implementing Search
    Algorithm: Tenary
    Array Size: 750000
    Searching for: 70
    Searching...
    Searching Completed
    Element Found!!!
    Runtime: 2500ns




SUMMARY OF TEST:    Runtime(ns) of Binary and Tenary Search on Sorted arrays (for Element wit
hin the Array)

Array Size  Binary Search Time  Tenary Search Time
50000           4200            3300
100000          3100            4300
250000          3000            4300
500000          5000            3300
750000          5100            2500
Data exported!!! test(search_70_in_Sorted_Array).csv
--------------------------------------------------------------------------------


Search Element: 120 (Element outside the Array)

--------------------------------------------------------------------------------
Implementing Binary Search...
    Implementing Search
    Algorithm: Binary
    Array Size: 50000
    Searching for: 120
    Searching...
    Searching Completed
    Element Not Found!!!
    Runtime: 6100ns


    Implementing Search
    Algorithm: Binary
```

```
    Array Size: 100000
    Searching for: 120
    Searching...
    Searching Completed
    Element Not Found!!!
    Runtime: 4500ns


    Implementing Search
    Algorithm: Binary
    Array Size: 250000
    Searching for: 120
    Searching...
    Searching Completed
    Element Not Found!!!
    Runtime: 5500ns


    Implementing Search
    Algorithm: Binary
    Array Size: 500000
    Searching for: 120
    Searching...
    Searching Completed
    Element Not Found!!!
    Runtime: 5300ns


    Implementing Search
    Algorithm: Binary
    Array Size: 750000
    Searching for: 120
    Searching...
    Searching Completed
    Element Not Found!!!
    Runtime: 5400ns


--------------------------------------------------
Implementing Tenary Search...
    Implementing Search
    Algorithm: Tenary
    Array Size: 50000
    Searching for: 120
    Searching...
    Searching Completed
    Element Not Found!!!
    Runtime: 4500ns


    Implementing Search
```

```
    Algorithm: Tenary
    Array Size: 100000
    Searching for: 120
    Searching...
    Searching Completed
    Element Not Found!!!
    Runtime: 3900ns


    Implementing Search
    Algorithm: Tenary
    Array Size: 250000
    Searching for: 120
    Searching...
    Searching Completed
    Element Not Found!!!
    Runtime: 4300ns


    Implementing Search
    Algorithm: Tenary
    Array Size: 500000
    Searching for: 120
    Searching...
    Searching Completed
    Element Not Found!!!
    Runtime: 4700ns


    Implementing Search
    Algorithm: Tenary
    Array Size: 750000
    Searching for: 120
    Searching...
    Searching Completed
    Element Not Found!!!
    Runtime: 4800ns




SUMMARY OF TEST:    Runtime(ns) of Binary and Tenary Search on Sorted arrays (for Element out
side the Array)

Array Size  Binary Search Time  Tenary Search Time
50000           6100            4500
100000          4500            3900
250000          5500            4300
500000          5300            4700
750000          5400            4800
Data exported!!! test(search_120_in_Sorted_Array).csv
```

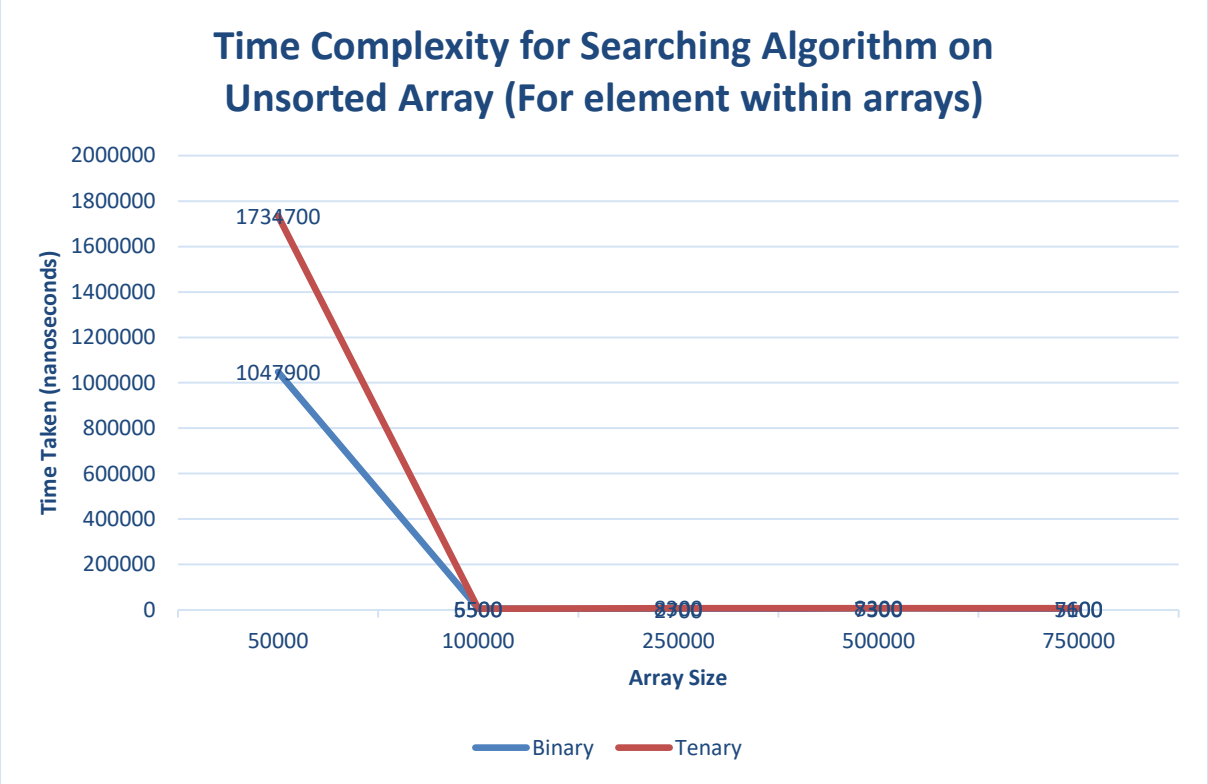
```
------------------------------------------------------------------------
_____________________________________________________________________________

END OF PROGRAM!!!
```

# Collected Data

## 1. For Unsorted Arrays

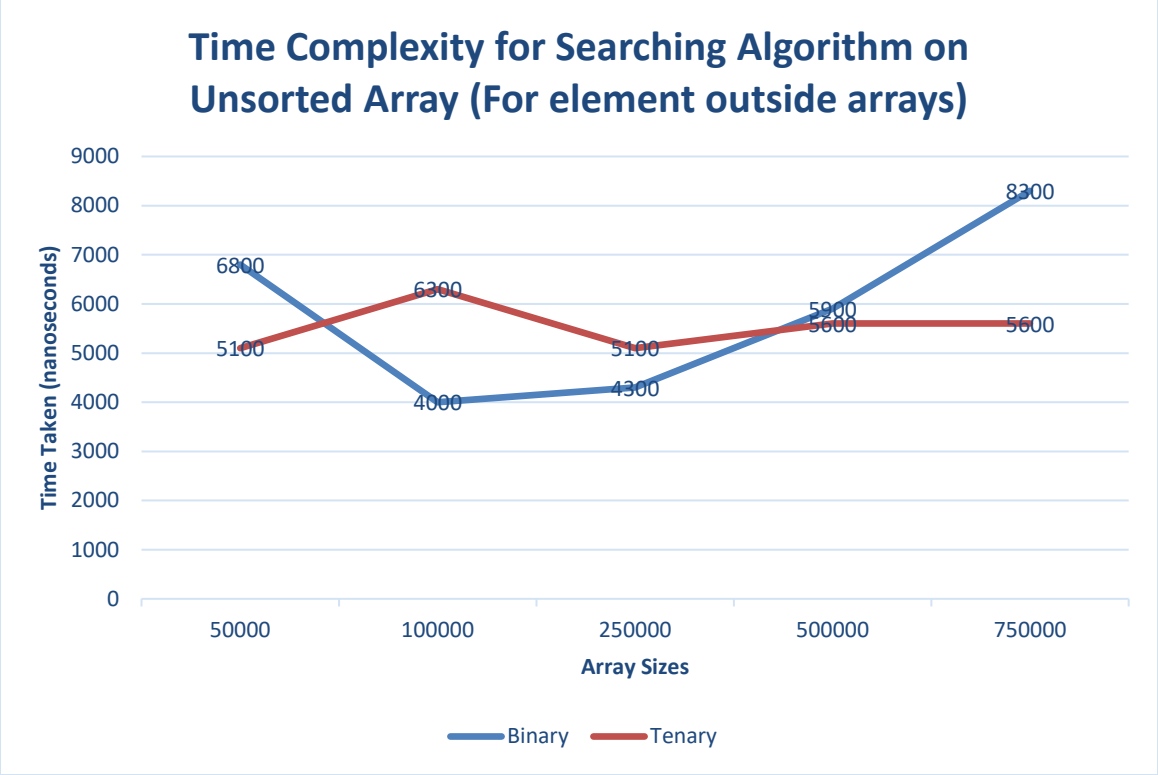
a. Runtime (in nanoseconds) of Searching Algorithms for element within the array

| ARRAY SIZES | 50000 | 100000 | 250000 | 500000 | 750000 |
|---|---|---|---|---|---|
| BINARY | 1047900 | 6500 | 2700 | 7300 | 5100 |
| TENARY | 1734700 | 5500 | 8300 | 8300 | 7600 |




b. Runtime (in nanoseconds) of Searching Algorithms for element outside the array

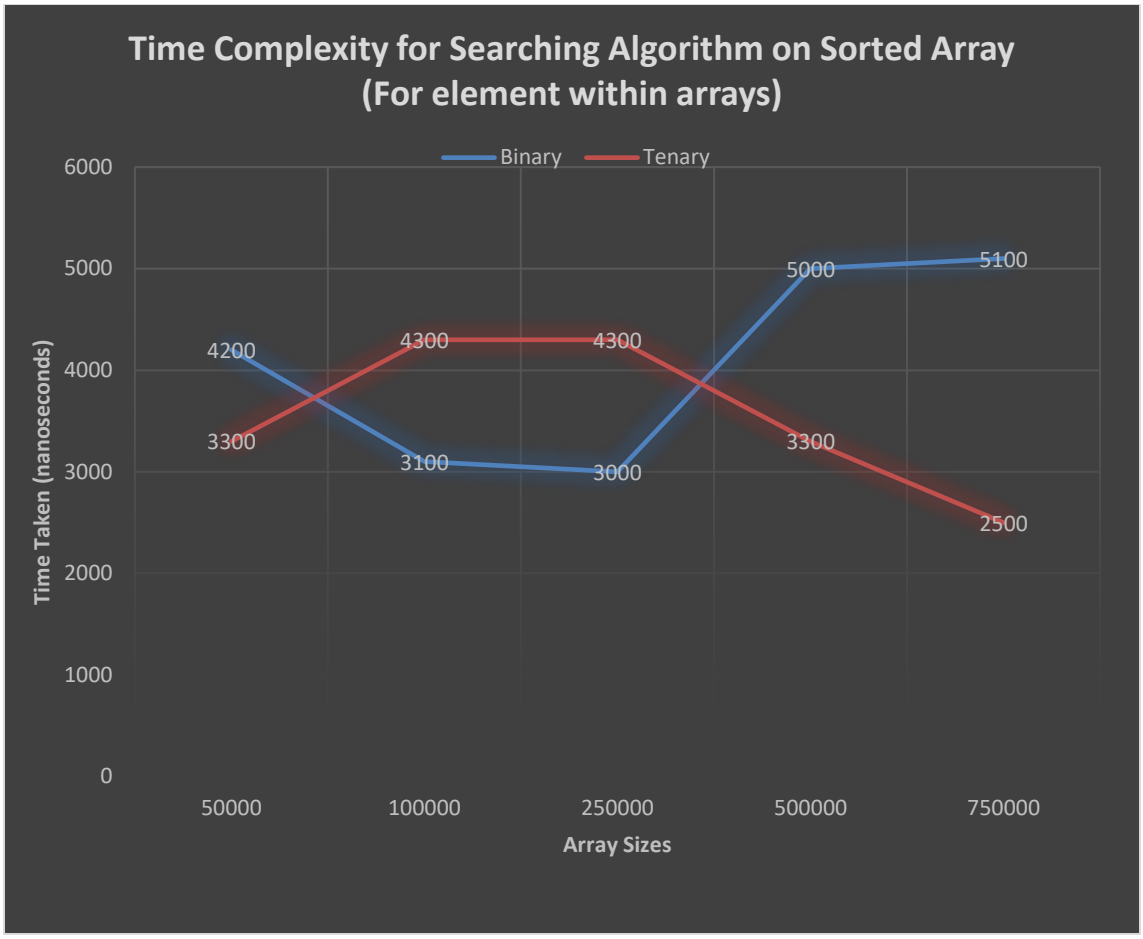
| ARRAY SIZES | 50000 | 100000 | 250000 | 500000 | 750000 |
|---|---|---|---|---|---|
| BINARY | 6800 | 4000 | 4300 | 5900 | 8300 |
| TENARY | 5100 | 6300 | 5100 | 5600 | 5600 |




## 2. For Sorted Arrays

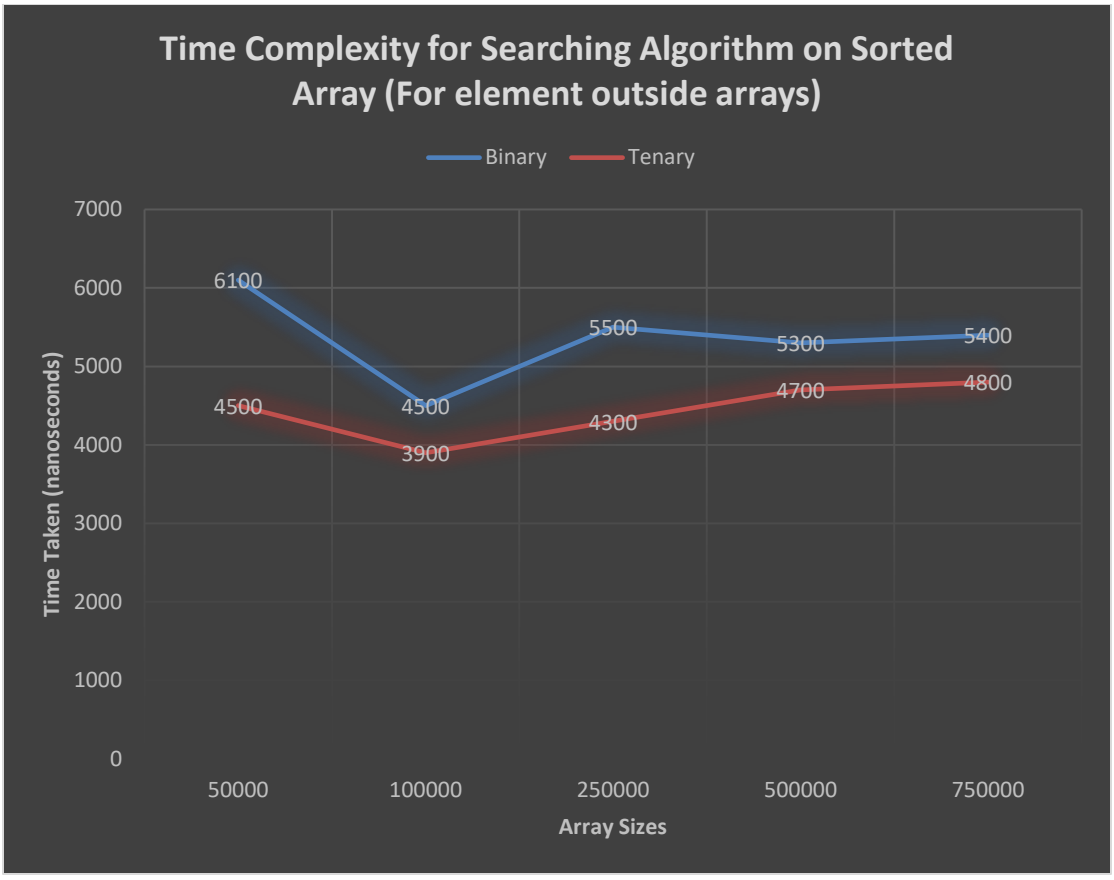
a. Runtime (in nanoseconds) of Searching Algorithms for element within the array

| ARRAY SIZES | 50000 | 100000 | 250000 | 500000 | 750000 |
|---|---|---|---|---|---|
| BINARY | 4200 | 3100 | 3000 | 5000 | 5100 |
| TENARY | 3300 | 4300 | 4300 | 3300 | 2500 |




b. Runtime (in nanoseconds) of Searching Algorithms for element outside the array

| ARRAY SIZES | 50000 | 100000 | 250000 | 500000 | 750000 |
|---|---|---|---|---|---|
| BINARY | 6100 | 4500 | 5500 | 5300 | 5400 |
| TENARY | 4500 | 3900 | 4300 | 4700 | 4800 |




OBSERVATIONS

- For the Unsorted Array, according to the output of the code pasted above, it is observed that both the Binary and Tenary Searching Algorithm reported "***Not Found***" for an element that is indeed present in the arrays. Its only on few occasion that both algorithms were able report a containing element was found and also, sometimes Binary might report '*found*' and Tenary "*not found*" and vice versa.
- The search time of both algorithms increases (by a very small fraction) as the array sizes increases for Sorted Array. Similar pattern in seen also in the Unsorted array.
- The search time of both Algorithms are relatively close for the Sorted Arrays

Algebraic Analysis of both the Binary and Tenary Searching Algorithms

To further analyze the algorithm in order to have a proper proof as to which algorithm is efficient I did further research online and found about the algebraic analysis.

It is believe that the time complexity of binary and ternary search is O($\log_2 n$) and O($\log_3 n$) respectively, evaluating these values.

For Binary $T_b(N) = C_b * \log_2 N$

For Tenary $T_t(N) = C_t * \log_3 N$

Where $C_b$ and $C_t$ are Number of Comparison in each recursive calls in binary and tenary respectively

$$T_b(N) = C_b * \frac{\log_e N}{\log_e 2} = \frac{C_b}{\log_e 2} \times \log_e N = 1.4426 C_b \times \log_e N$$

$$T_t(N) = C_t * \frac{\log_e N}{\log_e 3} = \frac{C_t}{\log_e 3} \times \log_e N = 0.9102 C_t \times \log_e N$$

It can be seen from the algorithm that

The number of Comparison in each recursive calls in binary ($C_b$) = 2 and

The number of Comparison in each recursive calls in tenary ($C_t$) = 4 and

Substituting these values in the equations above, we have

$$T_b(N) = 1.4426 \times 2 \times \log_e N = 2.885 \log_e N$$

$$T_t(N) = 0.9102 \times 4 \times \log_e N = 3.6408 \log_e N$$

Comparing we can see that $T_b(N) < T_t(N)$

INFERENCE

- With the first observation above, Binary and Tenary Searching Algorithm are not suitable to be used on Unsorted arrays, for proper and correct implementation. Both algorithms should be used on SORTED ARRAYS.
- According to the algebraic analysis done above, its proved that, Binary search is somewhat faster than the Tenary search because binary search has a smaller number of comparison done in each recursive calls compared to Tenary. Although tenary will have a smaller number of recursive calls compared to binary but these does not significantly reduce the runtime of the algorithm as the number of comparison in each recursive calls is more.

  The can be assume that binary search is faster than tenary but the difference is not very significant judging by the values of the coefficient of $\log_e N$ in the evaluation above

Final Information for John:

It is better to use Binary Searching algorithm as you data size increases, there would be more time efficiency with Binary Search.