Searching algorithms : Linear VS Binary VS Final

Dhruba Das

Spring Fall 2023

Introduction: Problems to solve

There are in total 3 problems I had to solve in this assignment: a) Implementing the Binary Search algorithm. b) Testing the Binary Search algorithm. c) Creating and testing all 3 algorithms including the Final version.

Binary Search

This was actually the easiest part of the assignment. I had learned about this algorithm in a previous course and could easily refresh my memory by looking back. Of course, I had to also read through the guidelines to see if any modifications were to be made. Below is the code snippet:

```
public static Boolean search(int[] array, int key)
    {
        int first = 0;
        int last = array.length-1;
        while (true)
        {
            int index = (last+first)/2;

            if (key == array[index])
                return true;
            if (key > array[index])
                first = index+1;
            else if (key < array[index])
                last = index-1;
            if (first > last)
                return false;
        }

    }
```

Testing the binary search algorithm

This bit was also quite easy as I could easily test it using the benchmark program which was provided. Here, we only benchmark the binary search against the linear search algorithm. Below are the values I obtained from the benchmarking program:

Size of array (n)	**Linear (ns)**	**Binary (ns)**
100	16	23
200	32	27
300	46	30
400	63	32
500	75	34
600	92	35
700	106	37
800	119	37
900	138	38
1000	153	38
1100	164	39
1200	181	40
1300	196	41
1400	212	42
1500	230	42
1600	240	42

Table 1: Binary and linear; same size for each ; 1000 runs each; runtime for 1 run in nanoseconds;

I tested the binary search algorithm for an array with a size of 1 million, and I got a runtime of 106 ns.

Creating and testing the final version

The final version

I hadn't seen this algorithm before, but some of my peers have worked with this algorithm before. It's known as the "smart search" algorithm. It was not difficult to implement, as I only had to keep track of items in two arrays, compare them, move on and at the end, return a counter which gave us the number of duplicates between the two arrays. Below is the code snippet for the algorithm:

```
public static int search(int[] array1, int[] array2)
    {
```

```
        int counter = 0;
        int i = 0;
        int j = 0;

        while (i<array1.length-1 && j<array2.length-1)
        {
            if (array1[i] > array2[j])
                j++;
            if (array1[i] < array2[j])
                i++;
            else
            {
                counter++;
                i++;
                j++;
            }
        }
        return counter;
    }
```

Testing all three algorithms

Here, I only had to make minor changes to the given benchmark program (to add the in the final search algorithm). Below is the snippet for the code I added in:

```
 for (int i = 0; i < k; i++) {
long t0 = System.nanoTime();
FinalSearch.search(array1, array2);
long t1 = System.nanoTime();
double t = (t1 - t0);
if (t < min)
    min = t;
    }

    System.out.printf("%8.0f\n" , (min/loop));
}
    }
```

Below I have listed the values I obtained from the benchmarking program (NOTE: All values shown as 0 are values of less than 1 ns).

Above is the time taken for all three of the algorithms to search for duplicates in two given sorted arrays.

Size of array (n)	**Linear (ns)**	**Binary (ns)**	**Final (ns)**
100	0	0	0
200	1	0	0
300	1	0	0
400	2	0	0
500	4	0	0
600	5	1	0
700	8	1	0
800	10	1	0
900	13	1	0
1000	15	1	0
1100	18	1	0
1200	22	1	0
1300	26	2	0
1400	30	2	0
1500	35	3	0
1600	39	3	0

Table 2: Binary, linear and final; same size for each ; 1000 runs each; runtime for 1 run in nanoseconds;

Conclusion

As it can be seen from the above data, Binary searching is far more efficient than Linear searching as we know approximately where to look, and smart search is even more so than Binary (when it comes to looking for duplicates or searching through an array in general). As of right now, sorting seems more cost efficient to me when it comes to searching through arrays/data.