# Implementing Linear and Binary Search Algorithms
# Homework #1

**By Logan Miles**

## 1. Objectives

The goal of this assignment was to implement linear and recursive binary search algorithms in order to evaluate their respective performance and time-complexities utilizing a high-level programming language.

## 2. Program Design

The assignment required us to implement linear and binary search algorithms by generating random arrays of integers from 2^4 to 2^20 then comparing them to a list of keys from a text file. To accomplish this, the following steps were integrated into the program:

a) Read the text file and add the keys into an array.

b) Iterate from n=4 to n=20 and create and an array for each power of 2^n.

c) Iterate through each array and add integers to them in numerical order.

d) Implement a recursive binary search algorithm.

e) Randomize the indices of the integers in the array.

f) Implement a linear search algorithm.

g) Record the system time in nanoseconds before and after the program has run to calculate the runtime.

The program first initializes an array of size 1000, as there are 1000 integer keys. It then creates file reader and scanner objects to read these keys from the text file and store them in to an array as strings which are then converted to integers in a for loop. The program utilizes a nested for loops to iterate through the different powers of 2^n, then iterate

through then iterate through the different keys and call the linear and binary search method on each key. Typically, the binary search algorithm would need the array to be sorted but since the array was generated in numerical order and is therefore already sorted, the program instead swaps the indexes of the integers in the arrays every iteration before the linear search. The system time is recorded using System.nanoTime(), which returns the system time in nanoseconds. Taking this before and after the search algorithm is called and then subtracting to get the difference results in the time the code ran.

**linearSearch()**

The linear search method is a simple linear search algorithm that iterates over every integer in the arrays generated by the main method within each iteration for each key. An if statement compares the integer at the current index to that of the key, returning the index if it matches the key.

**binarySearch()**

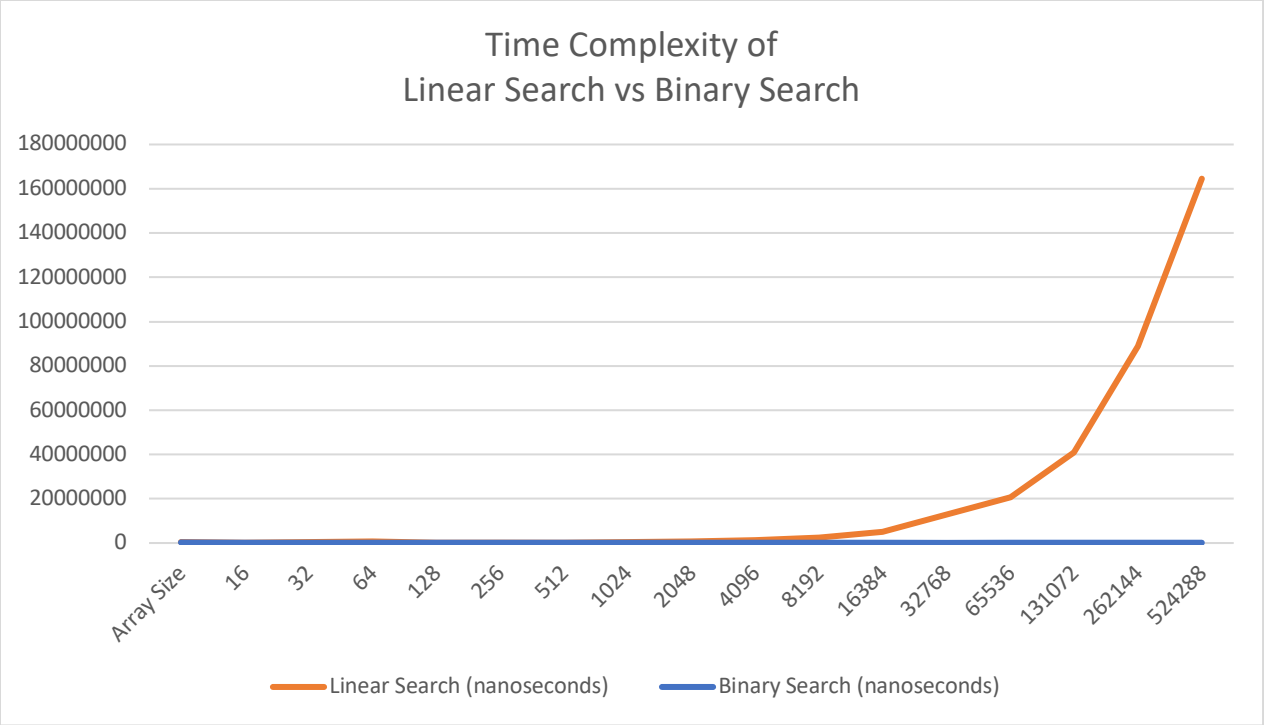
The binary search method is also a recursive search algorithm, but instead of iterating over every integer, a while loop checks the middle index to see if it matches the key. The middle index is calculated by averaging the starting index and the ending index. An if statement checks if the value of the middle index matches that of the key, and if it is it the program returns that index. If the key is lesser than the middle index value, the program reassigns the end index to the index before that of the mid index, essentially eliminating the other half of the array. This is because if the array is sorted, the target value could not be in that half. The same goes for if the key is greater than middle index, only it is the first half that is eliminated.

## 3. Testing

In order to test the program, there are print statements with every iteration detailing the search time and data set size for both search algorithms. This is for the purpose of easily seeing the difference between the two algorithms and comparing the two as the assignment requires.

Below is a table detailing the result of a test run:

| Array Size | Linear Search (nanoseconds) | Binary Search (nanoseconds) |
|---|---|---|
| 16 | 312389 | 269692 |
| 32 | 244544 | 100614 |
| 64 | 342904 | 111404 |
| 128 | 614001 | 166240 |
| 256 | 197291 | 149910 |
| 512 | 162065 | 79602 |
| 1024 | 195163 | 107088 |
| 2048 | 358947 | 108392 |
| 4096 | 630609 | 110151 |
| 8192 | 1189760 | 111928 |
| 16384 | 2371013 | 115920 |
| 32768 | 5031612 | 120121 |
| 65536 | 12778266 | 65536 |
| 131072 | 20522530 | 129764 |
| 262144 | 40847096 | 115633 |
| 524288 | 88743421 | 103579 |
| 1048576 | 164547576 | 97093 |




## 4. Analysis and Conclusions

Judging by the time complexity of the two search algorithms, binary search seems much more efficient. The time taken for the linear search followed its expected curved based on its time complexity, as did the binary search, which are O(n) and O(log(n)) respectively. It is worth noting, however, that this difference was only apparent in larger data sets. This means that if working with smaller data sets linear search could be easier to implement, but when working with larger ones, binary search will be the better option. This program specifically could be improved in the future by making it more abstract and open to the implementation of different data sets or search algorithms.

## 5. References

https://www.w3schools.com/java/java_files_create.asp
https://www.educative.io/answers/how-to-generate-random-numbers-in-java
https://stackoverflow.com/questions/26948421/how-to-create-multiple-arrays-with-a-loop
https://www.digitalocean.com/community/tutorials/shuffle-array-java