1. Running Environment
   1. Computer: MacBook Pro 2020
   2. Memory: 512gb
   3. Version: 18.0.1
   4. Processor: Apple M1 Chip
   5. OS: MacOS Ventura 13.2.1
2. Description
   1. For this experiment I coded three searching algorithms (linear, binary iterative, binary recursive), each of them goes through the randomly generated array of one million elements per epoch (1\_000\_000) amounts of time. We add the time it took for each of them and took the individual averages for each epoch.
   2. I reran the program 5 times to get a sense of performances.

Linear – A. Sizes (100\_000, 1\_000\_000, 10\_000\_000)

1\_000 0.000029 milliseconds

10\_000 0.000030 milliseconds

100\_000 0.000033 milliseconds

Binary Iterative – A. Sizes (100\_000, 1\_000\_000, 10\_000\_000)

1\_000 0.000032 milliseconds

10\_000 0.000036 milliseconds

100\_000 0.000042 milliseconds

Binary Recursive – A. Sizes (100\_000, 1\_000\_000, 10\_000\_000)

1\_000 0.000038 milliseconds

10\_000 0.000041 milliseconds

100\_000 0.000055 milliseconds

Array sort method

1\_000 0.000000 milliseconds

10\_000 1.000000 milliseconds

100\_000 6.000000 milliseconds

1. Results
2. Factors

Factors that may contribute to performance.

* Code
* Computer
* Resources

1. Big-O time complexity
   1. Linear Search -> O(n)
   2. Binary Search -> O(log2(n))
2. Big-O time complexity estimation (sorting array)
   1. It seems to look like a linear function, thus O(n)
3. Runtime relations to big O
   1. The plots relate to complexity in terms of the linearity of the function, whether is linear, logarithmic, exponential, quadratic and so on.