Abstract

With the goal of further understanding arrays in Python compared to those of other languages, this experiment will focus on the performance of the sorting algorithm binary search with varying input sizes. The test was inconclusive due to the predictable pattern in the array running binary search and an input size that was not large enough.

Experiments

*Note: for easy comparison the same tests were done in languages: Python, Java, JavaScript, and C++

- Package used
    - `time` package used in order to measure how long the specific test took to run in nanoseconds
- One dimensional array
    - Array input sizes:
        - 1,875,000
        - 3,750,000
        - 7,500,000
        - 15,000,000
        - 30,000,000
    - Binary Search value
        - 500
    - Array content
        - Contains only index+1
            - Arr[0] = 1
            - Arr[127] = 128 ...etc.

A one dimensional array will be created and populated based on the input size given. The time it took for each array to complete the binary search on the value 500 is recorded each time as data. The time spent waiting is due to the creation of an array with the specified size and data.

Results

Array input size	Time (nanoseconds)
1,875,000	0

3,750,000	0
7,500,000	0
15,000,000	0
30,000,000	0

Despite varying input sizes, the runs for binary search were a bit different from what was originally expected. It seems that bigger input may have been needed for the test.

Conclusion

Despite going up to 30 million the runtime for binary search continued to have an average runtime of zero. Bigger input sizes may have been needed. However, the creation of bigger arrays may cause a memory error, additional tests may be done in order to further investigate. The binary search algorithm was checked against various code sources numerous times, however it could still be possible that due to a personal bias I failed to catch an error in the implementation.