ADS1 - BUCKET SORT

JAKUB KUKA

Setup

The test environment was set up in Java with several custom classes to manage array generation, sorting, runtime measurement, and graph plotting. *BucketSort* was the class responsible for the sorting algorithm. *EvenlyDistributedArrayGenerator* and *UnevenlyDistributedArrayGenerator* created test cases. The *SortAndPrintRuntime* class recorded the runtime, and *ChartHelper* visualized the results.

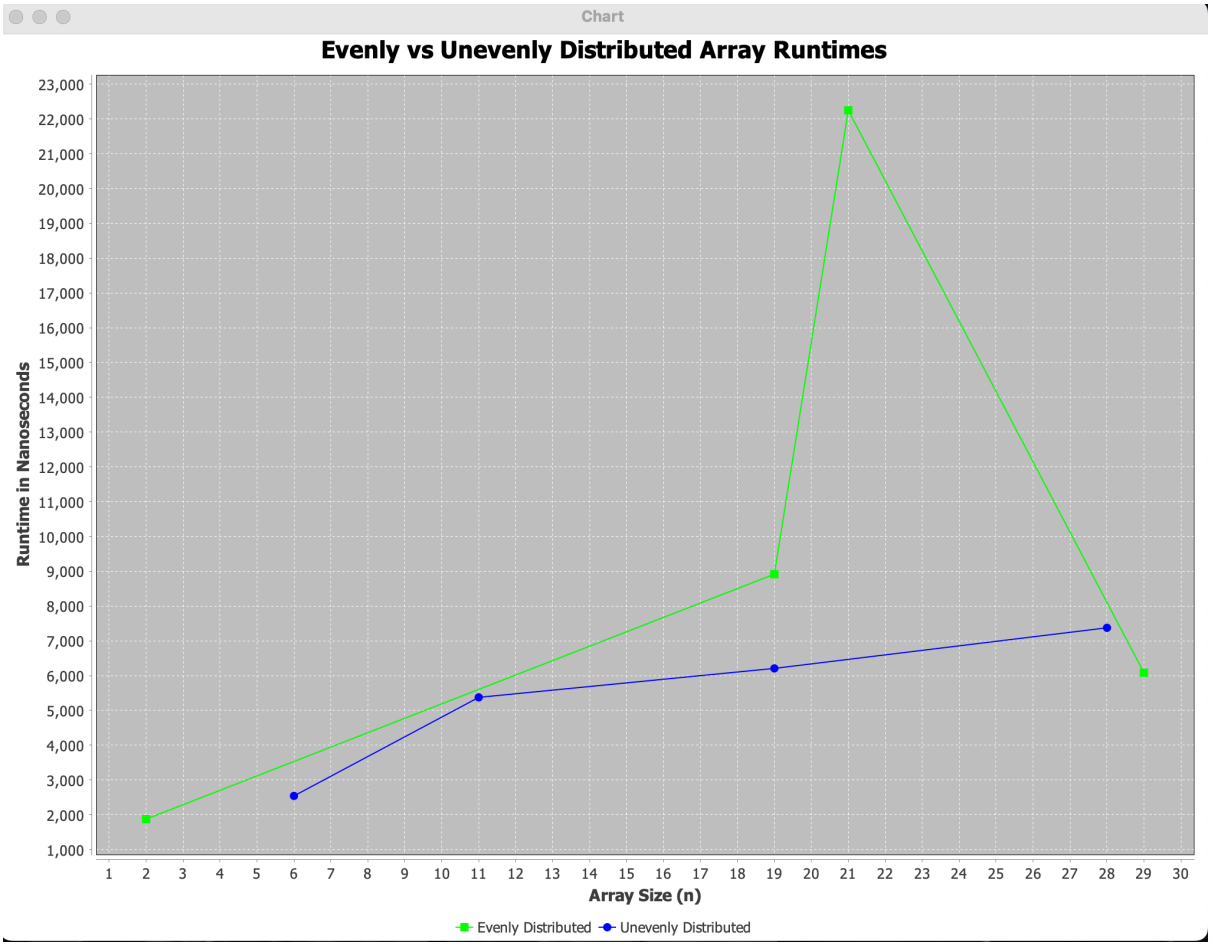
Test cases

Test cases were generated using two distinct methods for creating integer arrays:

- *EvenlyDistributedArrayGenerator* produced arrays where integers were incremented evenly and small random values were added to each element.
- *UnevenlyDistributedArrayGenerator* created arrays where every integer was multiplied by a random value, leading to a less consistent distribution.

Each generator created multiple arrays with varying sizes to ensure a comprehensive set of test cases.

Recorded runtimes



- The graph shows two lines for both evenly and unevenly distributed arrays

Evenly distributed arrays (green) - the runtime seems stable or increases slightly as the array size grows. There is a spike that is caused by external factors

Unevenly distributed arrays (blue) - appear to have different pattern, with runtime increasing gradually as the array size increases

Results

Runtime was recorded by capturing the system's nanotime before and after the sort method invocation in the *BucketSort* class. The *SortAndPrintRuntime* class differentiated the runtimes by using a flag indicating the array type. Two separate arrays, evenRuntimes and unevenRuntimes, stored the recorded times. This class provided methods to retrieve the runtime arrays for analysis.

Analysis of Results:

Upon running the experiment, the Main class invoked the sorting and runtime recording methods for both evenly and unevenly distributed arrays. It then called the *ChartHelper* to visualize the results in a scatter plot with lines connecting the points.

Observed Trends:

The generated graph, "Evenly vs Unevenly Distributed Array Runtimes," depicted the following trends:

- Evenly Distributed Arrays: Displayed a linear increase in runtime with the array size
- Unevenly Distributed Arrays: Demonstrated more variability in runtime, with some cases showing significant spikes. This is likely due to the distribution of data causing inefficient sorting, as Bucket Sort has a dependency on the input distribution.