This module enables us to test our knowledge on various aspects involving I/O operations and having a better understanding of computation and memory mechanisms by using efficient data structures and java programming.

1. Data Cleaning

Cleaning the data before performing the required tasks is always an integral part for performing operations efficiently. The code for cleaning data is in `CleaningData.java`. It accesses datasets in the same directory, using `replaceAll` to refine data as required. Output is stored with the same folder structure.

Commands

javac CleaningData.java

java CleaningData

2. Word Count

This problem statement enables us to push our minds in the direction of using an efficient data structure that can provide a lower time complexity to have linear write operations, so we have chosen Hashmap over here.

The code for word count is in `Unique.java`. It reads input from previous execution's output, counting words using a HashMap for efficient insertion/updating and retrieval.

Commands

javac UniqueWords.java

java UniqueWords

3. Sort Words

The code for sorting of the words obtained from the previous code should be written in such a way that it should have the maximum utilization of the throughput of the code and give a better understanding of I/O intensive operations and how we should co-relate all the factors like computation mechanisms and memory operations to make the data retrieval quicker. The code extracts word counts from input files, sorts using a TreeMap for efficient key-value pair storage. Output follows the same folder structure.

Commands

javac WordSort.java

java SortWordSort

Size of Datasets

The size of datasets can be calculated using `DatasetSize.java`, helping evaluate throughput and plot graphs.

Commands

javac DatasetSize.java

java DatasetSize