



**COMP 6521: Advanced Database Technology and Applications**

**Dept. of Computer Science and Software Engineering**

**Lab Assignment 1**

**Winter 2019-2020**

**Group 01**

**Kishan Bhimani (40081118)**

**Rajan Shah (40081117)**

**Tirth Shah (40115911)**

## 1. Explanation and Implementation of the Algorithm:

### **Main.java**

This class includes the main function of the program which calls various phases of our implementation during its runtime.

### **PhaseOne.java**

This class performs phase one of our proposed solution. In phase one, we have merged two given input files before passing it to phase two for sorting.

### **PhaseTwo.java**

The class performs phase two of our solution. In which, we read the merged file generated by phase one and sort based on given memory constraints. We have used a comparator based on employeeid and date to perform this sorting.

Initially, blocks will be sorted individually using the comparator.

In the sorting process, the primary comparison is based on the employeeid and if ids are the same then it compares based on the date and sorts based on that. Tuple with the latest date comes first in case of duplicate employeeid.

After sorting the blocks individually, we saved them as file objects.

Subsequently, we have created custom file handler objects for those files while bringing them back into memory for merging.

We have used the priority queue for merging all the blocks using their handler class objects.

After completing the merging process, we have all tuples sorted according to id and date. But still, we have to remove duplicate tuples from it.

### **Handler.java**

It is a wrapper class to all the blocks saved as file objects. It includes various methods like getCurrentLine, removeLine, isEmpty, etc. These methods are useful during the merge process of these blocks.

## DuplicateHandler.java

This file removes duplicate tuples from our output file and stores the latest distinct tuples in our final output file.

## 2. Group member contribution:

Everyone was equally involved while designing the solution,

It took a lot of time to choose an approach because there were multiple ways to get it done, but due to given memory constraint, we had to narrow down the options and select the generic best one.

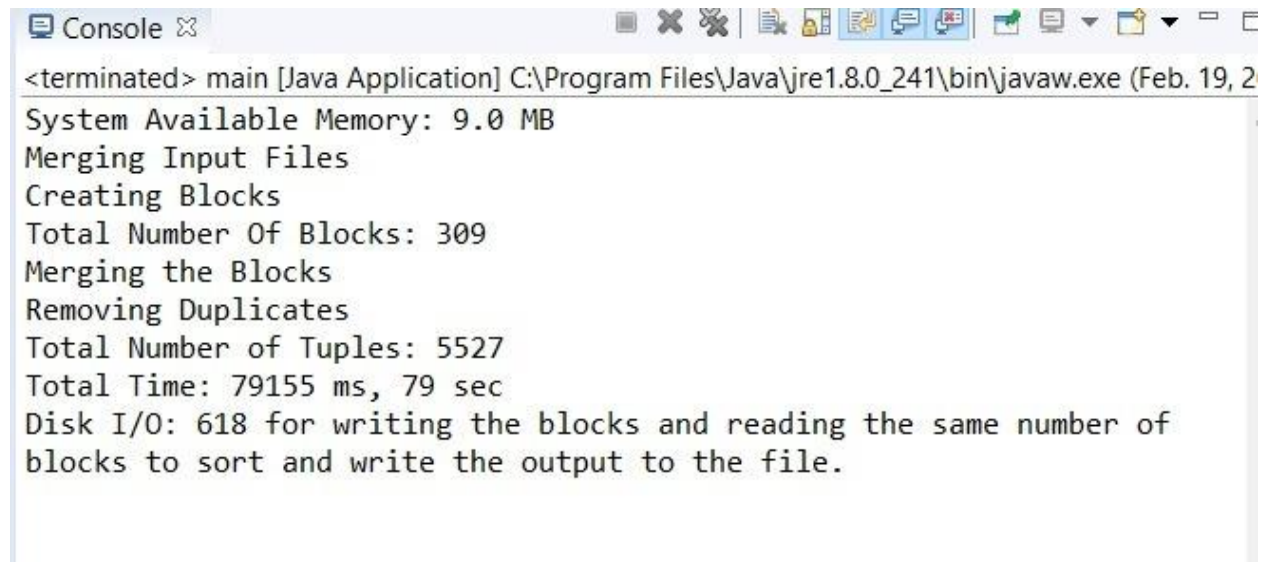
While testing the implementation, everyone came up with their own set of datasets to analyze the efficiency.

## 3. Output

### (i) With Block size of ~512 Kilobytes:

Our proposed solution for the two files, input1.txt (~1000000 tuples) and input2.txt (~500000 tuples), with available memory of 9 MB, generated 309 blocks and completed the sorting process in 79 seconds. And, with the available memory of 20 MB, our algorithm took 70 seconds to successfully complete the sorting process.

[bitbucket link to above implementation](#) (Private Repository)



```
<terminated> main [Java Application] C:\Program Files\Java\jre1.8.0_241\bin\javaw.exe (Feb. 19, 2018)
System Available Memory: 9.0 MB
Merging Input Files
Creating Blocks
Total Number Of Blocks: 309
Merging the Blocks
Removing Duplicates
Total Number of Tuples: 5527
Total Time: 79155 ms, 79 sec
Disk I/O: 618 for writing the blocks and reading the same number of
blocks to sort and write the output to the file.
```

```
Console
<terminated> main [Java Application] C:\Program Files\Java\jre1.8.0_241\bin\javaw.exe (Feb. 19, 202
System Available Memory: 20.0 MB
Merging Input Files
Creating Blocks
Total Number Of Blocks: 309
Merging the Blocks
Removing Duplicates
Total Number of Tuples: 5527
Total Time: 70505 ms, 70 sec
Disk I/O: 618 for writing the blocks and reading the same number of
blocks to sort and write the output to the file.
```

**(ii) With Block size of 4 Kilobytes:**

Our proposed solution for the two files, input1.txt (474192 tuples) and input2.txt (1106448 tuples), with available memory of 9 MB, generated 39515 blocks and completed the sorting process in 1244 seconds. And, with the available memory of 20 MB, our algorithm took 418 seconds to complete the sorting process.

```
Total Number Of Blocks: 39516
bigger blocks
Removing Duplicates
Total Number of Tuples: 5527
Total Time: 1244849 ms, 1244 sec
Disk I/O: 79032 for writing the blocks and reading the same number of
blocks to sort and write the output to the file.
```

```
Total Number of Tuples: 5527
Total Time: 418273 ms, 418 sec
Disk I/O: 79032 for writing the blocks and reading the same number of
blocks to sort and write the output to the file.
```