COMP - 6521

Advanced Database Technology and Applications

Project Report

on

TPMMS and Bag Difference

Professor

Dr. Nematollaah Shiri

Team Members

Student Name	Student Id	Email Id
Sagar Vetal	40071979	s_vetal@encs.concordia.ca
Himanshu Kohli	40070839	h_kohli@encs.concordia.ca
Nayana Raj Cheluvaraju	40071318	n_cheluv@encs.concordia.ca
Balpreet Singh	40070817	s_balpre@encs.concordia.ca

Contents:

1. How to run the program :

- Place the data sets in the 'inputfiles' folder and set up the eclipse environment.
- Setup main memory size 5MB or 10MB.
- Run the program with 'TpmmsDataController.java'
- The program will read the input files based on memory size, sort them into sublists, merge that sublists into final sorted relation and perform bag difference operation on the sorted relations.
- The sub lists can be seen in the 'sublists' folder.
- The output (i.e. sorted relations named T1_sorted.txt, T2_sorted.txt and bag difference named Bag_Difference.txt) will be generated in the 'outputfiles' folder.

2. Program Description:

The program takes the data sets T1 and T2 one after the other and processes through TPMMS phase 1 and phase 2. The sub-lists created in phase 1 can be seen in the 'sublists' folder and the complete sorted output can be seen in 'outputfiles' folder as well as the the the data set T1 after bag difference operation.

The calculations for Total numbers of blocks,number of runs in phase 1 and number of blocks main memory can hold are shown below:
As we are holding data in bytes thought out Program, calculation is done using same,
- Size of Input data T1 = 121,247,672 bytes.
- Size of Input data T2 = 120,974,467 bytes.
- Memory size= 5MB (50% of given memory size for datasets and remaining 50% for program execution. i,e 2.5MB each).
- Memory Size=10MB (50% of given memory size for datasets and remaining 50% for program execution. i,e 5MB each).

- Therefore, Below calculation is showed for 2.5MB and 5MB

- For 5MB available memory size is 2.5MB,
  - No of Blocks main memory can hold
    = (memory size / block size)
    = (2.5MB / 4KB)
    = 625 Blocks.

  - No of Runs
    = (T1 size / available memory size)
    = (121247672 / 2.5*10^6)
    = 49 runs

- For 10MB available memory size is 5MB,
  - No of Blocks main memory can hold
    = (memory size / block size)
    = (5MB / 4KB)
    = 1250 Blocks.

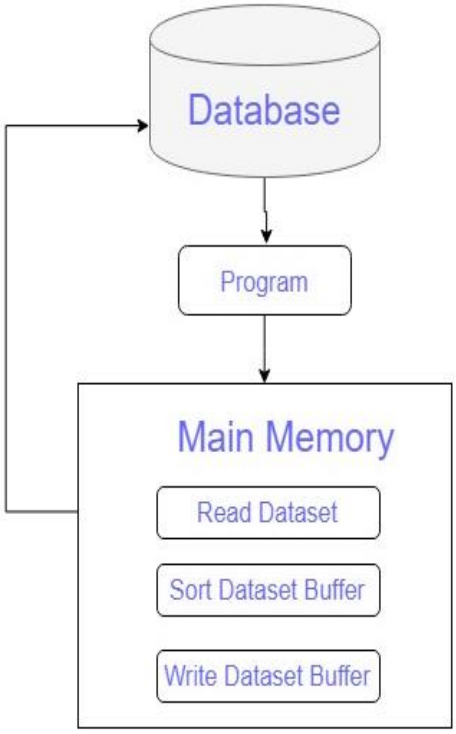
- No of Runs
  = (T1 size / available memory size)
  = (121247672 / 5*10^6)
  = 25 runs

➢ Total number of blocks required for T1 and T2
For T1 = (T1 size/block size) = (121247672/4K) = 30312 Blocks.
For T2 = (T2 size/block size) = (120974467/4K) = 30244 Blocks.

The program calculates the following which are displayed in console:

- Main Memory Size.
- Total tuples in a relation.
- Total Block Write for sorting of each relation.
- Total Block Read for sorting of each relation
- Total DISK I/O for sorting of both relations.
- Total Time taken for sorting of both relations.
- Total Blocks Read for Bag Difference.
- Total Time taken for Bag Difference.
- Total Execution Time for complete process.

3. Architecture Diagram:



Fiig. 1: Phase 1 - Sort

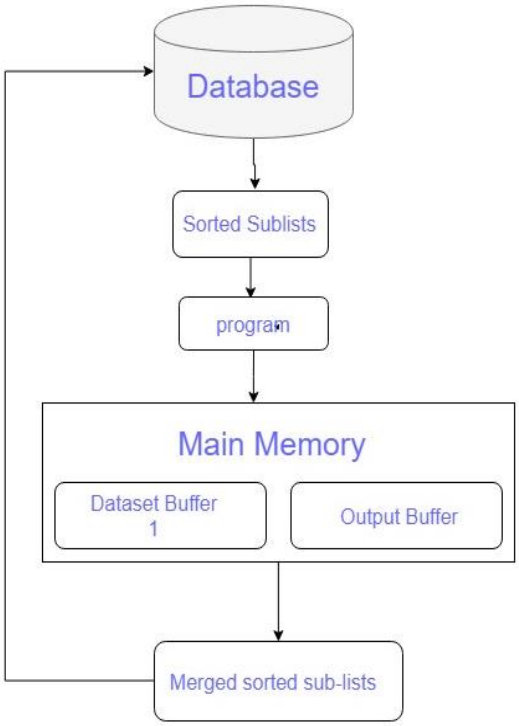


Fig. 2: Phase 2 - Merge

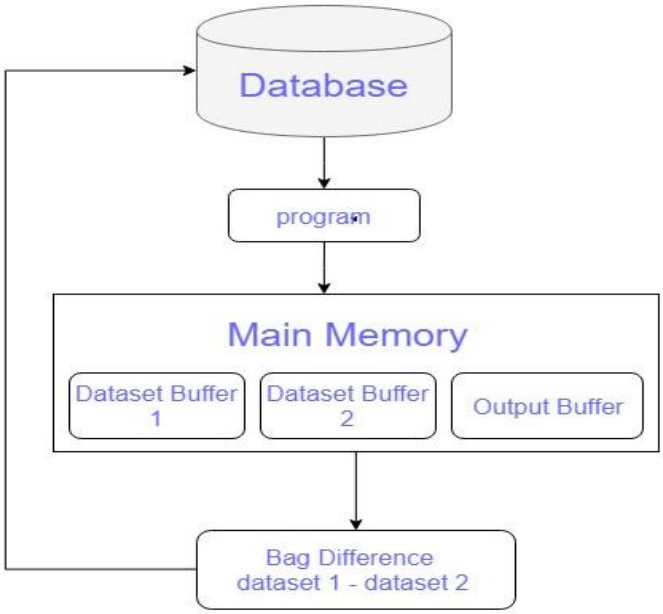


Fig. 3: Phase 3 - Bag Difference

4. Algorithms:

Here, We are allocating 50% of memory space for data and remaining 50% for program execution. So now consider we have 50% main memory available.

PHASE - 1 (TPMMS Sort):

1. Start.
2. Create input buffer and allocate available main memory space to read tuples from relation.
3. Calculate total number of main memory fills required to sort relation.
4. Load the input buffer.
5. Calculate total blocks read.
6. Apply Quick sort to sort the tuples within input buffer.
7. Write input buffer to sublist file.
8. Calculate total blocks written.
9. Repeat above steps from 4 to 8 for calculated number of memory fills.
10. End.

PHASE - 2 (TPMMS Sort):

1. Start
2. Allocate 70% of available main memory for input buffers of sublists and 30% to output buffer.
3. Create array of input buffers for sublists and divide that 70% of available main memory space among sublist buffers equally.
4. Create output buffer with 30% of available main memory space.
5. Load the input sublist buffers and get array length.
6. Read first tuple from all the sublist buffers and find the smallest one.
7. Swap that buffer having smallest tuple with the 0th position buffer.
8. Popout smallest tuple from 0th position buffer and put into output buffer, and check refill is required for that buffer. If yes, refill the buffer.

9. If entire sublist has been read, put its buffer at end of the sublist buffer array and decrease array length by 1.
10. If output buffer is full, write the data onto the disk and clear Output buffer.
11. Repeat the process from step 6 to 10 until all sublists and their corresponding buffers are empty.
12. End.

PHASE-3 (Bag Difference):

1. Start.
2. Allocate 35% of available main memory space for dataset T1 buffer and dataset T2 buffer each and remaining 30% for output buffer.
3. Fill the T1 buffer and T2 buffer.
4. If T1.tuple > T2.tuple, then move to next tuple in T2 buffer.
5. Else if T1.tuple < T2.tuple, move that tuple from T1 buffer to output buffer and move to next tuple.
6. Else if T1.tuple = T2.tuple, then move to next tuple of both datasets T1 buffer and T2 buffer.
7. If buffer becomes empty, refill them.
8. If output buffer is filled, write the data into disk.
9. Repeat process until one of the following conditions is met.
   - If T1 dataset read is finished, transfer T1 buffer data to output buffer and write\\ output buffer to disk).
   - If T2 dataset read is finished, move all elements from T1 datasets to output buffer and write the output buffer to disk.
10. End.

5. Technical Details:

The program executes in three phases;
Phase-1 includes reading the data from input file and store sorted data into sublist.
Phase-2 includes fetching of records from sublist and merge the sorted list into output file.Repeat the Phase1 and Phase2 for both datafiles T1 and T2.
Phase-3 will give the bag difference of T1 and T2.

The above process is followed for 5MB and 10MB memory size,later the execution time for performing the whole operation which includes sorting and computing bag difference is compared.

6. Coding Standards used:

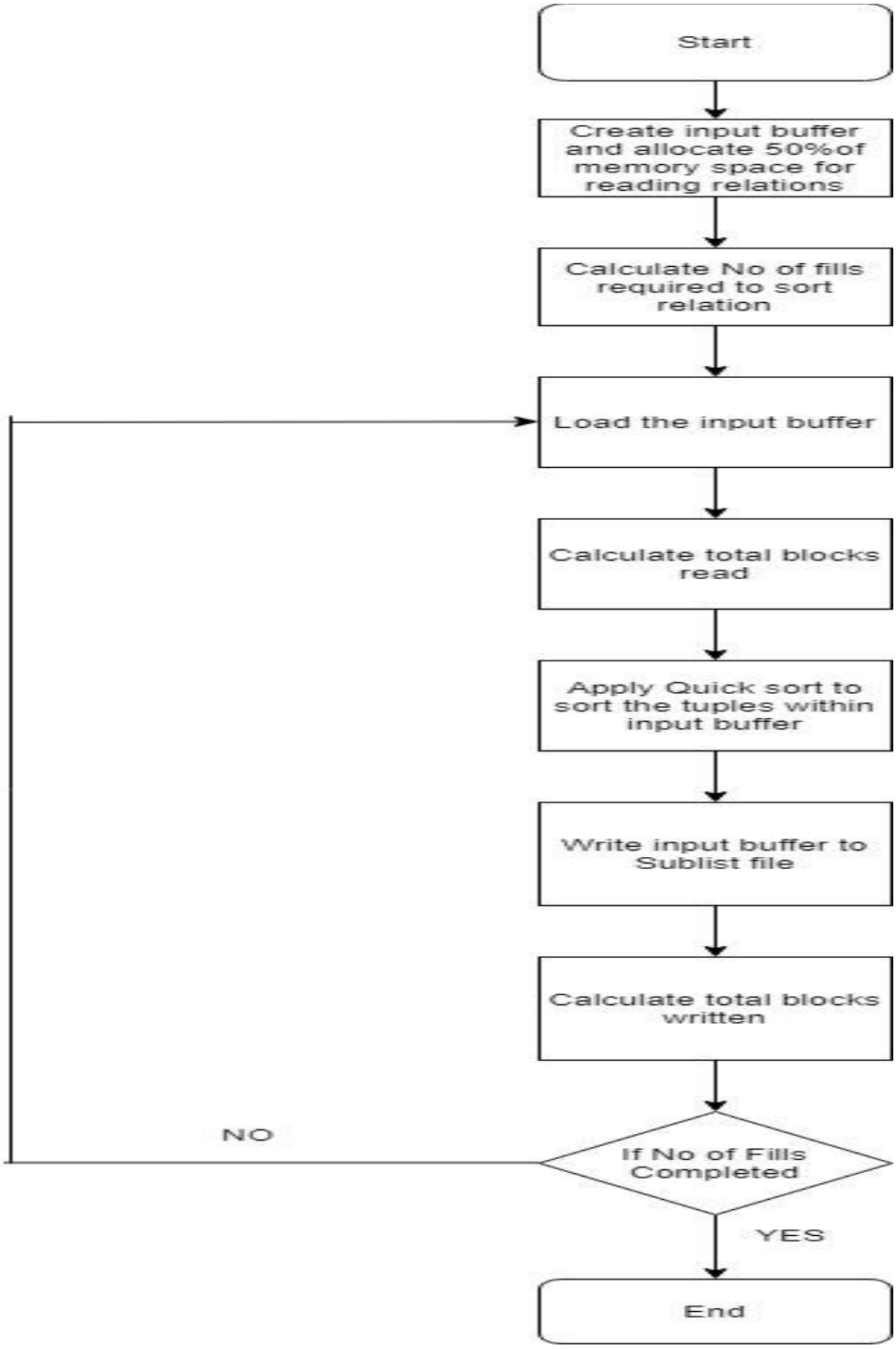
The most general coding conventions have been followed while developing the codes which are as follows,

- The name of the classes start with a upper case character.
  E.g.: TpmmsDataController.java
- Constants are named with upper case characters and include underscore between two words (if applicable).
- The name of the variables are descriptive and are written in lower case including a capital letter to separate between words.

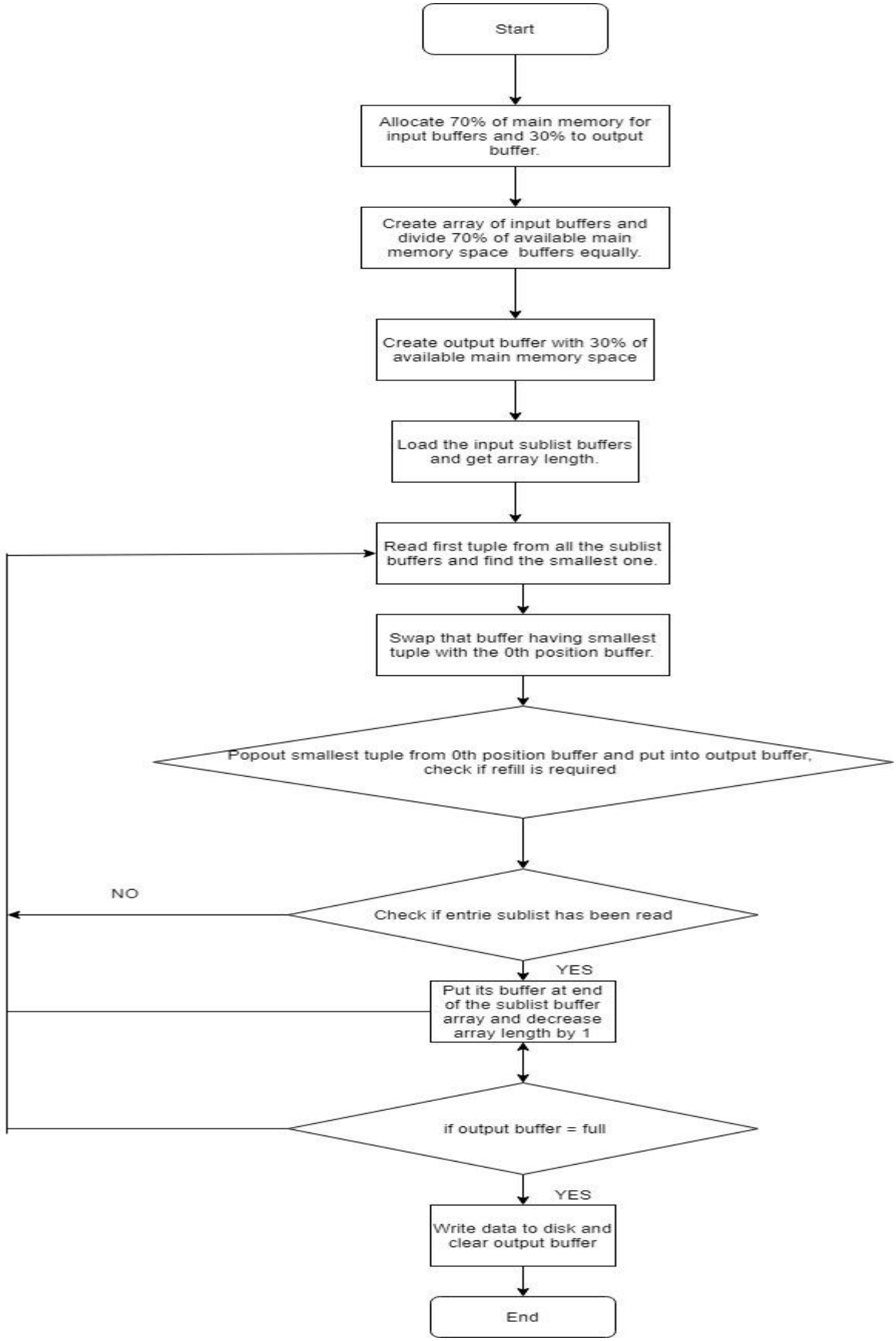
- The name of the methods start with a lower case character and use uppercase letters to separate words.

7. Flow Diagrams:

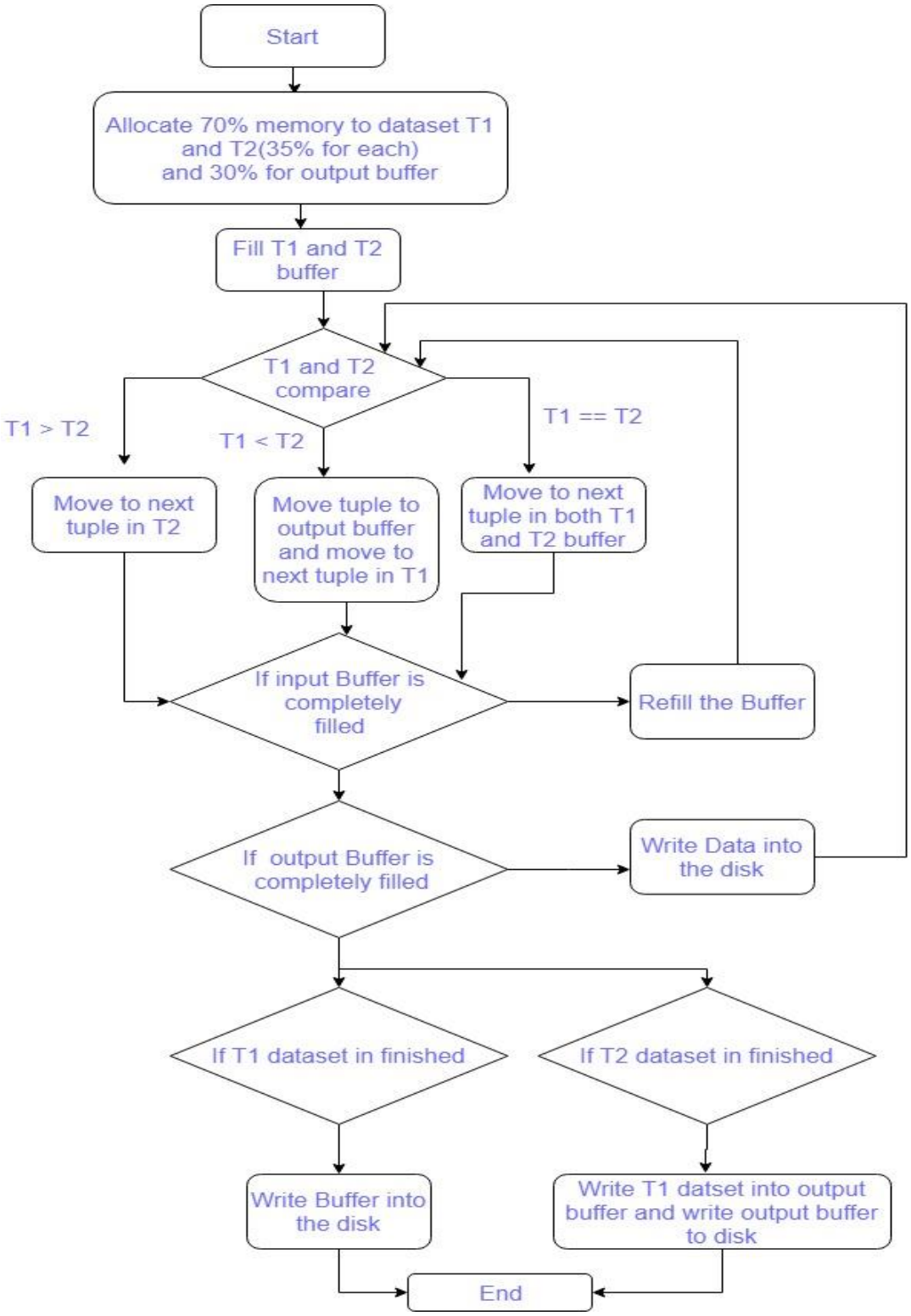
- **Phase 1:**



Phase 2:


Start
Allocate 70% of main memory for input buffers and 30% to output buffer.
Create array of input buffers and divide 70% of available main memory space buffers equally.
Create output buffer with 30% of available main memory space
Load the input sublist buffers and get array length.
Read first tuple from all the sublist buffers and find the smallest one.
Swap that buffer having smallest tuple with the 0th position buffer.
Popout smallest tuple from 0th position buffer and put into output buffer, check if refill is required
NO
Check if entrie sublist has been read
YES
Put its buffer at end of the sublist buffer array and decrease array length by 1
if output buffer = full
YES
Write data to disk and clear output buffer
End

Phase 3:


Start
Allocate 70% memory to dataset T1 and T2(35% for each) and 30% for output buffer
Fill T1 and T2 buffer
T1 and T2 compare
T1 > T2
T1 < T2
T1 == T2
Move to next tuple in T2
Move tuple to output buffer and move to next tuple in T1
Move to next tuple in both T1 and T2 buffer
If input Buffer is completely filled
Refill the Buffer
If output Buffer is completely filled
Write Data into the disk
If T1 dataset in finished
If T2 dataset in finished
Write Buffer into the disk
Write T1 datset into output buffer and write output buffer to disk
End

8. Description of the Classes:

- **Buffer.java:**
  file manages the buffers in the which are placed in the main memory and handles comparison operations and position management as data taken is in bytes form so every tuple must be regarded as 100 bytes.

- **TpmmsDataController.java:**
  This file is the main driver of the system and holds the data flow all the main processing is called in this system such as sorting, managing buffers and bag difference.

- **TpmmsService.java:**
  This file maintains the Tpmms data flow, it inputs the data and convert the data into blocks and manages buffers. It creates and sorts the sub-lists and in second phase it merges the sub-lists together into a collective sorted output

- **FileService.java**
  This file handles all the file management methods such as inputting, outputting, manipulation of data

- **FileConstants.java**
  This file holds all the constant values for the project such as input file names, output files names and bag difference filename and the directory locations.

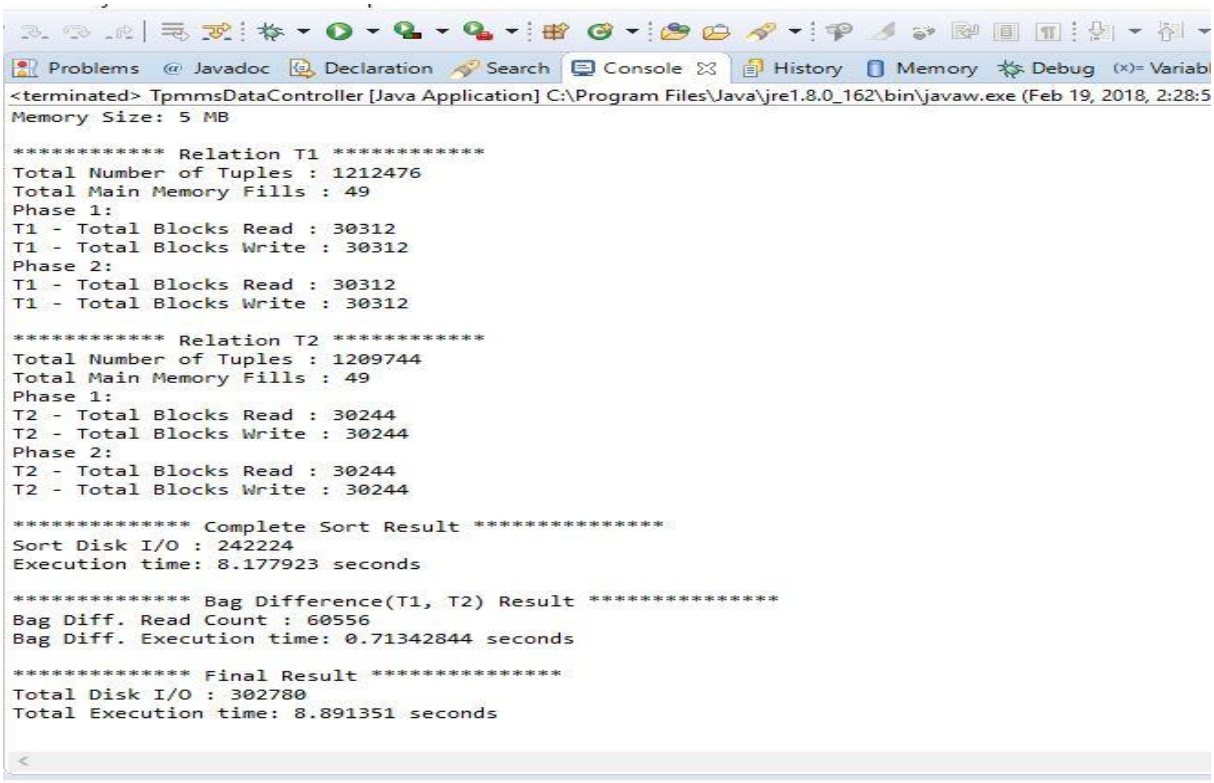
- **SizeConstatnts.java**
  This file holds the constants values for main memory sizes for 5 and 10MB and the block and tuple sizes.

- **QuickSort.java**
  This file holds the logic for quick sort which is adapted in the form of bytes so as the flow of data can be smooth.

- **CommonUtility.java**
  This file holds the data for values calculations such as total number of tuples, total number of tuple per block, total number of blocks, number of main memory blocks and fills and main memory size in MB’s.

9. Results:

For 5 MB:



For 10 MB: