COMP 9102 Data Management and Information Retrieval
Assignment 1
Implementation of query operators
Due Date: September 30th, 2022, 5:00pm

Summary

The goal of this assignment is the development and testing of algorithms for database operators (join, union, intersection, difference, and aggregation).
You are going to use synthetically produced data which are given to you at Moodle. These data are in the form of text files and each text file can be seen as a relational table with two fields: A (string) and B (integer). Each line in a file corresponds to a tuple, where the values of A and B are tab-separated. File R_sorted.tsv has the same tuples as R.tsv, but it is sorted. File S_sorted.tsv includes the tuples of a relation S, sorted. Attention: the tuples in a file are not necessarily unique, i.e., the same tuple may appear multiple times (bag semantics). Open the files and make sure that you understand their contents.

1. Merge-join (25%)

Write a program, which reads files R_sorted.tsv and S_sorted.tsv, and computes and writes to a file RjoinS.tsv the natural join result of R and S, assuming that their common attribute is their first field (A). For example, the join between tuple `('aaa', 710)` from R_sorted.tsv with tuple `('aaa', 151)` from S_sorted.tsv should produce output tuple `('aaa', 710, 151)`. The output tuples of the join should be written to file RjoinS.tsv separated with tabs; for example:

```
aaa  710  151
aaa  710  292
aaa  710  336
...
```

Attention: Your program should implement the merge-join algorithm described at the notes and not any variant of it. This means that:

1) The lines of files R_sorted.tsv and S_sorted.tsv should be read once only
2) You will not read the entire files in data structures in memory (e.g., arrays) before the join algorithm starts. For each line that you read from each file you should make sure that you find the results that correspond to that line.
3) The only thing that is allowed is to collect in an array (buffer) the lines of S that match with the current line *curline_r* of R. This way, if the next line of R has the same value on the join attribute as the previous one, we do not need to read again the lines from S that match with *curline_r*. At the end of your program, you should print the maximum size of this buffer (counted in number of lines).

Programs that violate the above guidelines will not receive full marks.

2. Union (15%)

Write a program that reads files R_sorted.tsv and S_sorted.tsv, and computes and writes to a file RunionS.tsv the union of R and S, assuming that the two relations have the same schema. Your program should read each line from R and S *just once* and compute the union at the same time, implementing a variant of the merge-join algorithm. In this case, you are not allowed to use a buffer and (of course) you are not allowed to load all data in memory before computing the union. Since R_sorted.tsv and S_sorted.tsv may include common tuples, you should care about eliminating duplicates in their union. So, the output file should not include a tuple more than once. Example output:

```
aaa 151
aaa 292
aaa 336
...
```

3. Intersection (15%)

Write a program that reads files R_sorted.tsv and S_sorted.tsv, and computes and writes to a file RintersectionS.tsv the intersection of R and S, assuming that the two relations have the same schema. Your program should read each line from R and S *just once* and compute the intersection at the same time, implementing a variant of the merge-join algorithm. Again, you are not allowed to use a buffer and (of course) you are not allowed to load all data in memory before computing the intersection. Since R_sorted.tsv and S_sorted.tsv may include duplicate tuples, you should care about eliminating them when computing the intersection. So, the output file should not include a tuple more than once. Example output:

```
abc 748
abq 511
ack 549
...
```

4. Set difference (15%)

Write a program that reads files R_sorted.tsv and S_sorted.tsv, and computes and writes to a file RdifferenceS.tsv the difference R – S, assuming that the two relations have the same schema. Your program should read each line from R and S *just once* and compute the difference at the same time, implementing a variant of the merge-join algorithm. Again, you are not allowed to use a buffer and (of course) you are not allowed to load all data in memory before computing the difference. Since R_sorted.tsv and S_sorted.tsv may include duplicate tuples, you should care about eliminating them when computing the difference. So, the output file should not include a tuple more than once. Example output:

```
aaa 710
aaa 821
aaa 862
...
```

5. Grouping and Aggregation (30%)

Write a program that reads unsorted file R.tsv and computes and writes to a file Rgroupby.tsv the result of grouping the tuples of R based on the first attribute and for each distinct value v of the first attribute includes the sum of all the second attribute values in all tuples having v as first attribute value. For example, records ('aaa', 710), ('aaa', 821), ('aaa', 862), ('aaa', 958) should be grouped together and tuple ('aaa', 3351) is written to the output file.

This time, in your implementation, you will read all the data from R.tsv in a main-memory data structure (e.g., array). Then you will implement and use the sort-merge algorithm in memory, however, your algorithm should be slightly different than the classic sort-merge: if two tuples (v,u), (v,u') that are merged have the same value in v the first attribute (sorting attribute), you will write to the output of the merge a single tuple, having value v in the first attribute and having as second attribute value the sum u+u'. With this change a merged list can never have duplicate v's and the final merged list will be the required aggregation. Example output:

```
aaa 3351
aab 2468
aac 3503
...
```

Deliverables: You should submit your 6 programs and a single PDF file which documents the programs and any special instructions for compiling and running them. Please submit a single **ZIP** file with all requested programs and documents to Moodle on or before 5:00pm, September 30th, 2022. Make sure all contents are readable. **Please do not submit any data files**. Please feel free to post your questions on **Moodle forum** or contact the TA of the course if you encounter any difficulty in this assignment. We would be happy to help.