COMP7106 – Big data management

Assignment 3 – Top-k queries

Deadline: April 21, 2023, 5pm

The goal of this assignment is to develop an algorithm for top-k queries, which combine atomic scores of objects from **three** sources: the first two sources seq1 and seq2 allow only sequential accesses (i.e., the objects there are sorted in decreasing order of their scores), whereas the third source rnd, allows only random accesses (i.e., the objects there are sorted by ID). The data of the three sources are given in files seq1.txt, seq2.txt, and rnd.txt. Each line is an object, where the first number is the object-ID and the second number is the object's atomic score (from 0.0 to 5.0). Assume that the objects are restaurants, and each source corresponds to a restaurant review site. The objective is to find the top-k restaurants by average rating (average = sum/3).

You are asked to **write a program**, which takes a **command-line argument k** and computes and prints the list of **top-k objects from the three sources, using aggregate function sum**. For example, the top-1 object is 50905 with total score 14.84.

Your program should first read file rnd.txt and add the scores of all objects to an array R, such that the position x of the array contains the atomic score of object-ID x in file rnd.txt. For example, R[0] = 2.07, R[1] = 2.33, etc. Then, your program should read lines from seq1.txt and seq2.txt in a round-robin fashion (**DO NOT** read the entire files in data structures in memory like arrays). For each object o that is accessed from seq1.txt or seq2.txt:

- If object o has not been seen before, then initialize the total score of o to be the score accessed by sequential accessing plus the score R[o] (we assume that one random access is done for this). This way, we derive **a lower bound** of the total score of o.
- If object o has been seen before, add to the total score of o the score accessed by sequential accessing. This would result in the complete total score of o (because we have seen o sequentially also from the other sequential input).

For example, when we first access 33136, i.e., the first object from seq1.txt, and its score 5.00 in seq1, we lookup R[33136]=4.40 and obtain its partial aggregate score 9.40 (this is a lower bound of the total score). Later, if we access 33136 again from seq2.txt, we will get score 4.28, which will be added to 9.40 to derive the total score of 33136, which is 13.68.

After having seen exactly k objects, you should initialize a min-heap W_k, which stores the top-k objects so far based on their lower bounds. The top of W_k should contain the smallest lower bound score of the top-k objects so far.

From thereon, you should continue round-robin accesses to seq1.txt and seq2.txt and after each access update the Threshold T = sum(last score seen in seq1.txt + last score seen in seq2.txt + 5.0). Threshold T represents the best possible score of any object not seen so far. While the score of the top element of heap W_k is smaller than T, we cannot terminate and we should do more sequential accesses. As soon as T becomes smaller than or equal to the top element of heap W_k, termination becomes possible. So, in this case, you should check if there exists any seen object x not in W_k, such that the upper bound of x is greater than the top element of W_k; if there exists such an object, we cannot terminate and we should continue the sequential accesses. Otherwise, we can terminate and report W_k as the top-k objects. The upper bound of x can be computed by adding to the partial score of x (lower bound known so far) the last score seen in the sequential input where x has not been seen yet.

Your program should output:

- The total number of sequential accesses to seq1.txt and seq2.txt. This corresponds to the number of lines that you have read from seq1.txt and seq2.txt until termination. Your program should minimize the number of lines read from seq1.txt and seq2.txt. You should not read all file contents in memory before starting the algorithm, but the algorithm should read the two files concurrently line-by-line until the correct result is guaranteed.
- The top-k objects and their total scores in descending score order.

Example of program running for k=5

```
Number of sequential accesses= 3018
Top k objects:
50905: 14.84
85861: 14.76
22652: 14.74
75232: 14.74
20132: 14.74
```

You may use any data structures or libraries/modules of your programming language (e.g., for heap). To check the correctness of your program, you are advised to implement a baseline method, which reads all data in memory and computes the top-k objects in a brute-force manner (i.e., compute the total scores of all objects, then sort them in descending order of their total scores, then print the top-k).

Deliverables: You should submit your program and a single PDF file which documents the program and includes any additional comments. You can use a programming language of your choice, but your program should be OS-independent. Please submit a single ZIP file with all requested programs and documents to Moodle on or before 5:00pm, April 21st, 2023. Make sure all contents are readable. Please do not submit any data files, your input files should be the same as given (please don't rename them or use any self-modified 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.