Homework 4: Query Execution

DSCI 551 – Fall 2024

Due: 11:59pm, November 15, 2024, Friday

Points: 100

In this homework, you are provided with a template program "sort_merge_join.py" for sort-merge join. Your task is to complete the missing codes in the template. The areas with missing codes are marked with **"#### fill in code …".**

The program essentially implements the following SQL query:

```
Select R1.proj_attrs1, R2.proj_attrs2

From R1, R2

Where R1.join_key1 = R2.join_key2
```

Where the content of R1 is stored in file1 which contains a collection of JSON objects/documents, one per row of the table R1. Similarly, the content of R2 is stored in file2. Proj_attrs are the list of attributes projected from the table. Join_keys refer to join attributes. The program assumes that there is only one join attribute from each table, but it may project multiple attributes from a table.

The code assumes that the details of the join query are specified in a parameter file called "parameters.json". For example,

```
{
        "file1": "country.json",
        "join_key1": "Code",
        "proj_attrs1": ["Name", "GNP", "Continent"],
        "file2": "countrylanguage.json",
        "join_key2": "CountryCode",
        "proj_attrs2": ["Language", "Percentage"],
        "chunksize": 100,
        "output_file": "join_output.json"
}
```

Note that the file also has a parameter chunksize which is used to split the input files into chunks/runs. For example, in the above example, both country and city rows will be split into chunk size of 100 (max). Note the last chunk of each table may contain fewer than chunksize of rows. The join output will be written in a file "join_output.json", where each line contains a pair of json objects, each containing the key-values for the projection attributes. See example output file for more details.

Recall that sort-merge join program works in two phases:

- **Sorting**: it will load the input file (e.g., country.json) one chunk at a time (e.g., loading first 100 documents, next 100, and so on), sort the chunk by the join key, and write it to the disk as run. In the program, these runs are stored in a folder "temp_runs" for storing temporary results of the join process.
- **Merging**: it will merge the runs created in the first phase, and find joining rows while merging. The program assumes that it loads only one row from each run at a time. That is, the input buffer page contains only one row. It assumes that it has enough input buffer holding one row from every run of both relations.
    - To facilitate the merging process, the program uses a heap for merging runs from each relation. Heap1 for relation1, heap2 for relation2. The heap is essentially a priority queue that spills out the rows in the runs in the ascending order of their join key values. For example, rows in the country table will be ordered by their Code values.
    - Since there may be multiple rows from each table that have the same value on their join attributes, the program extracts such rows together and produces joined rows accordingly. See the function nexts_from_runs(heap, join_key).
    - The details on heap queue can be found here:
        - https://docs.python.org/3/library/heapq.html
        - https://en.wikipedia.org/wiki/Heap_(data_structure)

Read below carefully before you start!

Requirements:

- You should not add new libraries into the code.
- Only fill in the missing code. Make sure you do not make any other modifications besides removing all print(None). For more details, see instructions commented in the template.
- For grading purposes, make sure you name the chunk files exactly like below:

  e.g., [temp_runs/**country_run0.json**, temp_runs/**country_run1.json**, ...]

  i.e., starting from index of 0
- In your submission, DO NOT add code for removing the temp_runs folder.

Submission details:

- Submit the complete code **sort_merge_join.py** (where all missing code are provided). DO NOT modify the file name.

Note: Any violation of the requirements will result in receiving 0 points for this assignment. No regrade requests relevant to this will be accepted.