Database Systems, CSCI 4380-01
Homework # 10
Due Thursday November 21, 2019 at 11:59:59 PM

Homework Statement. This homework is worth 3.5% of your total grade. If you choose to skip it, Final Exam will be worth 3.5% more. This homework is for understanding mechanics of indexing and searching indexes.

In this homework, you are given a simulation of a B-tree index for `listings(region, host_id, room_type)` for the full Airbnb database (48K tuples). Each node represents a disk page and is stored as a separate file. An index node id is the same as file name.

The B-tree index has the following format:

Internal nodes: Each line alternates between a pointer to a B-tree node and a key value. For example if the internal node starts as:

```
idxpage165.txt
A1
idxpage166.txt
A2
idxpage167.txt
...
```

We can read this as follows:

For values less than `A1`, go to `idxpage165.txt`.

For values greater than or equal to `A1` but less than `A2`, go to `idxpage166.txt`, etc.

Note that multiple tuples can have the same key value, so if `A1=A2`, then follow the first value you encounter that has the correct range.

Leaf nodes: Each line in a leaf nodes represents a tuple. We store the key values and the page id of a tuple for that key value. The last line of the index stores the page id of the next leaf node (sibling pointer). Here is an example:

```
Bedford-Stuyvesant|58633574|Private room|1472
Bedford-Stuyvesant|58994794|Private room|2160
...
Bedford-Stuyvesant|72957132|Private room|2424
Bedford-Stuyvesant|73066515|Entire home/apt|2131
NEXT:28
```

Problem Description. Write a program, either in C++ or Python that takes as input a query file:

```
hw10_adalis.py hw10input.txt
```

in which each line is a separate query (values separated by `|`) with ranges for each indexed attribute `region, host_id, room_type` in this order. Each range specifies a low value and a high value (both inclusive) for that attribute, wild value (`*`) specifying don't care. For example:

```
NoHo|SoHo|*|*|Private room|*
```

This translates to the following query:

```
FROM
   listings
WHERE
   region >= 'NoHo'
   and region <= 'SoHo'
   and room_type >= 'Private room'
```

Your program should scan the index, starting with root to find all the tuples satisfying these conditions, by following the B-tree structure: root, internal node, the first leaf node and follow all relevant leaf nodes using sibling pointers.

Your program should print out the following information for each query on a separate line (no spaces):

```
X1|X2|X3|X4|condition
```

where `X1` is the number of leaf nodes scanned, `X2` the number of matching tuples found, `X3` number of distinct pages the tuples reside in and `X4` is number of different relation pages read and `condition` is the search condition used for scanning the index.

To compute `X4`, suppose you are reading a disk page (for accessing a tuple found in the index) as soon as you encounter it. As you find new tuples and the relation page id that stores it, you want to read that page. However, if the page id is the same as the previous page id, then the page is already in memory. If it is not, you may end up re-reading a page. For example, if you encounter the following relation pages as you find the matching tuples:

```
P1, P2, P3, P3, P2, P1, P1, P4
```

Then, you will be reading pages: `P1, P2, P3, P2, P1, P4`, hence a total of 6 pages (`X4`) even though the number of distinct pages is 4 (`X3`).

SUBMISSION INSTRUCTIONS. Submit the following files into Submitty:

- A single file containing your source code.
- A README.txt that describes how to run your code.
- An output file containing the output to the test cases. The input file to be used as test cases will be provided by tuesday the latest.