CS 409 - UG SOFTWARE LAB
Mini-Project 3 Weightage: 30%
Submission Deadline: 3-May-2021 23:55pm

General Instructions:

- You must use only C, C++, Python or JAVA for this assignment.
- Prescribed specifications must be strictly followed. Failure to do so may lead to substantial loss of points.
- Make sure your code is well written (self explanatory variable names) and documented. You are likely to lose points if your TA cannot understand your code.
- You are not allowed to use any specialized libraries. You are allowed to use only the primitive data types in the language such as arrays, structures, classes, hash_map, lists, associative array, etc.
- **This project needs to be done in teams of size 2. Contact me in case you are working solo on the project.**

Question 1 (100 points):

In the final project, you would be developing an implementation of Grid files for storing and querying 2-dimensional data points.

Dataset for evaluation:

We would be using synthetic datasets for evaluation. Each data point in the dataset is a triple defined as follows: <id> <x-coordinate> <y-coordinate>. Here <id> is a unique number between 1 and number-points-in-dataset. Note that the spatial coordinates of the points in the datasets could be repeated. In other words, two different data-points (i.e., two different ids) can have the same x and y coordinates.

Dataset A: 30000 points generated in a uniformly random fashion where the x and y coordinates are random integers between 0 and 400.

Index structures need to be implemented:

(a) **Grid files.** Please refer to the class notes on this topic.

Simulating Buckets:

Buckets are to be implemented as text files in your code. Bucket size is defined as the number of data points it can accommodate. For instance, if the bucket-size is defined to be 30, it means that the file simulating the bucket can store 30 data records. **Very Imp:** Bucket sizes must be taken as input, should not be hard coded. The value of bucket-size would be varied in the experiments.

Query Algorithm to be implemented on index structures:

Range query on GRID files: Given the lower left and the upper right coordinates of a query rectangle, retrieve all points which fall inside the rectangle. Correctness of the algorithm must be ensured. Grossly inefficient techniques and naive solutions would not be accepted.

Implementation specifications:

(a) Input dataset should be read from a file.

(b) Buckets must be shared whenever necessary and the splits must be axis parallel lines, i.e., they must be through and throughout.

(c) Points should be inserted one by one into the Grid file. Your code should have a separate function for insertion. Also, there should be separate functions for splitting the scales and splitting the buckets. You may choose to split at the median data point (choice of axis is up to you). Appropriate implementation logic should be present for rearrangement of buckets.

(d) Your implementation should have appropriate global data structures for the following: (1) X and Y scales which denote the current grid structure (i.e., the split points on x and y axis). (2) A “mapper” which maps a grid cell to a bucket.

(e) Your implementation should have appropriate print functions for the Grid files. This function should print the entire data structure and the bucket contents. Your TA will use this function to evaluate the correctness of your code on small datasets. Output of this print function should be comprehensive and formatted appropriately. Scalability testing would be done on large datasets.

(f) In addition to the range query algorithm on Grid files, also implement a naive approach for solving range queries which just traverses through the dataset to return the points which are inside the query rectangle. TAs would be using this function to evaluate the correctness of your range query algo on Grid files.