PG Software Lab - CSP 509 / CSP 609

Final Project

Due date: Nov 29 2018, 2018 11:55pm

Total Weightage 38 %

Important Instructions:

- All submissions must be made through the Moodle site for this course.
- Any assumptions made while solving the problem should be clearly stated in the solution. Reasonable ones would be accepted and graded. Trivial and naïve algorithms for computing range queries will not be accepted.
- You may use existing libraries for basic data structures like vectors and associative array (map in C++ STL libraries). You may also use math libraries as needed for tasks like, generating random numbers, etc. Specialized libraries for managing hash tables must not be used.
- As always correctness of the algorithm must be ensured.
- Badly formatted output, which the TA is unable to understand, would lead to significant loss of points.
- The TA would be quizzing you on your code. You must understand each and every line of your submitted code.
- **Implementation specifications must be strictly followed. Failure to do so may lead to substantial loss of points.**
- **Very imp: There should be no directory structure in your code. All the files should be in just one directory.**
- **Question 2 is for teams of size-3 only. This question would not be graded for teams of size-2.**
- Your implementation would be evaluated for space and time efficiency. Unnecessary replication of data in the code and wastage of main memory space at the run-time may lead to loss of points.

Question 1 (200 Points)

In the final project, you would be developing an secondary memory based implementations for storing and querying 2-dimensional data points.

Datasets for evaluations:

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 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.

Dataset B: 12000 points generated in a uniformly random fashion where x and y coordinates are integers between 0 and 120. And another 20000 points generated over the ranges of 120 – 400 (integer x and y coordinates).

Index structures need to be implemented:

(a) **Grid files.** Please refer the class notes on this topic. More details can be found in the following paper Nievergelt et. al.: The Grid File: An Adaptable, Symmetric Multikey File Structure. ACM Trans. Database Syst. 9, 1 (March 1984).

(b) 2-Dimensional Regular-Grid-Array: Here, we would create a 2-dimensional grid to store the data points. Points in each grid cell should be stored as a bucket. For example, if your grid cell size is 100-by-100, all the points whose x-coordinate is between 0 – 100 and y-coordinate is between 0 – 100 would go into one bucket. Similarly, all the points whose x-coordinate is between 100 – 200 and y-coordinate is between 0 – 100 would go into another bucket, and so on. Once a bucket overflows, you need to create a new bucket and link it to current one (similar to chaining). Grid cell size needs to be taken in as an input.

Simulating Buckets:

Buckets are to be implemented as text files in your code. Bucket size is defined as the number of data point 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. In case of bucket overflows, you need to maintain the name of next bucket (also simulated as a file) in the current buket. **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:

Knn query: Given a query point q (x and y coordinate) and a value of k, retrieve the k closest points (from the dataset) for the input query point q. 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) In case of Grid files, Buckets must be shared whenever necessary and the splits must be axis parallel lines, i.e., they must be through and throughout.

(c) In grid files, points should be inserted one by one. 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) In case of overflow (only for 2 dimensional regular grid array) at a grid cell, the mapper would point to the first bucket. This first bucket should internally point to the next one in the overflow chain.

(f) Your implementation should have appropriate print functions for the 2-Dimensional Grid Array and the Grid files. This function should print the entire data structure. Your implementation should also have print functions for the knn query algorithms. Your TA will use these functions to evaluate the correctness of your code on small datasets. Output of these print functions should be comprehensive and formatted appropriately. Scalability testing would be done on large datasets.

(g) **Implementation for counting bucket access and simulating main/sec memory:** Assume that the main memory can hold only 1 bucket at a time. So if you algorithm is trying to access a grid cell (a,b), it would first determine its bucket number (say X). Then it should see if bucket X is located in the "main memory," if not then it has to be "fetched." Once you fetch a bucket you need to increment the #buckets-accessed counter.

Experiments:

Experiment 1-1

Algorithms: (a) Knn query algorithm on Grid-File,
(b) Knn query algo on 2D-Regular-Grid (Grid cell size 100-by-100).

Bucket Size: 30

Dataset A

X-axis -- value of k (5, 20, 50, 100)

Y-axis --- Number of buckets accessed

Note: For a given value of k (e.g., 5), create several random query points inside the dataset boundary (i.e., between 0,0 and 400,400). Pass each of these as a knn query (with the given k value) and measure the number of buckets accessed; finally take an average of those values and plot it as the y-coordinate (x-coordinate is the value of k, 5 in this example). Plot the results of both the algorithms in the same plot.

Experiment 1-2

Algorithms: (a) Knn query algorithm on Grid-File,
(b) Knn query algo on 2D-Regular-Grid (Grid cell size 100-by-100).

Bucket Size: 100

Dataset A

X-axis -- value of k (5, 20, 50, 100)

Y-axis --- Number of buckets accessed

Note: For a given value of k (e.g., 5), create several random query points inside the dataset boundary (i.e., between 0,0 and 400,400). Pass each of these as a knn query (with the given k value) and measure the number of buckets accessed; finally take an average of those values and plot it as the y-coordinate (x-coordinate is the value of k, 5 in this example). Plot the results of both the algorithms in the same plot.

Experiment 1-3

Repeat Experiment 1-1 with Dataset B.

Experiment 1-4

Repeat Experiment 1-2 with Dataset B.

Things to be submitted:

(a) **All the code. Note that file names of code should have your roll numbers in the prefix. Code should not have a directory structure.**

(b) **Each experiment should be separate plot. Plots must be labeled properly (x axis, y axis and legends) and the legends must be clearly visible. In case the TAs are not able to read and understand the plots, points will be deducted.**

(c) **Put all the plots in a report and briefly comment on the observed trends. You should explain the observed trends appropriately. This report must be written in LaTeX (with page margin as 1-inch).**

Question 2 (for team size-3 only) (70 points)
In addition to Question 1, teams of size-3 need to do the following as well.

(a) **Range query on 2 dimensional grid Array and the 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. These range query algorithms must be implemented over the data-structures developed in Question 1. In addition, an appropriate print function must be implemented to evaluate the correctness of the code on small datasets.

(b) **Knn query on the KD-trees.** Given a query point q (x and y coordinate) and a value of k, retrieve the k closest points (from the dataset) for the input query point q. Correctness of the algorithm must be ensured. Grossly inefficient techniques and naive solutions would not be accepted. For this, you can use the kd-tree implementation from your miniproject. In addition, an appropriate print function must be implemented to evaluate the correctness of the code on small datasets.

Experiment 2-1

Algorithms: (a) Range query algorithm of Grid-File,
(b) Range query algo of 2D-Regular-Grid (Grid cell size 100-by-100).
Bucket Size: 30
Dataset A
X-axis -- area of the range query (50, 100, 150, 200)
Y-axis --- Number of buckets accessed
Note: given an area (e.g., 50), you can create multiple rectangles of this area all over the dataset. Some of the rectangles should be long and thin and others more balanced. Pass each one of them as a range query and measure the number of buckets accessed for each one of them; finally take an average of those values and plot it as the y-coordinate (x-coordinate is the area, 50 in this example). Create your query rectangles such that all of them are completely inside the dataset boundary of 0,0 and 400,400. Plot the results of both the algorithms in the same plot.

Experiment 2-2

Algorithms: (a) Range query algorithm of Grid-File,
(b) Range query algo of 2D-Regular-Grid (Grid cell size 100-by-100).
Bucket Size: 100
Dataset A
X-axis -- area of the range query (50, 100, 150, 200)
Y-axis --- Number of buckets accessed
Note: given an area (e.g., 50), you can create multiple rectangles of this area all over the dataset. Some of the rectangles should be long and thin and others more balanced. Pass each one of them as a range query and measure the number of buckets accessed for each one of them; finally take an average of those values and plot it as the y-coordinate (x-coordinate is the area, 50 in this example). Create your query rectangles such that all of them are completely inside the dataset boundary of 0,0 and 400,400. Plot the results of both the algorithms in the same plot.

Experiment 2-3

Repeat Experiment 2-1 with Dataset B.

Experiment 2-4

Repeat Experiment 2-2 with Dataset B.

Things to be submitted:

- **(a)** **All the code. Note that file names of code should have your roll numbers in the prefix. Code should not have a directory structure.**
- **(b)** **Each experiment should be separate plot. Plots must be labeled properly (x axis, y axis and legends) and the legends must be clearly visible. In case the TAs are not able to read and understand the plots, points will be deducted.**
- **(c)** **Put all the plots in a report and briefly comment on the trends. You should explain the observed trends appropriately. This report must be written in LaTeX (with page margin as 1-inch).**