Introduction to Spatial Computing- CSE 555

Homework 3

Due date: Oct 22, 2016 6:00am

Important Instructions:

- All submissions must be made through usebackpack site for this course.
- Only one submission per team would be considered and graded. It would be assumed that all members of the team have participated equally and same score would be given to all members of the team.
- Your submission should have names of all the members of your team.
- Only one submission should be uploaded per team.
- 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 KNN and Range queries will not be accepted.
- As always correctness of the algorithm must be ensured.
- TAs would be quizzing you on your code. You must understand each and every line of your submitted code. Also the implementation specifications mentioned in the questions need to be strictly followed. Failure to adhere to these requirements would result in substantial loss of points.
- Question 2 is for teams of size 3. This questions will not be graded for teams for size 2 or less.

Question 1 (100 Points)

In this question you would be comparing the performance of Z-order curves, Hilbert curves and Grid files.

Index structures need to be implemented:

(a) Z-order curves. You are strongly suggested to use existing code for B+ tree. In other words, just code the z-order values and insert them into a B+ tree created from an existing code found online.
(b) Hilbert curves. Similar to part (a), you are strongly suggested to use existing code for B+ tree.
(c) Grid files.

Datasets for evaluations:

Following three synthetic datasets should be created and used for evaluation:

Dataset A: 50000 points generated in a uniformly random fashion where the x and y coordinates are integers between 0 and 500.

Dataset B: 25000 points generated in a uniformly random fashion where x and y coordinates are integers between 400 and 600. And another 25000 points generated over the ranges of 0 – 400 (integer x and y coordinates)

Bucket sizes:

Option-one: 20 points *Option-two:* 60 points

Query Algorithm to be implemented on index structures

K-nearest neighbor (Knn) query: Given the coordinates of a query point retrieve k points which are closest to it than any other points in the datasets. Your answer should include a detailed description of the Knn algorithm. Correctness of the algorithm must be ensured.

Some Implementation specifications:

(a) As you remember from class, Z-order and Hilbert-curves do not help so much with KNN queries as they did with range queries. For this reason, you are strongly advised to take advantage of the fact that points in the datasets are from an integer space. For every query point, enumerate its potential neighbors and cross-check their presence in the data structure. For e.g., if your query point is (a,b), you should first check presence of (a,b), (a-1, b), (a+1,b), (a, b-1), (a,b+1), (a+1, b+1), (a-1, b-1), (a+1, b-1), (a-1, b+1) in the dataset and compute their distance (and sort) from the query point to see if k is satisfied. In case k is not satisfied, then continue this process of enumerating the neighbors of the query point in "concentric rectangles". Your code should clearly have this logic implemented.

(b) For sake of uniformity, use the same algorithm described in part (a) for Grid files as well.

(c) 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. Bucket size should be taken in as an input parameter. Though your implementation puts these buckets in main memory, it will be assumed that these are sitting in secondary memory.

(d) **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 point (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. You must also have a small tag or data structure to simulate the "main memory" part for counting bucket access.

(e) 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.

Experiments to be conducted

Randomly generate Knn queries and determine the number of buckets examined to retrieve the final results. Following plots should be submitted with the answer. All plots should have proper legends.

Plot 1a containing 3 curves:

X-axis: Value of k. Pick k= 10, 50, 90, 130
Y-axis: Average number of buckets retrieved for 10 randomly generated Knn queries
Bucket size: 20 points
Curve 1: Z-order curves on Dataset A
Curve 2: Hilbert curves on Dataset A
Curve 3: Grid files on Dataset A

Plot 2a containing 3 curves:

X-axis: Value of k. Pick k=10, 50, 90 and 130
Y-axis: Average number of buckets retrieved for 10 randomly generated Knn queries
Bucket size: 20 points
Curve 1: Z-order curves on Dataset B
Curve 2: Hilbert curves on Dataset B
Curve 3: Grid files on Dataset B

Plot 1b: repeat the experiment in Plot 1a with a bucket size of 60 points

Plot 2b: repeat the experiment in Plot 2a with a bucket size of 60 points

Question 2 (20 Points)

Implement a range query algorithm on the grid files. Input should be taken in as the coordinates of lower left and upper right points of the query rectangle. As always correctness of the algorithm must be ensured. Trivial and naïve algorithms will not be accepted.

Things to be submitted:

A zipped folder containing the following:

(a) Code for Z-order, Hilbert and Grid files (Question 1)
(b) Query algorithm for KNN search (Question 1)
(c) The mentioned six plots in Question 1 and a 500 word interpretation of trends obtained.
(d) Query algorithm for Range search (Question 2)