**Boston University**
**CAS CS 562: Advanced Database Applications**
**Homework #1**
**Fall 2018**

**Due Date: Feb 9, 2018 at 5PM, using gsubmit**

**Problem 1. (Space Filling Curves)**

Write a program to experimentally compare the performance of z-values versus the performance of the Hilbert values (h-values) for range queries. We will provide two datasets that contain 2-dimensional points and a set of range queries for each dataset. You have first to compute for each point, the cell that it belongs using the precision p, and then compute the z- or h- value for each point. That is, given a precision p, each dimension is divided into $2^p$ intervals and the data space is divided into $2^p x 2^p$ cells. Now, given a 2-d range query Q, you have to write a function that returns the set of 1-d ranges that the query is mapped to and estimate how many I/Os (pages) need to be performed in order to answer the query assuming that the z- or h-values are stored into a B-tree. You can use an existing real B-tree in your experiments or you can use a model to estimate the number of pages to retrieve the answer using a logical B-tree. We will provide the B-tree characteristics together with the datasets and the queries. Show the average cost in number of I/Os (number of reads) of different sets of range queries for different datasets and different precision p. Submit both the results and your code.

HINT: Methods to compute z- and Hilbert- values can be found in the following paper:
H. V. Jagadish: *Linear Clustering of Objects with Multiple Attributes.* **ACM SIGMOD Conference** 1990, pages 332-342.

**Problem 2. (R-trees)**

Let $D$ be a 2-dimensional point dataset and $p = (x, y)$ a point in that set. The coordinates of all points are positive. Consider the function: $f(p) : D \rightarrow R$, where $f(p) = a_1 x + a_2 y$ and $a_1 + a_2 = 1$. The values for $a_1$ and $a_2$ are given by the user. The idea is that each user gives different importance (weight) to different attributes. We want to find the point (or points) that maximize(s) this function. This type of queries are called *preference* queries. Now, assume that an R-tree is used to store the dataset $D$.

(a) Design an efficient search procedure that uses the R-tree to find the point(s) that maximize the function $f$. Give the pseudo-code of the algorithm and explain how it works.

(b) What is the property that allows the design and guarantees the correctness of your algorithm? Explain.

**Problem 3. (General)**

Answer the following questions about some of the methods that we discussed so far:

1. Consider a *range counting query*, where the query asks for the number of points inside a range. Assume that you have a dataset of 2-d points and you want to create an index based on kd-trees to answer efficiently range counting queries. Explain how you will modify the original kd-tree data structure and give an algorithm that will use this modified kd-tree to answer efficiently range counting queries. What is the worst case performance of your data structure for range counting queries?

2. Discuss the different splitting policies of the Grid File and what are the pros and cons of each method. Which method you would use in your implementation? Why? Explain.

3. What is the **paging algorithm** of the LSD-tree and how we use it?