Carnegie Mellon University
15-826 – Multimedia Databases and Data Mining
Spring 2009, C. Faloutsos
Assignment 1, Due Date: Feb 3, in class

Reminders

- Due date: Feb. 3, 3:00pm, hard copy in class and soft copy e-mailed to the TA (*bziebart+826 at cs*) in a single e-mail (along with your source code).
- Please turn in a **typed** report – handwritten material may not be graded, at the grader's discretion.
- All homeworks including this one are to be done INDVIDUALLY.
- Expected effort for this homework (approximate times):
  - Q1: 5 hours
    - 1 hour to download the datasets and MySQL
    - 1 hour to install MySQL and read the manual
    - 2 hours to formulate all the queries
    - 1 hour to execute all queries and answer the questions
  - Q2: 5 hours
    - 1 hour to download and make the package
    - 3 hours to write and debug your algorithm
    - 1 hour to run your algorithm and answer questions
  - Q3: 10 hours
    - 1 hour to download and make the package
    - 7 hours to write and debug your algorithm
    - 2 hours to run your algorithm and answer questions

Q1 – DBMS and SQL [30pts]

Problem Description: The American Time Use-Survey (ATUS) is a minute-by-minute survey of how thousands of individuals spend their time over a 24-hour period (from 4:00AM to 4:00AM). We will use a DBMS and SQL queries to analyze statistical properties of this data. First, load the following tables into your DBMS (MySQL preferred).

Tables:

- DEMOGRAPHICS[1] (userID, age)
- ACTIVITIES[2] (userID, startTime, endTime, activityID, duration)

Part I. Basic Queries

Turn In: Please provide both **(a) the SQL query** and **(b) the resulting answer** for each of the following questions:

1. [2 points] What is the average age of participants in this survey?
2. [2 points] What are the ages of the 10 oldest participants?
3. [6 points] What is the average amount of sleep (*activityID = 10101*) in the 24 hour survey period for participants in this survey? (Note: 6 hours and 1 hour for a single participant equates to 7 hours)

Part II. Join Queries

4. Consider this question: What is the average amount of sleep reported over the 24 hour survey by those in their 20's (i.e, age 20 to 29 inclusive)?
   (a) [3 points] What is the query that obtains the answer?
   (b) [3 points] How is the join executed without indexing?
   (c) [3 points] How should the table(s) be indexed to make this query fast?
   (d) [3 points] How is the join executed after indexing?
   (e) [3 points] What is the answer to the query?
5. [5 points] What is the average age of those in the survey who report being asleep at 22:00 (i.e., 10:00PM)? Please provide both the query and the answer to the query. **Note:** An activity can start before 22:00 and end after 0:00.

Hints:

- SQL has a built-in type for time of day: *TIME*.
- Using *EXPLAIN* (and *LIMIT*) and appropriately indexing each table may dramatically change the run time of your queries.
- Feel free to use views or addition tables to compute intermediary results.

[1] http://www.cs.cmu.edu/~bziebart/15826-S09/hw1/demographics.csv
[2] http://www.cs.cmu.edu/~bziebart/15826-S09/hw1/activities.csv

Q2 – KD-Trees [30pts]

Problem Description: We will add new functionality to an existing KD-Tree package to find the element with the maximum x_1 value. Please build the KD-Tree Package[3] (`tar xvf; make`). Running `make hw1` should return an *algorithm not implemented* message after loading the appropriate dataset.

Consider a point, $p = (x_1, x_2, ...)^\top$. We are interested in finding the point from a set of points in a KD-tree with the maximum x_1 value.

Implementation: Implement a new command, `m`, that finds the element with the maximum x_1 value. It should print out the node ID and coordinates of that element.

Turn In:

1. [8 points] The index and values for the maximum x_1-valued datapoint when applied to the KD-Tree constructed with: (a) dataset 1 (using `make hw1`), (b) dataset 2 (using `make hw2`), and (c) dataset 3 (using `make hw3`).

2. [8 points] The number of nodes explored by your algorithm when applied to the KD-Tree constructed with: (a) dataset 1, (b) dataset 2, and (c) dataset 3.

3. [14 points] A tarball (`kdtree_YOURUSERNAME.tar`) of your code emailed to the TA (*bziebart+826 at cs*) and a hard copy of your code included in your submitted homework document.

Q3 – R-Trees [40pts]

Problem Description: We will add new functionality to an existing R-Tree package to perform a Skyline Query. Please build the R-Tree Package[4] (`tar xvf; make demo`). This creates the `bin/DRmain` program and runs it on some small datasets. It has been tested on the Unix platform in the andrew machines along with Cygwin on Windows. Running `make hw1` should return an *algorithm not implemented* message after loading the appropriate dataset.

Consider two data points: $p_1 = (x_1, y_1)$ and $p_2 = (x_2, y_2)$. p_1 is said to *dominate* p_2 if $x_1 < x_2$ and $y_1 < y_2$. More formally, p_1 must be better than p_2 in at least one dimension, but can be equal in all other dimensions[5] to dominate p_2. A point will dominate a whole range of points (as shown in Figure 1). A Skyline Query finds all *Leader* points, which are all the points in the dataset that are not dominated by another point.

[3] http://www.cs.cmu.edu/~bziebart/15826-S09/hw1/kdtree.tar
[4] http://www.cs.cmu.edu/~bziebart/15826-S09/hw1/rtree.tar
[5] If p_1 and p_2 are equal, neither dominates the other.

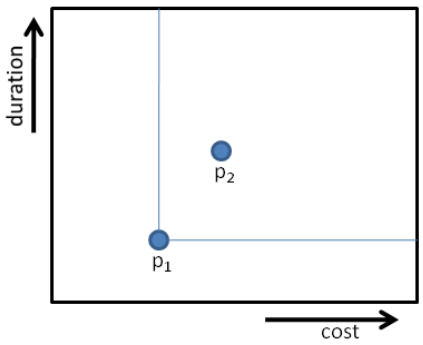

Figure 1: Point p_1 dominating point p_2.

To make this concrete, let x_1 and x_2, correspond to *cost* and *travel time* for flights to Sydney, Australia. Flights that are less expensive and shorter are universally preferred. However, depending on the person's exact trade-off (e.g., is CMU paying for the flight?), different flights will be best (that is, Leaders). The Skyline Query finds all data points that could be the best.

Implementation: Implement a new command, `y`, for performing a sk(y)line query of the data. Your program should print the node ID and coordinates of each of the non-dominated points (Leaders).

Turn In:

1. [4 points] A list of the skyline points (Leaders) for dataset 1 (2-d). This should run using `make hw1`.
2. [4 points] A list of the skyline points (Leaders) for dataset 2 (2-d). This should run using `make hw2`.
3. [8 points] The average running time and standard deviation of the running time (averaged over 10 executions) of your algorithm for the skyline query when run on dataset 1 and and dataset 2.
4. [4 points] A list of the skyline points (Leaders) for dataset 3 (2-d). This should run using `make hw3`.
5. [4 points] A list of the skyline points (Leaders) for dataset 4 (3-d). This should run using `make hw4`.
6. [16 points] A tarball (`rtree_YOURUSERNAME.tar`) of your code emailed to the TA (*bziebart+826 at cs*) and a hard copy of your code included in your submitted homework document.

Hints:

- A correct semi-efficient method is to find the skyline points for each child node and merge them.
- An even faster method is to exclude children that are clearly dominated by the contents of some other child. Either method will win full credit.