Carnegie Mellon University
15-826 – Multimedia Databases and Data Mining
Fall 2013, C. Faloutsos
Homework 1, Due Date: Oct 1st, at classroom 1:30pm
Prepared by: Seunghak Lee

Reminders

- All homeworks are to be done **INDIVIDUALLY**.
- For code submission to blackboard, make three directories /Q1, /Q2, /Q3, and then put your code for question 1, 2, 3 to /Q1, /Q2, /Q3, respectively. Then submit a single file ([andrew-id].tar.gz) that contains /Q1, /Q2, /Q3.
- Expected effort for this homework (approximate times):
  - Q1: 2-5 hours (To be graded by Alex Beutel)
    - 1-3 hours to write and debug all sql queries
    - 1-2 hours to run them queries and record the answers.
  - Q2: 4-9 hours (To be graded by Seunghak Lee)
    - 30' to download and `make` the package
    - 3-8 hours to write and debug your algorithm
    - 30' - 1 hour to run your algorithm and answer questions
  - Q3: 4-9 hours (To be graded by Vagelis Papalexakis)
    - 1 hour to download and `make` the package
    - 3-7 hours to write and debug your algorithm
    - 1 hours to run your algorithm and answer questions

Q1 – DBMS and SQL [30pts]

Problem Description: For each question in this part, provide both the SQL statement(s) and the resulting answer(s), unless specified otherwise. You'll be using the *movie reviews from Amazon* dataset [1] [2] [3] [4]
Hint: Please use SQLite3; version 3.6.20 is already on the andrew cluster machines. You may use your own machine and your own sqlite3 installation, as long as your submitted code runs correctly on the andrew cluster machines.

[1]movie data: `https://snap.stanford.edu/data/movies.txt.gz`.

[2]To save bandwidth, short version (`movies-short.txt.gz`) without text comments is available at `http://www.cs.cmu.edu/~seunghak/movies-short.txt.gz`

[3]information about the movie data: `https://snap.stanford.edu/data/web-Movies.html`

[4]make file: `http://www.cs.cmu.edu/~christos/courses/826.F13/HOMEWORKS/HW1/Q1/makefile-q1.tar.gz`

Implementation Details: Write sql code for the following queries.

1. [**0 pt**] *Line number:* Download the movies-short.txt.gz[2] from `http://www.cs.cmu.edu/~seunghak/movies-short.txt.gz` and `make` file[3] from `http://www.cs.cmu.edu/~christos/courses/826.F13/HOMEWORKS/HW1/Q1/makefile-q1.tar.gz` and unzip it. Then count the number of lines of movies-mini.csv (try '`make` linenum'). Next, try '`make`', and it will show you what to implement.
2. [**2 pt**] *Parsing:* Extract the following five columns from `movies-short.txt`: productId, userId, helpfulness, score, time. (Run `wc -l movies-short.txt`, to check if the number of lines of the movie data is 47470103.)
3. [**2 pt**] *Loading:* Create and load the following table, using the parsed data above.
   - `movies(productId, userId, helpfulness,score, time)`

   (Hint: check `.help` and `.import` in SQLite3)
4. [**2 pt**] *Movie size:* How many distinct movies does this dataset contain?
5. [**4 pt**] *Popular movie:* What is the most popular movie in this dataset (the movie with the largest number of reviews)? Report both productId and its real movie name by searching 'www.amazon.com/dp/[productId]' in the web.
6. [**4 pt**] *Active reviewer:* Who is the most active reviewer? (the reviewer with the largest number of reviews)
7. [**4 pt**] *Top movies:* What are the top 5 movies with the highest average score among the movies with more than 100 reviews? Report productId. For fun, manually find its real movie name by searching 'www.amazon.com/dp/[productId]' in the web.
   [**2 pt**] *Report* the wall-clock running time of your query, using, e.g., the `time` Linux/Unix command.
8. [**2 pt**] *Index impact:* Create indices on the columns `productId`, and `score` (follow the syntax shown here).
   [**2 pt**] Run the query from the last question again ("Top movies"), and *report* the wall-clock running time.
9. [**2 pt**] *Query optimization:* Show the output of `explain select` for the previous query in the last two questions, i.e., with, and without indices.
   [**4 pt**] Justify the speed-up with the indices.
   Hint: See `http://www.sqlite.org/draft/eqp.html` for a (rough) description of the output of `explain`.

What to turn in:

- **Code:** Submit a text file to blackboard in a file name `hw1.q1.tar.gz` (without the data such as movies.db or movies-short.txt), with all the SQL queries and commands you need to answer the queries above.. Make sure it runs: we will grade it using

  ```
  make all
  ```

(See reminder on the front page regarding how to organize your code files for submission to blackboard.)

- **Answers:** Submit a text file to blackboard with your results and responses. Call it `hw1.q1.output.txt`; create it with
  `make all` > hw1.q1.output.txt.
  and append query results for "popular movies", "Top movies" , the wall-clock times, and your justification to Q9 to this text file. (Submit answers for Q1 on separate page; hardcopy for each question should be submitted separately.)

Q2 – KD-Trees [40pts]

Problem Description: Consider the k-d-tree package, in C, at KD-Tree Package [5] (tar xvf; make). We want to augment the nearest neighbor search, so that it returns the k=10 nearest neighbors, instead of the 1 nearest neighbor that it returns now. If the tree has fewer than k=10 nodes, then return all the nodes, with a warning: "only <number> nodes found".

1. **[20 pt]** write kdtree with k=10 nearest neighbors (k should be a parameter in your code).
2. **[20 pt]** a hard copy of the output of your code, on the two included input scripts (2d-input.txt,3d-input.txt; they are included in 'kdtree.tar.gz'). In your output, keep only the lines that are the result of the query, to save paper. For your convenience, try 'make hw1'

What to turn in:

- **Code:** Submit your code to blackboard in a file `hw1.q2.tar.gz`. (Your `make` should print your responses to the two input files, i.e., 2d-input.txt and 3d-input.txt.)
  (See reminder on the front page regarding how to organize your code files for submission to blackboard.)
- **Answers:** Submit a hard copy for Q2.2. (Submit answers for Q2 on separate page; hardcopy for each question should be submitted separately.)

Q3 – Hilbert and z-order [30pts]

Problem Description: Write the code to compute the z-value and the hilbert-value of a 2-d point, as well as the inverses. The command-line syntax should be, e.g. for the 'zorder' version: zorder -n <order-of-curve> <xvalue> <yvalue>
Thus:

[5] `http://www.cs.cmu.edu/~christos/courses/826.F13/HOMEWORKS/HW1/Q2/kdtree.tar.gz`

- zorder -n 2 0 0 # should return '0'
- zorder -n 3 0 1 # should return '1'
- horder -n 2 0 0 # should return '0'

Write four programs to do these computations; hand in your source code on hard copy; and submit your source code to blackboard.

1. [**5 pt**] zorder should return the z-value of the given (x,y) point.
2. [**5 pt**] izorder should give the inverse:
   - izorder -n 5 0 # should return the 'x' and 'y' values, ie, 0 0
   - izorder -n 2 15 # should return '3 3'
3. [**5 pt**] horder should compute the hilbert order of the given point - specifically
   - horder -n 2 0 0 # should return '0'
   - horder -n 1 0 1 # should return '1'
   - horder -n 2 0 1 # should return '3'

   STRONG HINTS: use or modify the code by Jagadish [SIGMOD 90]; see the algorithm by Roseman+ [PODS 89]
4. [**5 pt**] ihorder should do the inverse - for example
   - ihorder -n 2 0 # should return '0 0'
   - ihorder -n 1 1 # should return '0 1'
   - ihorder -n 2 3 # should return '0 1'
5. [**0 pt**] Download a file from [`http://www.cs.cmu.edu/~christos/courses/826.F13/HOMEWORKS/HW1/Q3/format-q3.tar.gz`], in which there are two files: hw1-q3.output.txt, and hw1-q3.input.txt.

   Make sure your results match 'hw1-q3.output.txt' when applied on 'hw1-q3.input.txt'. (Hint: check `diff`) If there is difference, change your code so that it gives outputs that match 'hw1.q3.output.txt', as we will use this format for grading.
6. [**5 pt**] Give the results of your programs on the input file:

   [`http://www.cs.cmu.edu/~christos/courses/826.F13/HOMEWORKS/HW1/Q3/inp.txt`]

   Make sure you echo the input, so that it is clear which answer refers to which question
7. [**2 pt**] Using your izorder, plot a z-curve of order 7 (128×128 grid) and hand in the hardcopy of the plot.

8. **[3 pt]** Using your ihorder, plot a Hilbert curve of order 7, and hand in the hardcopy of the plot.

NOTES:

- It is fine to copy and/or modify code from the web, or from existing papers - but you have to make sure that your programs follow all the above specifications.
- Make sure you detect arithmetic overflows

What to turn in:

- **Code:** Submit your code to blackboard in a file `hw1.q3.tar.gz` for Q3.1,Q3.2,Q3.3,Q3.4. (your `make` should print your responses to the input file, i.e., inp.txt)
(See reminder on the front page regarding how to organize your code files for submission to blackboard.)
- **Answers:** Submit a hard copy for Q3.6,Q3.7,Q3.8 (Submit answers for Q3 on separate page; hardcopy for each question should be submitted separately.)