Homework #1: Query Processing

This homework has three questions that are intended to:

1. reinforce the simple IO cost models for joins as an example of envelope algorithmic calculations,
2. reinforce the strengths and limitations of System R; and
3. explore algorithmic and implementation ideas for a simple, but popular, query: triangles!

A more indirect goal of this homework is to make you familiar with the tools that can be used for graph and join processing. As all of you love data processing, part of the homework really is to browse through the solutions out there. Try not to get too sucked in!

Logistics Each person is expected to hand in their own copy of the homework by 11:59pm Pacific on February 15. Each student's work is expected to be their own—except where explicitly marked in question 3.

1 Shapiro: IO-based Joins (30 Points)

The point here is to do an engineering back-of-the-envelope calculation—not something overly detailed. One part of this question is to find the right level of abstraction to express the cost. In particular, your cost should be in terms of the number of IOs and we are only counting block-wise accesses.

1a: Sort Cost. Suppose you have B buffer pages and a relation R with N pages. What is the smallestq number of buffer pages B such that you can sort R in two passes using a sort-merge join? Please show your calculation and justify it algorithmically.

1b: Hash Cost. Suppose you have B buffer pages and a relation R with N pages. What is the smallest number of buffer pages B such that you can hash R into B sized buckets? Please show your calculation. *To simplify your life, you may assume a uniform distribution of values, i.e., there is no skew. If you do make this assumption, comment on how skew would complicate your answer.*

1c: Hash- versus Sort-Join Describe two scenarios: in the first scenario, a hash join is preferred to a sort-merge join and the other scenario is vice versa. Please make your argument using simple cost formulas (only count IO). *My goal is to get you to define the smallest set of parameters that gives a nontrivial result.*

2 System R & Joins (30 Points)

Each of these questions should have one-to-two sentence answers.

2a: Interesting Orders What is an interesting order? What does it mean that System R keeps interesting orders?

2b: Left-deep Plans What does "left-deep plan" mean? What are the pros and cons of keeping these such plans around?

2c: Y's Algorithm Can you give an example where Yannakakis's algorithm should be much faster than any plan that System R can explore? *Hint: think of an alpha acyclic query where pairwise joins have large intermediate results unless you go "back" in the semijoin reducers.*

3 Triangle Processing (40 Points)

The purpose of this assignment is for you to gain some experience with real systems for a simple, but challenging, task. Your goal is simple: count the number of triangles in the dataset as fast as you can.

Your running times will be posted, and the winner will be excused from the next homework (with full credit) and will get a treat. For this portion of the homework, you may form teams of no more than 2. Homework groups should have only singleton overlap with your project group.

Datasets The datasets are

- `http://snap.stanford.edu/data/higgs-twitter.html`, and
- `http://snap.stanford.edu/data/com-Amazon.html`.

These datasets are small enough to run on your laptop, but large enough to see some interesting trends.

3.1 Cardinality Bounds (10 points)

Question 3a. Suppose that you are joining three relations $R(AB), S(AC), T(BC)$ and $|R| = N^{3/2}$, $|S| = N$ and $|T| = N^{1/3}$ for some $N \geq 1$. What is maximum number of output tuples (as a function of N)? Please show your work.

Question 3b Suppose you have selection conditions on a "labeled graph" (below you'll see many graphs have labels, i.e., they are relations). Describe how you might take advantage of this information in a calculation like the one above.

3.2 Triangles for Speed!!! (30 Points)

You will compete with your classmates in two classes, called Daytona and F1; your goal is to be the fastest to list the triangles in a graph. The winners of each class will earn rewards.

(10 Points) In the Daytona class, you report only single-threaded performance for your algorithm. You must use a standard programming language (Java, Scala, C/C++, Python, etc.). You cannot use external libraries that deal with graph processing, but you can use data structures and parsing code, etc. Specifically, you will do two tasks:

- Write a nested loops version of the algorithm (no indexing)
- Write a version that computes some kind of index.

To get full credit, your algorithm with indexing must run at least 5× faster than one with a simple nested loops join in your chosen language for one of the two datasets above. You should demonstrate this with experimental results. Please run the single threaded configurations on a farm share machine (and report which machine).

(10 Points) In the F1 class, the wheels come off the bus: your F1 solution can be whatever configuration and system you want. To get full credit, your solution must be 5× faster than both your Daytona class solutions.

Systems for Graph Processing You are free to use whatever system you want for this exercise. The following are both easy to use and well documented.

- GraphLab (http://graphlab.org/)
- SNAP (http://snap.stanford.edu/snap/index.html)
- Giraph (http://giraph.apache.org/)
- Your own code or framework!

What you turn in and how you will be graded. Your code listing (in a zip) and a brief (one paragraph) writeup of your approach for each of the two problems. You should also produce a complete experimental description. You should divide your runtime into file reading, preprocessing (sorting or hashing), computation, and output time. Your grade depends on your ability to explain your results—not on your absolute speed. The remaining 10 points for this section will be for with how concisely you can explain your results.