Database Lab2

Lilin Wang & Yanjing Zhang

Fetch Data:

- We parsed input file "CA_ROADNET.txt"(http://snap.stanford.edu/data/roadNet-CA.html) and create a table ca_roadnet with 2 columns. Each row represents an edge linked with 2 nodes.
- **Method to run:** java CreateDB CA_ROADNET.txt
- **Optimization:** We execute Batches of SQL Statements to insert each row to table

Queries over Network

1. Neighbor count:
   Method to run: java NetworkAnalysis NeighbourCount <id>
   Optimization: Since most of our queries are applied to column 1. So we add an index on column 1 to optimize the queries process.
   Time cost reduced from 2s to 0.001s.

2. Reachability count:
   Method to run: java NetworkAnalysis ReachabilityCount <id>
   Optimization: We tried 2 methods on this problem.
   1) Store all data into memory, then run algorithm against memory data.
      Time cost: 2s, **Space complexity**: O(n), n=no of nodes
      Algorithm: Firstly, we transform all the network data into Adjacency table, and store the Adjacency table into memory. Then we apply breadth first search algorithm on the Adjacency table, to find reachable nodes count.
   2) Query database:
      Time cost: 181s, **Space complexity**: O(1)
      Algorithm: For each node, we fetch its neighbor nodes via querying database. Then, we apply breadth first search algorithm to find reachable nodes.

   Conclusion: The time cost of first method is much less than the second, while the memory cost of first method is much bigger than second method. Considering this is a database course lab assignment, we choose the second method to solve this problem.

3. Discover cliques
   Method to run: java NetworkAnalysis DiscoverCliques <k>
   Optimization: This problem requires us to return all sets of nodes, which leads to a huge result set. So we decide to store the results into database. So in the future we can simply get the result from database.
   (Result Table format: Cliques4 will store 4-size Cliques results, each row of Coliques4 represents a clique set. Cliques4 has 4 columns, with each column store a node.)
   Algorithm: Take k = 3 as an example:
   1-node Cliques contains each node in the graph, while 2-node Cliques contains each edge in the graph. To find 3-node Cliques, we just need to add a third node into each set of 2-node Cliques.
   For example:
   1) We have set {2,3} as a 2-node Clique.
   2) Node {2} has neighbors set A{4,5,7}
   Node {3} has neighbors set B{4,5,6}
   3) We get the intersection set C{4,5} of A & B.
   4) Then, we get 3-node Cliques set {2,3,4} and {2,3,5}
   With 3-node Cliques, we apply the same method from 1) to 4) to get 4-node Cliques. With k-1-node Cliques, we apply the same method from 1) to 4) to get k-node Cliques.
   Time cost for k=3: 600s
   Time cost for k=4: 1055s

4. Graph diameter
   Method to run: java NetworkAnalysis NetworkDiameter <k>
   Algorithm & Optimization: We found an algorithm in a research paper to find the diameter of a graph:
   1) Randomly pick a node A.
   2) Find A's reachable node B, which has the longest path to A.
   3) Find B's reachable node C, which has the longest path to B.
   4) Return the length of path between B and C, as diameter.
   Time cost: 367s