Recursion Graph-Analysis Exercises

You're tasked with performing graph analyses using a relational database system. You quickly recognize that recursion is needed to analyze arbitrary graphs using SQL queries. Fortunately, recursive SQL is available in some systems. In these exercises, you'll develop queries on a small graph with colored nodes and weighted edges. A SQL file to set up the schema and data for these exercises is downloadable here. This schema and data can be loaded as specified in the file into SQLite, MySQL, or PostgreSQL, but currently only PostgreSQL supports recursion. See our quick guide for installing and using PostgreSQL.

Schema:
Node (nID, color)
Edge (n1, n2, weight) // n1 and n2 identify nID's in table *Node*

As a guide to test the accuracy of your SQL queries, the correct query results over the provided data can be seen by pressing the button at the bottom of the page. You can return results in any order, but you may find it convenient to sort them in order to compare your results against the correct ones.

1. Find all node pairs n1,n2 that are both red and there's a path of length one or more from n1 to n2.

2. If your solution to problem 1 first generates all node pairs with a path between them and then selects the red pairs, formulate a more efficient query that incorporates "red" into the recursively-defined relation in some fashion.

3. If your solution to problem 2 incorporates the "red" condition in the recursion by constraining the start node to be red, modify your solution to constrain the end node in the recursion instead. Conversely, if your solution to problem 2 incorporates the "red" condition in the recursion by constraining the end node to be red, modify your solution to constrain the start node in the recursion instead.

4. Modify one of your previous solutions to also return the lengths of the shortest and longest paths between each pair of nodes. Your result should have four columns: the two nodeID's, the shortest path, and the longest path.

5. Modify your solution to problem 3 to also return (n1,n2,0,0) for every pair of nodes (n1≠n2) that are both red but there's no path from n1 to n2.

6. Find all node pairs n1,n2 that are both red and there's a path of length one or more from n1 to n2 that passes through exclusively red nodes.

7. Find all node pairs n1,n2 such that n1 is yellow and there is a path of length one or more from n1 to n2 that alternates yellow and blue nodes.

8. Find the highest-weight path(s) in the graph. Return start node, end node, length of path, and total weight.

9. Add one more edge to the graph: "insert into Edge values ('L','C',5);"
Your solution to problem 7 probably runs indefinitely now. Modify the query to find the highest-weight path(s) in the graph with total weight under

100. Return the number of such paths, the minimum length, maximum length, and total weight.

10. Continuing with the additional edge present, find all paths of length exactly 12. Return the number of such paths and their minimum and maximum total weights.

Hide Query Results

All query results can be returned in any order and still be correct.

1. (A,D), (A,G), (A,J), (D,G), (D,J)

2. (A,D), (A,G), (A,J), (D,G), (D,J)

3. (A,D), (A,G), (A,J), (D,G), (D,J)

4. (A,D,1,1), (A,G,3,3), (A,J,2,6), (D,G,2,2), (D,J,1,5)

5. (A,D,1,1), (A,G,3,3), (A,J,2,6), (D,A,0,0), (D,G,2,2), (D,J,1,5), (G,A,0,0), (G,D,0,0), (G,J,0,0), (J,A,0,0), (J,D,0,0), (J,G,0,0)

6. (A,D), (A,J), (D,J)

7. (E,I), (E,K), (E,L), (H,I), (H,K), (H,L), (K,L)

8. (A,L,7,19)

9. (197,28,34,99)

10. (157,32,43)