Assignment-1

CS698F Advanced Data Management

Medha Atre
Dept of Computer Science and Engineering
IIT Kanpur

Due Aug 15, 2016 23:59

Run-length encoding of a bitvector is such that given a bitvector of a kind 1111000011100000 it is represented as [1] 4 4 3 (ignoring the last 0s because they anyway indicate non-existence of values so need not be recorded). As we saw in our class, the 3D bitcube representing a graph is usually quite sparse, hence after *slicing* the bitcube in each dimension to create 2D bit matrices, we apply run-length-encoding on each row. Note that the run-length-encoding maintains the row boundaries.

Given an adjancency matrix A of a graph, A^2, i.e., multiplying the matrix by itself gives us a matrix in which a cell has 1 bit if there is a 2-length path between the corresponding graph nodes. Note that you are supposed to use Boolean matrix multiplication, so that the resultant matrix is also a matrix of 0/1 bits and no cell has value larger than 1.

You should **not** assume that the adjancency matrix is symmetric. It is an adjancency matrix of a directed graph where an edge (a, b) does **not** imply reverse edge (b, a).

(10 marks) 1. Given a list of edges of a directed graph without edge-labels or edge-weights, build the run-length-encoded adjancency matrix directly without building the intermediate uncompressed matrix. Note that your code will be tested on graphs where it is impossible to build intermediate uncompressed matrix in memory or on disk, hence your code should be able to build this compressed matrix directly from the given list of edges of a graph.

(15 marks) 2. Given such a compressed bit-matrix, write a program for multiplying it with another similarly compressed bit-matrix, without completely uncompressing any the two matrices or without completely uncompressing any row. You are expected to use methods that do not require the given compressed matrix data in completely uncompressed form because the data is so large that a commodity computer cannot hold the entire uncompressed matrix in memory.

For the input method assume that you have a simple text file where each line has an entry of type "src-node-id,dest-node-id" (without the double quotes), where src and dest node IDs are integers of less than 2^{32} values.

For the output, you should have a "switch" which decides whether to keep the resultant matrix in memory or write it out to the disk. There has to be a function through which

the resultant matrix stored on the disk can be read back in memory methodically and thus retrieved for further operations like computing A^3 matrix.

You are expected to write your code using C or C++ programming laguage and it should be compilable with G++ compiler and run on the Linux platform.

Instructions for the submission will be updated here and on the course webpage soon.