CO322: Data Structures and Algorithms

Lab 05: Simple Sparse Matrix Implementation

23rd March, 2017

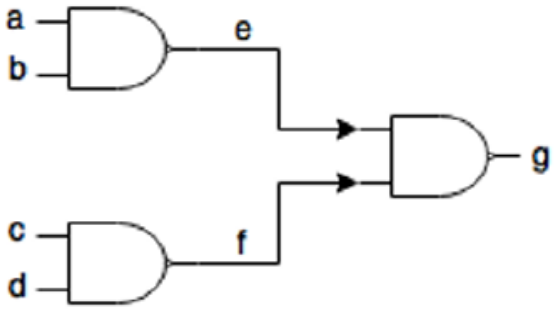
- **Aim**

The aim of this lab class is to design a simple data structure for a particular application and to embrace the notion of abstract data types.

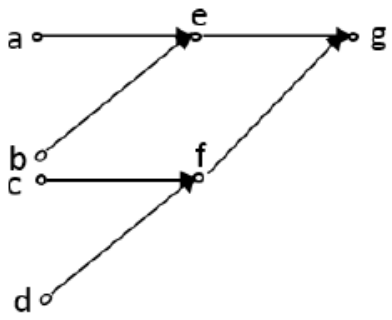
- **Background**

This week's lab class is based on a final year project done by one of the E11 students. The EDA – or Electronic Design Automation – is a multimillion dollar industry that develops software for automating electronic chip design. Complicated logic circuits which are to be implemented within the chip is first designed and analyzed using these software.

One of the main challenges in these tools is the size of the logic circuit which can have millions if not billions of gates. One of the analysis done on such a circuit is called the timing analysis. To understand timing analysis consider the *part i)* of the following diagram. The circuit contains 3 AND gates. Once a signal is given these AND gates will produce an output, however only after sometime. When one gives a signal to *a, b, c* and *d* after some ∂t the two gates will produce the correct logic signal on lines *e* and *f*. Then after another ∂t the correct output will be produced at output g.



(i)



(ii)

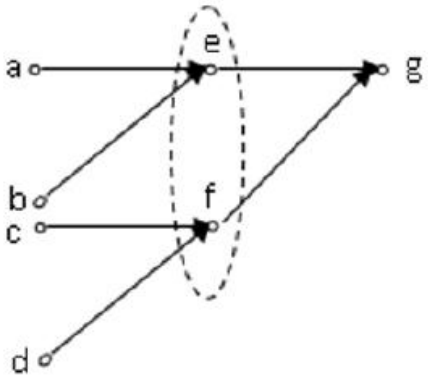
Logically, the above circuit can be displayed as *graph* as shown in part *ii)* of the above diagram. Note that for the timing analysis we do not need the type of the gate, only need to know that there exists a gate which will introduce a delay. The graph can then be represented as a matrix.

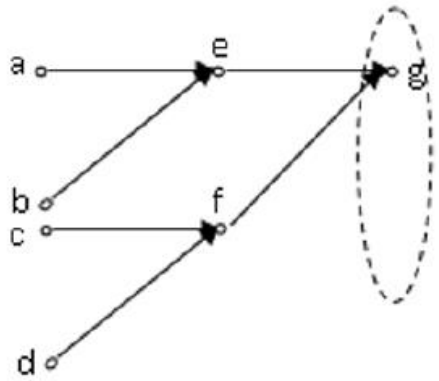
$$A = \begin{array}{c} \begin{matrix} a & b & c & d & e & f & g \end{matrix} \\ \begin{bmatrix} 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \end{array}$$

The matrix present the lines; in the above example a, b, c, d, e, f, g. One means there is some gate connecting the lines. For example there is a gate that connects a, b to e.

The obvious issue with representing this matrix is its size and the sparsity. For a typical logic circuit the rows and columns can be in billions (large size). Further most of the elements will be zero (sparsity).

Once this matrix is generated it can be multiplied with the input signal and the output will dictate where the signals would propagate. For example initially there is a signal at *a, b, c* and *d.* Once this input is multiplied with the transpose of the above matrix you see that output will propagate to *e* and *f.* If you multiply that then you see that signal will reach *g.*





$$A^T x = \begin{bmatrix} 0&0&0&0&0&0&0\\ 0&0&0&0&0&0&0\\ 0&0&0&0&0&0&0\\ 0&0&0&0&0&0&0\\ 1&1&0&0&0&0&0\\ 0&0&1&1&0&0&0\\ 0&0&0&0&1&1&0 \end{bmatrix} \begin{bmatrix} 1\\1\\1\\1\\0\\0\\0 \end{bmatrix} = \begin{bmatrix} 0\\0\\0\\0\\2\\2\\0 \end{bmatrix} \quad A^T(A^T x) = \begin{bmatrix} 0&0&0&0&0&0&0\\ 0&0&0&0&0&0&0\\ 0&0&0&0&0&0&0\\ 0&0&0&0&0&0&0\\ 1&1&0&0&0&0&0\\ 0&0&1&1&0&0&0\\ 0&0&0&0&1&1&0 \end{bmatrix} \begin{bmatrix} 0\\0\\0\\0\\2\\2\\0 \end{bmatrix} = \begin{bmatrix} 0\\0\\0\\0\\0\\0\\4 \end{bmatrix}$$

More details of this can be found here http://www.ce.pdn.ac.lk/ESCaPe/2016/papers/abs11.pdf

- **Task**

Your task is to implement the given interface for the sparse matrix using Java. You can use what you learned during CO225 for this implementation. Your interface should have

1. Function to create a new empty square matrix of given size m.
2. Function to add a connection between two given points i, j – simply make (i,j) element 1.
3. Function, given i,j that would return the stored value (either 0 or 1).

You are given some skeleton code to get started and test your code.

Note: You cannot use a two dimensional array for the matrix because you will run out of memory.

- **Submission**

Submit your code as a tar ball via Moodle. Your submission **must** contain the files **SparseMatrix.java**, **Main.class** that is provided to you and any other java classes that you need for the implementation.

Deadline: 30th March, 11.55 pm