Challenge 3
Sparse Matrix Multiplication

VerifyThis at ETAPS 2019
Organizers: Claire Dross, Carlo A. Furia,
Marieke Huisman, Rosemary Monahan, Peter Müller

6–7 April 2019, Prague, Czech Republic

How to submit solutions: send an email to `verifythis19@googlegroups.com` with your solution in attachment. Remember to clearly identify you, stating your group's name and its members.

We represent sparse matrices using the coordinate list (COO) format. In this format, the non-zero values of the matrix are stored in a sequence of triplets containing the row, the column, and the corresponding value. The sequence is sorted, first by row index and then by column index, for faster lookup. Here is an example. The matrix:

$$\begin{pmatrix} 0 & 0 & 1 & 0 \\ 5 & 8 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 3 & 0 & 0 \end{pmatrix}$$

is encoded into the following sequence (using row and column indexes that start from 1):

$$[(1,3,1),(2,1,5),(2,2,8),(4,2,3)]$$

We now consider an algorithm which computes the multiplication of a vector of values (encoded as a sequence) with a sparse matrix. It iterates over the values present inside the matrix, multiplies each of them by the appropriate element in the input vector, and stores the result at the appropriate position in the output vector. Here is the algorithm in pseudo-code, which multiplies a sparse matrix m with an input vector x and stores it in an output vector y:

```
y <- (0, ..., 0)
for every element (r, c, v) of m do
   y (c) <- y (c) + x (r) ∗ v
done
```

Tasks

Implementation tasks.

1. Implement the algorithm to multiply a vector with a sparse matrix in your language of choice.
2. We want to execute this algorithm in parallel, so that each computation is done by a different process/thread/task. Add the necessary synchronization steps in your sequential program, using the synchronisation feature of your choice (lock, atomic block, ...).

   You can choose how to allocate work to processes. For example:

   - each process computes exactly one iteration of the for loop,
   - there is a fixed number of processes, each taking an equal share of the total number of for loop iterations,
   - work is assigned to processes dynamically (for example using a work stealing algorithm).

Verification tasks.

1. Verify that the sequential muplitplication algorithm indeed performs standard matrix multiplication (that is, it computes $y_i = \sum_k x_k \times m_{k,i}$).
2. Verify that the concurrent algorithm does not exhibit concurrency issues (data-races, deadlocks, ...).
3. Verify that the concurrent algorithm still performs the same computation as the sequential algorithm. If time permits, you can also experiment with different work allocation policies and verify that they all behave correctly.