APPLIED LINEAR ALGEBRA MTH343: PREPARATORY PROGRAMMING TASK

PANAYOT S. VASSILEVSKI

Abstract. This is a recommended programming task. It deals with reading a matrix from a file, storing it in three possible formats, and performing matrix times a vector operations.

1. Efficient way of storing sparse matrices

Now we describe an efficient way to store matrices, taking advantage that in a row, we may have only few non-zero entries. We describe next the CSR format (CSR stands for "compressed sparse row"). The CSR format is a popular way to store sparse matrices. For an $n \times m$ sparse matrix $A = (a_{ij})$, the CSR format exploits two one-dimensional integer arrays I and J and if the matrix is not Boolean (as the relation tables discussed in class) a real array "Data" is needed in addition to store the values/the actual entries a_{ij} of A.

Let A have in row i, $m_i \geq 1$ non–zero entries at positions $(i, j_1^{(i)}),\ \ldots,\ (i, j_{m_i}^{(i)})$.

The one-dimensional array I has length $n + 1$. With $I[0] = 0$, we set

$$I[i] = I[i-1] + m_i \text{ for } i \geq 1.$$

The array J has length $I[n]$, which is the total number of all nonzero entries of A. Similarly, the data array, Data, has the same length $I[n]$.

For each row $i = 1,\ \ldots,\ n$ of A, we list consecutively in the one–dimensional array J the indices $j_s^{(i)}$, $s = 1,\ \ldots,\ m_i$ starting at position $I[i-1]$ till position $I[i] - 1$, that is

$$J[I[i-1] + s - 1] = j_s^{(i)}, \text{ for } s = 1,\ \ldots,\ m_i.$$

The data array is filled-in similarly, i.e., we let

$$\text{Data}[I[i-1] + s - 1] = a_{i,\, j_s^{(i)}} \text{ for } s = 1,\ \ldots,\ m_i.$$

Having sparse matrices stored in CSR format in practice it is useful to have algorithms that implement matrix operations such as A^T, matrix–matrix multiply $C = AB$. I.e., if A is stored in CSR format we need to store A^T in CSR format using only $\mathcal{O}(n)$ operations. Similarly, if the sparse matrices A and B are represented in CSR format with $\mathcal{O}(n)$ non–zero entries, we want to find an algorithm that computes and stores C in CSR format for $\mathcal{O}(n)$ storage and operations. All this is feasible for matrices corresponding to sparse relation tables.

Date: March 6, 2018.

2. Creating input and sample tests

- Read a matrix from a file. The file will have for each row two integers, i and j and a real number, which will be the (i,j)-th entry a_{ij} of the matrix A. The file may contain per row only the two integers i and j, then we assume that $a_{ij} = 1$. In both cases, all other entries of A that are not read from the file are assumed zero.
- Choose a way to store the matrix A. Possible options:
  (1) Use two dimensional array $A[.,.]$. Initialize all entries of A with zeros. Then, for each pair of indices (i,j) that you have read, you set $A[i,j] = a_{ij}$ (the value which you have read, or if it is missing in the file, you set it to 1).
  (2) A more efficient way to store A, is to use two integer arrays, $I[.]$ and $J[.]$ and one real array $data[.]$ all of size nnz, the number of nonzero entries of A. When you read the triplet $i, j, value$ from the file at step $k = 1, 2, \ldots, nnz$, you set $I[k] = i$, $J[k] = j$ and $data[k] = value$. This is sometime referred to as the $(i,j)j$ (or adjacency) matrix format.
  (3) Alternatively, you can use the more efficient way of storing A (the CSR format described above) as triplet of three one-dimensional arrays; two integer arrays $I[.]$, $J[.]$ and one array of reals $data[.]$.
  (4) Use two one-dimensional arrays, $v[.]$ and $w[.]$, of corresponding length. Initialize $w[.]$ with some values (choose random values or unit values).
  (5) Compute the product $v = Aw$.
  (6) Compare the timings for the various formats of A (two-dimensional, (i,j), or the CSR) for fairly large matrices. Document your observation(s) with some conclusions.
  (7) A simple file of any length $n > 1$ is (assuming that indices run from 0)

$$
\begin{array}{llr}
0 & 0 & 2.0 \\
0 & 1 & -1.0 \\
1 & 0 & -1.0 \\
1 & 1 & 2.0 \\
1 & 2 & -1.0 \\
\vdots & \vdots & \vdots \\
i & i-1 & -1.0 \\
i & i & 2.0 \\
i & i+1 & -1.0 \\
\vdots & \vdots & \vdots \\
n & n-1 & -1.0 \\
n & n & 2.0
\end{array}
$$

  (8) A large set of matrix files is found at
  http://www.cise.ufl.edu/research/sparse/matrices/list_by_id.html

Professor, Fariborz Maseeh Department of Mathematics and Statistics, Portland State University, Portland OR 97207
E-mail address: panayot@pdx.edu