Additional Notes on Sheet 7 - Execise 3
Compressed Storage Schemes

Michael Lieb

15.01.2014

1 Compressed Sparse Column

- interprete matrix as a sequence of columns
- store the number of non-zeros (nnz) per column in index vector JA (offset of columns within data sequence)
- store tupels value/row in data stream

$$\left(\begin{array}{c|c|c|c|c} 1 & 0 & 0 & 2 & 0 \\ 3 & 4 & 0 & 5 & 0 \\ 6 & 0 & 7 & 8 & 9 \\ 0 & 0 & 10 & 11 & 0 \\ 0 & 0 & 0 & 2 & 12 \end{array}\right)$$

$$\begin{array}{lccc|c|cc|cccc|cc|c} AA: & 1 & 3 & 6 & 4 & 7 & 10 & 2 & 5 & 8 & 11 & 9 & 12 & \\ IA: & 1 & 2 & 3 & 2 & 3 & 4 & 1 & 2 & 3 & 4 & 3 & 5 & \perp \\ JA: & 1 & 4 & 5 & 7 & 11 & 13 & & & & & & & \end{array}$$

- storage complexity $\mathcal{O}(|JA| + 2nnz) = \mathcal{O}(n + 1 + 2nnz)$
- Note: Within one column order of entries is not specified
- naming conventions:

  IA hold row indices i of A

  JA hold column indices j of A

  AA hold matrix entries a_{ij} of A
- always a standard question in exams ☺
- Improvement by diagonal extraction → See CSR improvement → Script!

1.1 Difference CSR vs. CSC

Think about parallelization where data is scattered row-wise or column-wise. It can have an effect on the communication between processes.
E.g.: matrix-matrix-multiplication: $C = AB$

$$\rightarrow c_{ij} = \text{row}_i(A) \cdot \text{col}_j(B)$$
$$\rightarrow \text{store A in CSR and B in CSC}$$

1.2 Matrix-vector-multiplication

In Algorithm 2, j is the column counter. The matrix is a sequence of columns of length n. i is the row counter.

Algorithm 1 Calculate $c = AB$

for $j = 1 : n$ **do**
  for $i = JA(j) : JA(j+1) - 1$ **do**
    $c[IA(i)] += \underbrace{AA_i}_{a_{IA(i)_{1j}}} b_i$
  end for
end for

2 Jagged Diagonal Front

- Reorder matrix rows such that: $\text{nnz}(\text{row}_i) \geq \text{nnz}(\text{row}_{i+1})$.
- store length of the offsets of jagged diagonals (JDIAG)
- interpret diagonal as sequence of row entries

$$\begin{pmatrix} 1 & 0 & 2 & 0 & 0 \\ 3 & 4 & 0 & 5 & 0 \\ & 6 & 7 & 0 & 8 \\ & & 9 & 10 & 0 \\ & & 0 & 11 & 12 \end{pmatrix} \xrightarrow{reorder} \begin{pmatrix} 3 & 4 & 0 & 5 & \\ & 6 & 7 & 0 & 8 \\ 1 & 0 & 2 & 0 & 0 \\ & & 9 & 10 & \\ & & & 11 & 12 \end{pmatrix} \begin{matrix} \} length:3 \\ \\ \} length:2 \end{matrix}$$

$AA:$	3	6	1	9	11	4	7	2	10	12	5	8	
$JDIAG:$	1	2	1	3	4	2	3	5	4	5	4	5	$\perp$
$IDIAG:$	1	6	11	13									

IDIAG is the pointer to the beginning of the j^{th} diagonal.

2.1 MV-Multiplication

Traverse the matrix diagonal-wise and compute AB diagonal-wise.

Algorithm 2 Calculate $c = AB$

```
for d = 1 : #diags do
  for a = IDIAG(d) : IDIAG(d+1) − 1 do
    c[a − IDIAG(d) + 1] += AA(a) · b(IDIAG(a))
  end for
end for
```

3 Converting CSR to CSC

Given

$$C = A \cdot B, \qquad A, B, C \in \mathbb{R}^{n \times n}.$$

Observation:

- best suited to store A in CSR and B in CSC as

$$c_{ij} = \mathrm{row}_i(A) \cdot \mathrm{col}_j(B)$$

- sometimes matrices should be available both in CSR and CSC

i) Convert and use coordinate form storage scheme as intermediate format (data redundancy)
CSR ⇆ Coordinate Form ⇆ CSC

ii)
- hold matrix as both CSR and CSC
- use pointers to avoid data redundancy

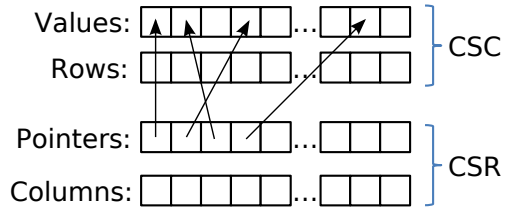

Figure 1: Transfer from CSR to CSC.