Matrix Data Structures

Shusen Wang

Stevens Institute of Technology

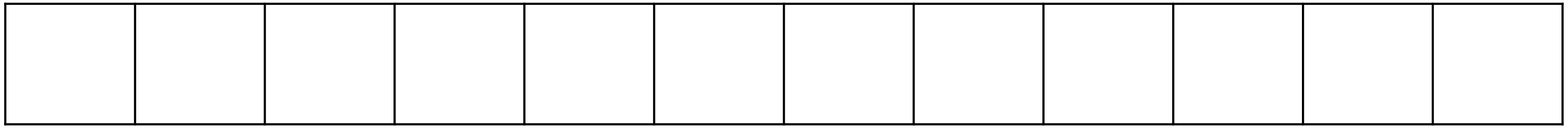
http://wangshusen.github.io/

Dense Matrix Data Structures

Dense Matrix Data Structures

- **Dense matrix:** most of the elements are non-zero.
- Dense matrix can be stored in a fixed-size array.

Array:



Dense Matrix Data Structures

Row-Major Order

$$\mathbf{A} = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \\ a_{41} & a_{42} & a_{43} \end{bmatrix}$$

Column-Major Order

$$\mathbf{A} = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \\ a_{41} & a_{42} & a_{43} \end{bmatrix}$$

Array:

Dense Matrix Data Structures

Row-Major Order

$$\mathbf{A} = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \\ a_{41} & a_{42} & a_{43} \end{bmatrix}$$

Column-Major Order

$$\mathbf{A} = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \\ a_{41} & a_{42} & a_{43} \end{bmatrix}$$

Array:

a_{11}	a_{12}	a_{13}									

Dense Matrix Data Structures

Row-Major Order

$$\mathbf{A} = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \\ a_{41} & a_{42} & a_{43} \end{bmatrix}$$

Column-Major Order

$$\mathbf{A} = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \\ a_{41} & a_{42} & a_{43} \end{bmatrix}$$

Array:

a_{11}	a_{12}	a_{13}	a_{21}	a_{22}	a_{23}						

Dense Matrix Data Structures

Row-Major Order

$$\mathbf{A} = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \\ a_{41} & a_{42} & a_{43} \end{bmatrix}$$

Column-Major Order

$$\mathbf{A} = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \\ a_{41} & a_{42} & a_{43} \end{bmatrix}$$

Array:

a_{11}	a_{12}	a_{13}	a_{21}	a_{22}	a_{23}	a_{31}	a_{32}	a_{33}			

Dense Matrix Data Structures

Row-Major Order

$$\mathbf{A} = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \\ a_{41} & a_{42} & a_{43} \end{bmatrix}$$

Column-Major Order

$$\mathbf{A} = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \\ a_{41} & a_{42} & a_{43} \end{bmatrix}$$

Array:

a_{11}	a_{12}	a_{13}	a_{21}	a_{22}	a_{23}	a_{31}	a_{32}	a_{33}	a_{41}	a_{42}	a_{43}

Dense Matrix Data Structures

Row-Major Order

$$\mathbf{A} = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \\ a_{41} & a_{42} & a_{43} \end{bmatrix}$$

Column-Major Order

$$\mathbf{A} = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \\ a_{41} & a_{42} & a_{43} \end{bmatrix}$$

Array:

a_{11}	a_{21}	a_{31}	a_{41}								

Dense Matrix Data Structures

Row-Major Order

$$\mathbf{A} = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \\ a_{41} & a_{42} & a_{43} \end{bmatrix}$$

Column-Major Order

$$\mathbf{A} = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \\ a_{41} & a_{42} & a_{43} \end{bmatrix}$$

Array:

a_{11}	a_{21}	a_{31}	a_{41}	a_{12}	a_{22}	a_{32}	a_{42}				

Dense Matrix Data Structures

Row-Major Order

$$\mathbf{A} = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \\ a_{41} & a_{42} & a_{43} \end{bmatrix}$$

Column-Major Order

$$\mathbf{A} = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \\ a_{41} & a_{42} & a_{43} \end{bmatrix}$$

Array:

a_{11}	a_{21}	a_{31}	a_{41}	a_{12}	a_{22}	a_{32}	a_{42}	a_{13}	a_{23}	a_{33}	a_{43}

Why does layout matter?

Row-Major Order

$$\mathbf{A} = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \\ a_{41} & a_{42} & a_{43} \end{bmatrix}$$

Column-Major Order

$$\mathbf{A} = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \\ a_{41} & a_{42} & a_{43} \end{bmatrix}$$

Array:

a_{11}	a_{12}	a_{13}	a_{21}	a_{22}	a_{23}	a_{31}	a_{32}	a_{33}	a_{41}	a_{42}	a_{43}

Why does layout matter?

Row-Major Order

$$\mathbf{A} = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \\ a_{41} & a_{42} & a_{43} \end{bmatrix}$$

Traversing a row is fast.

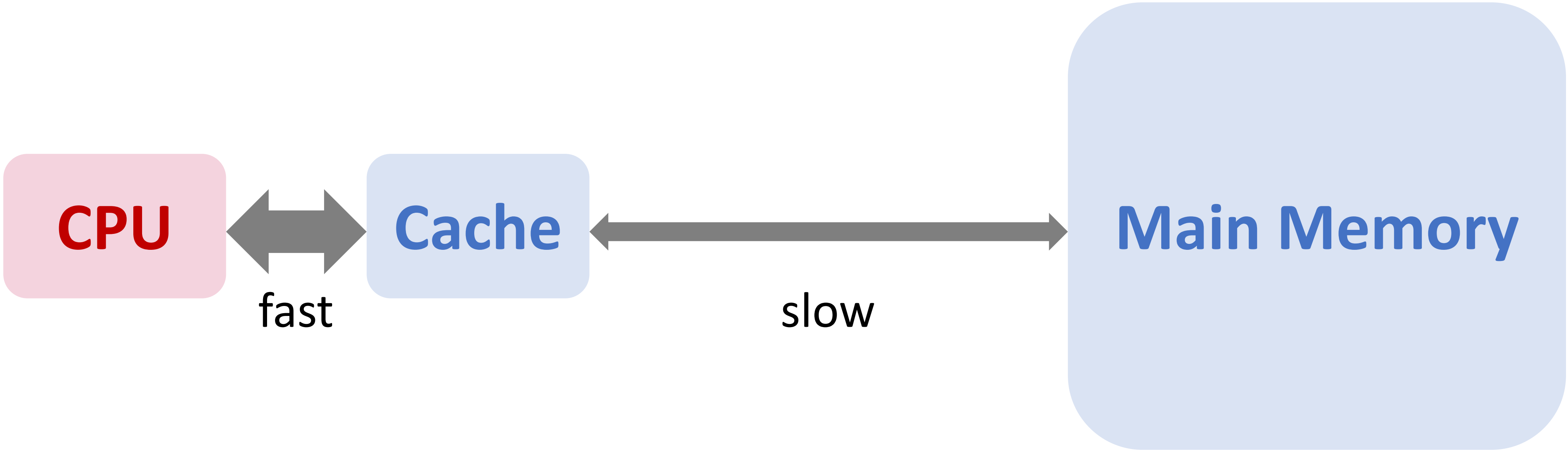
Column-Major Order

$$\mathbf{A} = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \\ a_{41} & a_{42} & a_{43} \end{bmatrix}$$

Array:

a_{11}	a_{12}	a_{13}	a_{21}	a_{22}	a_{23}	a_{31}	a_{32}	a_{33}	a_{41}	a_{42}	a_{43}

Why does layout matter?



Traversing a row is fast.

Array:

a_{11}	a_{12}	a_{13}	a_{21}	a_{22}	a_{23}	a_{31}	a_{32}	a_{33}	a_{41}	a_{42}	a_{43}

Why does layout matter?

Row-Major Order

$$\mathbf{A} = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \\ a_{41} & a_{42} & a_{43} \end{bmatrix}$$

Traversing a column is slow.

Column-Major Order

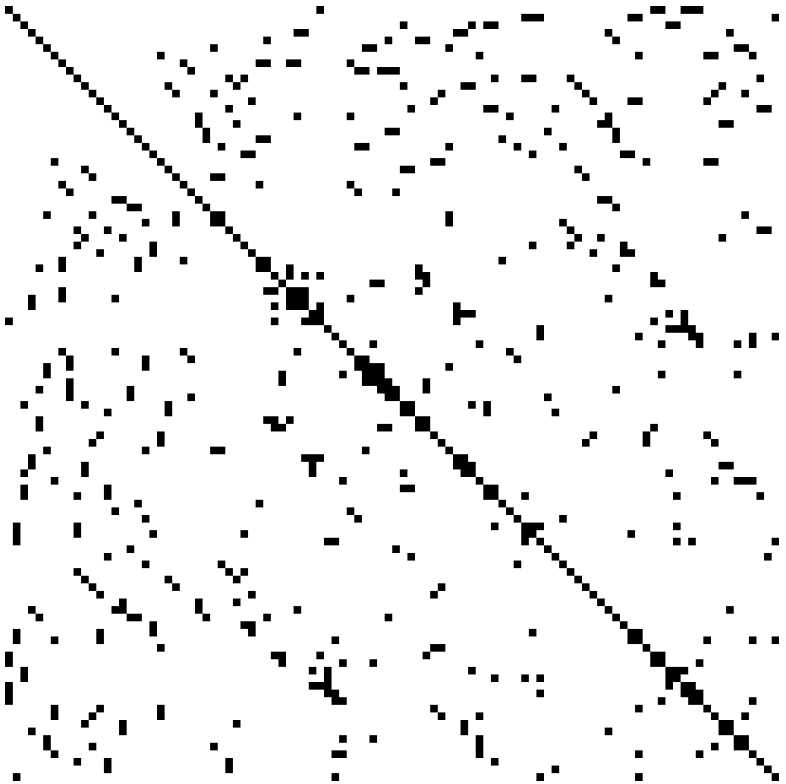
$$\mathbf{A} = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \\ a_{41} & a_{42} & a_{43} \end{bmatrix}$$

Array:

a_{11}	a_{12}	a_{13}	a_{21}	a_{22}	a_{23}	a_{31}	a_{32}	a_{33}	a_{41}	a_{42}	a_{43}

Sparse Matrix Data Structures

Sparse Matrices



Example of sparse matrix

- **Sparse matrix:** A matrix in which most elements are zeros.
- **Question:** How to store a sparse matrix?
- **Bad solution:** As a dense matrix.
- **Good solution:** Storing only the nonzero elements and their indices.

Sparse Matrices

- Compressed Sparse Row (**CSR**).
- Compressed Sparse Column (**CSC**).
- There are different schemes. This lecture introduces only one representation.

Compressed Sparse Row (CSR)

$$\mathbf{A} = \begin{bmatrix} 10 & 5.5 & 0 & 0 & 0 & 0 \\ 0 & 0 & 30 & 40 & 0 & 7.1 \\ 0 & 2 & 0 & 0 & 70 & 0 \\ 0 & 0 & 0 & 9.2 & 0 & 26 \end{bmatrix}$$

Compressed Sparse Row (CSR)

$$\mathbf{A} = \begin{bmatrix} 10 & 5.5 & 0 & 0 & 0 & 0 \\ 0 & 0 & 30 & 40 & 0 & 7.1 \\ 0 & 2 & 0 & 0 & 70 & 0 \\ 0 & 0 & 0 & 9.2 & 0 & 26 \end{bmatrix}$$

Value:

10	5.5	30	40	7.1	2	70	9.2	26

Compressed Sparse Row (CSR)

$$\mathbf{A} = \begin{bmatrix} 10 & 5.5 & 0 & 0 & 0 & 0 \\ 0 & 0 & 30 & 40 & 0 & 7.1 \\ 0 & 2 & 0 & 0 & 70 & 0 \\ 0 & 0 & 0 & 9.2 & 0 & 26 \end{bmatrix}$$

Value:

10	5.5	30	40	7.1	2	70	9.2	26

Compressed Sparse Row (CSR)

$$\mathbf{A} = \begin{bmatrix} 10 & 5.5 & 0 & 0 & 0 & 0 \\ 0 & 0 & 30 & 40 & 0 & 7.1 \\ 0 & 2 & 0 & 0 & 70 & 0 \\ 0 & 0 & 0 & 9.2 & 0 & 26 \end{bmatrix}$$

Value:

$$\underbrace{\boxed{10}\,\boxed{5.5}\,\boxed{30}\,\boxed{40}\,\boxed{7.1}\,\boxed{2}\,\boxed{70}\,\boxed{9.2}\,\boxed{26}}_{\mathrm{nnz}(\mathbf{A})}$$

Compressed Sparse Row (CSR)

$$\mathbf{A} = \begin{bmatrix} 10 & 5.5 & 0 & 0 & 0 & 0 \\ 0 & 0 & 30 & 40 & 0 & 7.1 \\ 0 & 2 & 0 & 0 & 70 & 0 \\ 0 & 0 & 0 & 9.2 & 0 & 26 \end{bmatrix}$$

Value:	10	5.5	30	40	7.1	2	70	9.2	26
Row Index:	1	1	2	2	2	3	3	4	4
Col Index:	1	2	3	4	6	2	5	4	6

Compressed Sparse Row (CSR)

$$A = \begin{bmatrix} 10 & 5.5 & 0 & 0 & 0 & 0 \\ 0 & 0 & 30 & 40 & 0 & 7.1 \\ 0 & 2 & 0 & 0 & \boxed{70} & 0 \\ 0 & 0 & 0 & 9.2 & 0 & 26 \end{bmatrix}$$

Value:	10	5.5	30	40	7.1	2	70	9.2	26
Row Index:	1	1	2	2	2	3	3	4	4
Col Index:	1	2	3	4	6	2	5	4	6

How to slice a row?

CSR Matrix: $A = \begin{bmatrix} 10 & 5.5 & 0 & 0 & 0 & 0 \\ 0 & 0 & 30 & 40 & 0 & 7.1 \\ 0 & 2 & 0 & 0 & 70 & 0 \\ 0 & 0 & 0 & 9.2 & 0 & 26 \end{bmatrix}$

Value:	10	5.5	30	40	7.1	2	70	9.2	26
Row Index:	1	1	2	2	2	3	3	4	4
Col Index:	1	2	3	4	6	2	5	4	6

How to slice a row?

CSR Matrix: $\mathbf{A} = \begin{bmatrix} 10 & 5.5 & 0 & 0 & 0 & 0 \\ 0 & 0 & 30 & 40 & 0 & 7.1 \\ 0 & 2 & 0 & 0 & 70 & 0 \\ 0 & 0 & 0 & 9.2 & 0 & 26 \end{bmatrix}$

Value:	10	5.5	30	40	7.1	2	70	9.2	26
Row Index:	1	1	2	2	2	3	3	4	4
Col Index:	1	2	3	4	6	2	5	4	6

How to slice a row?

CSR Matrix: $\mathbf{A} = \begin{bmatrix} 10 & 5.5 & 0 & 0 & 0 & 0 \\ 0 & 0 & 30 & 40 & 0 & 7.1 \\ 0 & 2 & 0 & 0 & 70 & 0 \\ 0 & 0 & 0 & 9.2 & 0 & 26 \end{bmatrix}$

Traversing a row is fast.

Value:	10	5.5	30	40	7.1	2	70	9.2	26
Row Index:	1	1	2	2	2	3	3	4	4
Col Index:	1	2	3	4	6	2	5	4	6

How to slice a column?

CSR Matrix: $\mathbf{A} = \begin{bmatrix} 10 & 5.5 & 0 & 0 & 0 & 0 \\ 0 & 0 & 30 & 40 & 0 & 7.1 \\ 0 & 2 & 0 & 0 & 70 & 0 \\ 0 & 0 & 0 & 9.2 & 0 & 26 \end{bmatrix}$

Value:	10	5.5	30	40	7.1	2	70	9.2	26
Row Index:	1	1	2	2	2	3	3	4	4
Col Index:	1	2	3	4	6	2	5	4	6

How to slice a column?

CSR Matrix: $\mathbf{A} = \begin{bmatrix} 10 & 5.5 & 0 & 0 & 0 & 0 \\ 0 & 0 & 30 & 40 & 0 & 7.1 \\ 0 & 2 & 0 & 0 & 70 & 0 \\ 0 & 0 & 0 & 9.2 & 0 & 26 \end{bmatrix}$

Value:	10	5.5	30	40	7.1	2	70	9.2	26
Row Index:	1	1	2	2	2	3	3	4	4
Col Index:	1	2	3	4	6	2	5	4	6

How to slice a column?

CSR Matrix: $\mathbf{A} = \begin{bmatrix} 10 & 5.5 & 0 & 0 & 0 & 0 \\ 0 & 0 & 30 & 40 & 0 & 7.1 \\ 0 & 2 & 0 & 0 & 70 & 0 \\ 0 & 0 & 0 & 9.2 & 0 & 26 \end{bmatrix}$

Traversing a column is slow.

Value:	10	5.5	30	40	7.1	2	70	9.2	26
Row Index:	1	1	2	2	2	3	3	4	4
Col Index:	1	2	3	4	6	2	5	4	6

Compressed Sparse Column (CSC)

$$\mathbf{A} = \begin{bmatrix} 10 & 5.5 & 0 & 0 & 0 & 0 \\ 0 & 0 & 30 & 40 & 0 & 7.1 \\ 0 & 2 & 0 & 0 & 70 & 0 \\ 0 & 0 & 0 & 9.2 & 0 & 26 \end{bmatrix}$$

Compressed Sparse Column (CSC)

$$\mathbf{A} = \begin{bmatrix} 10 & 5.5 & 0 & 0 & 0 & 0 \\ 0 & 0 & 30 & 40 & 0 & 7.1 \\ 0 & 2 & 0 & 0 & 70 & 0 \\ 0 & 0 & 0 & 9.2 & 0 & 26 \end{bmatrix}$$

Value:

10	5.5	2	30	40	9.2	70	7.1	26

Compressed Sparse Column (CSC)

$$\mathbf{A} = \begin{bmatrix} \boxed{10} & \boxed{5.5} & 0 & 0 & 0 & 0 \\ 0 & 0 & \boxed{30} & \boxed{40} & 0 & \boxed{7.1} \\ 0 & \boxed{2} & 0 & 0 & \boxed{70} & 0 \\ 0 & 0 & 0 & \boxed{9.2} & 0 & \boxed{26} \end{bmatrix}$$

Value:

10	5.5	2	30	40	9.2	70	7.1	26

Compressed Sparse Column (CSC)

$$\mathbf{A} = \begin{bmatrix} 10 & 5.5 & 0 & 0 & 0 & 0 \\ 0 & 0 & 30 & 40 & 0 & 7.1 \\ 0 & 2 & 0 & 0 & \boxed{70} & 0 \\ 0 & 0 & 0 & 9.2 & 0 & 26 \end{bmatrix}$$

Value:	10	5.5	2	30	40	9.2	70	7.1	26
Row Index:	1	1	3	2	2	4	3	2	4
Col Index:	1	2	2	3	4	4	5	6	6

Compressed Sparse Column (CSC)

$$\mathbf{A} = \begin{bmatrix} 10 & 5.5 & 0 & 0 & 0 & 0 \\ 0 & 0 & 30 & 40 & 0 & 7.1 \\ 0 & 2 & 0 & 0 & 70 & 0 \\ 0 & 0 & 0 & 9.2 & 0 & 26 \end{bmatrix}$$

Value:	10	5.5	2	30	40	9.2	70	7.1	26
Row Index:	1	1	3	2	2	4	3	2	4
Col Index:	1	2	2	3	4	4	5	6	6

Memory Cost

- 8 Bytes for a double-precision floating-point number (a value).
- 4 Bytes for a long integer (an index).

Memory Cost

- 8 Bytes for a double-precision floating-point number (a value).
- 4 Bytes for a long integer (an index).
- Memory cost (Bytes) of CSR or CSC:

$$(8+4+4) \cdot \mathrm{nnz}(\mathbf{A}) = 16 \cdot \mathrm{nnz}(\mathbf{A}).$$

- Memory cost (Bytes) of an $m \times n$ dense matrix:

$$8mn.$$

- If over 50% elements are zeros, then CSR and CSC save memory.

Questions

From CSR to dense matrix

Value:	9	8.2	29	2	3.1	5	2	1.5	7	10
Row Index:	1	1	1	1	2	2	3	4	4	4
Col Index:	2	4	5	6	1	2	2	3	4	6

Convert the CSR matrix to dense matrix:

$$\mathbf{A} = \begin{bmatrix} ? & ? & ? & ? & ? & ? \\ ? & ? & ? & ? & ? & ? \\ ? & ? & ? & ? & ? & ? \\ ? & ? & ? & ? & ? & ? \end{bmatrix}$$

Matrix L1 Norm

Value:	3	2	1	7	4	3	5	1	2
Row Index:	1	1	2	2	2	3	3	4	4
Col Index:	1	2	3	4	6	2	5	4	6

- The 4×6 matrix $\mathbf{A}$ is stored as CSR matrix (in the above).
- **Question:** What is the ℓ_1-norm of $\mathbf{A}$?
- **Hint:** The matrix ℓ_1-norm is $\boxed{\left|\left|\mathbf{A}\right|\right|_1 = \sum_{i=1}^{4} \sum_{j=1}^{6} \left|a_{ij}\right|.}$

Thank You!

Solution

From CSR to dense matrix

Value:	9	8.2	29	2	3.1	5	2	1.5	7	10
Row Index:	1	1	1	1	2	2	3	4	4	4
Col Index:	2	4	5	6	1	2	2	3	4	6

Convert the CSR matrix to dense matrix:

$$\mathbf{A} = \begin{bmatrix} 0 & 9 & 0 & 8.2 & 29 & 2 \\ 3.1 & 5 & 0 & 0 & 0 & 0 \\ 0 & 2 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1.5 & 7 & 0 & 10 \end{bmatrix}$$