Arrays and Structures

The Sparse Matrix Data Type

Joseph Chuang-Chieh Lin (林莊傑)

Department of Computer Science & Engineering,
National Taiwan Ocean University

Fall 2025

Outline

1. The Sparse Matrix ADT

Outline

An $m \times n$ Matrix

- An $m \times n$ matrix with m rows and n columns.

$$\begin{bmatrix} -27 & 3 & 4 \\ 6 & 82 & -2 \\ 109 & -64 & 11 \\ 12 & 8 & 9 \\ 48 & 27 & 47 \end{bmatrix}$$

An $m \times n$ Matrix

- An $m \times n$ matrix with m rows and n columns.

$$\begin{bmatrix} -27 & 3 & 4 \\ 6 & 82 & -2 \\ 109 & -64 & 11 \\ 12 & 8 & 9 \\ 48 & 27 & 47 \end{bmatrix}$$

- Totally 15 elements with 15 nonzero entries.

An $m \times n$ Matrix

- An $m \times n$ matrix with m rows and n columns.

$$\begin{bmatrix} 15 & 0 & 0 & 22 & 0 & -15 \\ 0 & 11 & 3 & 0 & 0 & 0 \\ 0 & 0 & 0 & -6 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 91 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 28 & 0 & 0 & 0 \end{bmatrix}$$

An $m \times n$ Matrix

- An $m \times n$ matrix with m rows and n columns.

$$\begin{bmatrix} 15 & 0 & 0 & 22 & 0 & -15 \\ 0 & 11 & 3 & 0 & 0 & 0 \\ 0 & 0 & 0 & -6 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 91 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 28 & 0 & 0 & 0 \end{bmatrix}$$

  - Totally 36 elements with only 8 nonzero entries.

An $m \times n$ Matrix

- An $m \times n$ matrix with m rows and n columns.

$$\begin{bmatrix} 15 & 0 & 0 & 22 & 0 & -15 \\ 0 & 11 & 3 & 0 & 0 & 0 \\ 0 & 0 & 0 & -6 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 91 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 28 & 0 & 0 & 0 \end{bmatrix}$$

  - Totally 36 elements with only 8 nonzero entries.
- How to efficiently store this sparse matrix?

Remarks & The Idea

- The standard representation of a matrix is a two-dimensional array defined as `a[MAX_ROWS][MAX_COLS]`.
  - We can locate quickly any element by writing `a[i][j]`.

Remarks & The Idea

- The standard representation of a matrix is a two-dimensional array defined as `a[MAX_ROWS][MAX_COLS]`.
    - We can locate quickly any element by writing `a[i][j]`.
- Let's consider alternative forms of representation for the matrix.

Remarks & The Idea

- The standard representation of a matrix is a two-dimensional array defined as `a[MAX_ROWS][MAX_COLS]`.
  - We can locate quickly any element by writing `a[i][j]`.
- Let's consider alternative forms of representation for the matrix.
  - Store only nonzero elements of the matrix.

The Sparse Matrix ADT (1/2)

$$\begin{bmatrix} 15 & 0 & 0 & 22 & 0 & -15 \\ 0 & 11 & 3 & 0 & 0 & 0 \\ 0 & 0 & 0 & -6 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 91 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 28 & 0 & 0 & 0 \end{bmatrix}$$

▷

A[0]	6	6	8

index, # rows, # cols, # nonzeros

	Row	Col	Value
A[0]	6	6	8
A[1]	0	0	15
A[2]	0	3	22
A[3]	0	5	−15
A[4]	1	1	11
A[5]	1	2	3
A[6]	2	3	−6
A[7]	4	0	−91
A[8]	5	2	−28

The Sparse Matrix ADT (2/2)

```c
#define MAX_TERMS 101
// maximum number of items + 1

typedef struct {
    int col;
    int row;
    int value;
} term;

term a[MAX_TERMS];
```

	Row	Col	Value
A[0]	6	6	8
A[1]	0	0	15
A[2]	0	3	22
A[3]	0	5	-15
A[4]	1	1	11
A[5]	1	2	3
A[6]	2	3	-6
A[7]	4	0	-91
A[8]	5	2	-28

Why use sparse matrix ADT?

- Space efficient.

Why use sparse matrix ADT?

- Space efficient.
- Next, we will discuss matrix transpose and matrix multiplication using sparse matrix ADT.

Discussions