Data Structures and Algorithms (12)

Instructor: Ming Zhang
Textbook Authors: Ming Zhang, Tengjiao Wang and Haiyan Zhao
Higher Education Press, 2008.6 (the "Eleventh Five-Year" national planning textbook)

https://courses.edx.org/courses/PekingX/04830050x/2T2014/

Chapter 12 Advanced Data Structure

- 12.1 Multidimensional array
  - 12.1.1 Basic Concepts
  - 12.1.2 Structure of Array
  - 12.1.3 Storage of Array
  - 12.1.4 Declaration of Array
  - 12.1.5 Special Matrices Implemented by Arrays
  - 12.1.6 Sparse Matrix
- 12.2 Generalized List
- 12.3 Storage management
- 12.4 Trie
- 12.5 Improved BST

Basic Concepts

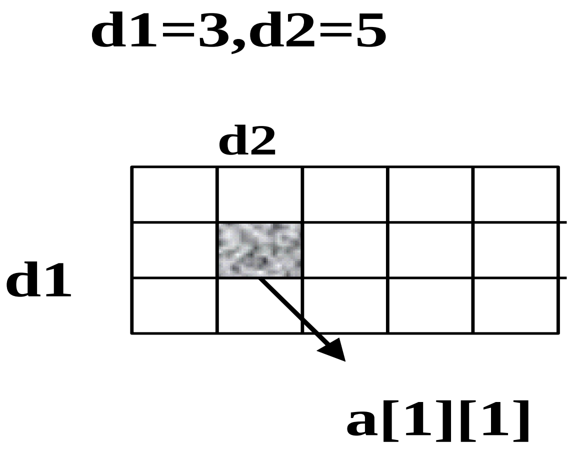
- Array is an ordered sequence with fixed number of elements and type.
- The size and type of static array must be specified at compile time
- Dynamic array is allocated memory at runtime

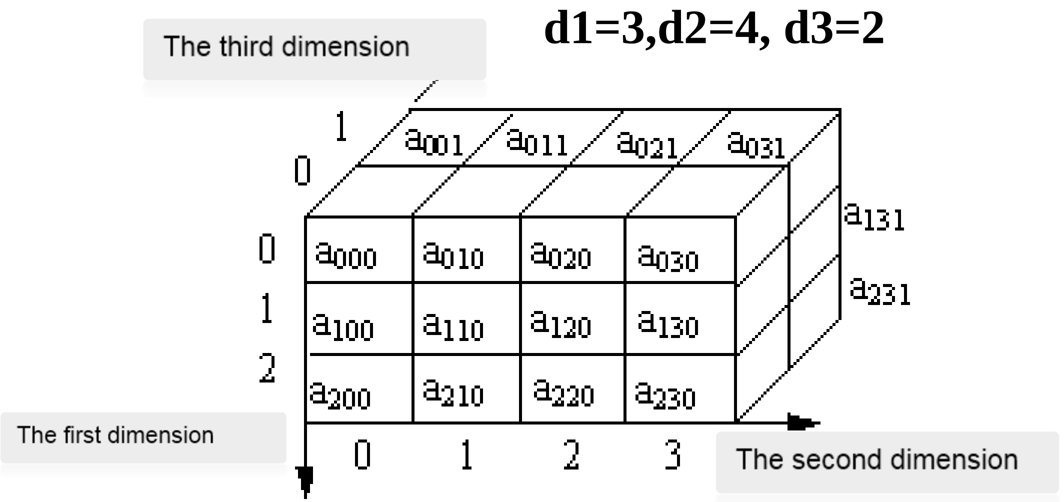
Basic Concepts

- Multidimensional array is an extension of one-dimensional array (vector).
- Vector of vectors make up an multidimensional array.
- Represented as

  ELEM $A[c_1..d_1][c_2..d_2]...[c_n..d_n]$
- c_i and d_i are upper and lower bounds of the indices in the i-th dimension. Thus, the total number of elements is: $\prod_{i=1}^{n}(d_i - c_i + 1)$

Structure of Array





2-dimensional array 3-dimensional array

d1[0..2], d2[0..3], d3[0..1] are the three dimensions respectively

Storage of Array

- Memory is one-dimensional, so arrays are stored linearly
  - Stored row by row (row-major)
  - Stored column by column (column-major)

$$X = \begin{vmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{vmatrix}$$

Row-Major in Pascal

a[1..k,1..m,1..n]

a_{111} a_{112} a_{113} … a_{11n}	a_{11*}
a_{121} a_{122} a_{123} … a_{12n}	a_{12*}

…………………………

a_{1m1} a_{1m2} a_{1m3} … a_{1mn}	a_{1m*}
a_{211} a_{212} a_{213} … a_{21n}	a_{21*}
a_{221} a_{222} a_{223} … a_{22n}	a_{22*}

…………………………

a_{2m1} a_{2m2} a_{2m3} … a_{2mn}	a_{2m*}

⋮

a_{k11} a_{k12} a_{k13} … a_{k1n}

a_{k21} a_{k22} a_{k23} … a_{k2n}

…………………………

a_{km1} a_{km2} a_{km3} … a_{kmn}

Column-Major in FORTRAN a[1..k, 1..m, 1..n]

a_{111} a_{211} a_{311} … a_{k11} a_{*11}
a_{121} a_{221} a_{321} … a_{k21} a_{*21}
……………………………
a_{1m1} a_{2m1} a_{3m1} … a_{km1} a_{*m1}
a_{**1}

a_{112} a_{212} a_{312} … a_{k12}
a_{122} a_{222} a_{322} … a_{k22}
……………………………
a_{1m2} a_{2m2} a_{3m2} … a_{km2}
a_{**2}

a_{11n} a_{21n} a_{31n} … a_{k1n}
a_{12n} a_{22n} a_{32n} … a_{k2n}
……………………………
a_{1mn} a_{2mn} a_{3mn} … a_{kmn}

- C++ multidimensional array

ELEM A[d_1][d_2]...[d_n];

$$loc(A[j_1, j_2, \ldots, j_n]) = loc(A[0,0,\ldots,0])$$
$$+d \cdot [\, j_1 \cdot d_2 \cdot \ldots \cdot d_n + j_2 \cdot d_3 \cdot \ldots \cdot d_n$$
$$+ \ldots + j_{n-1} \cdot d_n + j_n \,]$$
$$= loc(A[0,0,\ldots,0]) + d \cdot [\sum_{i=1}^{n-1} j_i \prod_{k=i+1}^{n} d_k + j_n]$$

Special Matrices Implemented by Arrays

- Triangular matrix (upper/lower)
- Symmetric matrix
- Diagonal matrix
- Sparse matrix

Lower Triangular Matrix

- One-dimensional array: list[0.. $(n^2+n)/2-1$]
  - The matrix element $a_{i,j}$ is stored in list[$(i^2+i)/2 + j$] ($i>=j$)

$$\begin{pmatrix} 0 & & & & & \\ 0 & 0 & & & & \\ 7 & 5 & 0 & & & \\ 0 & 0 & 1 & 0 & & \\ 9 & 0 & 0 & 1 & 8 & \\ 0 & 6 & 2 & 2 & 0 & 7 \end{pmatrix}$$

Symmetric Matrix

$$\begin{bmatrix} 0 & 3 & 0 & 15 \\ 3 & 0 & 4 & 0 \\ 0 & 4 & 0 & 6 \\ 15 & 0 & 6 & 0 \end{bmatrix}$$

- Satisfies that $a_{i,j} = a_{j,i}$, $0 \le i, j < n$

  The matrix on the right is a (symmetric) adjacent matrix for a undirected graph

- Store the lower triangle in a 1-dimensional array

  sa[0..n (n+1) /2-1]

  - There is a one-to-one mapping between sa[k] and $a_{i,j}$:

$$k = \begin{cases} j(j+1)/2 + i, i < j \\ i(i+1)/2 + j, i \ge j \end{cases}$$

Diagonal Matrix

- Diagonal matrix: all non-zero elements are located at diagonal lines.
- Band matric: a[i][j] = 0 when |i-j| > 1
  - A band matrix with bandwidth 1 is shown as below

$$
\begin{pmatrix}
a_{0,0} & a_{0,1} & \cdots\cdots & & 0 \\
a_{1,0} & a_{1,1} & a_{1,2} & \cdots\cdots & \\
 & & & & \\
\cdots\cdots & & & & a_{n-2,n-1} \\
0 & & & a_{n-1,n-2} & a_{n-1,n-1}
\end{pmatrix}
$$

Sparse Matrix

- Few non-zero elemens, and these elements distribute unevenly

$$\mathbf{A}_{6\times 7} = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 5 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 11 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

12.1 Multidimensional Array

- Sparse Factor
  - In a m×n matrix, there are t non-zero elements, and the sparse factor is:

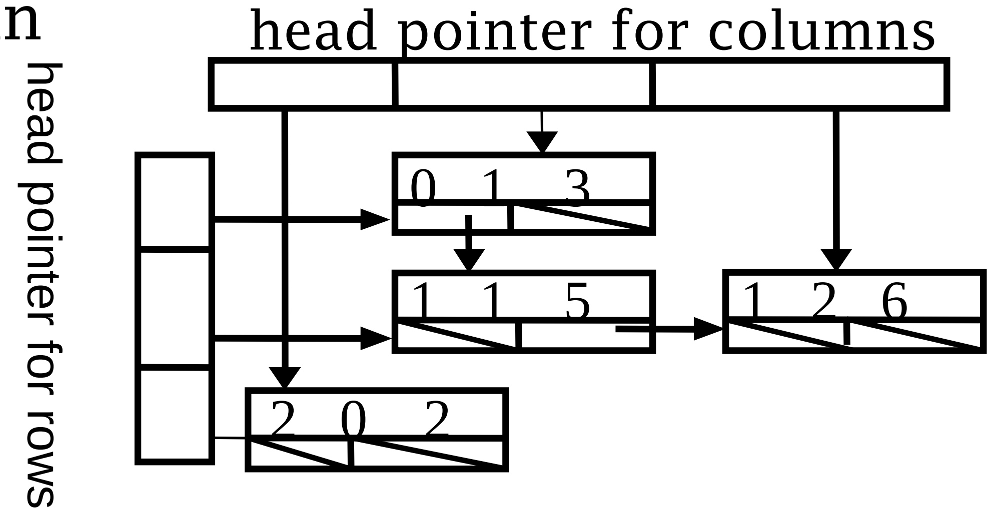
$$\delta = \frac{t}{m \times n}$$

  - When this value is lower than 0.05, the matrix could be considered a sparse matrix.
- 3-tuple (i, j, a_{ij}): commonly used for input/output
  - i is the row number
  - j is the column number
  - a_{ij} is the element value

Orthogonal Lists of a Sparse Matrix

- An orthogonal list consists of two sets of lists
  - pointer sequense for rows and columns
  - Each node has two pointers: one points to the successor on the same row; the other points to the successor on the same column

$$\begin{bmatrix} 0 & 3 & 0 \\ 0 & 5 & 6 \\ 2 & 0 & 0 \end{bmatrix}$$



Classic Matrix Multiplication

- A[c1..d1][c3..d3] , B[c3..d3][c2..d2], C[c1..d1][c2..d2]。

$$C = A \times B \quad (C_{ij} = \sum_{k=c3}^{d3} A_{ik} \cdot B_{kj})$$

-

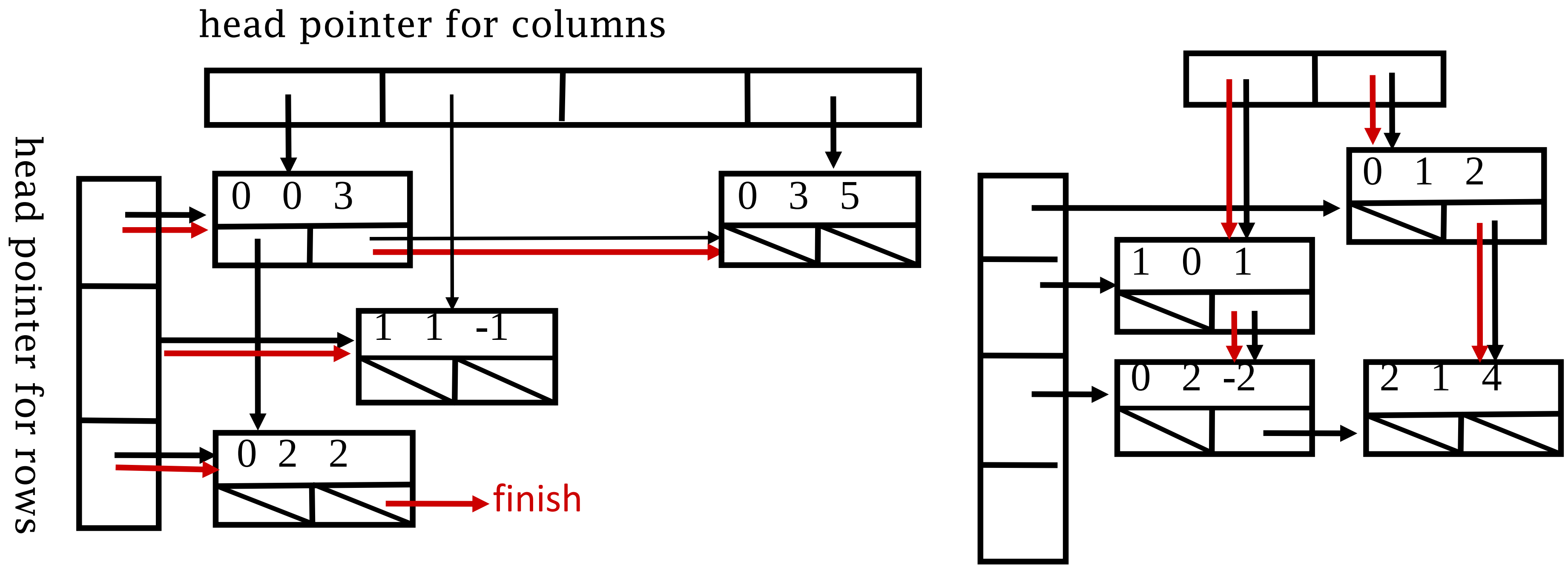
Time Cost of Classic Matrix Multiplication

- p=d1-c1+1 , m=d3-c3+1 , n=d2-c2+1 ;
- A is a p×m matrix, B is a m×n matrix, resulting in C, a p×n matrix
- So the time cost of the classic matrix multiplication is O (p×m×n)

```
for  (i=c1; i<=d1; i++)
    for  (j=c2; j<=d2; j++){
        sum = 0;
        for  (k=c3; k<=d3; k++)
            sum = sum + A[i,k]*B[k,j];
        C[i , j] = sum;
    }
```

Sparse Matrix Multiplication

$$\begin{bmatrix} 3 & 0 & 0 & 5 \\ 0 & -1 & 0 & 0 \\ 2 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} 0 & 2 \\ 1 & 0 \\ -2 & 4 \\ 0 & 0 \end{bmatrix} = \begin{bmatrix} 0 & 6 & 6 \\ -1 & 0 & \\ 0 & 4 & 4 \end{bmatrix}$$



Time Cost of Sparse Matrix Multiplication

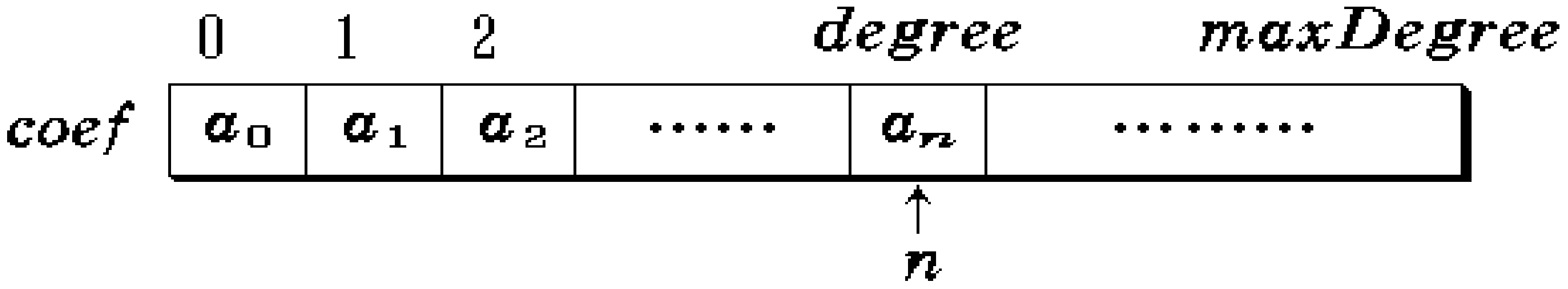
- A is a p×m matrix, B is a m×n matrix, resulting in C, a p×n matrix.
  - If the number of non-zero elements in a row of A is at most t_a
  - and the number of non-zero elements in a column of B is at most t_b
- Overall running time is reduced to O ((t_a+t_b) ×p×n)
- Time cost of classic matrix multiplication is O (p×m×n)

Applications of Sparse Matrix

polynomial of one indeterminate

$$P_n(x) = a_0 + a_1 x + a_2 x^2 + \cdots + a_n x^n$$

$$= \sum_{i=0}^{n} a_i x^i$$



Data Structures and Algorithms

Thanks

the National Elaborate Course (Only available for IPs in China)
http://www.jpk.pku.edu.cn/pkujpk/course/sjjg/

Ming Zhang, Tengjiao Wang and Haiyan Zhao
Higher Education Press, 2008.6 (awarded as the "Eleventh Five-Year" national planning textbook)