Data Structures and Algorithms (7)

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 7. Graphs

- 7.1 Definition and terms of graphs
- 7.2 Abstract data type of graphs
- 7.3 Storage structure of graphss
- 7.4 Traversals of graphs
- 7.5 The shortest paths
- 7.6 Minimum-cost spanning trees

Adjacency matrix

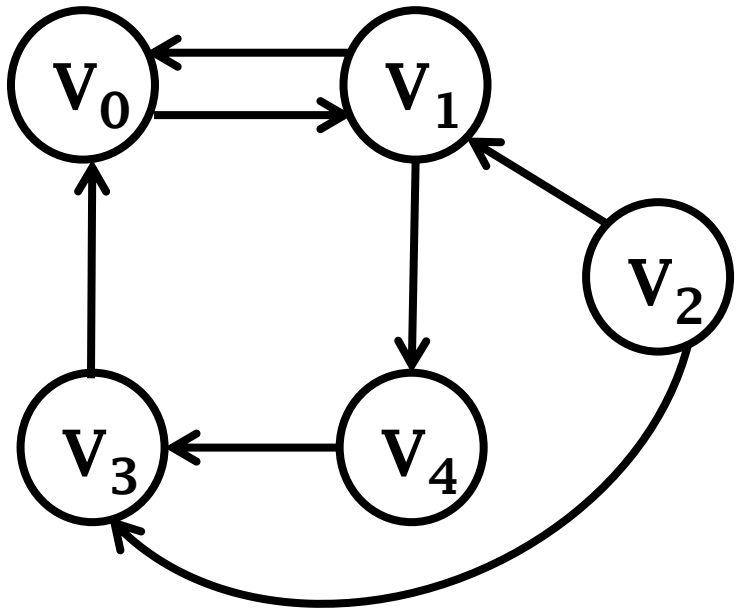
- **An Adjacency matrix of a graph represents the adjacency relation of vertices, an element of which indicates whether an edge exists**
- Let G = <V, E> be a graph with n vertices, its adjacency matrix it is a two-dimensional array A[n, n], defined as:

$$A[i,j]=\begin{cases}1, if\ (v_i,v_j)\in E\ \ or <v_i,v_j>\in E\\ 0, if\ (v_i,v_j)\notin E\ \ or <v_i,v_j>\notin E\end{cases}$$

- For a graph with n vertices, its adjacency matrix consumes $O(n^2)$ storage space, no matter how many edges the graph has

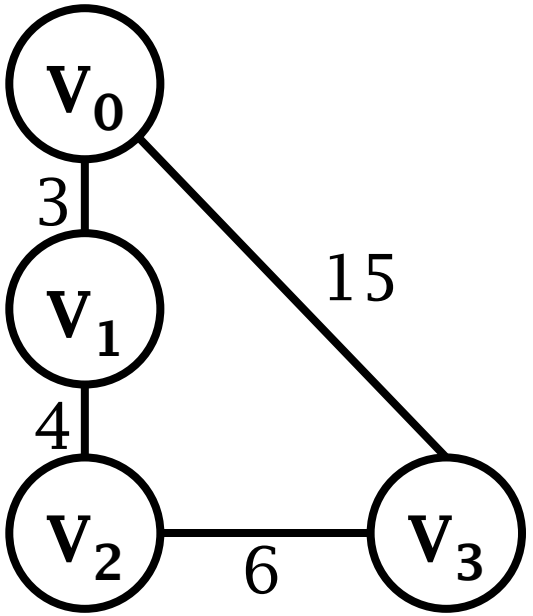
Adjacency matrixes of directed graphs

$$A = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \end{bmatrix}$$



Adjacency matrixes of undirected graphs

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



Adjacency matrix

```
class Edge  {                       // class of an edge
public:
    int from,to,weight ;            // start, end, weight of the edge
    Edge()  {                       // default constructor
        from = -1;  to = -1;  weight = 0;    }
    Edge(int f,int t,int w){        // constructor with given
parameters
        from = f;   to = t;   weight = w;     }
};
class Graph  {
public:
    int numVertex;                  // the number of vertices
    int numEdge;                    // the number of edges
    int *Mark;                      // visit marks of vertices of graph
    int *Indegree;                  // indegrees of vertices of graph
};
```

- Sparsity factor
  - In a matrix of size m × n, there are t nonzero elements, the sparsity factor δ is:

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

  - If δ is less than 0.05, then we say the matrix is a sparse matrix

Adjacency lists

- For a sparse graph, we can use adjacency list for storage
  - If there are few edges, the adjacency matrix will contains many zero elements,
  - which consume much space and time
- Linked list is the storage structure of adjacency list
  - Two fields of the vertex entry for v_i: vertex data field and pointer field which points to the edge list of the vertex
  - The edge list maintains all the adjacent edges (nonzero elements in a row of the adjacency matrix) to vertex v_i, and forms a single linked list. It consists of two main fields:
    - The index number of the vertex adjacent to v_i
    - The pointer specifying the next edge entry in the edge list

Adjacency lists

- The information of vertices and edges are as follows:

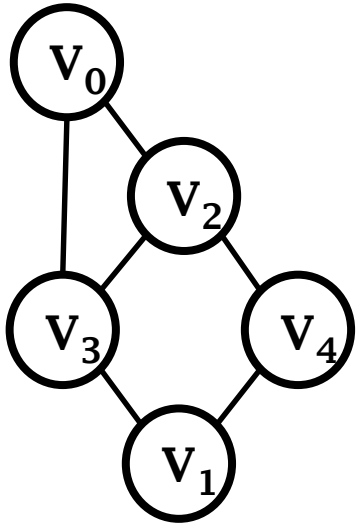
Vertex node

data	firstarc

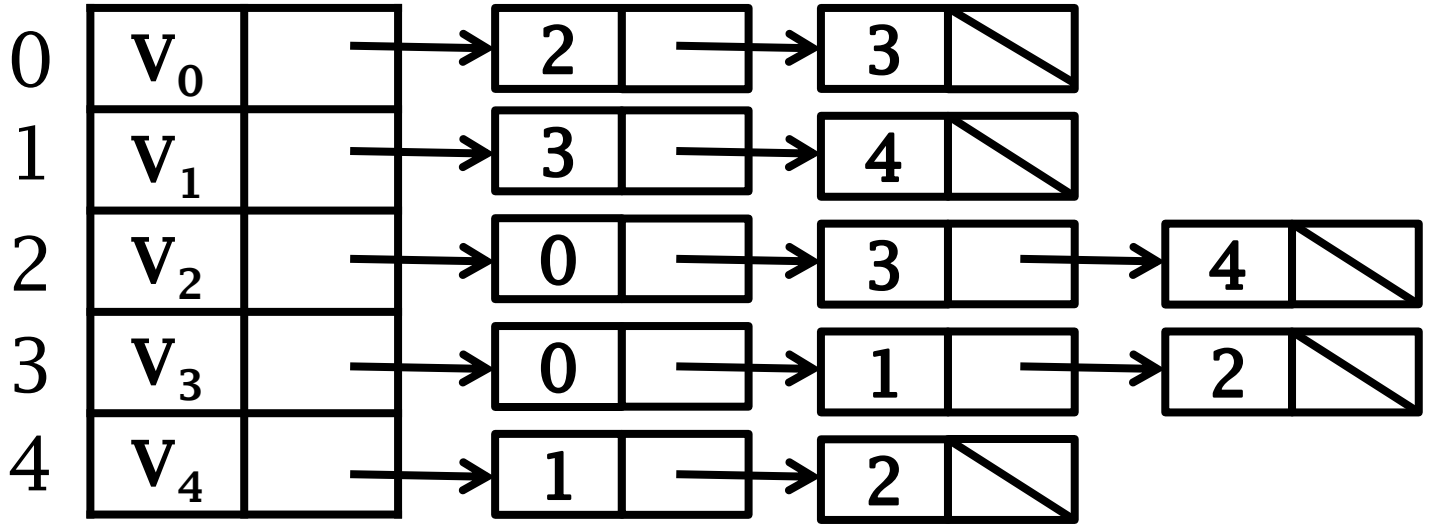
Edge node

adjvex	nextarc	Info

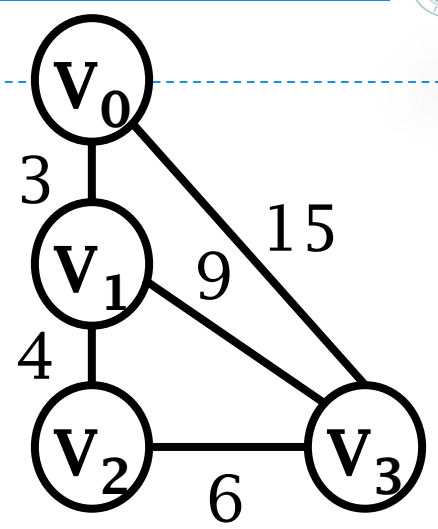
Adjacency list representation of undirected graphs

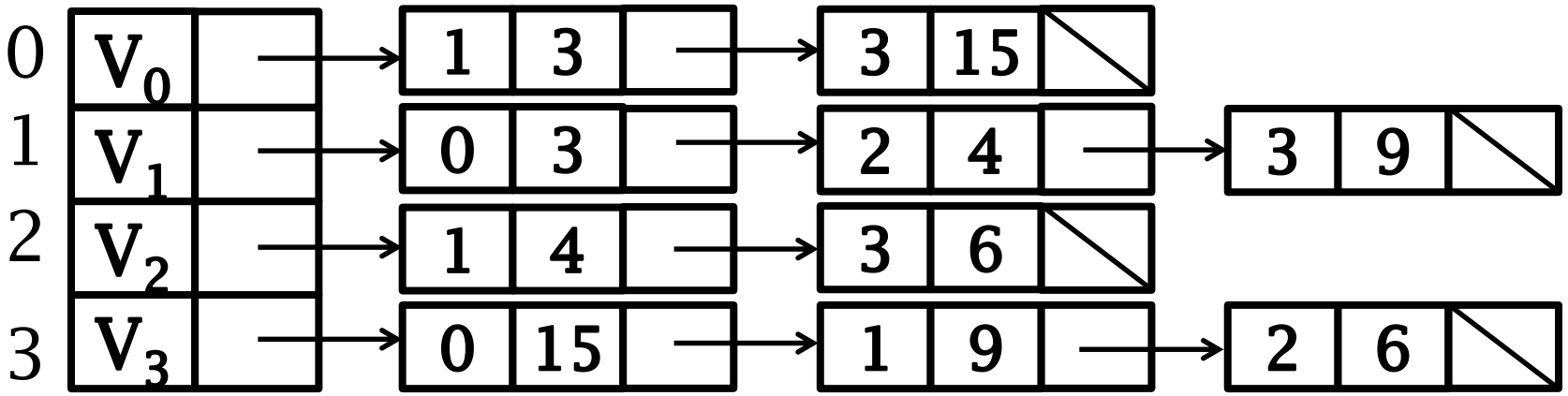


One edge appears twice in the adjacency list of undirected graph

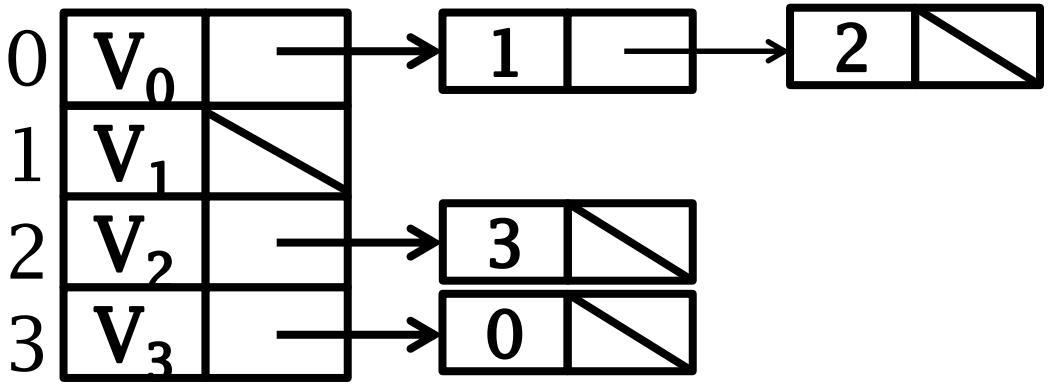


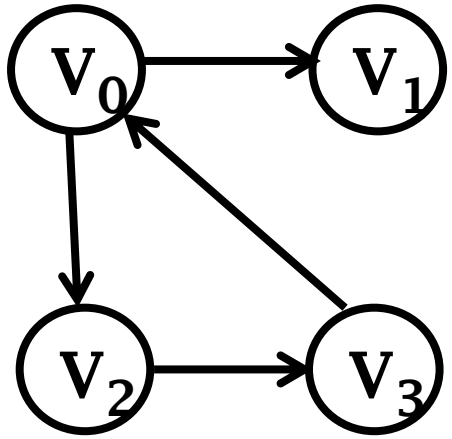
Adjacency list representation of weighted graphs



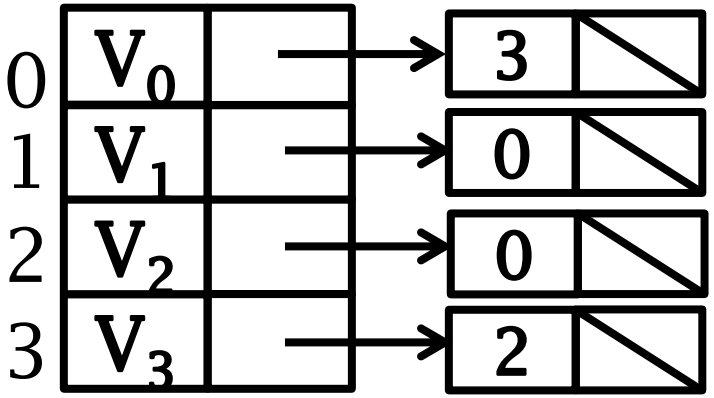


An Adjacency list of digraph (list of outgoing edge)





Inverse adjacency list of digraph(list of incoming edge)

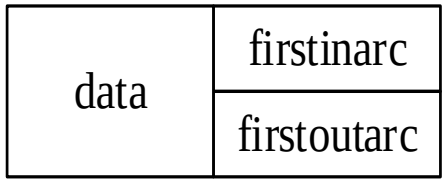


Space of adjacency lists

- undirected graph with n vertices and e edges
  - Need $(n + 2e)$ storage units.
- Digraph with n vertices and e edges.
  - Need $(n + e)$ storage units.
- When e is small, it saves much storage space
- The entries of edges list are usually sorted from small to large according to the vertex indices.

Orthogonal lists

- **An Orthogonal List can be seen as combination of an adjacency list and an inverse adjacency list.**
- Each entry corresponds to one arc of digraph, contains five fields:
  - head headvex、tail tailvex 、next arc with same tail tailnextarc; next arc with same head headnextarc ; Information, such as weight of arc, etc
- The vertex entry consists of 3 fields: data field; firstinarc is the first arc using this vertex as the end point; firstoutarc is the first arc using this vertex as the start point.



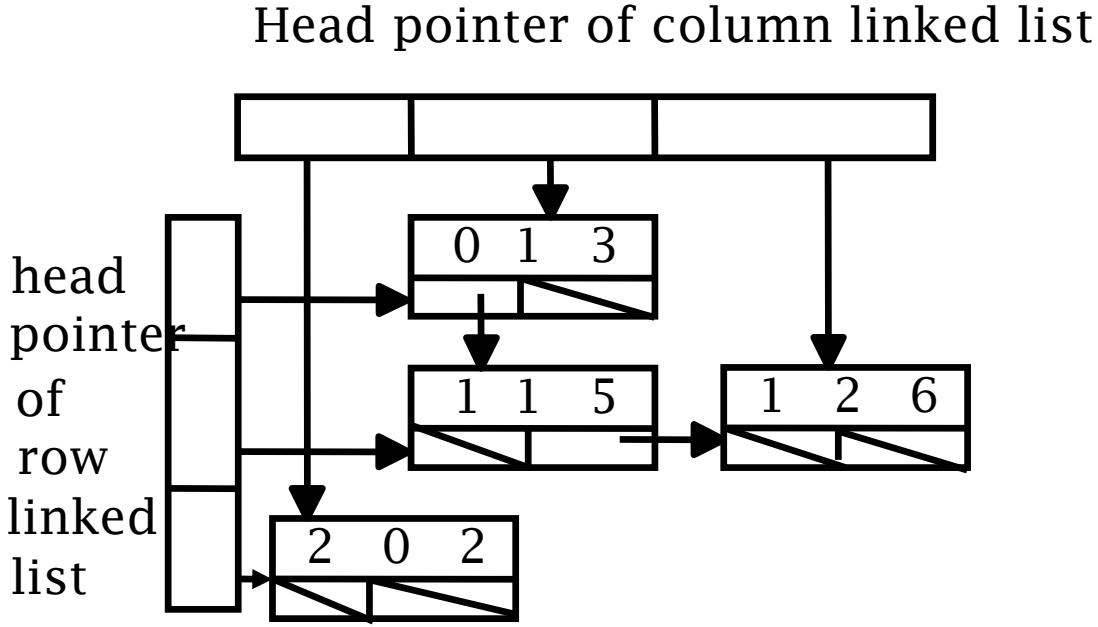
vertex node

arc(directed graph)node

Orthogonal lists of sparse matrix

- Orthogonal consists of two linked list.
  - Pointer sequence of row and column.
  - Each vertex contains two pointers: the successor of the same row and the successor of the same column

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



Sudoku

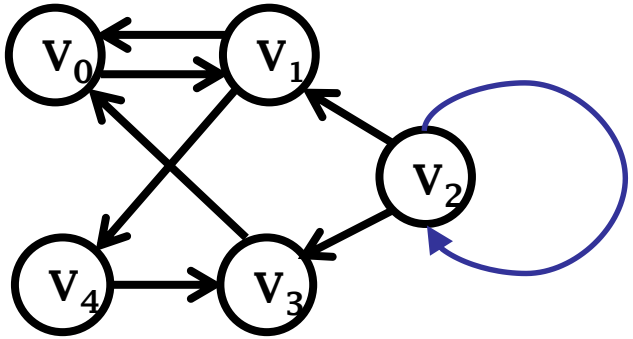
- Consists of n×n matrixes of size n×n
  - Digits of each row or each column are not repeated
  - Digits of each sub matrix are not repeated

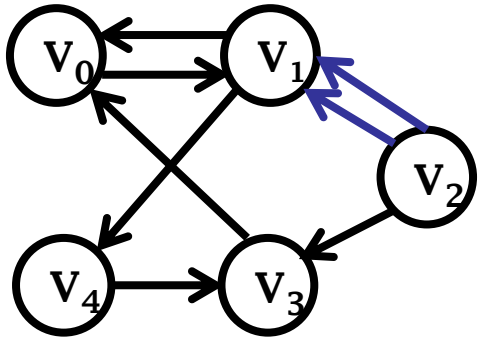
5						3		
	9		5			4		
		4				7		
	5	1		3	7	2	8	9
3		2		8		6		4
		8		5	2	1	3	7
	3	5				9		
6		9				8	2	3
	8			2	3			6

			14	13		6		1			9		5		8
			7			11	5		10	16		1			
			1			8	7		3				6		12
3	11	10	9		14					6				2	
			2	1		3		5					4		15
5	12					2	11			1	8		16		
		16	15				4		12			10		14	9
			10	15	12				2	13					11
4					6	12				7	2	16			
16	3		12			5		8				2	15		
		15		9	4			16						1	13
2		6					16		15		1	8			
	7				16					8		5	10	12	3
10		4				1		9	13			6			
			8		15	4		7	5			14			
15		1		10			8		6		16	7			

Thinking

- For the following two extended complex graphs, how should the storage structure be designed?





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)