# 图的实现

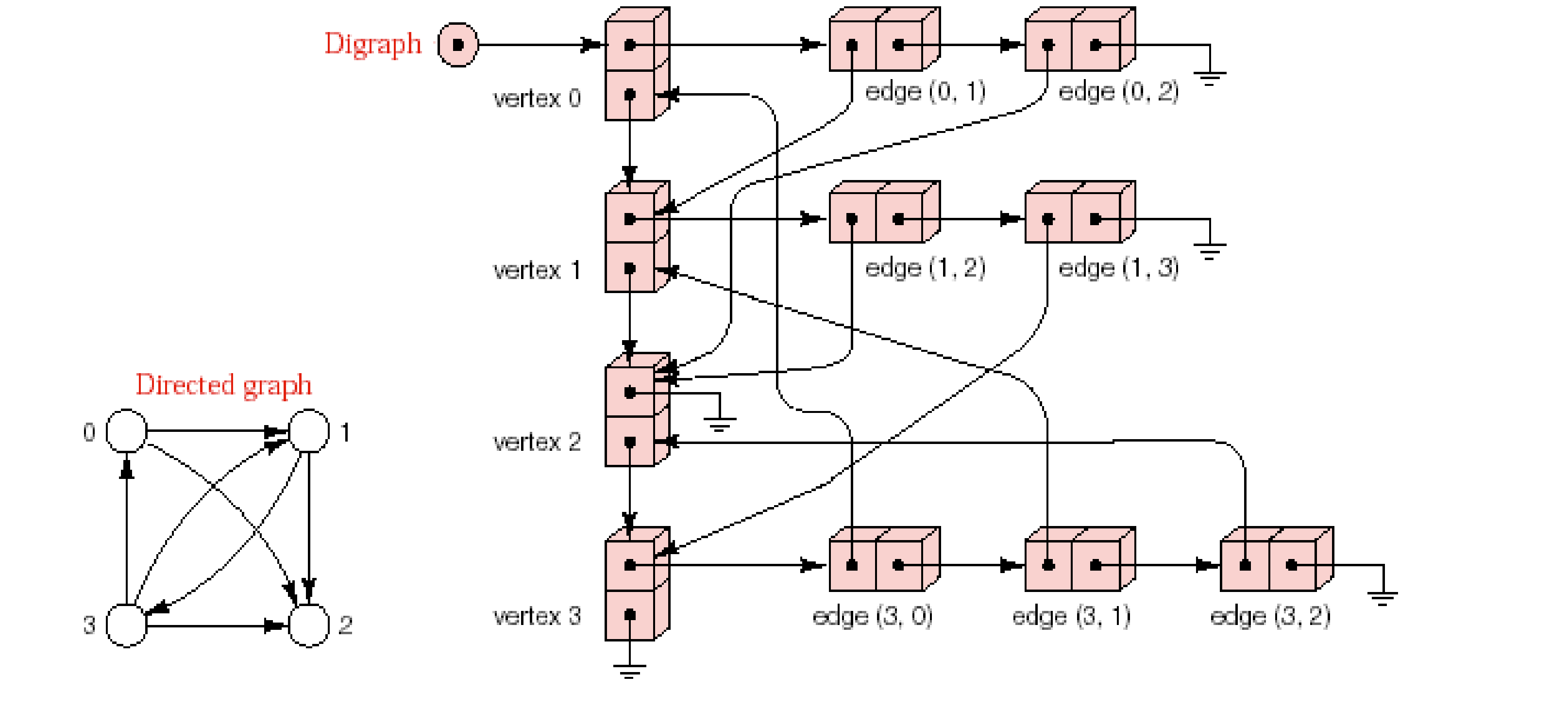
# 图的实现

- 图的线性表形式的表示：
  - 顶点：形成线性表，其结点数据域指向该顶点参与的边形成的线性表
  - 边：形成线性表，其结点数据域指向该边对应的另一个顶点
  - 图：指向第一个顶点

```
class Edge; //forward declaration(向前声明)
class Vertex {
        Edge *first_edge; // start of the adjacency list
        Vertex *next_vertex; // next vertex on the linked list
};
class Edge {
        Vertex *end_point; // vertex to which the edge points
        Edge *next_edge; // next edge on the adjacency list
};
class Digraph {
        Vertex *first_vertex; // header for the list of vertices
};
```

# 图的实现

- 图的线性表表示：



(a) Linked lists

# 图的实现

- 图的集合形式的表示：

  - DEFINITION： A *digraph G* consists of a set *V* , called the *vertices* of *G*, and, for all *v* ∈ *V* , a subset $A_v$ of *V* , called the set of vertices *adjacent* to v. （$A_v$*称为v的邻接点集合）*。

  - Set as a bit string（**位串**）：

```
template <int max_set>
struct Set {
          bool is_element[max_set];
};
```

# 图的实现

- 图的集合形式的表示：（续）

  - Digraph as a bit-string set

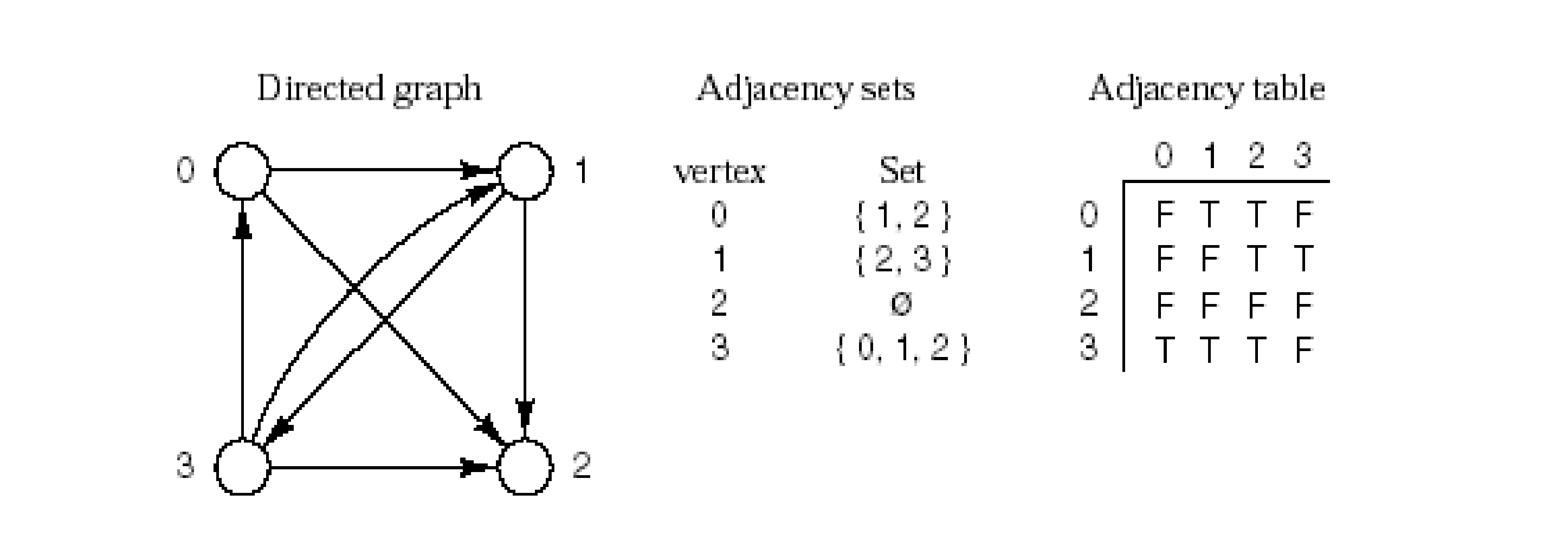
```
template <int max_size>
class Digraph {
        int count; //number of vertices, at most max_size
        Set<max_size> neighbors[max_size];
};
```

  - Digraph as an **adjacency table**（**邻接表格、邻接矩阵**）

```
template <int max_size>
class Digraph {
        int count; //number of vertices, at most max_size
        bool adjacency[max_size][max_size];
};
```

# 图的实现

- 图的集合形式的表示：（续）



Adjacency sets

| vertex | Set |
|---|---|
| 0 | { 1, 2 } |
| 1 | { 2, 3 } |
| 2 | ∅ |
| 3 | { 0, 1, 2 } |

Adjacency table

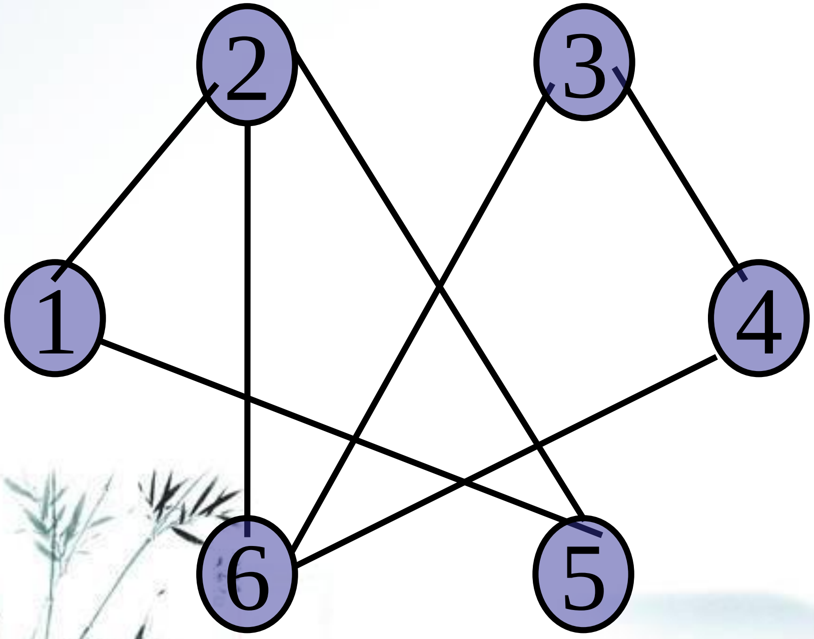
|   | 0 | 1 | 2 | 3 |
|---|---|---|---|---|
| 0 | F | T | T | F |
| 1 | F | F | T | T |
| 2 | F | F | F | F |
| 3 | T | T | T | F |

# 图的实现

定义:矩阵的元素为

$$A_{ij}=\begin{cases} F(0) & (i,j)\notin E \\ T(1) & (i,j)\in E \end{cases}$$

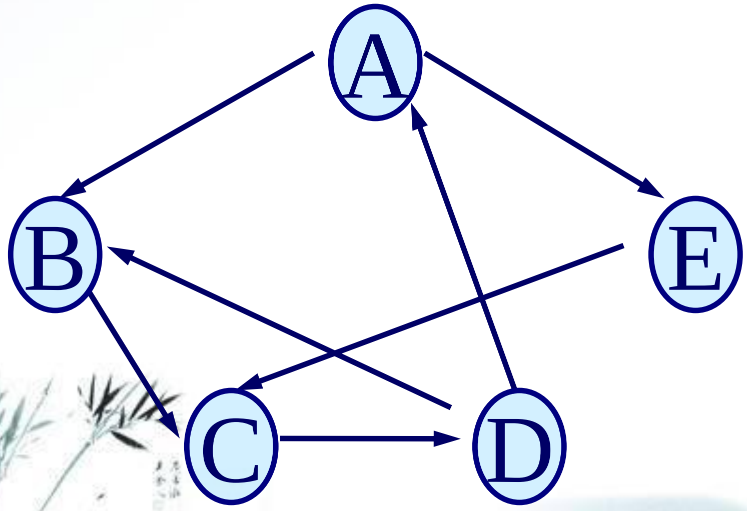
——无向图的邻接矩阵为对称矩阵



| | | | | | |
|---|---|---|---|---|---|
| 0 | 1 | 0 | 0 | 1 | 0 |
| 1 | 0 | 0 | 0 | 1 | 1 |
| 0 | 0 | 0 | 1 | 0 | 1 |
| 0 | 0 | 1 | 0 | 0 | 1 |
| 1 | 1 | 0 | 0 | 0 | 0 |
| 0 | 1 | 1 | 1 | 0 | 0 |

# 图的实现

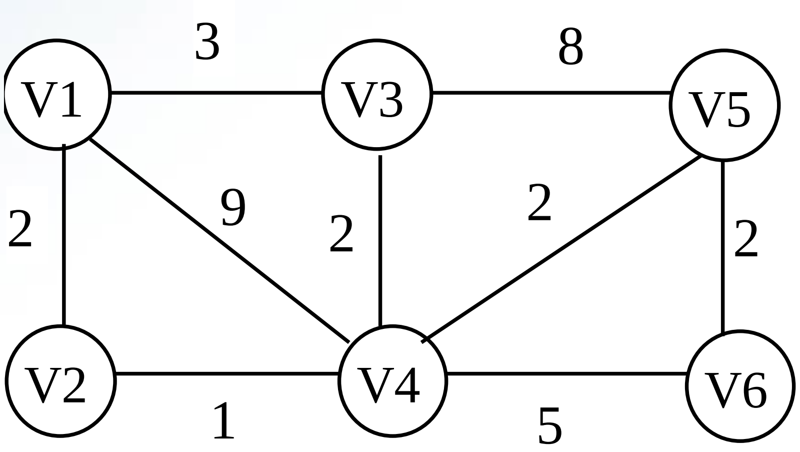
无向图的邻接矩阵为对称矩阵，有向图的邻接矩阵通常为非对称矩阵



| | | | | |
|---|---|---|---|---|
| 0 | 1 | 0 | 0 | 1 |
| 0 | 0 | 1 | 0 | 0 |
| 0 | 0 | 0 | 1 | 0 |
| 1 | 1 | 0 | 0 | 0 |
| 0 | 0 | 1 | 0 | 0 |

# 图的实现

## Weighted Graph Representation



$$\begin{matrix}
\infty & 2 & 3 & 9 & \infty & \infty \\
2 & \infty & \infty & 1 & \infty & \infty \\
3 & \infty & \infty & 2 & 8 & \infty \\
9 & 1 & 2 & \infty & 2 & 5 \\
\infty & \infty & 8 & 2 & \infty & 2 \\
\infty & \infty & \infty & 5 & 2 & \infty
\end{matrix}$$

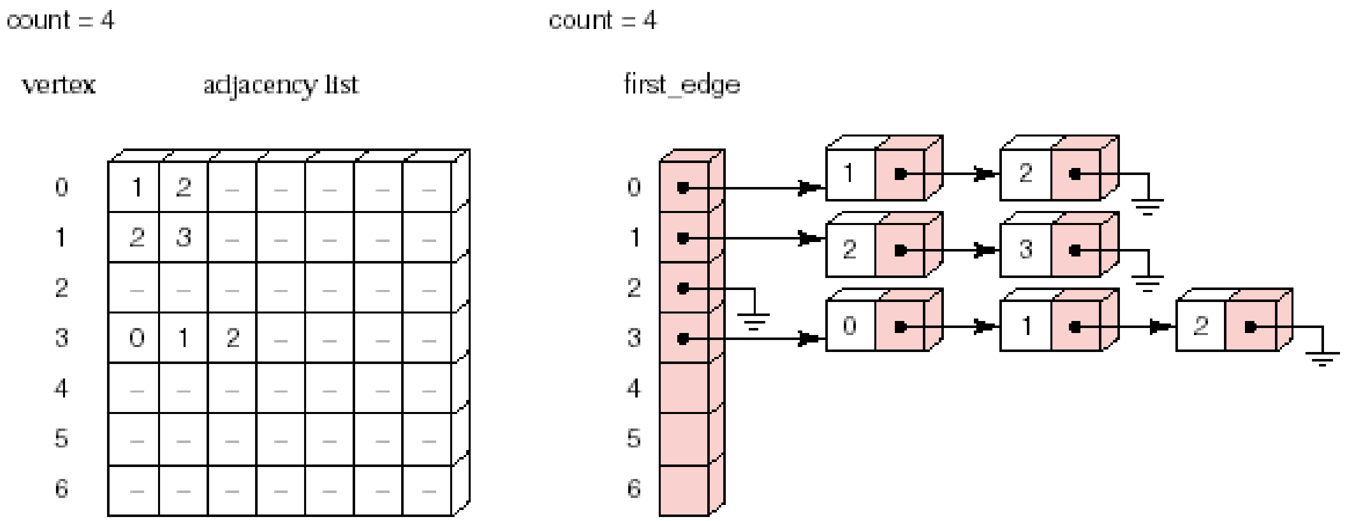
# 图的实现

- 图的邻接表表示（**adjacency list**）

```
typedef int Vertex;
        template <int max_size>
        class Digraph {
              int count; //number of vertices, at most max_size
              List<Vertex> neighbors[max_size];
        };
```

# 图的实现

- 图的邻接表表示（**adjacency list**）



count = 4

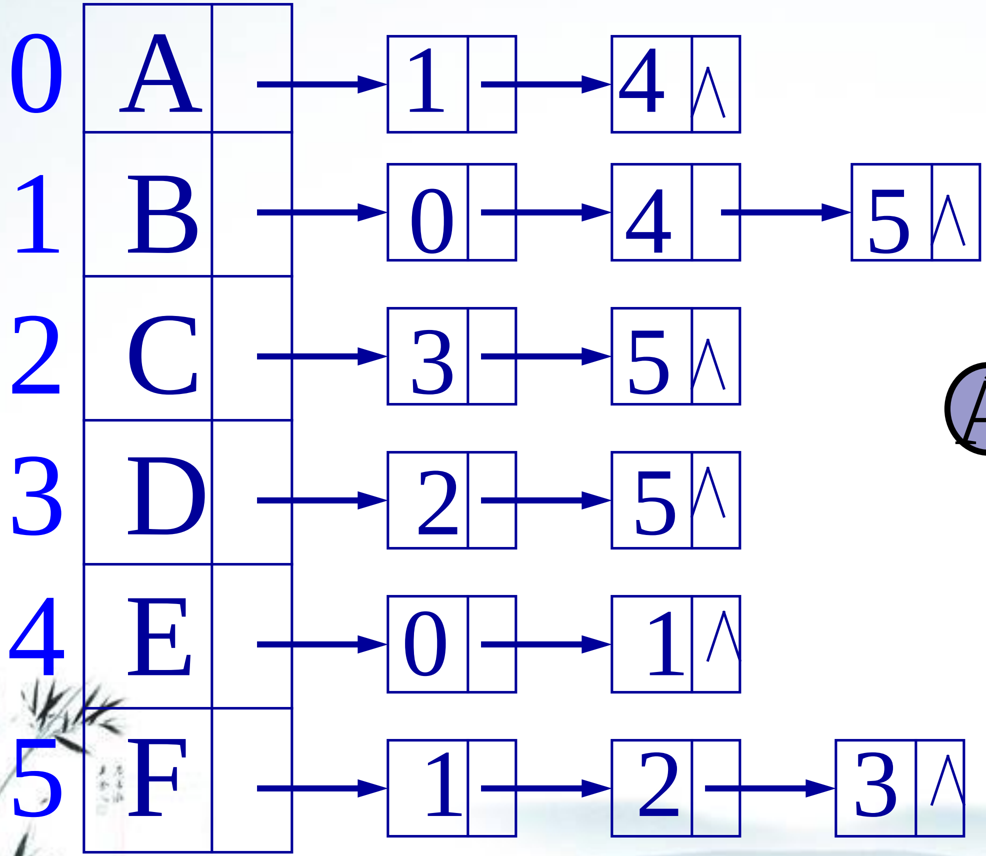
| vertex | adjacency list | | | | | | |
|---|---|---|---|---|---|---|---|
| 0 | 1 | 2 | – | – | – | – | – |
| 1 | 2 | 3 | – | – | – | – | – |
| 2 | – | – | – | – | – | – | – |
| 3 | 0 | 1 | 2 | – | – | – | – |
| 4 | – | – | – | – | – | – | – |
| 5 | – | – | – | – | – | – | – |
| 6 | – | – | – | – | – | – | – |

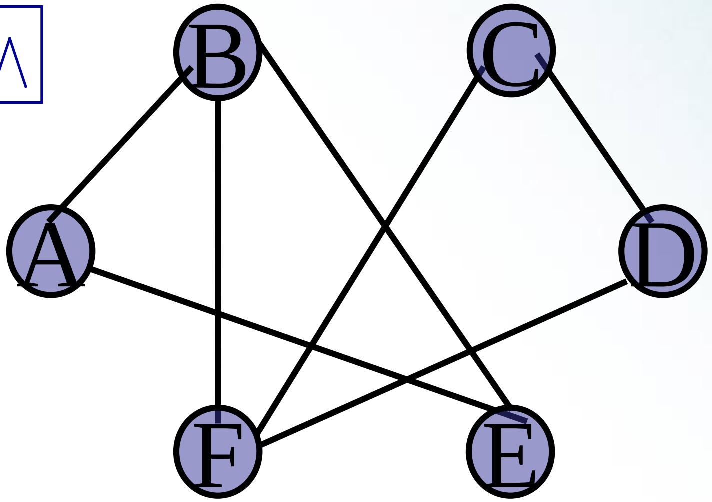
(b) Contiguous lists

(c) Mixed

# 图的实现

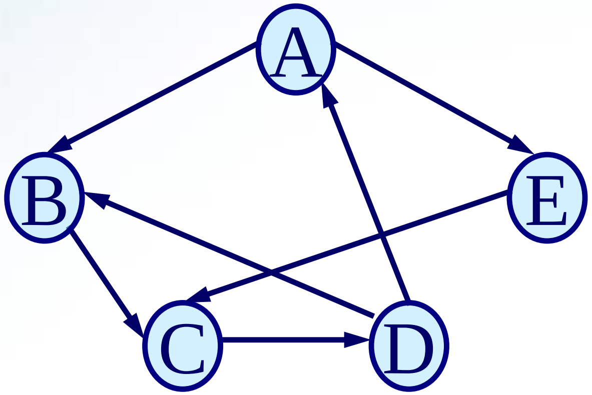
- 图的邻接表表示（**adjacency list**）



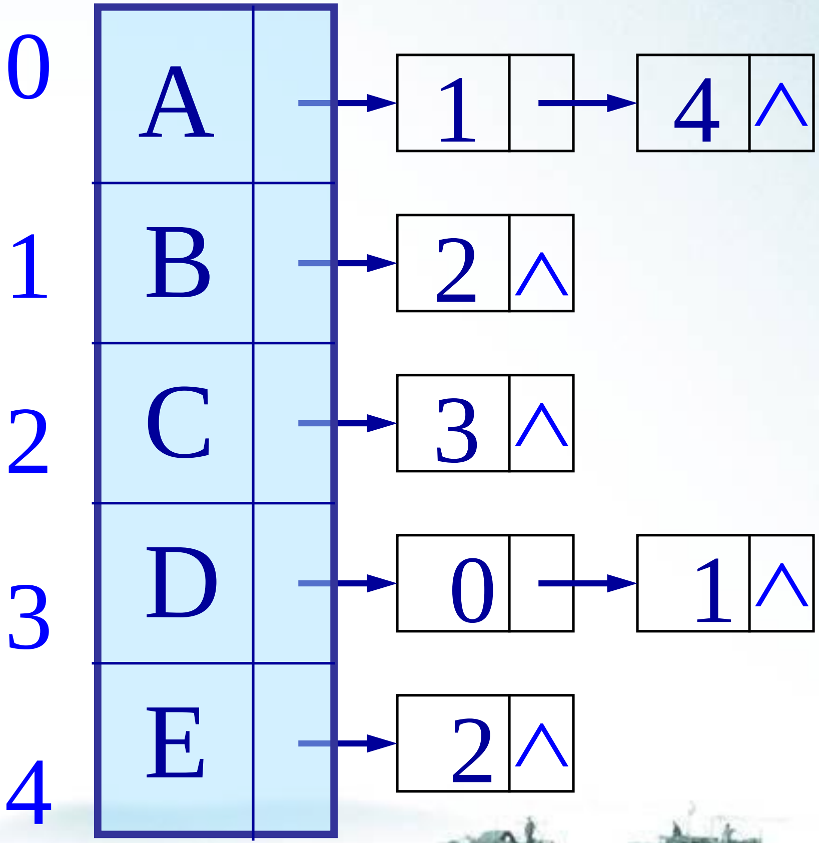


# 图的实现

- 图的邻接表表示（**adjacency list**）



——可见，在有向图的邻接表中不易找到指向该顶点的弧。


0
A
1
4
∧
1
B
2
∧
2
C
3
∧
3
D
0
1
∧
4
E
2
∧