

本节内容

图的存储

邻接表法

知识总览

图的存储

邻接矩阵

数组实现的顺序存储，
空间复杂度高，不适合
存储稀疏图

邻接表

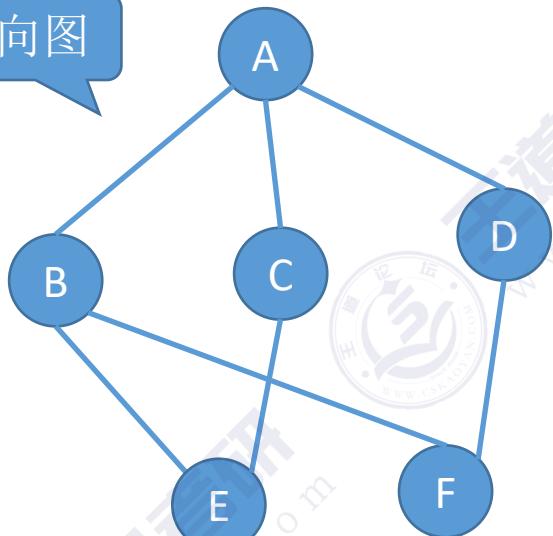
顺序+链式存储

十字链表

邻接多重表

邻接表法（顺序+链式存储）

无向图

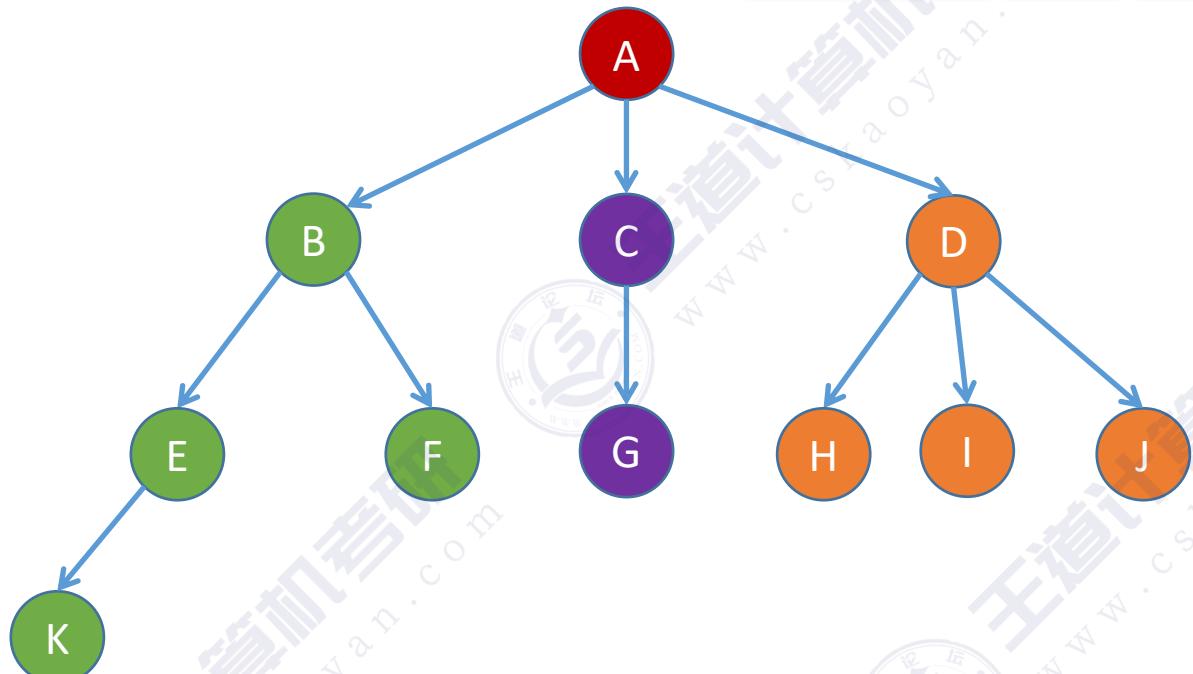


```
//用邻接表存储的图
typedef struct{
    AdjList vertices;
    int vexnum,arcnum;
} ALGraph;
```

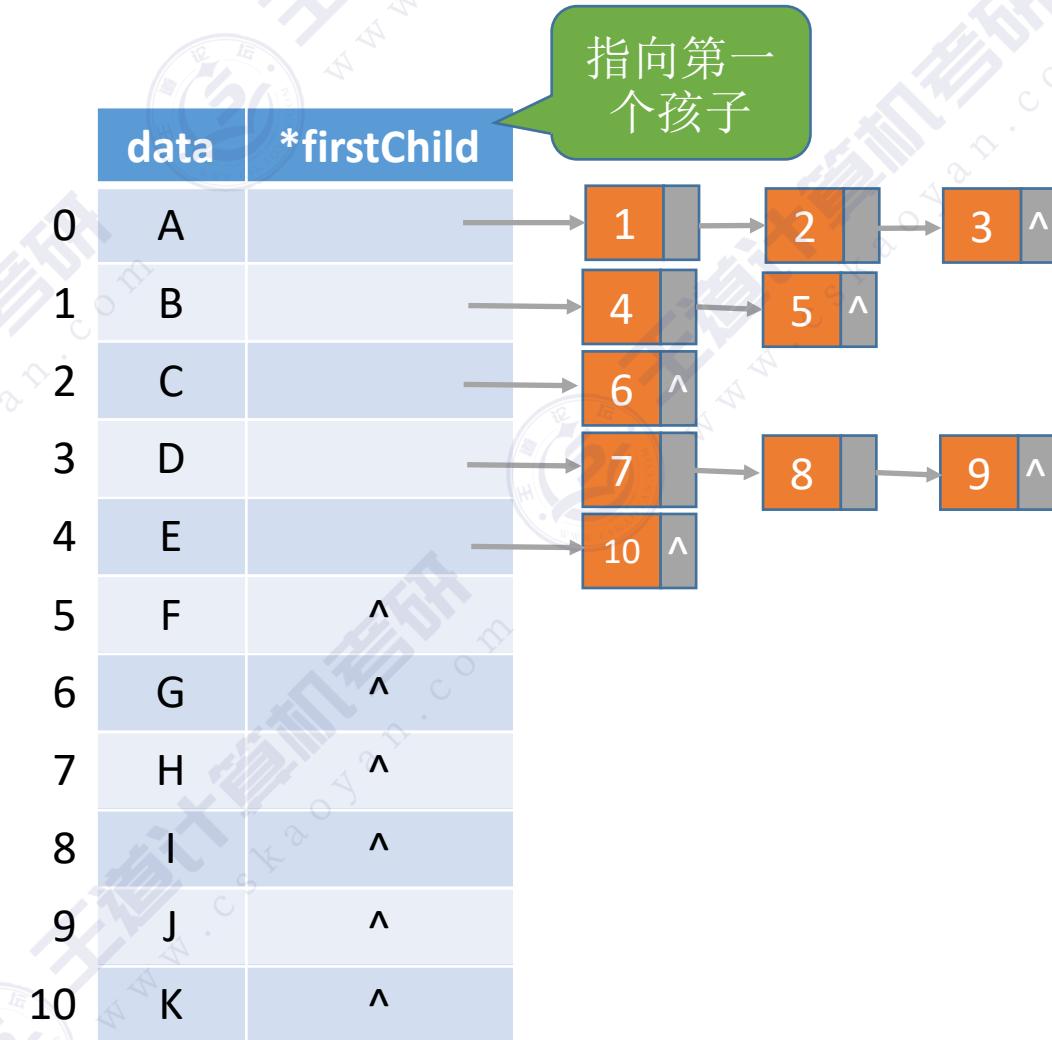
```
///"边/弧"
typedef struct ArcNode{
    int adjvex;           //边/弧指向哪个结点
    struct ArcNode *next; //指向下一条弧的指针
    //InfoType info;      //边权值
}ArcNode;
```

```
///"顶点"
typedef struct VNode{
    VertexType data;      //顶点信息
    ArcNode *first;        //第一条边/弧
}VNode,AdjList[MaxVertexNum];
```

对比：树的孩子表示法

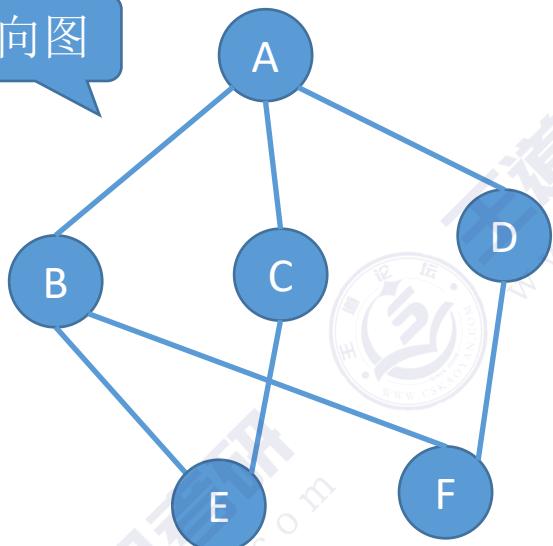


孩子表示法：顺序存储各个节点，每个结点中保存孩子链表头指针

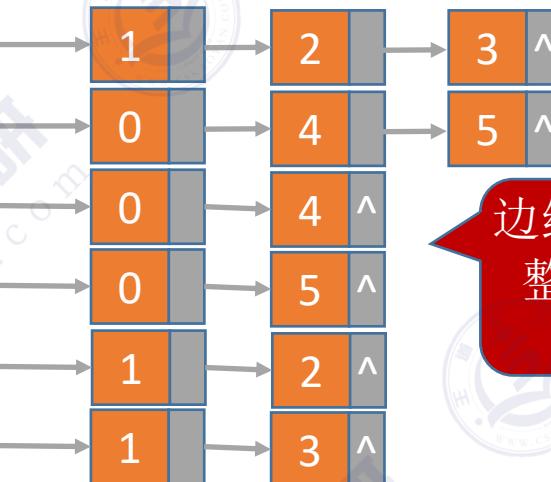


邻接表法

无向图



	data	*first
0	A	1
1	B	0
2	C	0
3	D	0
4	E	1
5	F	1



边结点的数量是 $2|E|$ ，
整体空间复杂度为
 $O(|V| + 2|E|)$

思考：如何求顶点的度、入度、出度？

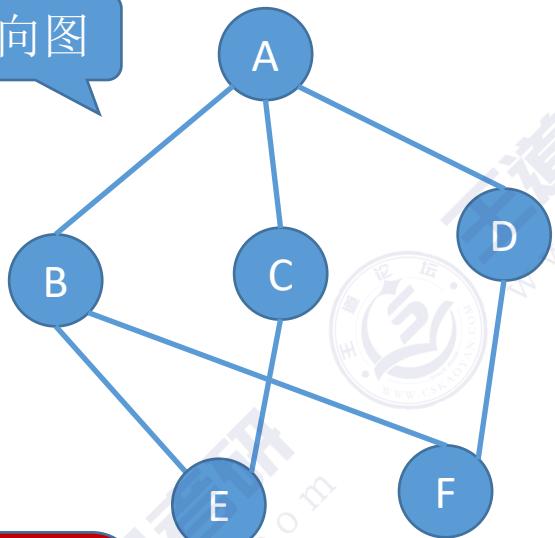
如何找到与一个顶点相连的边/弧？



边结点的数量是 $|E|$ ，
整体空间复杂度为
 $O(|V| + |E|)$

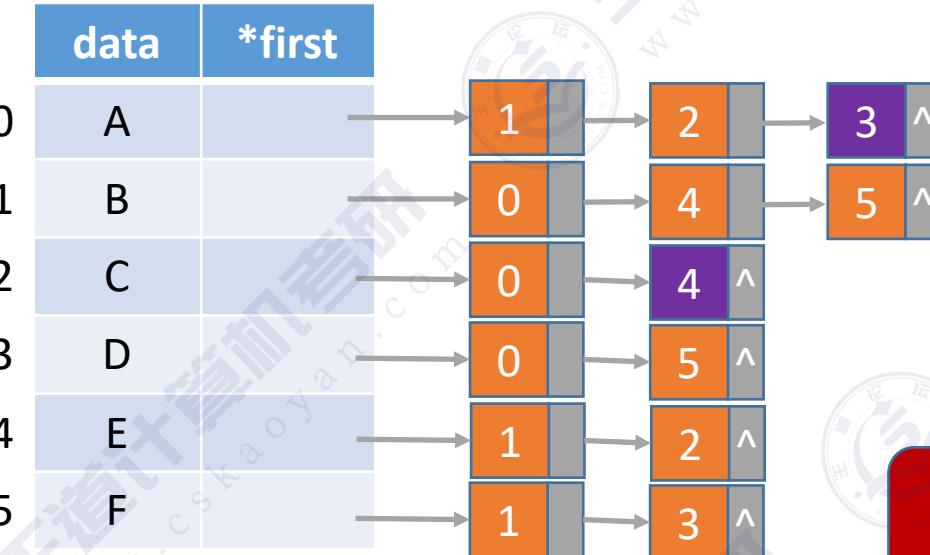
邻接表法

无向图

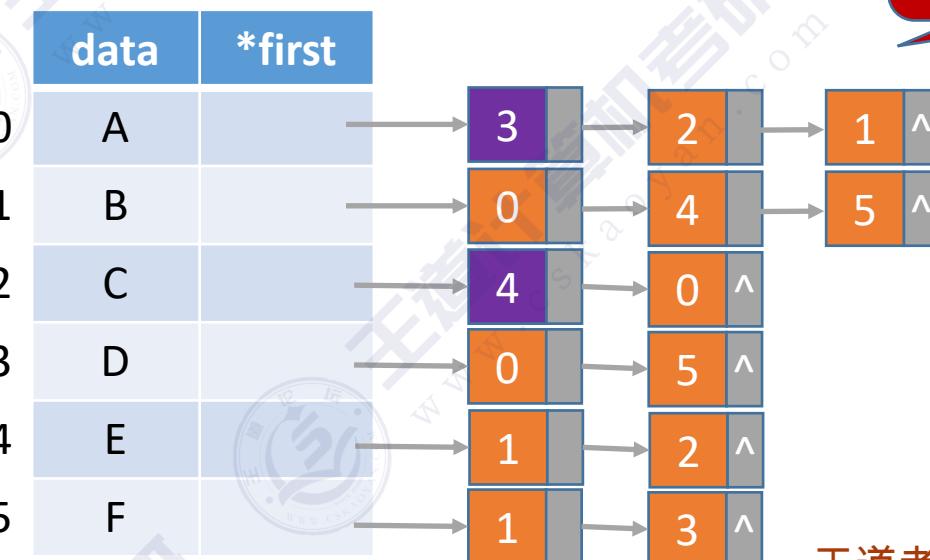


只要确定了顶点编号，图的邻接矩阵表示方式唯一

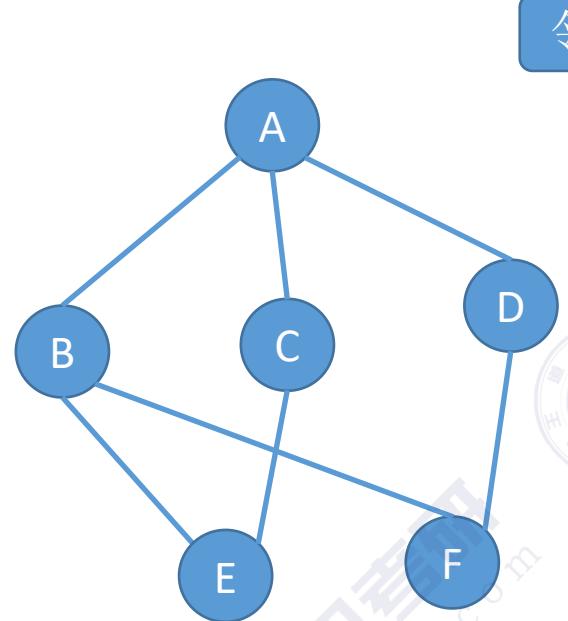
	A	B	C	D	E	F
A	0	1	1	1	0	0
B	1	0	0	0	1	1
C	1	0	0	0	1	0
D	1	0	0	0	0	1
E	0	1	1	0	0	0
F	0	1	0	1	0	0



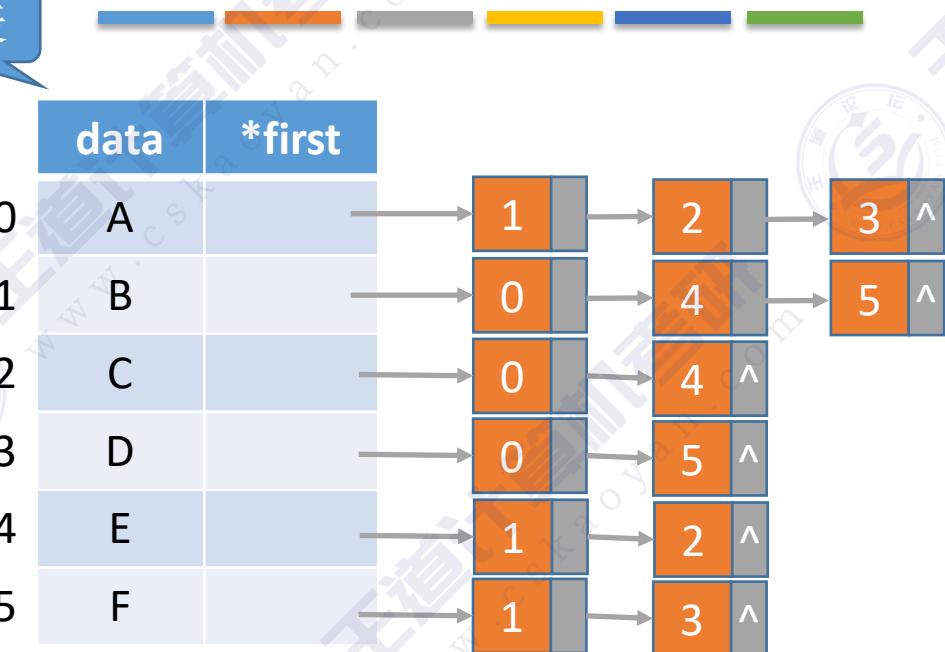
图的邻接表表示方式并不唯一



知识回顾与重要考点



邻接表



邻接矩阵

	A	B	C	D	E	F
A	0	1	1	1	0	0
B	1	0	0	0	1	1
C	1	0	0	0	1	0
D	1	0	0	0	0	1
E	0	1	1	0	0	0
F	0	1	0	1	0	0

	邻接表	邻接矩阵
空间复杂度	无向图 $O(V + 2 E)$ ；有向图 $O(V + E)$	$O(V ^2)$
适合用于	存储稀疏图	存储稠密图
表示方式	不唯一	唯一
计算度/出度/入度	计算有向图的度、入度不方便，其余很方便	必须遍历对应行或列
找相邻的边	找有向图的入边不方便，其余很方便	必须遍历对应行或列