

10-10

图

邻接表

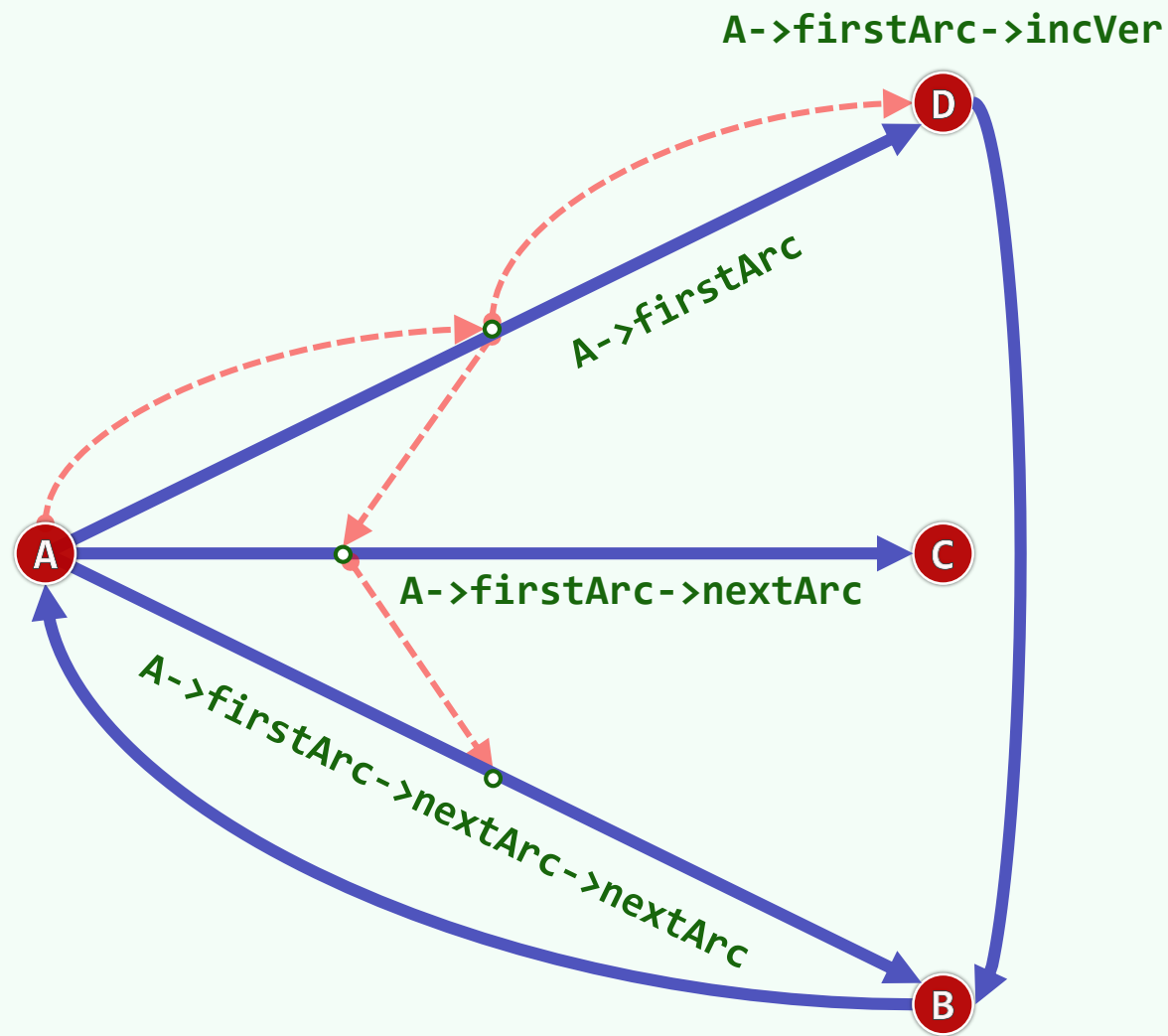
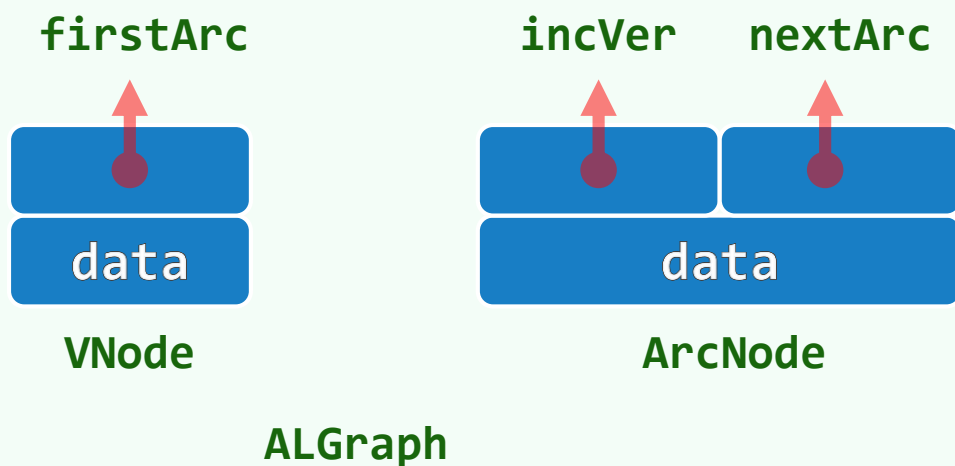
邓俊辉

deng@tsinghua.edu.cn

# 邻接表

- ❖ 如何避免邻接矩阵的空间浪费?
- ❖ 将邻接矩阵的各行组织为列表, 只记录存在的边
- ❖ 等效于, 每一顶点 $v$ 对应于列表:

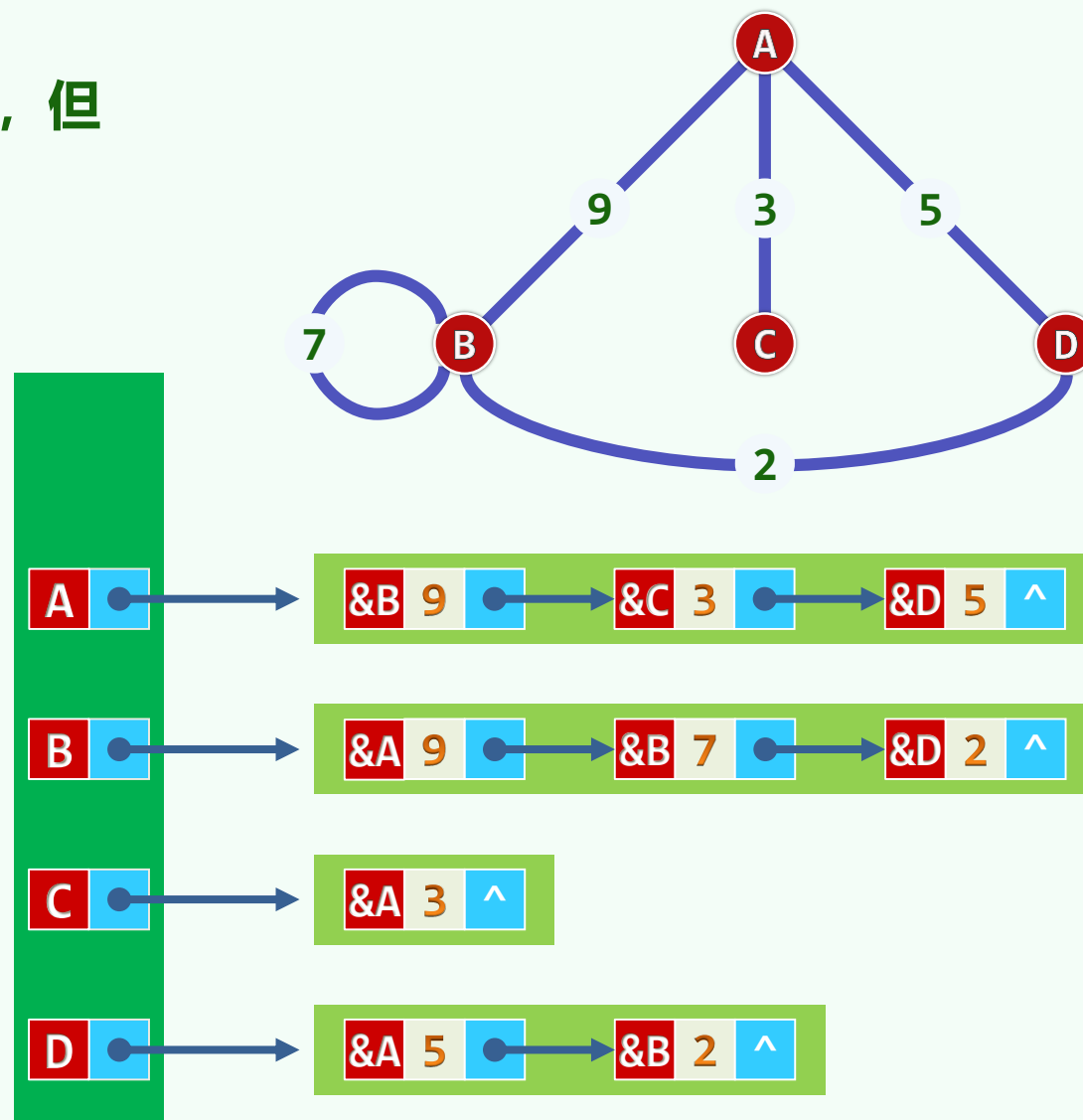
$$L_v = \{ u \mid \langle v, u \rangle \in E \}$$



# 实例

❖ 4个顶点，5条弧：不必占用 $4 \times 4 = 16$ 个单元，但  
还是占用了9个单元，另加4个表头

$\infty$	A	B	C	D
A		9	3	5
B	9	7		2
C	3			
D	5	2		



# 空间复杂度

❖ 有向图 =  $O(n + e)$

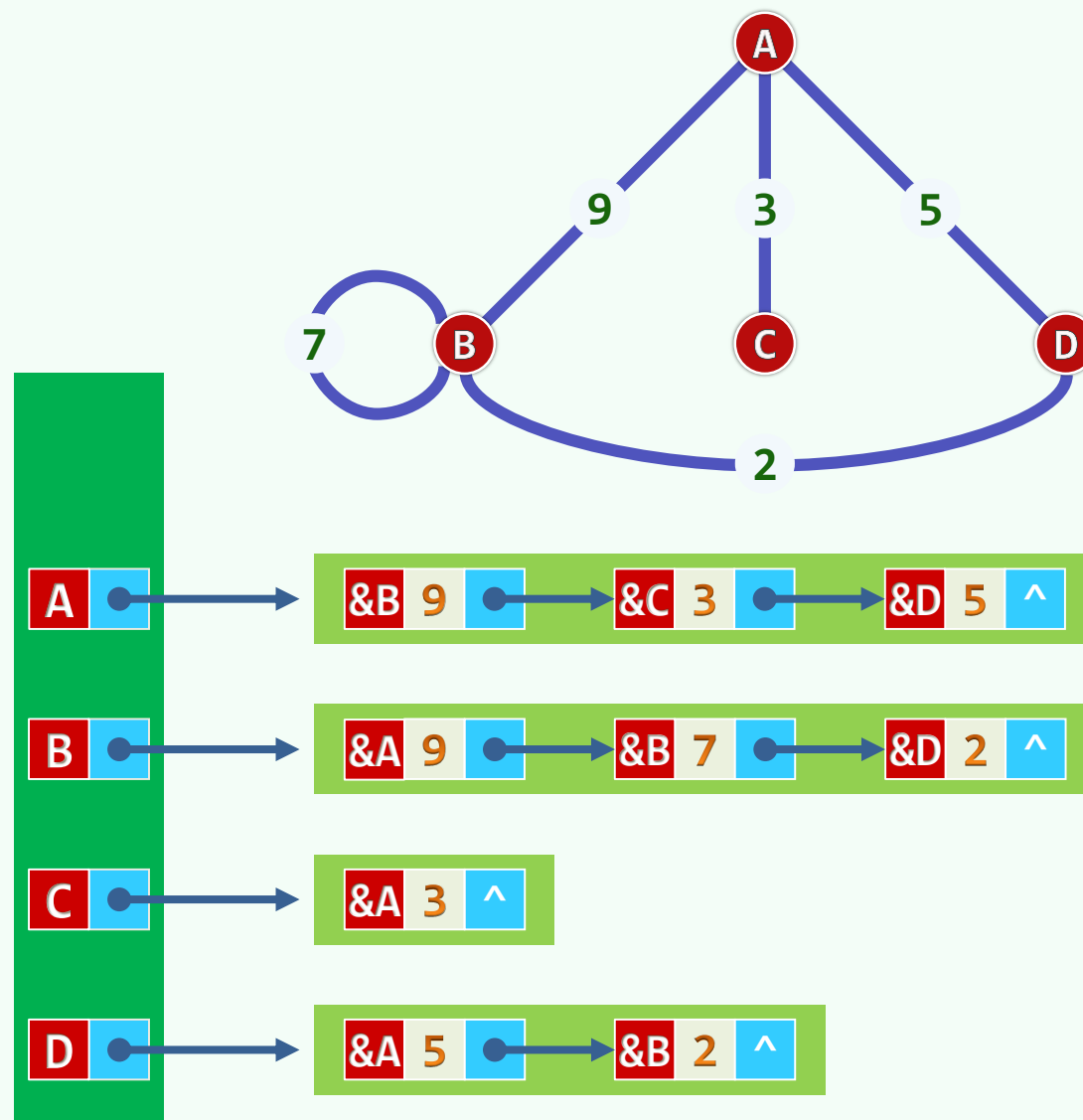
❖ 无向图 =  $O(n + 2 \times e) = O(n + e)$

- 注意：无向弧被重复存储
- 问题：如何改进？

❖ 适用于稀疏图

❖ 平面图 =  $O(n + 3 \times n) = O(n)$

较之邻接矩阵，有极大改进



## 时间复杂度 (1/2)

- ❖ 建立邻接表（递增式构造）： $O(n + e)$  //如何实现
- ❖ 枚举所有以顶点 $v$ 为尾的弧： $O(1 + \deg(v))$  //遍历 $v$ 的邻接表
- ❖ 枚举（无向图中）顶点 $v$ 的邻居： $O(1 + \deg(v))$  //遍历 $v$ 的邻接表
- ❖ 枚举所有以顶点 $v$ 为头的弧： $O(n + e)$  //遍历所有邻接表  
可改进至 $O(1 + \deg(v))$  //建立逆邻接表——为此，空间需增加多少？
- ❖ 计算顶点 $v$ 的出度/入度：
  - 增加度数记录域： $O(n)$ 附加空间
  - 增加/删除弧时更新度数： $O(1)$ 时间 //总体 $O(e)$ 时间
  - 每次查询： $O(1)$ 时间！

## 时间复杂度 (2/2)

❖ 给定顶点 $u$ 和 $v$ , 判断是否  $\langle u, v \rangle \in E$

- 有向图: 搜索 $u$ 的邻接表,  $\mathcal{O}(\deg(u)) = \mathcal{O}(e)$
- 无向图: 搜索 $u$ 或 $v$ 的邻接表,  $\mathcal{O}(\max(\deg(u), \deg(v))) = \mathcal{O}(e)$
- “并行” 搜索:  $\mathcal{O}(2 \times \min(\deg(u), \deg(v))) = \mathcal{O}(e)$

能够达到邻接矩阵的 $\mathcal{O}(1)$ 吗?

❖ 散列! 如果装填因子选取得当 //保持兴趣

- 弧的判定: expected- $\mathcal{O}(1)$ , 与邻接矩阵 “相同”
- 空间:  $\mathcal{O}(n + e)$ , 与邻接表相同

❖ 为何有时仍使用邻接矩阵? 仅仅因为实现简单? 不, 有更多用处! 比如, 可处理

Euclidean graph和intersection graph之类的隐式图 (implicitly-represented graphs)

# 取舍原则

❖ 空间/速度

❖ 顶点类型

- bit
- int
- float
- struct
- class
- ...

❖ 弧类型 (方向 / 权值)

❖ 图类型 (稀疏 / 稠密)

	邻接矩阵	邻接表
适用场合	<p>经常检测边的存在</p> <p>经常做边的插入/删除</p> <p>图的规模固定</p> <p>稠密图</p>	<p>经常计算顶点的度数</p> <p>顶点数目不确定</p> <p>经常做遍历</p> <p>稀疏图</p>