

03-A

列表

循位置访问

Don't lose the link.

- Robin Milner

邓俊辉

deng@tsinghua.edu.cn

# 从静态到动态

❖ 根据是否修改数据结构，所有操作大致分为两类方式

- 静态： 仅读取，数据结构的内容及组成**一般**不变： get、search
- 动态： 需写入，数据结构的局部或整体将改变： put、insert、remove

❖ 与操作方式相对应地，数据元素的存储与组织方式也分为两种

- 静态： 数据空间整体创建或销毁

数据元素的**物理次序**与其**逻辑次序**严格一致；可支持高效的**静态**操作

比如向量，元素的物理地址与其逻辑次序线性对应

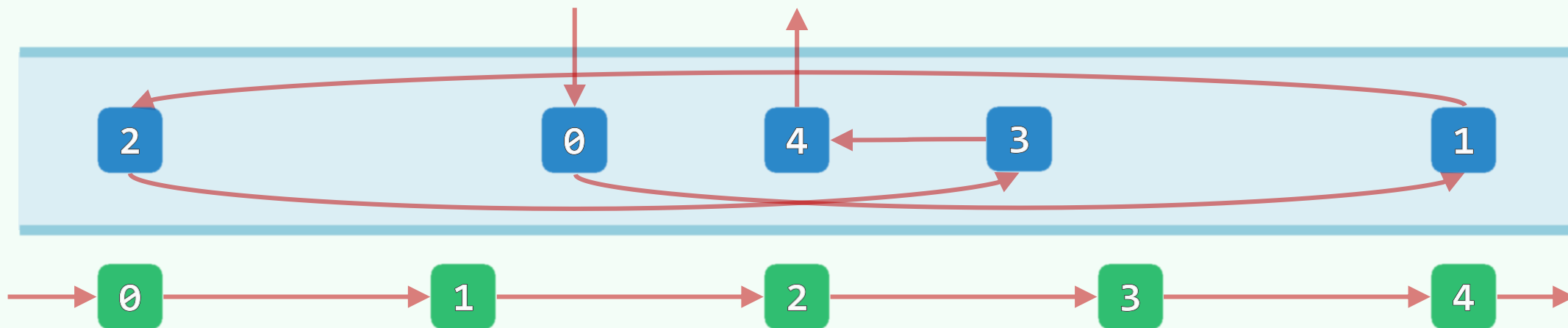
- 动态： 为各数据元素动态地分配和回收的物理空间

相邻元素记录彼此的物理地址，在逻辑上形成一个整体；可支持高效的**动态**操作

# 从向量到列表

❖ 列表 (list) 是采用**动态**储存策略的典型结构

- 其中的元素称作节点 (node) , 通过指针或引用彼此联接
- 在**逻辑**上构成一个线性序列:  $L = \{ a_0, a_1, \dots, a_{n-1} \}$

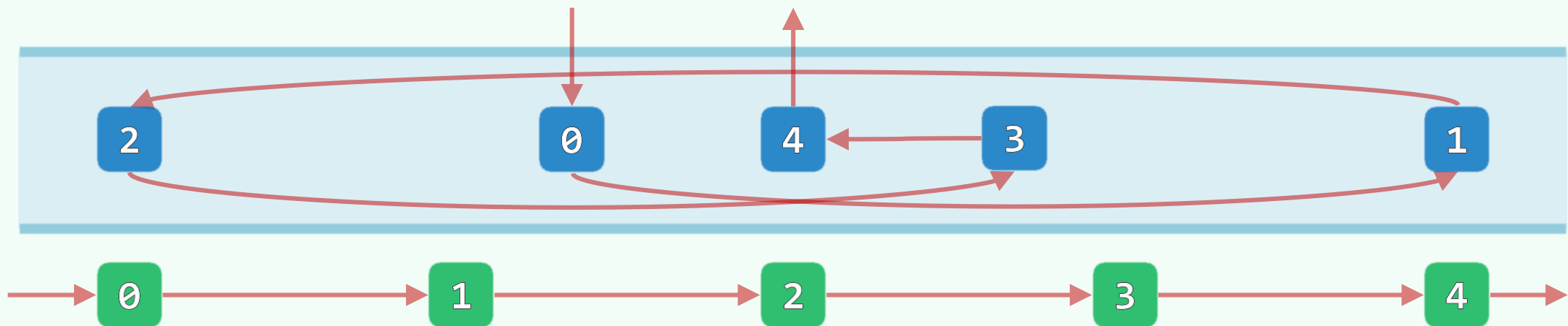


❖ 相邻节点彼此互称前驱 (predecessor) 或后继 (successor)

没有前驱/后继的节点称作**首** (first/front) /**末** (last/rear) 节点

## Call-By-Position

- ❖ 如此，列表中各元素的物理地址将**不再**决定于逻辑次序  
动态操作可以在**局部**完成，复杂度有望控制在 $O(1)$
- ❖ 循**位置**访问：利用节点之间的相互**引用**，找到特定的节点



- ❖ 顺藤摸瓜：找到我的...朋友A的...亲戚B的...同事C的...战友D的...同学Z  
如果是按逻辑次序的**连续**访问，单次也是 $O(1)$