4.3 链接表 Linked Lists

郝家胜

hao@uestc.edu.cn

自动化工程学院

链接存储的线性表

- 1.2、 链接存储的线性表（链表）的定义
- 1.2.1、 链表的引入
  - 顺序存储的缺点:
    - 1、在插入、删除时要移动大量的节点
    - 2、存储空间的大小固定
  - 原因:
    - 存放的连续性
  - 突破
    - 离散存放
    - 用指针来表示元素之间的关系

链接存储的线性表

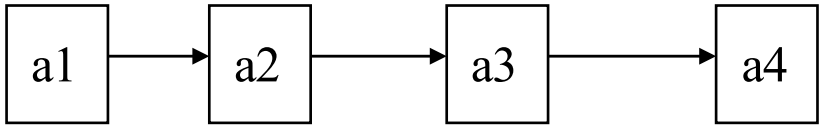
- **用链表实现线性表（非连续存储）**

线性表元素：a1、a2、a3、a4....

↕

链表链点

线性关系：

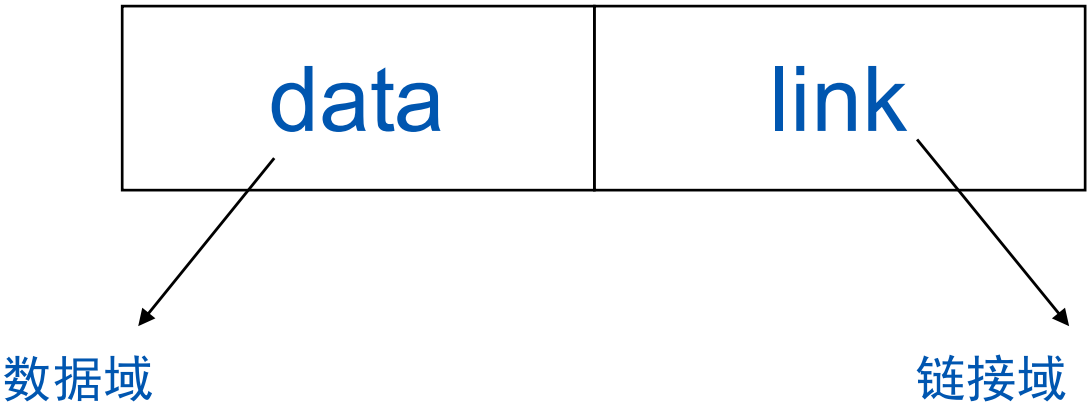


↕

指针域，指针关系

链表的定义

- **1.2.2 链表的定义**
- **链点**

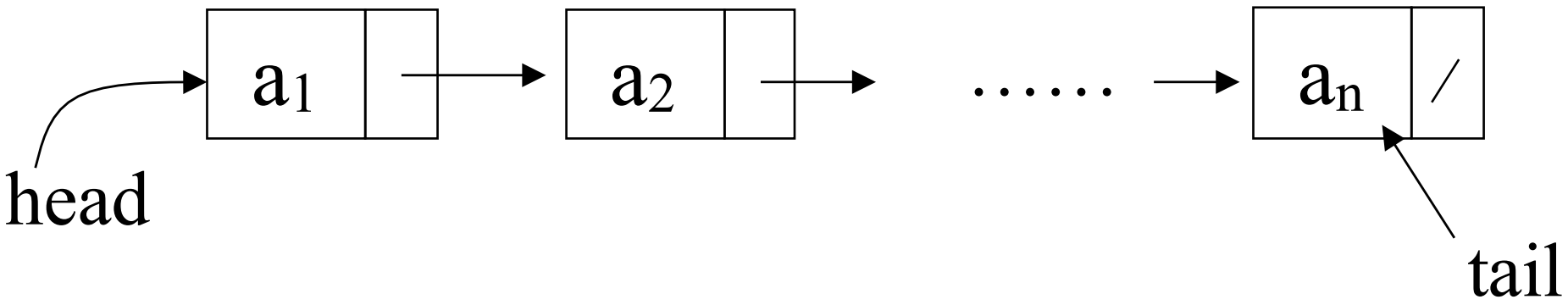


数据域：存放数据。
链接域：存放指向下一个结点的指针

——结点间的关系。

元素域 + 链接域 = 结点（链点）

抽象描述



- 链表的抽象描述
  - head(L)，length(L)，tail(L)
- 链点的抽象描述
  - key(*x*)，next(*x*)，prev(*x*)
  - NIL表示空

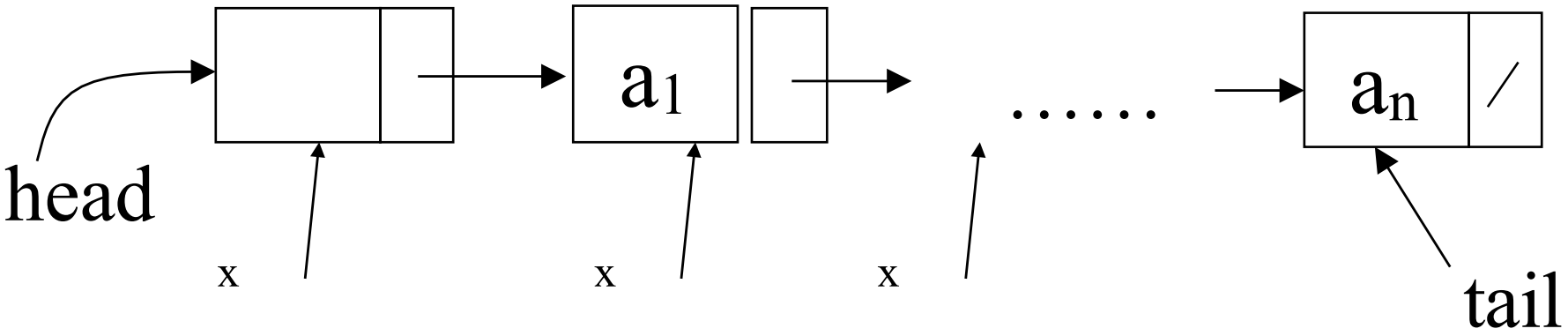
线性表的链接存储实现

- 数据表示
  - L: 数据链，长度k
- 接口定义
  - LENGTH(L)
  - GET(L, i)
  - INSERT(L, i, x)
  - DELETE(L, i)
- 举例

链表的访问操作GET(*L*, *i*)

- 访问操作
  - 问题描述：访问链表的第i个节点
  - 问题分析：
    - 输入：链表，i
    - 输出：链点——指向链点的指针
  - 算法实现分析：

只能从链表头开始，一个一个“数”下去，直到第i个。



链表的访问操作GET(*L*, *i*)

```
x ← head(L)
n ← 0

while x ≠ NIL and n < i do
     x ← next(x)
     n ← n + 1
end

return x
```

访问操作

- **注意**

  1、x ← next(x) ;

  沿链表前进

  2、循环结束条件

  counter == i 或 node == NIL

思 考

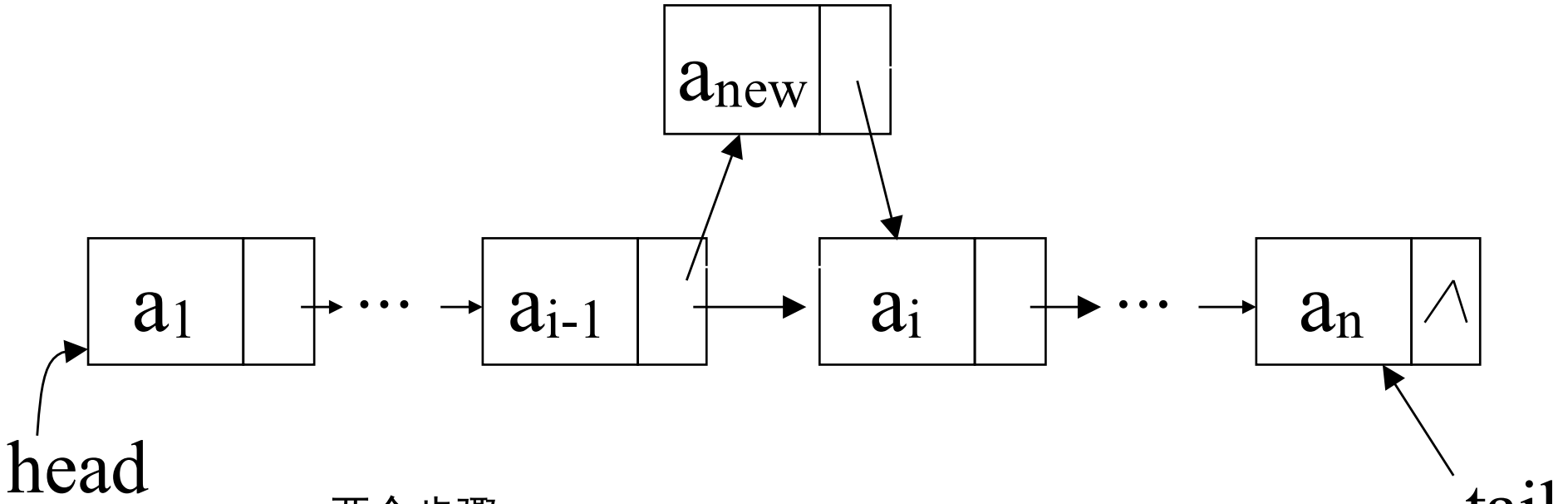
如果希望获得值为k的元素，如何实现？

while(x ≠ NIL and key(x) != k)

插入操作INSERT(L, i, k)

- 链表插入操作
  - 问题描述：
    - 在元素a_i前插入新的元素new_node；
  - 问题分析：
    - 输入：链表，location，x
    - 输出：插入新元素后的链表。
  - 算法实现分析

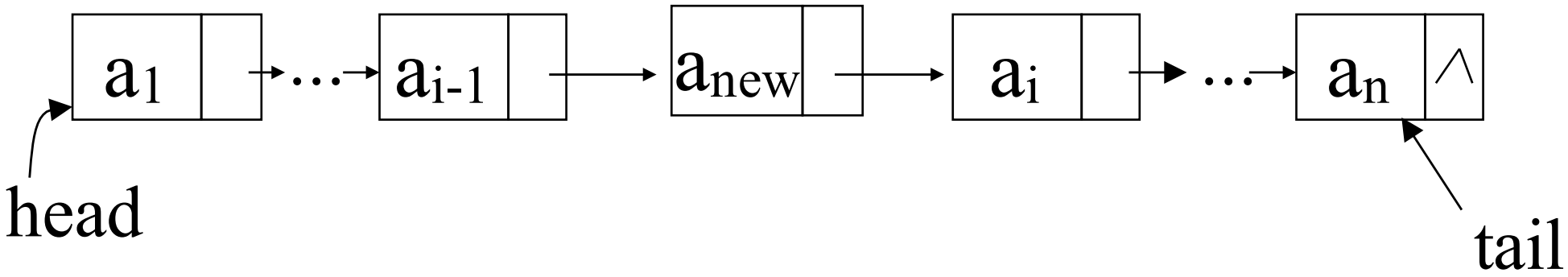
插入操作



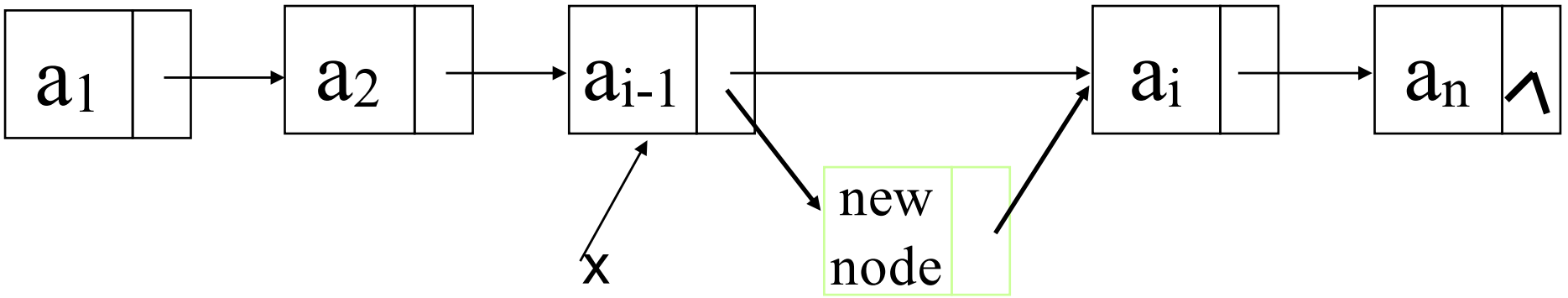
两个步骤：

$next(a_{new}) \leftarrow a_i$

$next(a_{i-1}) \leftarrow a_{new}$



插入操作 INSERT(L, i, k)




```
x ← GET(L, i - 1)

next(k)← next(x)
next(x)← k

LENGTH(L)← LENGTH(L) + 1
```

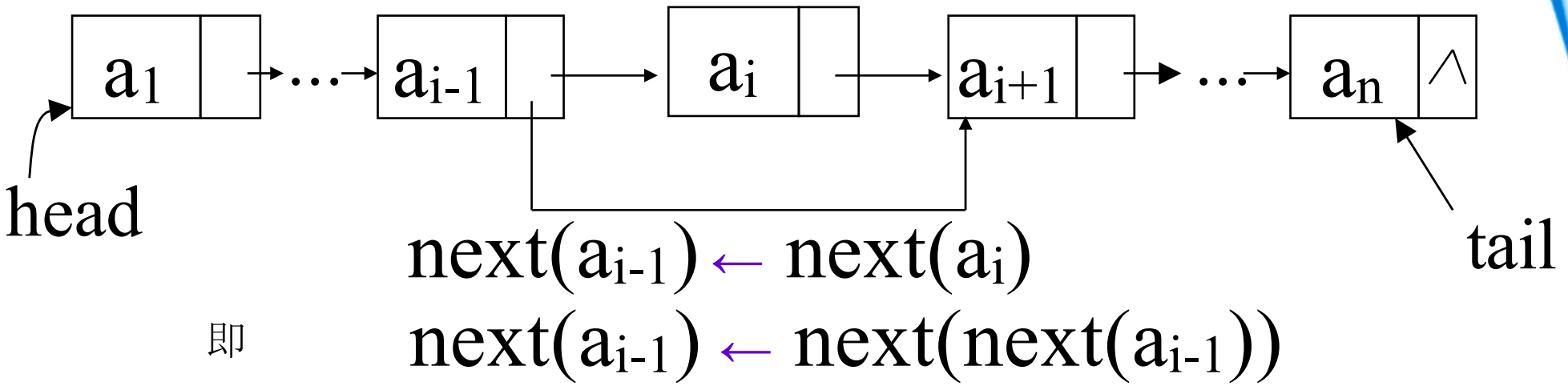
从插入算法中对链表操作的体会

- 1、链表操作往往从表头开始，逐个找到需要的链点
- 2、链表操作的有向性

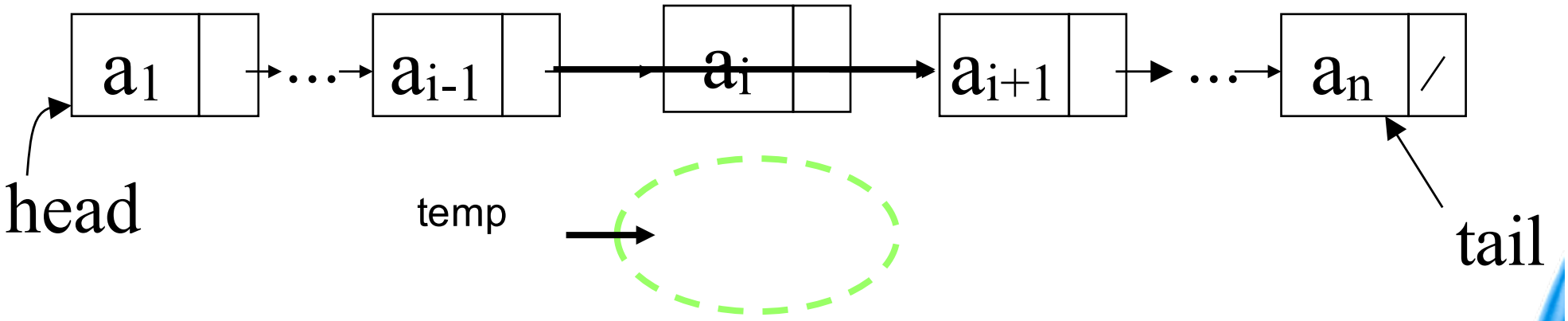
  不能回退；
- 3、链表指针小心使用，谨防丢失。
- 4、插入过程没有进行链点内容进行搬移。

删除操作

- 链表的删除操作
  - 问题描述：删除元素a_i；
  - 问题分析：
    - 输入：链表，location
    - 输出：删除元素后的链表。
  - 算法实现分析



删除操作 DELETE(L, i)




```
x ← GET(L, i - 1)

next(x)← next(next(x))

LENGTH(L)← LENGTH(L) - 1
```

删除操作

- **注意:**
  - **对删除链点的处理**

    从链表上取下的链点a_{i-1}需要挂在一个指针上，否则可能丢失
  - **一般需要释放存储空间**

思考

如果希望删除值为x的元素，如何实现？

链表的特点

- 1.2.4 链表的特点
  - 1、操作的顺序性
    - 有平均N／2次查找过程。
  - 2、离散存放
    - 不受链表大小限制
    - 不进行链点内容的搬移
  - 访问操作：数组效率优于链表
  - 插入、删除操作：链表效率优于数组

线性表抽象数据类型

- 数学模型
- 数据定义

$$L:(a_1, a_2, \cdots a_{i-1}, a_i, a_{i+1}, \cdots, a_n)$$

- 接口声明
  - `LENGTH(L)`
  - `GET(L, i)`
  - `INSERT(L, i, x)`
  - `DELETE(L, i)`

线性表抽象数据类型的两种存储实现

- 抽象数据类型的意义
  - 使用与其存储结构（内部表示）分离
  - 问题分解
  - 代码重用
- 顺序存储
  - 简单
  - 随机访问高效
- 链接存储
  - 灵活
  - 增减结点方便