软件技术基础

4.2 顺序表 Sequential Lists

郝家胜

hao@uestc.edu.cn

自动化工程学院

电子科技大学

1

内容提要

- 线性表的顺序存储
  - 顺序表的表示
  - 顺序表的操作
- 线性表的链接存储
  - 链表的表示
  - 链表的操作

2

内容回顾

- 线性表的概念
- 线性表的抽象数据类型
- 线性表抽象数据类型的运算示例

3

线性表的存储

- 顺序存储
- 连接存储

4

顺序表

线性表的顺序表示又称为**顺序存储结构**或顺序映像。

顺序存储定义：把逻辑上相邻的数据元素存储在物理上相邻的存储单元中的存储结构。

特点：逻辑上相邻的元素，物理上也相邻

顺序存储方法用一组地址连续的存储单元依次存储线性表的元素。

可以用数组类型来实现

5

线性表顺序存储特点：

- 逻辑上相邻的数据元素，其物理上也相邻；
- 若已知表中首元素在存储器中的位置，则其他元素存放位置亦可求出。

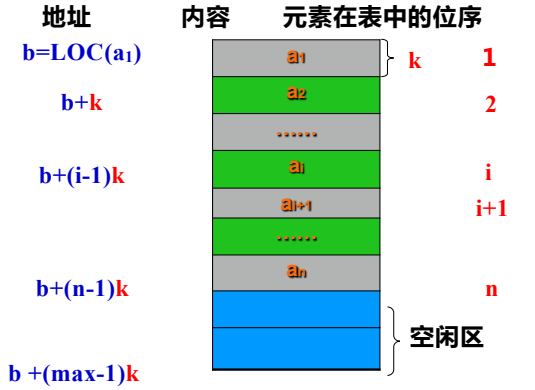
设首元素a_1的存放地址为$LOC(a_1)$（称为首地址），设每个元素占用存储空间（地址长度）为k字节，则表中任一数据元素的存放地址为：

$$LOC(a_i) = LOC(a_1) + k * (i-1)$$

对上述公式的解释如图所示

6

线性表的顺序存储结构示意图



$$LOC(a_i) = LOC(a_1) + k * (i-1)$$

7

例1 设有一维数组M，下标的范围是0到9，每个数组元素用相邻的8个字节存储。存储器按字节编址，设存储数组元素M[0]的第一个字节的地址是128，则M[3]的第一个字节的地址是多少？

152

解：已知地址计算通式为：

$$LOC(a_i) = LOC(a_1) + L * (i-1)$$

但此题要注意下标起点略有不同：

$LOC(M_{[3]}) = 128 + 8 \times 3 = 152$

8

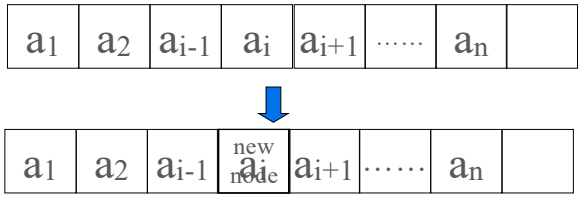
顺序表的实现

- 数据表示
  - L: 数组 a[n]，长度k
- 接口定义
  - LENGTH(L)
  - GET(L, i)
  - INSERT(L, i, x)
  - DELETE(L, i)

9

顺序表的插入操作 INSERT(L, i, x)

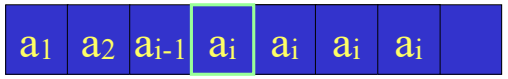
- 问题描述
  - 以a_1开始的线性表中
  - 在第i个元素前插入一个新元素new_node。



10

插入操作

- 从a_i开始向后移动



- 从a_n开始向前每个元素向后移一格

11

插入操作

在线性表的第i个位置前插入一个元素

算法步骤：

- 将第n至第i 位的元素向后移动一个位置；
- 将要插入的元素写到第i个位置；
- 表长加1。

注意：事先应判断: 插入位置i 是否合法?表是否已满?

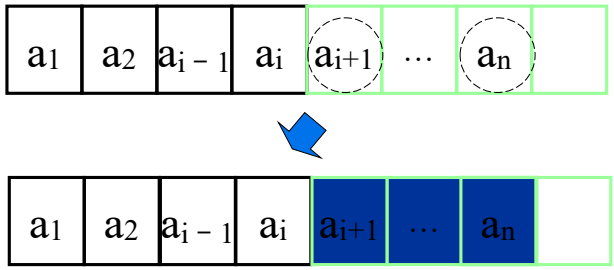
伪代码描述

```
j ← LENGTH(L)
while j >= i do
    a[j + 1] ← a[j]
    j ← j - 1
end
LENGTH(L)← LENGTH(L) + 1
```

12

删除操作 DELETE(L, i)

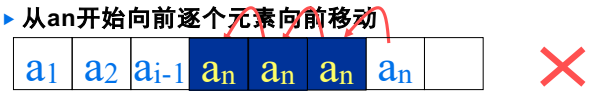
- 问题描述：删除第i个元素
  - 算法实现分析



13

删除操作

- 算法实现分析
  - 从an开始向前逐个元素向前移动



  - 从a_{i+1}开始向后逐个元素向前移动

14

删除操作

删除线性表的第i个位置上的元素

算法步骤：

- 将第i+1 至第n 位的元素向前移动一个位置；
- 表长减1。

注意：事先需要判断，删除位置i 是否合法?

伪代码描述

```
j ← i + 1
while j <= LENGTH(L) do
    a[j - 1] ← a[j]
    j ← j + 1
end
LENGTH(L)← LENGTH(L) - 1
```

15

顺序表的运算效率分析

时间效率分析:

- 算法时间主要耗费在移动元素的操作上，因此计算时间复杂度的基本操作（最深层语句频度）

$T(n)= f$ (移动元素次数)

而移动元素的个数取决于插入或删除元素的位置。

讨论1：若在长度为 n 的线性表的第 i 位前 插入一个元素，则向后移动元素的次数f(n)为:

$$f(n) = n - i + 1$$

思考：若插入在尾结点之后，则根本无需移动（特别快）；

若插入在首结点之前，则表中元素全部要后移（特别慢）；

应当考虑在各种位置插入（共n+1种可能）的平均移动次数。

16

推导：假定在每个元素位置上插入x的可能性都一样（即概率P相同），则应当这样来计算平均执行时间：

将所有位置的执行时间相加，然后取平均。

若在首结点前插入，需要移动的元素最多，后移n次；
若在a_1后面插入，要后移n-1个元素，后移次数为n-1；
……
若在a_{n-1}后面插入，要后移1个元素；
若在尾结点a_n之后插入，则后移0个元素；

所有可能的元素移动次数合计： $0+1+\ldots+n = n(n+1)/2$

共有多少种插入形式?——连头带尾有n+1种!

故插入时的平均移动次数为：$n(n+1)/2 \div (n+1) = n/2$，即 O(n)

想一想：删除结点的平均执行时间是多少？

17

顺序表小结

线性表顺序存储结构特点：逻辑关系上相邻的两个元素在物理存储位置上也相邻；

优点：

1.存储密度大；
2.可以随机存取表中任一元素，方便快捷；

缺点：

1.在插入或删除某一元素时，需要移动大量元素;
2.需要一片连续的存储空间；
3.有“溢出”问题。

解决问题的思路：改用另一种线性存储方式：

链式存储结构

18

小 结

- 线性表的概念
- 线性表抽象数据类型的运算
- 顺序表实现
- 顺序表的操作效率

19