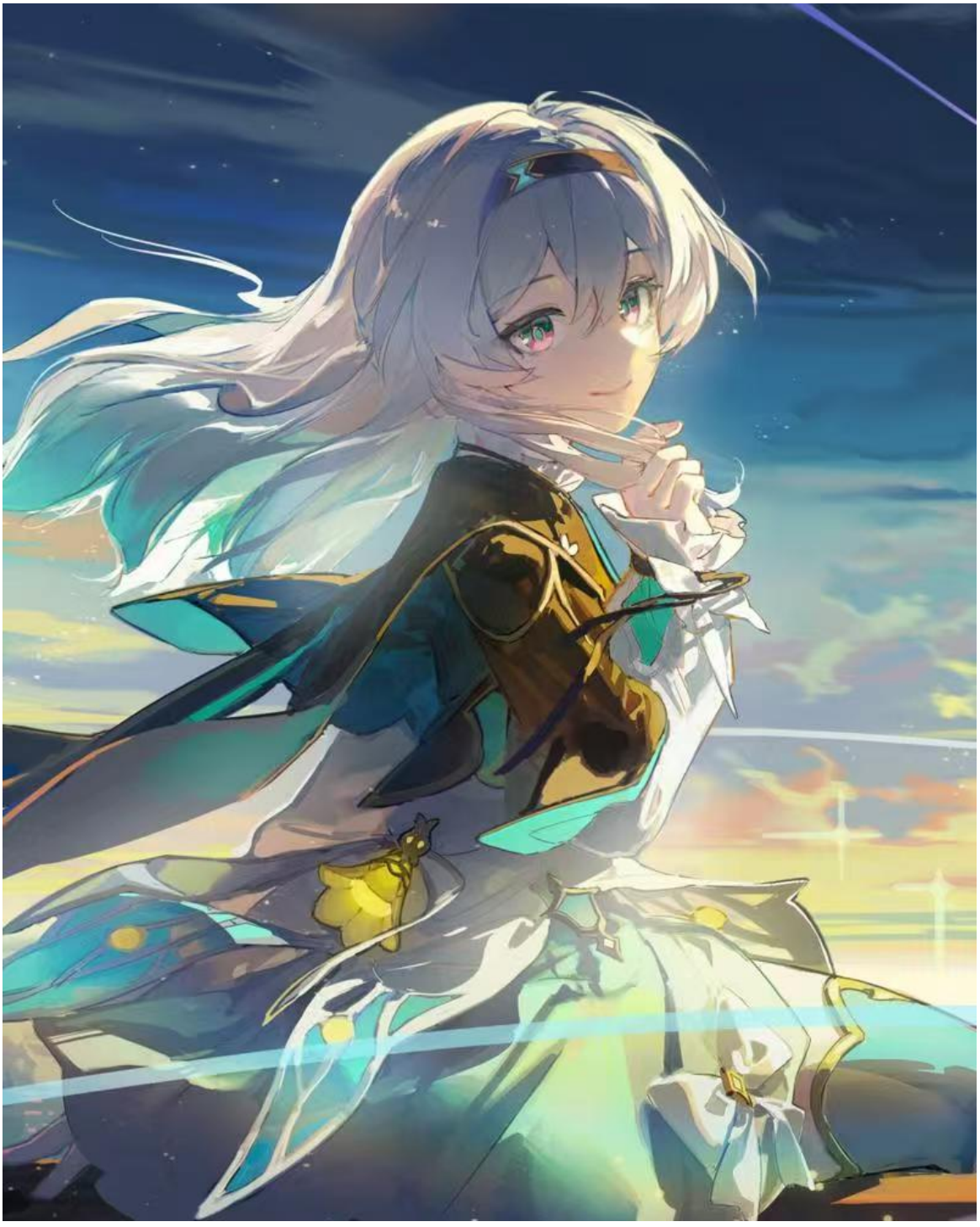
线性表

Tan Yiqing

2025 年 10 月 31 日



1 线性表的逻辑结构

1.1 定义

线性表，亦简称表，是 n 个句有相同类型的数据元素的有限序列。　　规定线性表的长度为表中元素个数 n，$n=0$ 时称为空表。写作:$L=(a_1,a_2,...,a_n)$，a_i 为表中第 i 个元素，1≤i≤n。空表写作:$L=()$。

1.2 特点

有限性：线性表中元素个数有限。
相同性：线性表中元素类型相同。
顺序性：线性表中元素是有序排列的。

1.3 功能

1.3.1 构造函数

- **前置条件**：表不存在
- **输入**：无
- **功能**：表的初始化
- **输出**：无
- **后置条件**：表存在且为空

1.3.2 析构函数

- **前置条件**：表存在
- **输入**：无
- **功能**：销毁表，释放动态申请的空间
- **输出**：无
- **后置条件**：已释放所有在这个表的操作申请过的动态空间

1.3.3 长度

- **前置条件**：表存在
- **输入**：无
- **功能**：回传表的长度
- **输出**：表的长度
- **后置条件**：表没有变化

1.3.4 是否空表

- **前置条件**：表存在
- **输入**：无
- **功能**：判断表是否为空 (若是空表则返回 true，否则返回 false)
- **输出**：如功能
- **后置条件**：表没有变化

1.3.5 插入元素

- **前置条件**：表存在
- **输入**：元素值 x，插入位置 i
- **功能**：把元素 x 插入到表的第 i 个位置。如 $L=(a,b,c)$,$i=1$, 则插入后 $L=(a,x,b,c)$。
- **输出**：若插入失败则抛出异常
- **后置条件**：若插入成功，表的第 i 个位置为 x，表的长度加 1。

1.3.6 删除元素

- **前置条件**：表存在
- **输入**：整数 i
- **功能**：把表的第 i 个位置的元素删除。如 $L=(a,b,c)$,$i=1$, 则删除后 $L=(a,c)$。
- **输出**：若删除成功，回传删掉的元素值；若删除失败则抛出异常
- **后置条件**：若删除成功，表中元素减少一个，表的长度减 1。

1.3.7 查找所在位置的元素

- **前置条件**：表存在
- **输入**：整数 i
- **功能**：回传表中第 i 个位置的元素。如 $L=(a,b,c)$,$i=1$, 则返回 b。
- **输出**：若 i 合法，回传表中第 i 个位置的元素；否则抛出异常
- **后置条件**：表没有变化

1.3.8 查找元素所在位置

- **前置条件**：表存在
- **输入**：元素值 x
- **功能**：查找元素 x 在表中的首次出现的位置；若不存在，回传-1。
- **输出**：如功能
- **后置条件**：表没有变化

2 顺序表

2.1 定义

顺序表的存储结构是用一个数列存储所有数据，另外用一个整数变量存储表的长度。这是一个连续的存储结构，无需特别写出前驱后继关系。

可以认为是一维数组加上一堆功能。

注：需要数列的长度，如长度为 1000，在插入第 1001 个元素时可能会出问题。

2.2 实现

2.2.1 初始化、销毁、长度、是否空表

- **初始化**：申请一个长度为 maxSize 的数组，length=0
- **销毁**：释放数组空间, 这里因为没东西，所以啥都不用做。
- **长度**：返回 length
- **是否空表**：判断 length 是否为 0 注：上述时间复杂度均为 O(1)

2.2.2 插入元素

- 先把 $(a_i, ..., a_{n-1})$ 后移一位，然后把 x 放到 a_i 的位置上，n++。注：时间复杂度为 O(n), 与位置 i 和表长 n 有关，具体来说是 $n-i+1$，平均复杂度（认为 i 是均匀分布）为 O(n/2)=O(n)。如果要追求效率应该让 i 尽量大。
  有些 i 是非法的，比如 i<0 或 i>n。

2.2.3 删除元素

- 先把 $(a_{i+1}, ..., a_{n-1})$ 前移一位，然后 n–。注：时间复杂度为 O(n), 与位置 i 和表长 n 有关，具体来说是 $n-i$，平均复杂度（认为 i 是均匀分布）为 O(n/2)=O(n)。如果要追求效率应该让 i 尽量小。
  有些 i 是非法的，比如 i<0 或 i>n-1。而且 a_{n-1} 的位置的值是不变的。

2.2.4 查找

- **查找所在位置的元素**：当位置合理，直接返回 a_i；否则抛出异常。时间复杂度为 O(1)。
- **查找元素所在位置**：从前往后遍历数组，找到第一个等于 x 的元素，返回其位置；否则返回-1。时间复杂度为 O(n)。

2.3 优缺点

2.3.1 优点

- 不用特别处理元素的前驱后继关系，逻辑简单。
- 随机存储，时间复杂度为 O(1)。
- 插入和删除操作相对简单，时间复杂度为 O(n)。

2.3.2 缺点

- 容量限制，需要预先分配内存，可能导致空间浪费。
- 插入和删除操作需要移动大量元素，效率较低。、
- 碎片化问题，频繁的插入和删除操作可能导致内存碎片化，影响性能。

3 向量表

3.1 定义

- 向量表是顺序表的一种改进形式，基本操作和顺序表相同，但解决了顺序表容量限制的问题。
- 数列是动态申请的，初始化时首次申请数列的空间；因此销毁时要释放空间。
- 需要一个新的变量来储存目前表的容量。
- 主要的区别在于插入的处理：当表满时，当表满时，申请一个新的数列空间（一般为原容量的双倍），复制所有数据到新数列，然后把旧的数列空间释放。

4 链表

4.1 定义

数据的物理存储不一定按顺序，要想办法表示数据的前驱后继。于是我们对每个数据同时配上若干个指针来达到目的。单链表就是我们给每个数据配上一个指针，指向它的后继，最后一个指向空白。

4.2 实现

4.2.1 是否空表, 遍历

- **是否空表**：判断首指针是否为空。
- **遍历**：从头指针开始，依次访问每个节点，直到遍历完整个链表，指针指向空白。

4.2.2 插入元素

- **空表**：做一个新的节点，其指针指向空白，放入数据，首指针指向新的节点。
- **非空表**：找到第 i-1 个节点，做一个新的节点，其指针指向第 i 个节点，放入数据，然后把第 i-1 个节点的指针指向新的节点。时间复杂度为 O(n)，与位置 i 有关，平均复杂度为 O(n/2)=O(n)。注：自己想想一下框框图。

4.3 构造与销毁

- **构造函数**：申请一个头节点，指针指向空白。长度设为 0。
- **析构函数**：从头节点开始，依次释放每个节点，直到指针指向空白。具体来说是：暂存首指针指向的节点至 K，首指针指向 K 的后继，释放 K，重复直到首指针指向空白。

4.4 长度

- 在没有长度的私有变量的情况下。从头节点开始，依次访问每个节点，直到遍历完整个链表，指针指向空白。时间复杂度为 O(n)。

4.5 插入

- 把数据 x 插入第 i 个位置。
- 遍历到第 i-1 个节点，记为 N。
- 做一个新的节点，其指针指向第 i 个节点，放入数据，然后把第 i-1 个节点的指针指向新的节点。注：时间复杂度为 O(n)，与位置 i 有关，平均复杂度为 O(n/2)=O(n)。

4.6 删除

- 把第 i 个节点删除。
- 遍历到第 i-1 个节点，记为 N。
- 用一个节点指针 K 暂存 N 的指针指向的节点 (即第 i 个节点)。
- 把 N 的指针指向 K 指向的下一个节点，即第 i 个节点的后继，释放第 i 个节点。注：时间复杂度为 O(n)，与位置 i 有关，平均复杂度为 O(n/2)=O(n)。

4.7 查找

- 每个节点遍历，直到找到第 i 个节点，返回其数据。

5 题外话：算法设计

- 确定输入输出。
- 写例子。
- 正反面考虑。
- 大致思路，边界考虑。
- 验证。

6 顺序表和链表的比较

- 存储结构：顺序表是连续存储，链表是非连续存储。
- 空间利用率：顺序表可能会浪费空间，链表空间利用率高。
- 访问速度：顺序表支持随机访问，链表只能顺序访问。
- 前者适合频繁查找，后者适合频繁插入删除。

7 链表算法提升

7.1 双链表

每个节点配两个指针，分别指向前驱与后继。提高查找链表结点前驱的效率。

7.2 静态链表

使用数组存取一堆结点，结点之间的指针用数组位置取代。

7.3 循环链表

把尾节点指向首节点，做成单循环链表；把尾的结点的后继指向首数据的结点，并把首数据的前驱指向尾结点，做成双循环链表。

7.4 带头节点的链表 (ppt 中的小技巧，也叫哨兵节点)

在链表的最前面加一个不存数据的节点，作为头节点。

7.5 间接寻址

这是对于顺序表的一个改版，生成的数列不直接存数据，而是存一个指向数据的指针。这个改版的好处是插入、删除移动的只是这堆指针，而不是整个数据。（对于每个数据元素比较庞大时，这个确保了移动量很小）。