顺序表和链表

本节目标

1.线性表

2.顺序表

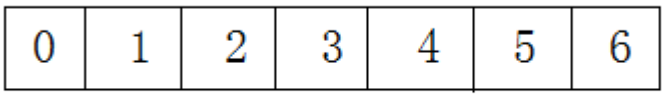
3.链表

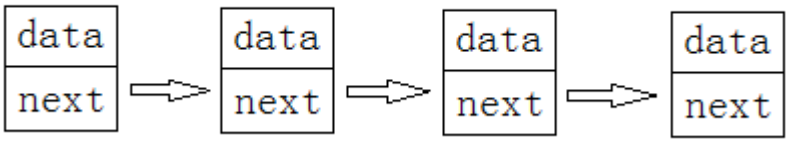
4.顺序表和链表的区别和联系

1. 线性表

线性表 *(linear list)* 是n个具有相同特性的数据元素的有限序列。 线性表是一种在实际中广泛使用的数据结构，常见的线性表：顺序表、链表、栈、队列、字符串...

线性表在逻辑上是线性结构，也就说是连续的一条直线。但是在物理结构上并不一定是连续的，线性表在物理上存储时，通常以数组和链式结构的形式存储。





2. 顺序表

2.1 概念及结构

顺序表是用一段**物理地址连续**的存储单元依次存储数据元素的线性结构，一般情况下采用数组存储。在数组上完成数据的增删查改。

顺序表一般可以分为：

- 静态顺序表：使用定长数组存储。
- 动态顺序表：使用动态开辟的数组存储。

静态顺序表适用于确定知道需要存多少数据的场景.

静态顺序表的定长数组导致N定大了，空间开多了浪费，开少了不够用.

相比之下动态顺序表更灵活, 根据需要动态的分配空间大小.

2.2 接口实现

我们来实现一个动态顺序表. 以下是需要支持的接口.

```
public class SeqList {
    // 打印顺序表
    public void display() {    }
    // 在 pos 位置新增元素
    public void add(int pos, int data) {  }
    // 判定是否包含某个元素
    public boolean contains(int toFind) { return true; }
    // 查找某个元素对应的位置
    public int search(int toFind) { return -1; }
    // 获取 pos 位置的元素
    public int getPos(int pos) { return -1; }
    // 给 pos 位置的元素设为 value
    public void setPos(int pos, int value) {    }
    //删除第一次出现的关键字key
    public void remove(int toRemove) {    }
    // 获取顺序表长度
    public int size() { return 0; }
    // 清空顺序表
    public void clear() {    }
}
```

2.3 顺序表的问题及思考

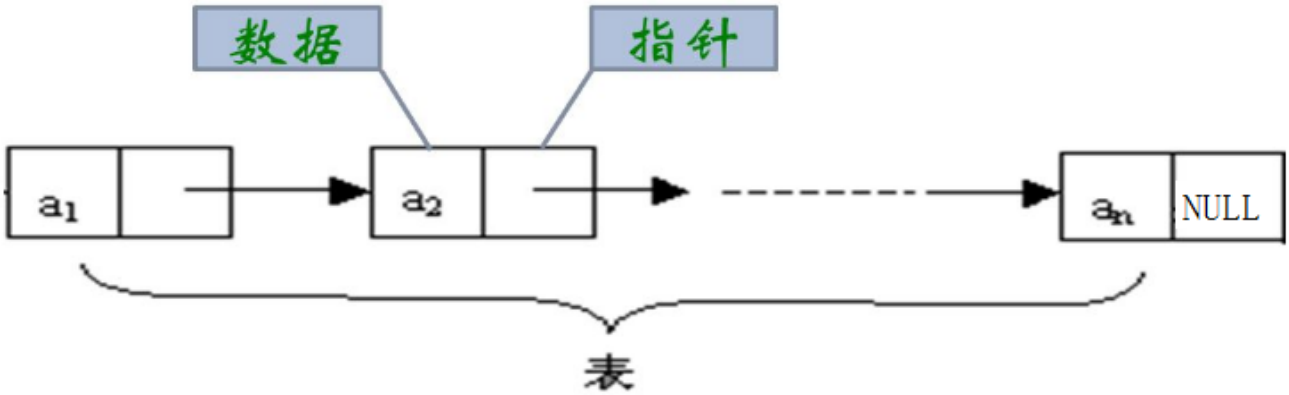
1. 顺序表中间/头部的插入删除，时间复杂度为O(N)
2. 增容需要申请新空间，拷贝数据，释放旧空间。会有不小的消耗。
3. 增容一般是呈2倍的增长，势必会有一定的空间浪费。例如当前容量为100，满了以后增容到200，我们再继续插入了5个数据，后面没有数据插入了，那么就浪费了95个数据空间。

思考： 如何解决以上问题呢？下面给出了链表的结构来看看。

3. 链表

3.1 链表的概念及结构

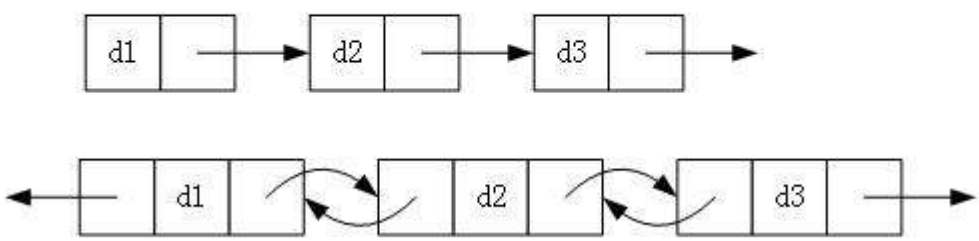
链表是一种**物理存储结构上非连续**存储结构，数据元素的**逻辑顺序**是通过链表中的**引用链接**次序实现的 。



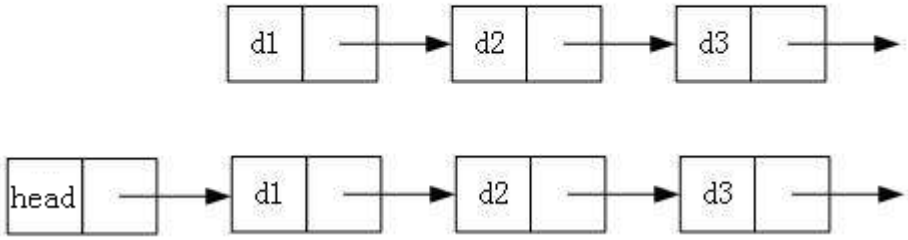
实际中链表的结构非常多样，以下情况组合起来就有8种链表结构：

- 单向、双向
- 带头、不带头
- 循环、非循环

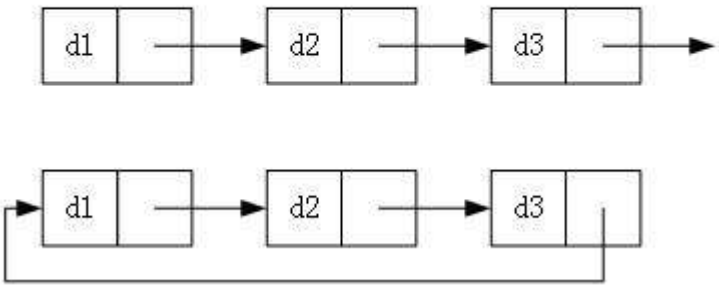
1. 单链表、双向链表



2. 不带头单链表、带头链表

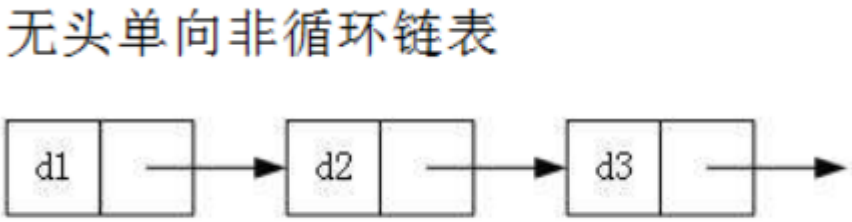


3. 单链表、循环单链表

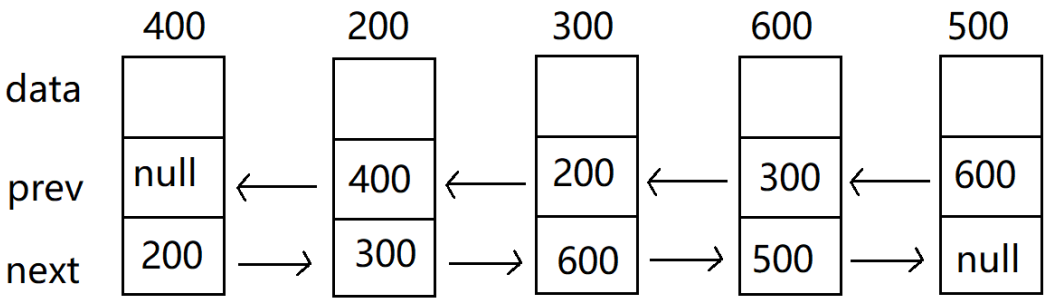


虽然有这么多的链表的结构，但是我们重点掌握两种:

- 无头单向非循环链表：**结构简单**，一般不会单独用来存数据。实际中更多是作为**其他数据结构的子结构**，如哈希桶、图的邻接表等等。另外这种结构在**笔试面试**中出现很多。



- 无头双向链表：在Java的集合框架库中LinkedList底层实现就是无头双向循环链表。



说明：400,200等均为节点的引用(地址)，本应是16进制，为了画图方便，这里进行了简写 .

3.2 链表的实现

```
// 1、无头单向非循环链表实现
public class SingleLinkedList {
    //头插法
    public void addFirst(int data);
    //尾插法
    public void addLast(int data);
    //任意位置插入,第一个数据节点为0号下标
    public boolean addIndex(int index,int data);
    //查找是否包含关键字key是否在单链表当中
    public boolean contains(int key);
    //删除第一次出现关键字为key的节点
    public void remove(int key);
    //删除所有值为key的节点
    public void removeAllKey(int key);
    //得到单链表的长度
    public int size();
    public void display();
    public void clear();
}
```

```
// 2、无头双向链表实现
public class DoubleLinkedList {
    //头插法
    public void addFirst(int data);
    //尾插法
    public void addLast(int data);
    //任意位置插入,第一个数据节点为0号下标
    public boolean addIndex(int index,int data);
    //查找是否包含关键字key是否在单链表当中
    public boolean contains(int key);
    //删除第一次出现关键字为key的节点
    public void remove(int key);
    //删除所有值为key的节点
    public void removeAllKey(int key);
    //得到单链表的长度
    public int size();
    public void display();
    public void clear();
}
```

3.3 链表面试题

1. 删除链表中等于给定值 **val** 的所有节点。 OJ链接
2. 反转一个单链表。 OJ链接

3. 给定一个带有头结点 head 的非空单链表，返回链表的中间结点。如果有两个中间结点，则返回第二个中间结点。OJ链接
4. 输入一个链表，输出该链表中倒数第k个结点。 OJ链接
5. 将两个有序链表合并为一个新的有序链表并返回。新链表是通过拼接给定的两个链表的所有节点组成的。OJ链接
6. 编写代码，以给定值x为基准将链表分割成两部分，所有小于x的结点排在大于或等于x的结点之前 。OJ链接
7. 在一个排序的链表中，存在重复的结点，请删除该链表中重复的结点，重复的结点不保留，返回链表头指针。 OJ链接
8. 链表的回文结构。OJ链接
9. 输入两个链表，找出它们的第一个公共结点。OJ链接
10. 给定一个链表，判断链表中是否有环。 OJ链接
11. 给定一个链表，返回链表开始入环的第一个节点。 如果链表无环，则返回 null OJ链接
12. 其他 。ps：链表的题当前因为难度及知识面等等原因还不适合我们当前学习，以后大家自己下去以后Leetcode OJ链接 + 牛客 OJ链接

https://leetcode-cn.com/problems/copy-list-with-random-pointer/submissions/

4. 顺序表和链表的区别和联系

顺序表：一白遮百丑

白：空间连续、支持随机访问

丑：1.中间或前面部分的插入删除时间复杂度O(N) 2.增容的代价比较大。

链表：一(胖黑)毁所有

胖黑：以节点为单位存储，不支持随机访问

所有：1.任意位置插入删除时间复杂度为O(1) 2.没有增容问题，插入一个开辟一个空间。

内容重点总结

- 掌握顺序表和链表的基本增删查改操作。
- 掌握课件所列的在线OJ题目的解答。

课后作业

将顺序表链表相关代码及OJ题型全部进行熟练编写，并在LeetCode上和牛课上Leetcode OJ链接 + 牛客 OJ链接进行正对性训练。