1. 列表

顺序存储、列表中存储的是地址，地址的长度都为4 byte.，所以列表中不要求元素类型相同，因为存的地址。长度可以无限扩。

Append:O(1)、insert:O(n)、delete:O(n)。

C的数组怎么存储：a[2]=首地址+4字节\*2，查找a[2]的时间复杂度为0(1)。不同之处：

1. 一个数组中元素类型一样。
2. 数组长度固定。
3. 栈Stack

是一个数据集合，只能在一端进行插入或者删除的列表。

后进先出（last-in,first-out,LIFO）。栈底（a[0]）、栈顶a[n]

进栈（压栈）push：li.append、 出栈pop：li.pop、 取栈顶gettop：li[-1]。

1. 队列

是一个数据集合，仅允许一端插入，另一端删除。

插入的一端叫队尾rear，进队、入队。 删除的一端叫队头队首front，出队。

先进先出（FIFO）

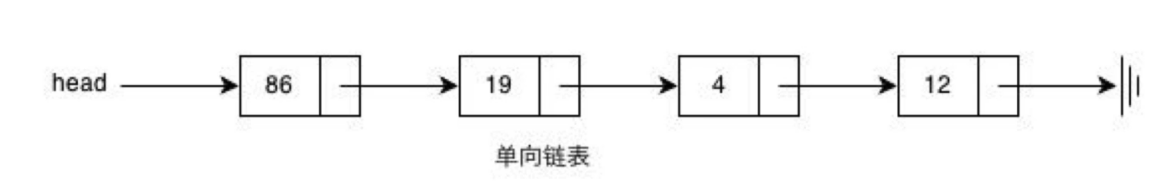
环形队列、队满（（rear+1）%队的长度=fronts牺牲一个空，用来区分队空和队满）；队空（rear=fronts）

双向队列：

队列的内置模块：

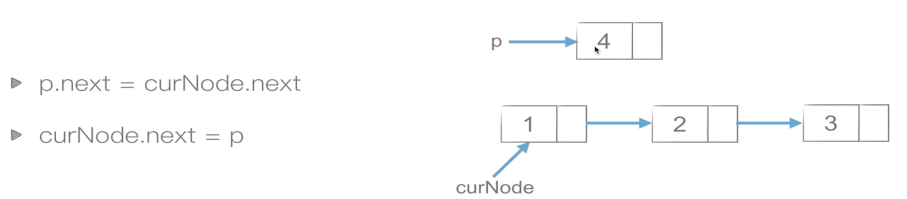
1. 链表

由一系列节点组成的元素集合。每个节点包含两部分：数据域item 和 指向下一个节点的指针next，，通过节点之间的相互连接，最终串联成一个链表。

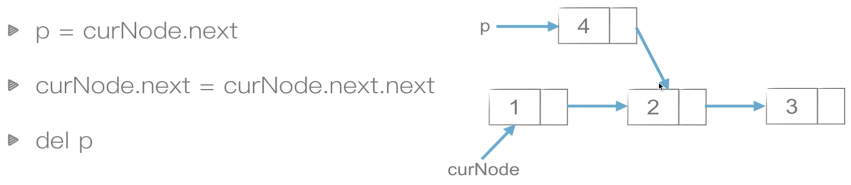


创建链表：头插法、尾插法

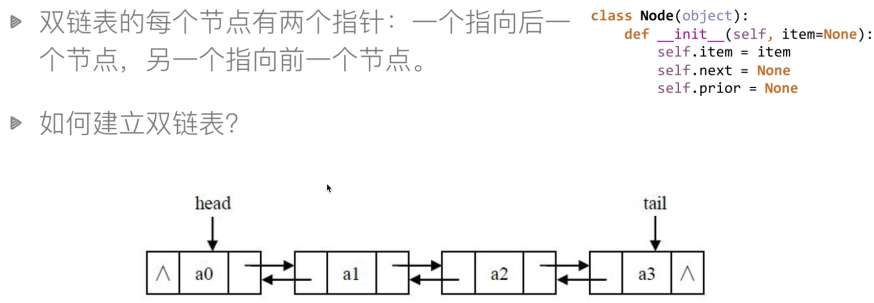
链表的插入：O(1)



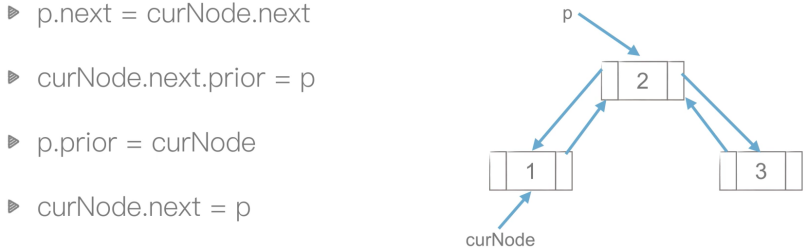
链表的删除：



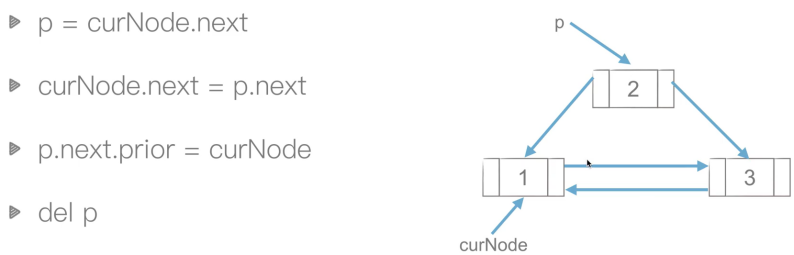
双链表：



双链表的插入：



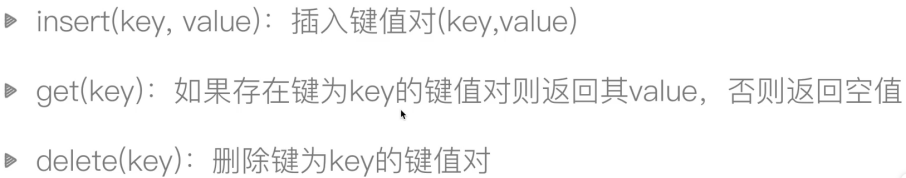
双链表的删除：



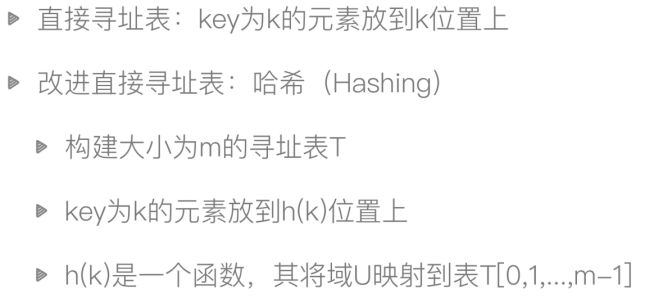
链表对图和树具有启发意义。

1. 哈希表（字典和集合）

通过哈希函数来计算数据存储位置的数据结构。



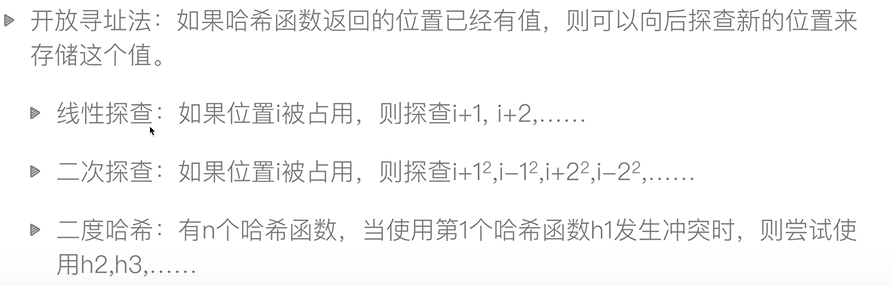
哈希表是改进的直接寻址：是一种线性表的存储结构，由一个直接寻址表和一个哈希函数组成。哈希函数h(k)将关键字k作为自变量，返回元素的存储下标。

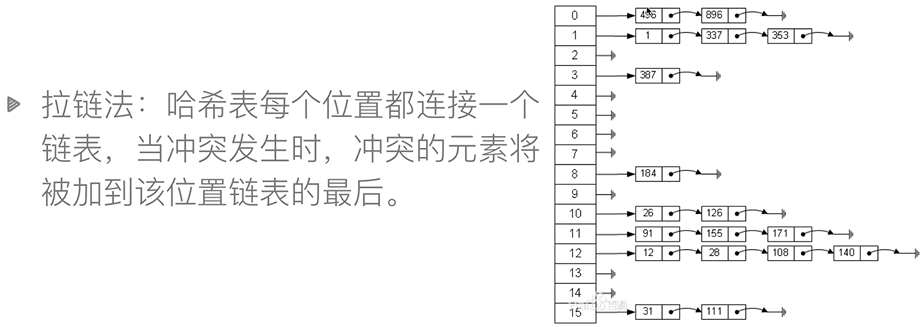


除法哈希、乘法哈希、全域哈希。

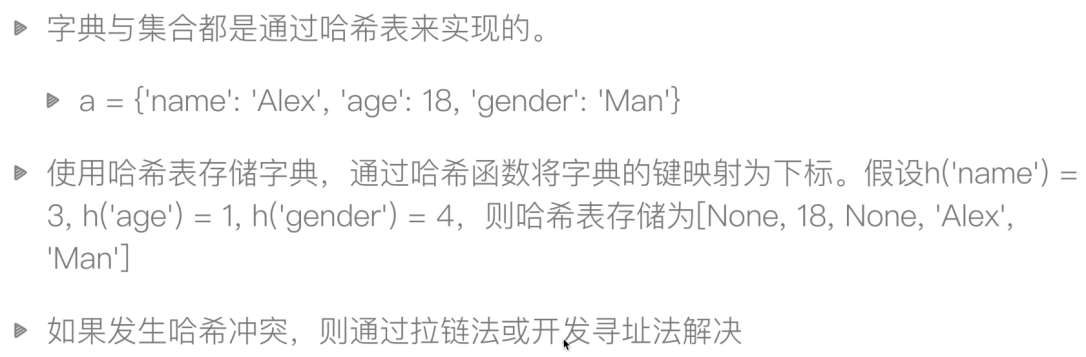
哈希冲突：对于任何哈希函数，都会出现两个不同元素映射到同一位置的情况。

解决哈希冲突的方法：开放寻址法、拉链法（遍历链表）

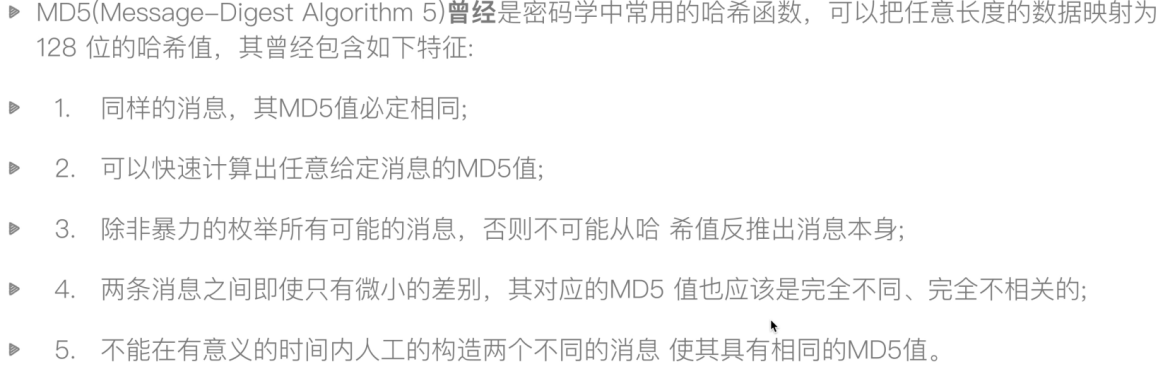




哈希表的应用：



应用——md5算法 Message-Digest：把任意长度的数据映射为128位的哈希值



有哈希就一定有哈希冲突，所以即使哈希值一样，也无法说两个原始数据一样。

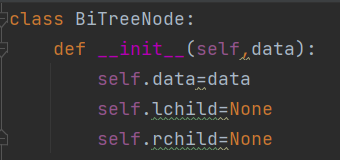
1. 树

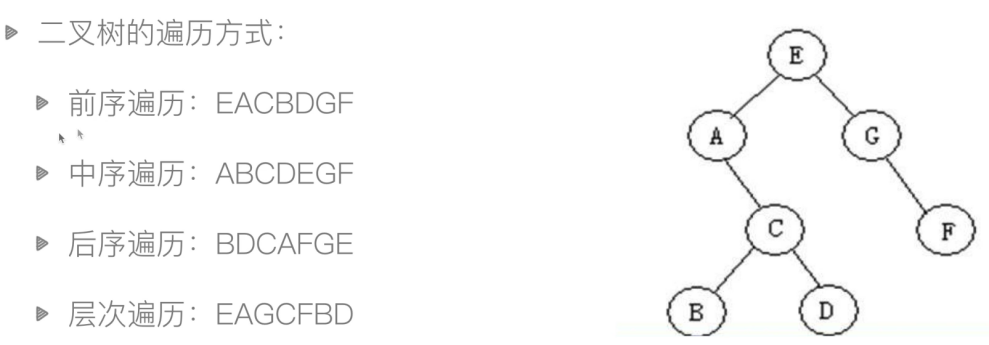
根节点、叶子节点、树的度、孩子节点/父节点、子树。

树的实例：模拟文件系统 cd、ls、mkdir，链式存储。

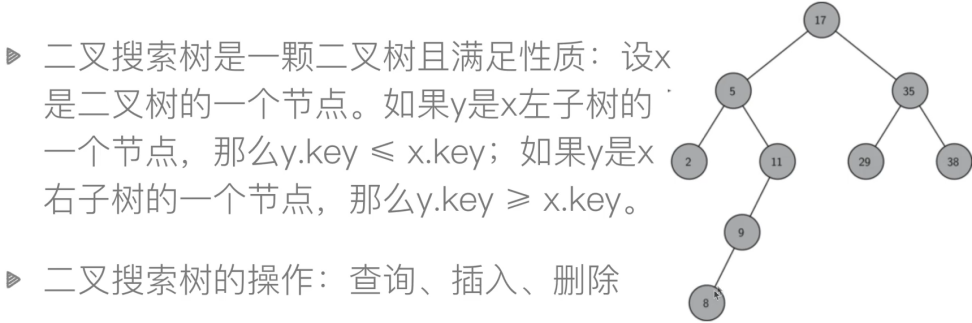
1. 二叉树

将二叉树的节点定义为一个对象，节点之间通过类似链表的连接方式来连接。



二叉树的遍历：一旦给了两个遍历，就能推出一整棵树。

二叉搜索树：



在这种情况下，中序遍历出来的相当于从小到大排序。