链表

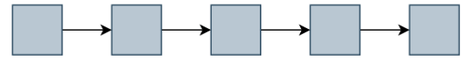
定义

链表（Linked List）是一种线性表数据结构，使用非连续的内存空间，存储相同或者不同类型的元素，但不要求元素大小相等

使用链表存储不同的元素视具体语言而定，如Java、Go等静态语言只能存储同类型元素，Python等动态语言可使用鸭子类型(Duck Typing)来存储不同类型的元素

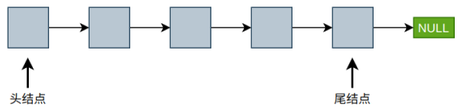
通常来说，链表内元素类型是固定的，元素大小可动态变化

链表就像使用一根绳子串起来的多个浮标一样，浮标之间的距离可以任意，大小也可以任意，只不过它们都在同一根绳子上



链表相较于数组的最大区别就在于链表使用非连续的进程空间，链表中的元素通常称为结点，结点中包含指向下一个结点的指针以及结点本身保存的数据

链表结构



在链表中，有两个结点较为特殊。一个是头结点，即链表的第一个结点，它没有前驱结点。一个是尾结点，它没有后驱结点，指针指向NULL

链表元素操作

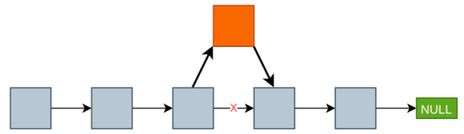
随机访问

链表的随机访问(例如，找到第五个元素)性能较差，其原因在于元素分布于非连续的内存空间。那么每次进行随机访问时，都只能从头开始遍历，直到第5个元素为止

平均复杂度为O(n)

插入元素

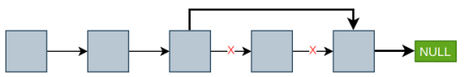
在链表中插入一个元素非常简单，不需要对元素进行相应的移动，只需要操作结点指向的指针即可



单就插入元素这一操作来说，其时间复杂度为O(1)

删除元素

同插入元素一样，链表元素的删除也只需要操作结点的指针指向



单就删除元素这一操作而言，其时间复杂度也是O(1)

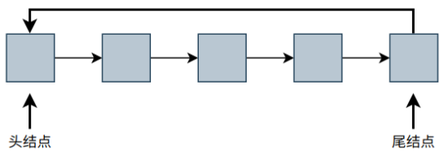
分类

单向链表

单向链表如上图所示，结点指针要么指向下一个结点，要么指向NULL

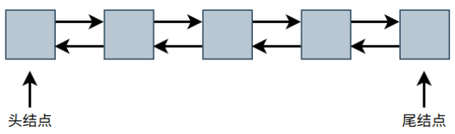
循环链表

循环链表相较于单链表而言，其尾结点指针并不指向NULL，而是指向头结点



双向链表

双向链表是实际应用中使用最为频繁的链表，每个结点有两个指针，一个指向下一个结点，另一个指向前一个结点



和数组的比较

随机访问

数组

数组内的元素存在于一连续的内存空间，且数组内元素大小相同，那么可以很方便地利用一次公式计算得到寻找元素的内存地址，进行直接读取

并且，由于CPU多级缓存的存在，下一个随机访问的数组元素很有可能已经存在于CPU缓存中，可进一步加快读取效率

CPU的多级缓存会缓存一页内存或者相邻的多页内存

链表

链表的元素由于可能分散在内存空间的各个位置，所以只能使用逐一遍历的方式进行查找，且大概率无法从CPU多级缓存中获益

查找特定元素

数组和链表在查找特定元素时的平均时间复杂度均为O(n)，都需要从头开始遍历

但是，如上文所提，数组的顺序访问可利用CPU多级缓存机制，而链表则大概率无法使用，所以数组的查找速度仍然优于链表

插入元素与删除元素

数组和链表在插入元素以及删除元素上，平均时间复杂度均为O(n), 数组更多的时间花费在移动元素上，而链表则将多数时间花费在随机访问元素上

链表的应用

实现双向队列

队列是一种先进先出数据结构，若采用数组实现，元素的插入或者是删除总是一个操作的平均时间复杂为O(n)，而采用双向链表，元素的入队和出队操作的时间复杂度均可降低为O(1)

实现LUR缓存淘汰算法

LRU(最近最少使用)算法多用于缓存数据操作，由于内存容量有限，所以当有新的数据进入缓存时，将最近最少使用的数据从缓存中移除

同样是利用了双向链表高效的头、尾插入与删除操作的特性