# ### Array vs.Linked List

Array: 通过index访问各个元素 --> O(1) <br>

# 数组 顺序存储

# 由于是紧凑连续存储,可以随机访问，通过索引快速找到对应元素，而且相对节约存储空间。但正因为连续存储，内存空间必须一次性分配够，所以说数组如果要扩容，需要重新分配一块更大的空间，再把数据全部复制过去，扩容时间复杂度 O(N)；而且你如果想在数组中间进行插入和删除，每次必须搬移后面的所有数据以保持连续，增删时间复杂度 O(N)、改查时间复杂度 O(1)

# 链表 链式存储

# 因为元素不连续，而是靠指针指向下一个元素的位置，所以不存在数组的扩容问题；如果知道某一元素的前驱和后驱，操作指针即可删除该元素或者插入新元素，增删时间复杂度 O(1)。但是正因为存储空间不连续，你无法根据一个索引算出对应元素的地址，所以不能随机访问；而且由于每个元素必须存储指向前后元素位置的指针，会消耗相对更多的储存空间，改查时间复杂度 O(n)

# Linked List

A linked list is a collection of nodes. The first node is called the head, and it’s used as the starting point

for any iteration through the list. The last node must have its next reference pointing to None to determine the

end of the list.

# 总结

* LC 206 reversed linked list:

iteration 快慢指针 + temp指针, 但一般这种题用递归（recursion）更好

* fast = fast.next 移动指针迭代到下一个
* fast.next = slow 反转指针

Reference:

* labuladong算法小抄
* Linked Lists in Python: An Introduction <https://realpython.com/linked-lists-python/>
* 图灵星球
* 古城算法基础数据结构 (五) -- 链表(上) 反转 + 合并 + 找 <https://www.youtube.com/watch?v=gDxv49BGz0w>
* labuladong算法框架 Python版 <https://www.codeleading.com/article/83145179559/>