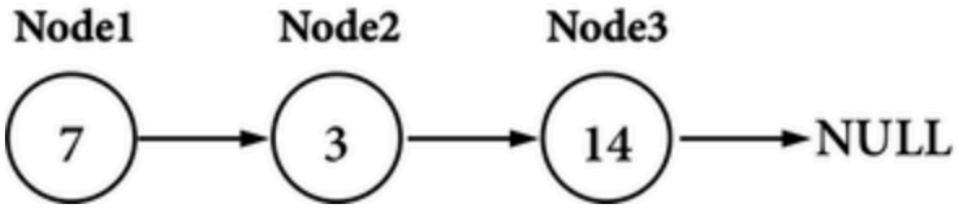
Linked list

Linked list(連結串列)是一種常見的資料結構，其使用node(節點)來記錄、表示、儲存資料(data)，並利用每個node中的pointer指向下一個node，藉此將多個node串連起來，形成Linked list，並以NULL來代表Linked list的終點。



Array與Linked list

Array(矩陣)也是常見的用來記錄一連串「具有相同形態的資料」的資料結構，以下便列舉Array與Linked list各自的優缺點，以及各自適合的問題情境。

Array

優點：

- 較Linked list為節省記憶體空間：因為Linked list需要多一個pointer來記錄下一個node的記憶體位置。

缺點：

- 新增/刪除資料很麻煩：若要在第一個位置新增資料，就需要O(N)時間把矩陣中所有元素往後移動。同理，若要刪除第一個位置的資料，也需要O(N)時間把矩陣中剩餘的元素往前移動。
- 若資料數量時常在改變，要時常調整矩陣的大小，會花費O(N)的時間在搬動資料(把資料從舊的矩陣移動到新的矩陣)。

適用時機：

- 希望能夠快速存取資料。
- 已知欲處理的資料數量，便能確認矩陣的大小。
- 要求記憶體空間的使用越少越好。

Linked list

優點：

- 新增/刪除資料較Array簡單，只要對O(1)個node(所有與欲新增/刪除的node有pointer相連的node)調整pointer即可，不需要如同Array般搬動其餘元素。
    - 若是在Linked list的「開頭」新增node，只要O(1)。
    - 若要刪除特定node，或者在特定位置新增node，有可能需要先執行O(N)的「搜尋」。
- Linked list的資料數量可以是動態的，不像Array會有resize的問題。

缺點：

- 因為Linked list沒有index，若要找到特定node，需要從頭(ListNode *first)開始找起，搜尋的時間複雜度為O(N)。
- 需要額外的記憶體空間來儲存pointer。

適用時機：

- 無法預期資料數量時，使用Linked list就沒有resize的問題。
- 需要頻繁地新增/刪除資料時。
- 不需要快速查詢資料。