第十三週

王子銜, 陳毅軒, 吳尚龍

電機通訊程式設計

May 13, 2024

Outline

Linked list

Linked list

Linked list 是一種常見的資料結構，用於儲存一系列節點。每個節點包含一個資料元素以及指向下一個節點的指標。相較於陣列（Array），Linked list 可以更容易地插入或刪除節點，而不需要移動其他節點。

Linked list 常見的種類

- Singly Linked List: 每個節點只包含一個指向下一個節點的指標，最後一個節點指向 null 表示結尾。
- Doubly Linked List: 每個節點除了指向下一個節點的指標外，還包含指向上一個節點的指標，使得它能夠從兩個方向遍歷。
- Circular Linked List: 最後一個節點的指標會指向第一個節點，形成一個循環結構。

Linked list 優缺點

優點

可以動態調整 list 的大小
插入和刪除更有效率，特別是在頭部或尾部

缺點

隨機訪問的效率較低，因爲必須從頭開始遍歷
需要額外的記憶體記錄下一個節點的位置

常見應用

stack(堆疊)
queue
graph

Linked list 實作

Linked list 基本結構

Node 定義爲 Linked list 的每個節點。
LinkedList 結構是紀錄 Linked list 整個結構的開頭和數量。

```
1 struct Node {
2     value // 節點的值
3     next  // 指向下一個節點的指標
4 }
5 struct LinkedList {
6     head // 指向第一個節點的指標
7 }
```

Linked list 初始化

初始化空的 Linked list

```
1 procedure initializeList():
2     list = new LinkedList
3     list.head = null
4     return list
```

Linked list 插入

```
1  procedure insertNode(list, value, position):
2      newNode = new Node
3      newNode.value = value
4      newNode.next = null
5      if list.head is null:     // 空列表，直接插入
6          list.head = newNode
7          return
8      if position == 0:         // 插入到第一個位置
9          newNode.next = list.head
10         list.head = newNode
11         return
```

Linked list 插入

```
    // 插入到尾部或中間
    current = list.head
    prev = null
    index = 0
    // 找到所需要插入位置或末尾
    while current is not null and index < position:
        prev = current
        current = current.next
        index += 1
    if current is null:
        prev.next = newNode//插入到末端
    else:
        newNode.next = current//插入在中間
        prev.next = newNode
```

Linked list 刪除

刪除 Linked list 中第一個匹配特定值的節點

```
procedure deleteNode(list, value):
    if list.head is null:
        return
    if list.head.value == value:    // 如果第一個節點就是要刪除的
        list.head = list.head.next
        return
    current = list.head
    while current.next is not null and current.next.value != value://尋找目標
        current = current.next
    if current.next is not null:
        current.next = current.next.next
```

Linked list Search

因爲 Linked list 只能一個一個檢查，所以搜尋的速度比 array 慢很多。

```
procedure search(list, value):
    current = list.head
    while current is not null:
        if current.value == value:
            return true
        current = current.next
    return false
```

Linked list 輸出

```
procedure printList(list):
    if list.head is null:        // 檢查是否爲空
        print("List is empty.")
        return
    current = list.head
    while current is not null:
        print(current.value)
        current = current.next
```

說明

★ 作業 Domjudge 用其他寫法通過我們不會檢查，但如果考試出 linked list 一定會檢查通過的程式碼