* Prompt: *“So sánh Array và Linked List theo tốc độ truy cập, bộ nhớ, tình huống áp dụng. Sau đó đưa code minh họa thao tác chèn phần tử trong mỗi cấu trúc.”*
* Nếu AI chỉ trả lời lý thuyết mà chưa có code → chỉnh Prompt để bổ sung.

BÀI LÀm

**- So sánh Array và Linked List**

| **Tiêu chí** | **Array (Mảng)** | **Linked List (Danh sách liên kết)** |
| --- | --- | --- |
| **Cấu trúc lưu trữ** | Các phần tử được lưu **liên tiếp** trong bộ nhớ. | Mỗi phần tử (node) lưu dữ liệu và **con trỏ đến node tiếp theo**. |
| **Tốc độ truy cập (Access time)** | Truy cập **rất nhanh** (O(1)) vì có thể tính trực tiếp chỉ số. | Truy cập **chậm hơn** (O(n)) vì phải duyệt tuần tự từ đầu. |
| **Thao tác chèn/xóa** | **Chậm** (O(n)) — cần dịch chuyển các phần tử. | **Nhanh hơn** (O(1) – O(n)), chỉ cần thay đổi con trỏ. |
| **Bộ nhớ sử dụng** | Tiết kiệm hơn, chỉ lưu dữ liệu. | Tốn thêm bộ nhớ cho con trỏ next. |
| **Tình huống áp dụng** | Khi cần **truy cập ngẫu nhiên nhanh** và kích thước **cố định hoặc ít thay đổi**. | Khi cần **chèn/xóa thường xuyên** và **kích thước linh hoạt**. |

**Code minh họa thao tác chèn phần tử**

**1. Chèn phần tử vào mảng (Array)**

# Dùng list trong Python (tương đương array động)

arr = [10, 20, 30, 40]

print("Trước khi chèn:", arr)

# Chèn phần tử 25 vào vị trí thứ 2 (chỉ số 1)

arr.insert(1, 25)

print("Sau khi chèn:", arr)

**Giải thích:**

* insert(index, value) chèn phần tử mới vào vị trí chỉ định.
* Python list là mảng động, nhưng vẫn phải **dịch chuyển các phần tử phía sau**, nên độ phức tạp là **O(n)**.

**Kết quả:**

Trước khi chèn: [10, 20, 30, 40]

Sau khi chèn: [10, 25, 20, 30, 40]

**2. Chèn phần tử vào danh sách liên kết (Linked List)**

# Định nghĩa Node

class Node:

def \_\_init\_\_(self, data):

self.data = data

self.next = None

# Định nghĩa LinkedList

class LinkedList:

def \_\_init\_\_(self):

self.head = None

# Hàm thêm phần tử vào đầu danh sách

def insert\_at\_beginning(self, data):

new\_node = Node(data)

new\_node.next = self.head # Con trỏ next trỏ tới node hiện tại

self.head = new\_node # Cập nhật node đầu là node mới

# Hàm in danh sách

def display(self):

current = self.head

while current:

print(current.data, end=" -> ")

current = current.next

print("None")

# --- Thử nghiệm ---

ll = LinkedList()

ll.insert\_at\_beginning(30)

ll.insert\_at\_beginning(20)

ll.insert\_at\_beginning(10)

print("Danh sách sau khi chèn:")

ll.display()

**Giải thích:**

* Mỗi node gồm data và next.
* Khi chèn, chỉ cần cập nhật con trỏ next, **không cần dịch chuyển phần tử** → O(1).

**Kết quả:**

Danh sách sau khi chèn:

10 -> 20 -> 30 -> None

**Tóm tắt**

| **Thao tác** | **Array** | **Linked List** |
| --- | --- | --- |
| Truy cập phần tử | O(1) | O(n) |
| Chèn đầu danh sách | O(n) | O(1) |
| Bộ nhớ phụ | Không | Có (con trỏ) |
| Khi nào dùng | Dữ liệu tĩnh, truy cập nhanh | Dữ liệu động, chèn/xóa thường xuyên |

**Nếu AI chỉ trả lời lý thuyết mà chưa có code**, bạn có thể **chỉnh prompt** thành:

“So sánh Array và Linked List (theo tốc độ truy cập, bộ nhớ, tình huống áp dụng) **và kèm code minh họa thao tác chèn phần tử** trong mỗi cấu trúc.”