Trong trường hợp của hai thuật toán sắp xếp **Chèn (Insertion Sort)** và **Chọn (Selection Sort)**, việc lưu trữ dữ liệu đầu vào trong **LinkedList (danh sách liên kết)** không phải là lựa chọn tốt, và có một số lý do chính sau đây:

### 1. ****Insertion Sort (Sắp xếp Chèn)****

**Cách hoạt động**: Insertion Sort hoạt động bằng cách duyệt qua từng phần tử trong mảng, sau đó chèn phần tử đó vào đúng vị trí của nó trong phần trước đã được sắp xếp. Với mỗi phần tử, thuật toán cần truy cập và chèn nó vào một vị trí cụ thể trong danh sách.

**Truy cập trong LinkedList**: Danh sách liên kết (LinkedList) không hỗ trợ truy cập ngẫu nhiên, có nghĩa là để truy cập vào phần tử thứ i, bạn phải duyệt qua từ đầu danh sách tới vị trí đó, dẫn đến độ phức tạp thời gian là **O(i)**. Trong khi đó, nếu sử dụng mảng, truy cập ngẫu nhiên là **O(1)**.

**Hiệu quả trên LinkedList**: Do việc truy cập từng phần tử trong LinkedList tốn thời gian **O(n)** với mỗi phần tử, điều này làm cho thời gian thực hiện của Insertion Sort trên LinkedList trở nên chậm, với độ phức tạp có thể lên tới **O(n²)** khi cần phải duyệt qua nhiều phần tử để chèn một giá trị vào đúng vị trí. Điều này không hiệu quả so với khi sử dụng mảng.

### 2. ****Selection Sort (Sắp xếp Chọn)****

**Cách hoạt động**: Selection Sort tìm phần tử nhỏ nhất trong phần chưa được sắp xếp của danh sách và hoán đổi nó với phần tử đầu tiên của phần chưa sắp xếp, lặp đi lặp lại quá trình này cho tới khi danh sách được sắp xếp.

**Hoán đổi trong LinkedList**: Selection Sort thường yêu cầu nhiều lần hoán đổi các phần tử. Trong LinkedList, việc hoán đổi các phần tử giữa hai vị trí khác nhau là tốn kém, vì không thể truy cập ngẫu nhiên, mà phải duyệt qua từ đầu danh sách để tìm phần tử cần hoán đổi. Điều này làm cho thời gian thực hiện của Selection Sort trên LinkedList rất chậm, với độ phức tạp thời gian là **O(n²)**.

**Hiệu quả trên LinkedList**: Giống như Insertion Sort, Selection Sort yêu cầu nhiều lần duyệt danh sách để tìm phần tử nhỏ nhất và hoán đổi, và điều này làm cho nó không hiệu quả khi dữ liệu được lưu trữ trong LinkedList.

### 3. ****Kết luận****

* Cả hai thuật toán **Insertion Sort** và **Selection Sort** đều hoạt động hiệu quả hơn khi sử dụng **mảng (array)** vì chúng dựa nhiều vào việc truy cập ngẫu nhiên và hoán đổi phần tử. Trong **LinkedList**, truy cập ngẫu nhiên tốn nhiều thời gian và làm cho độ phức tạp của thuật toán tăng lên, đặc biệt là khi danh sách có nhiều phần tử.
* **LinkedList** không phù hợp cho các thuật toán sắp xếp yêu cầu nhiều lần truy cập và hoán đổi như Insertion Sort và Selection Sort.