

TRƯỜNG ĐẠI HỌC TRÀ VINH  
TRƯỜNG KỸ THUẬT VÀ CÔNG NGHỆ  
KHOA CÔNG NGHỆ THÔNG TIN



**THỰC TẬP ĐỒ ÁN CƠ SỞ NGÀNH**  
**HỌC KỲ I, NĂM HỌC 2025-2026**  
**ĐỀ TÀI : ỨNG DỤNG MỘT SỐ PHƯƠNG PHÁP**  
**SẮP XẾP TRÊN DANH SÁCH LIÊN KẾT ĐƠN**

*Giảng viên hướng dẫn:*

**ThS : Lê Minh Tự**

*Sinh viên thực hiện:*

**Họ tên : Phạm Thảo Nguyên**

**MSSV : 110123143**

**Lớp : DA23TTC**

*Vĩnh Long, tháng 1 năm 2026*

TRƯỜNG ĐẠI HỌC TRÀ VINH  
TRƯỜNG KỸ THUẬT VÀ CÔNG NGHỆ  
KHOA CÔNG NGHỆ THÔNG TIN



**THỰC TẬP ĐỒ ÁN CƠ SỞ NGÀNH**  
**HỌC KỲ I, NĂM HỌC 2025-2026**  
**ĐỀ TÀI : ỨNG DỤNG MỘT SỐ PHƯƠNG PHÁP**  
**SẮP XẾP TRÊN DANH SÁCH LIÊN KẾT ĐƠN**

*Giảng viên hướng dẫn:*

**ThS : Lê Minh Tự**

*Sinh viên thực hiện:*

**Họ tên : Phạm Thảo Nguyên**

**MSSV : 110123143**

**Lớp : DA23TTC**

*Vĩnh Long, tháng 1 năm 2026*

## NHẬN XÉT CỦA GIẢNG VIÊN HƯỚNG DẪN

[illegible]

Vĩnh Long, ngày ..... tháng ..... năm 2026

## Giảng viên hướng dẫn

(Ký tên và ghi rõ họ tên)



---

## LỜI CẢM ƠN

Trong quá trình thực hiện đề án “Ứng dụng một số phương pháp sắp xếp trên danh sách liên kết đơn”, em đã nhận được rất nhiều sự quan tâm, hướng dẫn và hỗ trợ quý báu từ thầy, bạn bè và những người xung quanh. Em xin được bày tỏ lòng biết ơn chân thành và sâu sắc nhất.

Trước hết, em xin gửi lời cảm ơn chân thành đến giảng viên đã trực tiếp giảng dạy và hướng dẫn em trong suốt quá trình học tập học phần. Những kiến thức nền tảng về cấu trúc dữ liệu, giải thuật, cùng với sự chỉ bảo tận tình của thầy đã giúp em hiểu rõ hơn về danh sách liên kết đơn cũng như cách áp dụng các thuật toán sắp xếp vào thực tế lập trình. Sự hướng dẫn tận tâm và những góp ý quý báu của thầy là nền tảng quan trọng để em hoàn thành đồ án này một cách đúng hướng và hiệu quả.

Em xin cảm ơn Khoa/Trường đã tạo điều kiện thuận lợi về môi trường học tập, tài liệu tham khảo và cơ sở vật chất, giúp em có cơ hội tiếp cận và rèn luyện kỹ năng lập trình một cách bài bản. Những kiến thức được học không chỉ phục vụ cho bài báo cáo này mà còn là hành trang quý giá cho quá trình học tập và làm việc sau này.

Bên cạnh đó, em cũng xin gửi lời cảm ơn đến bạn bè và các bạn trong lớp đã luôn sẵn sàng trao đổi, thảo luận, chia sẻ kinh nghiệm và hỗ trợ em trong quá trình học tập và thực hiện đề án. Những ý kiến đóng góp, sự động viên tinh thần từ các bạn đã giúp em vượt qua những khó khăn khi gặp lỗi chương trình hay chưa hiểu rõ thuật toán.

Cuối cùng, em xin bày tỏ lòng biết ơn sâu sắc đến gia đình – những người luôn bên cạnh, động viên và tạo điều kiện tốt nhất để em yên tâm học tập và hoàn thành bài báo cáo này.

Mặc dù đã rất cố gắng, nhưng do thời gian và kiến thức còn hạn chế, bài báo cáo không tránh khỏi những thiếu sót. Em rất mong nhận được sự góp ý và chỉ dẫn thêm từ thầy để bài báo cáo được hoàn thiện hơn.

Em xin chân thành cảm ơn!

## MỤC LỤC

### Contents

<b>NHẬN XÉT CỦA GIẢNG VIÊN HƯỚNG DẪN .....</b>	<b>i</b>
<b>NHẬN XÉT CỦA THÀNH VIÊN HỘI ĐỒNG .....</b>	<b>ii</b>
<b>LỜI CẢM ƠN.....</b>	<b>iii</b>
<b>MỤC LỤC .....</b>	<b>iv</b>
<b>DANH MỤC HÌNH ẢNH.....</b>	<b>vi</b>
<b>DANH MỤC BẢNG BIỂU.....</b>	<b>vii</b>
<b>TÓM TẮT ĐỒ ÁN CƠ SỞ NGÀNH.....</b>	<b>viii</b>
<b>MỞ ĐẦU .....</b>	<b>ix</b>
<b>CHƯƠNG 1: TỔNG QUAN NGHIÊN CỨU .....</b>	<b>1</b>
1.1. Lý do chọn đề tài.....	1
1.2. Mục tiêu nghiên cứu .....	1
1.3. Đối tượng và phạm vi nghiên cứu .....	2
1.4. Phương pháp nghiên cứu .....	2
<b>CHƯƠNG 2: NGHIÊN CỨU LÝ THUYẾT .....</b>	<b>4</b>
<b>NỘI DUNG 1: CƠ SỞ LÝ THUYẾT VỀ CẤU TRÚC DỮ LIỆU.....</b>	<b>4</b>
1.1. Khái niệm cấu trúc dữ liệu.....	4
1.2. Phân loại cấu trúc dữ liệu .....	4
1.3. Danh sách liên kết và đặc điểm.....	5
1.4. Danh sách liên kết đơn .....	5
1.5. Ưu và nhược điểm của danh sách liên kết đơn .....	5
<b>NỘI DUNG 2: CƠ SỞ LÝ THUYẾT VỀ THUẬT TOÁN.....</b>	<b>6</b>
<b>SẮP XẾP .....</b>	<b>6</b>
2.1. Khái niệm thuật toán sắp xếp .....	6
2.2. Vai trò của thuật toán sắp xếp.....	6
2.3. Phân loại thuật toán sắp xếp.....	7
2.4. Tiêu chí đánh giá thuật toán sắp xếp .....	7
2.5. Ảnh hưởng của cấu trúc dữ liệu đến thuật toán sắp xếp .....	8
2.6. Mức độ phù hợp của thuật toán với danh sách liên kết .....	8
<b>NỘI DUNG 3: CÁC THUẬT TOÁN SẮP XẾP TRÊN.....</b>	<b>9</b>
<b>DANH SÁCH LIÊN KẾT ĐƠN .....</b>	<b>9</b>
3.1. Thuật toán Bubble Sort.....	9
3.2. Thuật toán Insertion Sort.....	10
3.3. Thuật toán Selection Sort.....	11
3.4. Thuật toán Merge Sort.....	12
3.5. Thuật toán Quick Sort.....	13
3.6. So sánh và nhận xét chung.....	13

<b>CHƯƠNG 3: HIỆN THỰC HÓA NGHIÊN CỨU</b>	<b>15</b>
3.1. Phân tích và đặc tả yêu cầu hệ thống	15
3.1.1. Bối cảnh và mục tiêu nghiên cứu	15
3.1.2. Đặc tả bài toán	15
3.1.3. Phân tích yêu cầu chức năng	15
3.2. Thiết kế tổng thể hệ thống	16
3.2.1. Mô hình thiết kế chương trình	16
3.2.2. Sơ đồ luồng xử lý	17
3.2.3. Thiết kế cấu trúc dữ liệu	18
3.3. Thiết kế chi tiết các module chức năng	18
3.3.1. Module quản lý danh sách	18
3.3.2. Module sắp xếp	19
3.4. Hiện thực chi tiết các thuật toán sắp xếp	20
3.4.1. Hiện thực thuật toán Bubble Sort	20
3.4.2. Hiện thực thuật toán Insertion Sort	21
3.4.3. Hiện thực thuật toán Selection Sort	22
3.4.4. Hiện thực thuật toán Merge Sort	23
3.4.5. Hiện thực thuật toán Quick Sort	23
3.5. Kiểm thử và đánh giá chương trình	24
3.5.1. Phương pháp kiểm thử	24
3.5.2. Kết quả kiểm thử	24
3.5.3. Đánh giá hiệu quả	24
3.6. Hạn chế và hướng mở rộng	24
<b>CHƯƠNG 4: KẾT QUẢ NGHIÊN CỨU</b>	<b>26</b>
4.1. Kết quả đạt được	26
4.2. Đánh giá tính đúng đắn của chương trình	26
4.3. Đánh giá hiệu năng của các thuật toán	27
4.3.1. So sánh thời gian thực thi	27
4.3.2. Đánh giá bộ nhớ sử dụng	27
4.4. Trải nghiệm người dùng	28
4.5. Minh họa kết quả thực thi	28
4.6. Nhận xét chung	28
<b>CHƯƠNG 5: KẾT LUẬN HƯỚNG PHÁT TRIỂN</b>	<b>29</b>
5.1. Kết luận	29
5.2. Hướng phát triển	29
<b>DANH MỤC TÀI LIỆU THAM KHẢO</b>	<b>31</b>
<b>PHỤ LỤC</b>	<b>32</b>

---

## DANH MỤC HÌNH ẢNH

<b>Hình 1 : Minh Họa Thuật toán Bubble Sort .....</b>	<b>9</b>
<b>Hình 2 : Minh Họa Thuật toán Insertion Sort .....</b>	<b>10</b>
<b>Hình 3: Minh Họa Thuật toán Selection Sort .....</b>	<b>11</b>
<b>Hình 4: Minh Họa Thuật toán Merge Sort.....</b>	<b>12</b>
<b>Hình 5 : Minh Họa Thuật toán Quick Sort .....</b>	<b>13</b>
<b>Hình 6 : Minh Họa cấu trúc dữ liệu danh sách liên kết đơn.....</b>	<b>18</b>

---

## DANH MỤC BẢNG BIỂU

<b>Bảng 1</b> Nội dung các chương .....	3
<b>Bảng 2</b> Phân loại cấu trúc dữ liệu .....	4
<b>Bảng 3</b> Minh họa nguyên lý hoạt động của thuật toán sắp xếp .....	7
<b>Bảng 4</b> Minh họa tiêu chí đánh giá của thuật toán sắp xếp .....	7
<b>Bảng 5</b> Minh họa mức độ phù hợp của thuật toán .....	8
<b>Bảng 6</b> Sơ đồ luồng xử lý .....	17
<b>Bảng 7</b> Module quản lý danh sách .....	19
<b>Bảng 8</b> Module sắp xếp .....	20
<b>Bảng 9</b> So sánh hiệu năng các thuật toán sắp xếp .....	27

---

## **TÓM TẮT ĐỒ ÁN CƠ SỞ NGÀNH**

Đồ án cơ sở ngành với đề tài “Ứng dụng một số phương pháp sắp xếp trên danh sách liên kết đơn” được thực hiện nhằm nghiên cứu và cài đặt các thuật toán sắp xếp cơ bản trên cấu trúc dữ liệu danh sách liên kết đơn bằng ngôn ngữ lập trình C. Nội dung đồ án tập trung vào việc tìm hiểu đặc điểm của danh sách liên kết đơn, phân tích nguyên lý hoạt động của các thuật toán sắp xếp và đánh giá hiệu quả của từng phương pháp khi áp dụng trên cấu trúc dữ liệu này.

Trong đồ án, năm thuật toán sắp xếp phổ biến gồm Bubble Sort, Insertion Sort, Selection Sort, Merge Sort và Quick Sort đã được lựa chọn để cài đặt và so sánh. Chương trình cho phép người dùng nhập dữ liệu, lựa chọn thuật toán sắp xếp thông qua menu và quan sát kết quả sắp xếp của danh sách liên kết. Các thuật toán được xây dựng dựa trên nguyên tắc thao tác con trỏ và cấp phát động bộ nhớ, phù hợp với đặc trưng của danh sách liên kết đơn. Riêng Selection Sort được cài đặt theo hướng giữ nguyên cấu trúc liên kết của các node và chỉ hoán đổi dữ liệu, giúp thuật toán đơn giản và an toàn khi thực hiện.

Kết quả thực nghiệm cho thấy mỗi thuật toán có những ưu điểm và hạn chế riêng. Bubble Sort, Insertion Sort và Selection Sort có cách cài đặt đơn giản nhưng hiệu suất thấp khi số lượng phần tử lớn. Ngược lại, Merge Sort và Quick Sort cho hiệu quả cao hơn với độ phức tạp thời gian trung bình là  $O(n \log n)$ , trong đó Merge Sort thể hiện tính ổn định và đặc biệt phù hợp với danh sách liên kết đơn. Thông qua quá trình thực hiện đồ án, sinh viên đã củng cố kiến thức về cấu trúc dữ liệu, giải thuật, con trỏ và lập trình C, đồng thời rèn luyện tư duy phân tích và kỹ năng xây dựng chương trình hoàn chỉnh.

Đồ án không chỉ mang ý nghĩa học thuật mà còn có giá trị thực tiễn, làm cơ sở cho việc nghiên cứu và phát triển các ứng dụng quản lý dữ liệu trong những học phần và đồ án tiếp theo.

---

## MỞ ĐẦU

Trong lĩnh vực công nghệ thông tin, cấu trúc dữ liệu và giải thuật đóng vai trò vô cùng quan trọng trong việc xây dựng các chương trình và hệ thống phần mềm. Việc lựa chọn cấu trúc dữ liệu phù hợp cùng với thuật toán xử lý hiệu quả không chỉ giúp chương trình hoạt động chính xác mà còn nâng cao hiệu suất và khả năng mở rộng của hệ thống. Trong số các bài toán cơ bản của cấu trúc dữ liệu, bài toán sắp xếp là một trong những vấn đề nền tảng, được ứng dụng rộng rãi trong nhiều lĩnh vực như quản lý dữ liệu, tìm kiếm thông tin và xử lý thống kê.

Danh sách liên kết đơn là một cấu trúc dữ liệu động, cho phép quản lý tập hợp dữ liệu có kích thước thay đổi linh hoạt trong quá trình thực thi chương trình. So với mảng, danh sách liên kết đơn có ưu điểm trong việc chèn và xóa phần tử mà không cần dịch chuyển dữ liệu. Tuy nhiên, do đặc điểm không cho phép truy cập ngẫu nhiên, việc áp dụng các thuật toán sắp xếp trên danh sách liên kết đơn đòi hỏi cách tiếp cận và cài đặt phù hợp, khác với các cấu trúc dữ liệu tuyến tính thông thường.

Xuất phát từ yêu cầu thực tiễn trong học tập và nhằm củng cố kiến thức đã được học trong học phần Cấu trúc dữ liệu và giải thuật, đồ án cơ sở ngành với đề tài “Ứng dụng một số phương pháp sắp xếp trên danh sách liên kết đơn” được thực hiện. Đề tài tập trung nghiên cứu, phân tích và cài đặt một số thuật toán sắp xếp tiêu biểu gồm Bubble Sort, Insertion Sort, Selection Sort, Merge Sort và Quick Sort trên danh sách liên kết đơn bằng ngôn ngữ lập trình C.

Thông qua việc xây dựng chương trình minh họa và so sánh hiệu quả của các thuật toán, đồ án giúp làm rõ ưu điểm, hạn chế cũng như phạm vi ứng dụng của từng phương pháp sắp xếp. Qua đó, sinh viên không chỉ nâng cao kỹ năng lập trình, tư duy thuật toán mà còn có cái nhìn tổng quan hơn về việc lựa chọn giải pháp phù hợp trong xử lý dữ liệu, làm nền tảng cho các học phần chuyên ngành và các đồ án tiếp theo.

# CHƯƠNG 1: TỔNG QUAN NGHIÊN CỨU

## 1.1. Lý do chọn đề tài

Trong lĩnh vực công nghệ thông tin, việc xử lý và quản lý dữ liệu một cách hiệu quả đóng vai trò vô cùng quan trọng. Trong đó, sắp xếp dữ liệu là một trong những bài toán cơ bản nhưng mang ý nghĩa nền tảng, được ứng dụng rộng rãi trong nhiều hệ thống phần mềm và chương trình máy tính như quản lý cơ sở dữ liệu, tìm kiếm thông tin và xử lý thống kê.

Danh sách liên kết đơn là một cấu trúc dữ liệu động, cho phép quản lý bộ nhớ linh hoạt và hiệu quả hơn so với mảng trong nhiều trường hợp, đặc biệt là khi thực hiện các thao tác chèn và xóa phần tử. Tuy nhiên, do đặc điểm không cho phép truy cập ngẫu nhiên, việc áp dụng các thuật toán sắp xếp trên danh sách liên kết đơn có những đặc thù riêng, đòi hỏi cách tiếp cận và cài đặt phù hợp so với sắp xếp trên mảng.

Xuất phát từ nhu cầu tìm hiểu sâu hơn về cấu trúc dữ liệu và giải thuật, cũng như rèn luyện kỹ năng lập trình, nhóm thực hiện lựa chọn đề tài “Ứng dụng một số phương pháp sắp xếp trên danh sách liên kết đơn” nhằm nghiên cứu, cài đặt và đánh giá hiệu quả của các thuật toán sắp xếp phổ biến như Bubble Sort, Insertion Sort, Selection Sort, Merge Sort và Quick Sort trên cấu trúc dữ liệu danh sách liên kết đơn.

## 1.2. Mục tiêu nghiên cứu

Mục tiêu chính của đề tài là tìm hiểu và áp dụng một số thuật toán sắp xếp cơ bản trên danh sách liên kết đơn, từ đó đánh giá ưu điểm và hạn chế của từng thuật toán khi áp dụng trên cấu trúc dữ liệu này. Cụ thể, đề tài hướng đến các mục tiêu sau:

- Nghiên cứu cơ sở lý thuyết về danh sách liên kết đơn và các thuật toán sắp xếp.
- Cài đặt các thuật toán sắp xếp gồm Bubble Sort, Insertion Sort, Selection Sort, Merge Sort và Quick Sort trên danh sách liên kết đơn.
- So sánh hiệu quả của các thuật toán thông qua kết quả thực nghiệm.
- Nâng cao kỹ năng lập trình bằng ngôn ngữ C và khả năng tư duy thuật toán.

---

### 1.3. Đối tượng và phạm vi nghiên cứu

#### \*\*\* Đối tượng nghiên cứu

Đối tượng nghiên cứu của đề tài là các thuật toán sắp xếp được áp dụng trên cấu trúc dữ liệu danh sách liên kết đơn.

#### \*\*\* Phạm vi nghiên cứu

Phạm vi nghiên cứu của đề tài bao gồm:

- Sử dụng danh sách liên kết đơn với dữ liệu kiểu số nguyên.
- Nghiên cứu và cài đặt năm thuật toán sắp xếp cơ bản: Bubble Sort, Insertion Sort, Selection Sort, Merge Sort và Quick Sort.
- Thực hiện chương trình trong môi trường lập trình ngôn ngữ C.
- Không đi sâu vào các kỹ thuật tối ưu hóa nâng cao hoặc các biến thể phức tạp của thuật toán..

### 1.4. Phương pháp nghiên cứu

Để thực hiện đề tài, nhóm sử dụng các phương pháp nghiên cứu sau:

- **Phương pháp nghiên cứu lý thuyết:** Tìm hiểu các tài liệu liên quan đến cấu trúc dữ liệu danh sách liên kết đơn và các thuật toán sắp xếp.
- **Phương pháp phân tích và thiết kế thuật toán:** Xây dựng giải thuật phù hợp cho từng phương pháp sắp xếp (Bubble Sort, Insertion Sort, Selection Sort, Merge Sort và Quick Sort) trên danh sách liên kết đơn.
- **Phương pháp lập trình thực nghiệm:** Cài đặt chương trình bằng ngôn ngữ C và kiểm tra kết quả với nhiều bộ dữ liệu khác nhau.
- **Phương pháp so sánh và đánh giá:** So sánh kết quả sắp xếp, đánh giá ưu điểm và hạn chế của từng thuật toán.

---

**1.5. Bố cục đề tài**

Nội dung của đề tài được trình bày gồm 5 chương chính:

Chương	Nội Dung Chính
Chương 1	Tổng quan đề tài
Chương 2	Cơ sở lý thuyết
Chương 3	Cài đặt và đánh giá
Chương 4	Kết quả nghiên cứu
Chương 5	Kết luận và hướng dẫn phát triển

**Bảng 1 Nội dung các chương**

## CHƯƠNG 2: NGHIÊN CỨU LÝ THUYẾT

### NỘI DUNG 1: CƠ SỞ LÝ THUYẾT VỀ CẤU TRÚC DỮ LIỆU

#### 1.1. Khái niệm cấu trúc dữ liệu

Cấu trúc dữ liệu là cách tổ chức, lưu trữ và quản lý dữ liệu trong bộ nhớ máy tính sao cho việc truy xuất, xử lý và cập nhật dữ liệu được thực hiện một cách hiệu quả nhất. Trong lập trình, cấu trúc dữ liệu đóng vai trò nền tảng, quyết định đến hiệu năng, độ phức tạp và khả năng mở rộng của chương trình.

Một chương trình có thuật toán tốt nhưng lựa chọn cấu trúc dữ liệu không phù hợp vẫn có thể hoạt động kém hiệu quả. Do đó, việc nghiên cứu cấu trúc dữ liệu là yêu cầu bắt buộc đối với sinh viên công nghệ thông tin.

#### 1.2. Phân loại cấu trúc dữ liệu

Loại cấu trúc dữ liệu	Ví dụ	Đặc điểm chính
Tuyến tính	Mảng, danh sách liên kết, ngăn xếp	Dữ liệu được tổ chức theo thứ tự
Phi tuyến tính	Cây, đồ thị	Biểu diễn mối quan hệ phức tạp
Tĩnh	Mảng	Kích thước cố định
Động	Danh sách liên kết	Kích thước linh hoạt

**Bảng 2 Phân loại cấu trúc dữ liệu**

#### Nhận xét:

Danh sách liên kết thuộc nhóm cấu trúc dữ liệu tuyến tính, động, rất phù hợp với các bài toán cần thêm, xóa phần tử thường xuyên.

### **1.3. Danh sách liên kết và đặc điểm**

Danh sách liên kết là cấu trúc dữ liệu gồm các node liên kết với nhau thông qua con trỏ. Mỗi node chứa dữ liệu và địa chỉ của node kế tiếp. Nhờ đặc điểm này, danh sách liên kết không bị giới hạn kích thước cố định như mảng.

Tuy nhiên, danh sách liên kết không hỗ trợ truy cập ngẫu nhiên, khiến một số thao tác trở nên chậm hơn so với mảng.

### **1.4. Danh sách liên kết đơn**

Danh sách liên kết đơn là dạng đơn giản nhất của danh sách liên kết. Mỗi node chỉ chứa một con trỏ trỏ đến node tiếp theo. Node đầu tiên gọi là head, node cuối cùng trỏ đến NULL.

Danh sách liên kết đơn thường được sử dụng trong giảng dạy và nghiên cứu cơ bản do cấu trúc đơn giản và dễ cài đặt.

### **1.5. Ưu và nhược điểm của danh sách liên kết đơn**

#### **\*\*\*Ưu điểm:**

- Linh hoạt trong việc thêm và xóa phần tử.
- Không cần vùng nhớ liên tiếp.
- Phù hợp với dữ liệu thay đổi thường xuyên.

#### **\*\*\*Nhược điểm:**

- Không truy cập trực tiếp phần tử bất kỳ.
- Tốn thêm bộ nhớ cho con trỏ.
- Khó tối ưu một số thuật toán.

---

## **NỘI DUNG 2: CƠ SỞ LÝ THUYẾT VỀ THUẬT TOÁN**

### **SẮP XẾP**

#### **2.1. Khái niệm thuật toán sắp xếp**

Thuật toán sắp xếp là tập hợp hữu hạn các bước nhằm sắp xếp dữ liệu theo một thứ tự xác định. Đây là một trong những bài toán cơ bản nhất trong khoa học máy tính và xuất hiện trong hầu hết các hệ thống phần mềm.

Sắp xếp giúp dữ liệu trở nên có tổ chức, từ đó hỗ trợ hiệu quả cho các thao tác tìm kiếm và xử lý.

#### **2.2. Vai trò của thuật toán sắp xếp**

Thuật toán sắp xếp đóng vai trò quan trọng trong:

- Tăng tốc độ tìm kiếm.
- Hỗ trợ thống kê và phân tích dữ liệu.
- Tối ưu hóa hiệu năng hệ thống.

Trong nhiều ứng dụng thực tế, sắp xếp là bước tiền xử lý bắt buộc.

**2.3. Phân loại thuật toán sắp xếp**

Thuật toán	Phương pháp	Nguyên lý hoạt động
Bubble Sort	So sánh	Đổi chỗ các phần tử kề nhau nếu sai thứ tự
Insertion Sort	Chèn	Chèn phần tử vào vị trí thích hợp trong dãy đã sắp xếp
Selection Sort	Chọn	Chọn phần tử nhỏ nhất và đưa về đúng vị trí
Merge Sort	Chia để trị	Chia danh sách thành các phần nhỏ, sắp xếp và trộn lại
Quick Sort	Phân hoạch	Chọn phần tử chốt (pivot) và phân chia danh sách

**Bảng 3 Minh họa nguyên lý hoạt động của thuật toán sắp xếp****2.4. Tiêu chí đánh giá thuật toán sắp xếp**

Tiêu chí	Ý nghĩa
Độ phức tạp thời gian	Tốc độ thực thi
Độ phức tạp bộ nhớ	Mức sử dụng RAM
Tính ổn định	Giữ nguyên thứ tự phần tử bằng nhau
Khả năng áp dụng	Phù hợp với cấu trúc dữ liệu

**Bảng 4 Minh họa tiêu chí đánh giá của thuật toán sắp xếp**

## 2.5. Ảnh hưởng của cấu trúc dữ liệu đến thuật toán sắp xếp

Cấu trúc dữ liệu ảnh hưởng trực tiếp đến cách cài đặt và hiệu quả của thuật toán sắp xếp. Các thuật toán hoạt động tốt trên mảng chưa chắc phù hợp khi áp dụng trên danh sách liên kết do hạn chế về truy cập ngẫu nhiên.

## 2.6. Mức độ phù hợp của thuật toán với danh sách liên kết

Thuật toán	Mức độ phù hợp	Nhận xét
Bubble Sort	Thấp	Chỉ phù hợp minh họa
Insertion Sort	Trung bình	Dễ cài đặt
Selection Sort	Thấp	Đơn giản nhưng hiệu suất không cao
Merge Sort	Cao	Hiệu quả nhất
Quick Sort	Trung bình	Cài đặt phức tạp

**Bảng 5 Minh họa mức độ phù hợp của thuật toán**

## NỘI DUNG 3: CÁC THUẬT TOÁN SẮP XẾP TRÊN

### DANH SÁCH LIÊN KẾT ĐƠN

#### 3.1. Thuật toán Bubble Sort

Bubble Sort hoạt động dựa trên nguyên tắc so sánh các phần tử kế nhau và hoán đổi nếu thứ tự chưa đúng. Thuật toán lặp lại quá trình này cho đến khi danh sách được sắp xếp hoàn toàn.

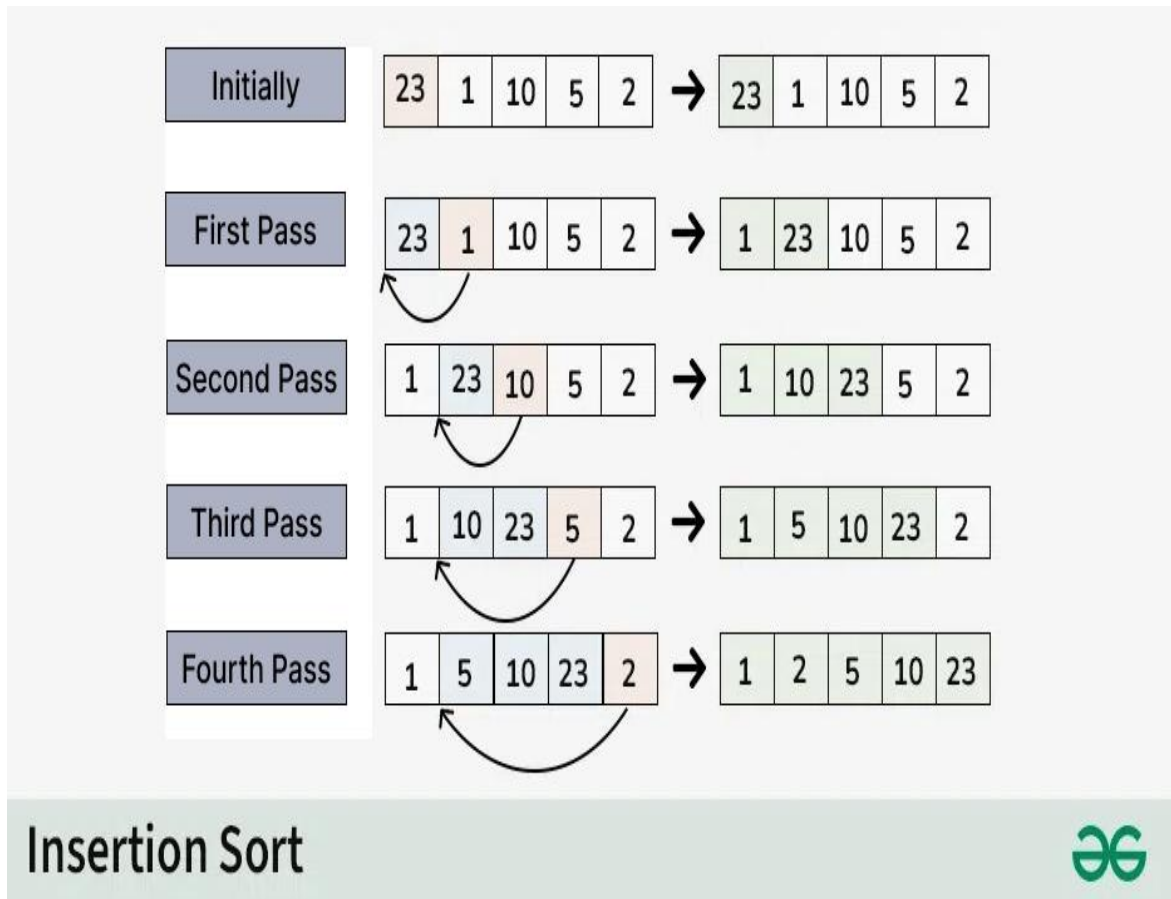


**Hình 1 : Minh Họa Thuật toán Bubble Sort**

**Nhận xét:** Bubble Sort có cấu trúc đơn giản nhưng độ phức tạp cao  $O(n^2)$ , chủ yếu phù hợp cho mục đích minh họa.

### 3.2. Thuật toán Insertion Sort

Insertion Sort xây dựng dần một danh sách đã được sắp xếp bằng cách chèn từng phần tử vào vị trí thích hợp. Thuật toán này tận dụng tốt đặc điểm của danh sách liên kết vì thao tác chèn node đơn giản.

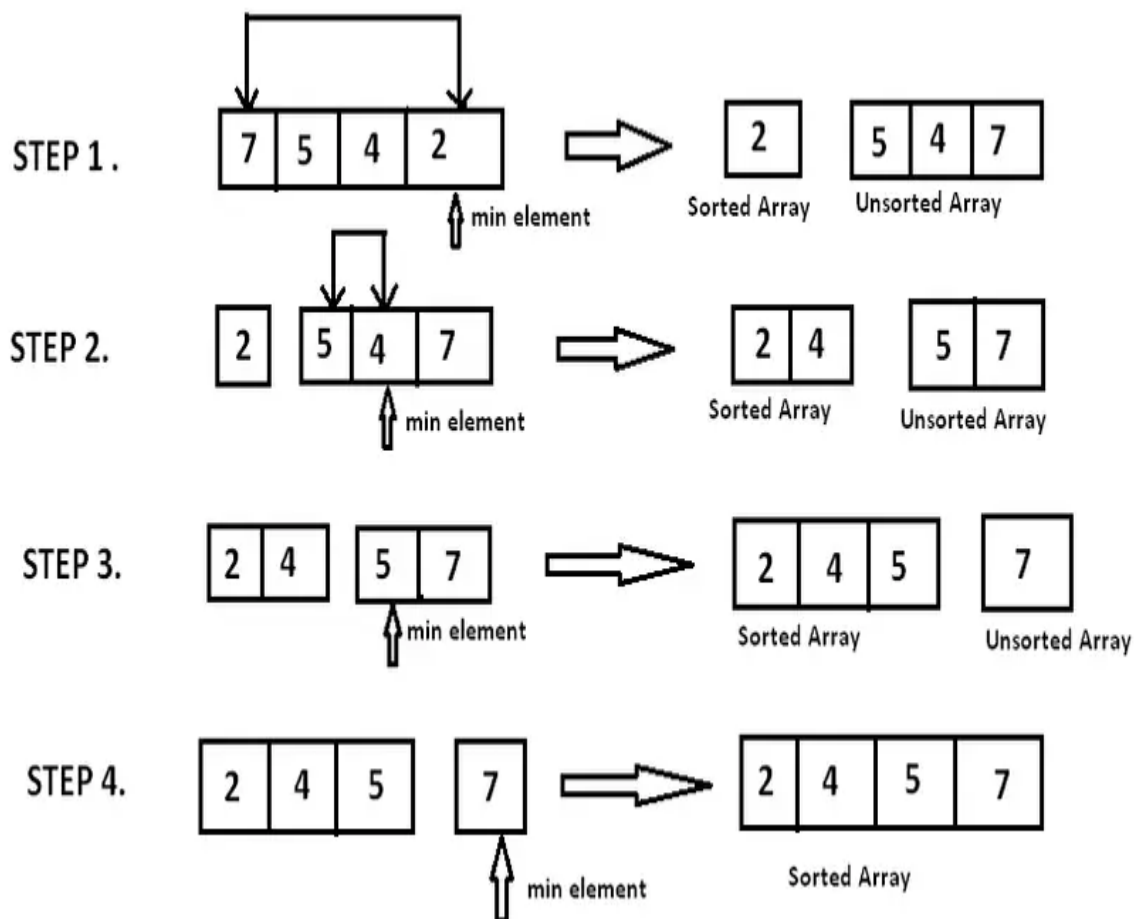


**Hình 2 : Minh Họa Thuật toán Insertion Sort**

**Nhận xét:** Insertion Sort phù hợp với dữ liệu nhỏ hoặc gần như đã được sắp xếp.

### 3.3. Thuật toán Selection Sort

Selection Sort hoạt động bằng cách lặp qua danh sách, mỗi lần chọn phần tử nhỏ nhất trong phần chưa sắp xếp và đưa về đúng vị trí. Trên danh sách liên kết đơn, thuật toán thường được cài đặt bằng cách giữ nguyên liên kết node và chỉ hoán đổi dữ liệu.

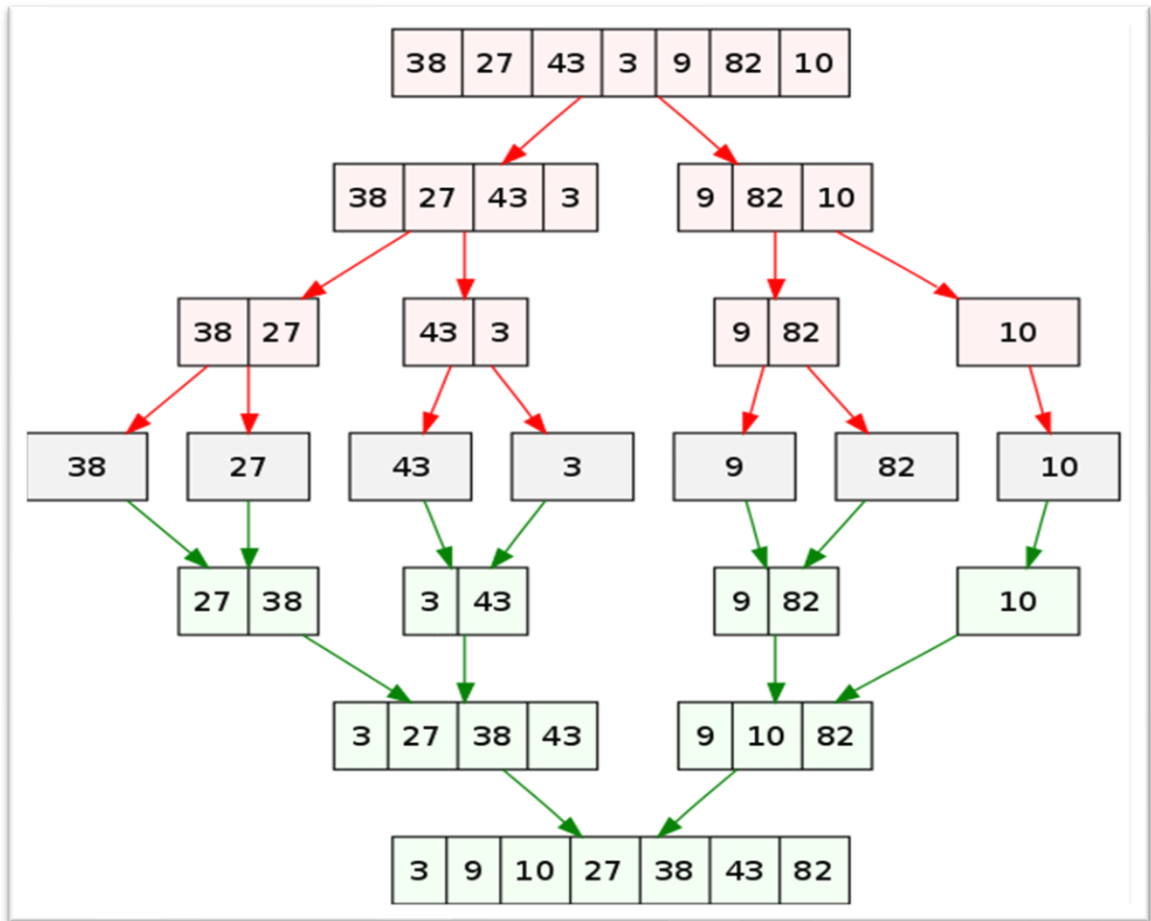


**Hình 3: Minh Họa Thuật toán Selection Sort**

**Nhận xét:** Selection Sort dễ cài đặt, phù hợp cho mục đích học tập nhưng có độ phức tạp  $O(n^2)$ , không hiệu quả với danh sách lớn.

### 3.4. Thuật toán Merge Sort

Merge Sort là thuật toán hiệu quả dựa trên phương pháp chia để trị. Thuật toán chia danh sách thành các phần nhỏ hơn, sắp xếp từng phần rồi trộn lại.

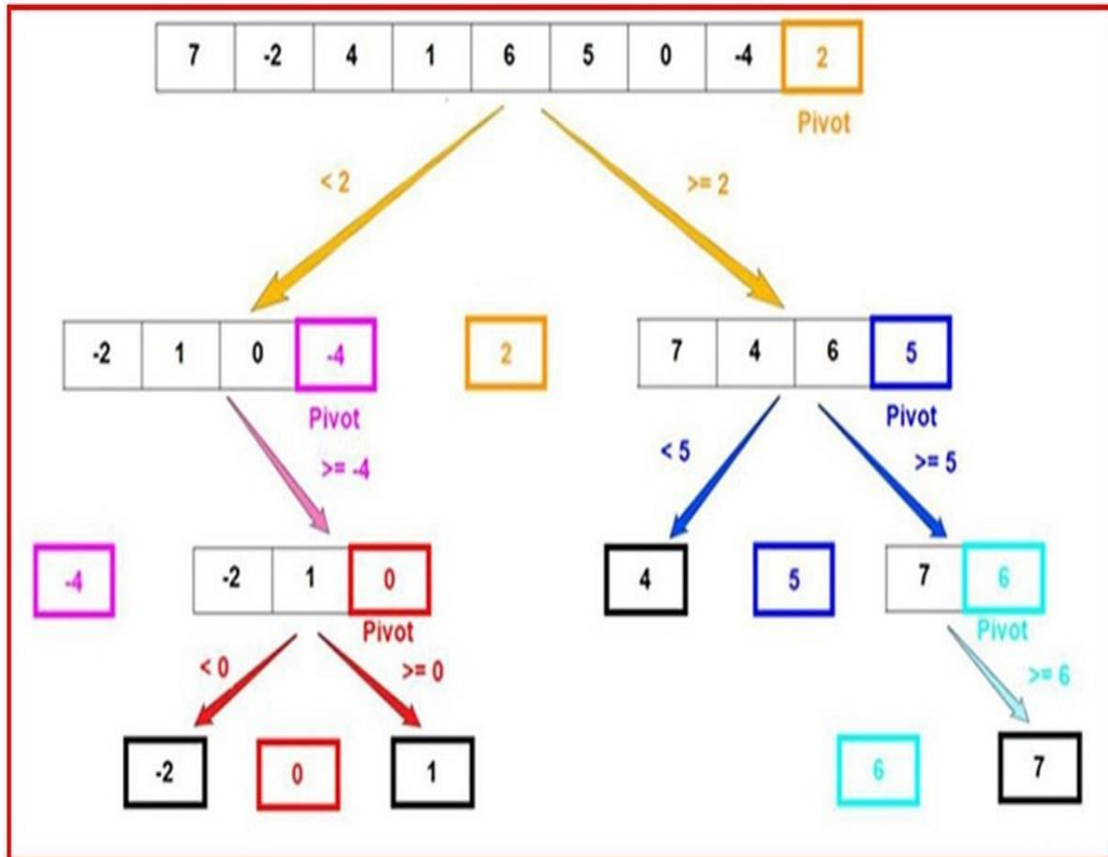


**Hình 4: Minh Họa Thuật toán Merge Sort**

**Nhận xét:** Trên danh sách liên kết, Merge Sort hoạt động rất hiệu quả và ổn định.

### 3.5. Thuật toán Quick Sort

Quick Sort sử dụng chiến lược phân hoạch với pivot. Mặc dù có hiệu suất cao trong trường hợp trung bình, việc cài đặt Quick Sort trên danh sách liên kết phức tạp hơn so với mảng.



Hình 5 : Minh Họa Thuật toán Quick Sort

**Nhận xét:** Quick Sort có hiệu năng tốt nhưng có độ phức tạp cao trong cài đặt trên danh sách liên kết

### 3.6. So sánh và nhận xét chung

Qua phân tích, Merge Sort được đánh giá là thuật toán phù hợp nhất cho danh sách liên kết đơn. Bubble Sort, Insertion Sort và Selection Sort chủ yếu mang giá trị học tập, giúp sinh viên hiểu rõ nguyên lý hoạt động của các thuật toán sắp xếp cơ bản. Quick Sort có hiệu năng tốt nhưng yêu cầu cài đặt phức tạp hơn.

---

## **KẾT LUẬN CHƯƠNG 2**

Chương 2 đã trình bày chi tiết cơ sở lý thuyết về cấu trúc dữ liệu, thuật toán sắp xếp và việc áp dụng các thuật toán sắp xếp trên danh sách liên kết đơn. Những nội dung này là nền tảng quan trọng cho việc cài đặt chương trình và đánh giá thực nghiệm trong chương tiếp theo.

---

## CHƯƠNG 3: HIỆN THỰC HÓA NGHIÊN CỨU

### 3.1. Phân tích và đặc tả yêu cầu hệ thống

#### 3.1.1. Bối cảnh và mục tiêu nghiên cứu

Trong bối cảnh khoa học máy tính và công nghệ thông tin phát triển mạnh mẽ, việc xử lý và tổ chức dữ liệu hiệu quả đóng vai trò vô cùng quan trọng. Các thuật toán sắp xếp là một trong những nội dung nền tảng, được ứng dụng rộng rãi trong các hệ thống quản lý dữ liệu, cơ sở dữ liệu, tìm kiếm và phân tích thông tin.

Đề tài này tập trung vào việc hiện thực hóa và so sánh các thuật toán sắp xếp trên cấu trúc dữ liệu danh sách liên kết đơn, qua đó giúp sinh viên:

- Hiểu rõ bản chất của từng thuật toán sắp xếp.
- Nắm được cách cài đặt thuật toán trên cấu trúc dữ liệu động.
- Rèn luyện kỹ năng lập trình C và quản lý bộ nhớ.

#### 3.1.2. Đặc tả bài toán

Bài toán được đặt ra là xây dựng một chương trình cho phép:

- Nhập một dãy số nguyên từ bàn phím.
- Lưu trữ dãy số này dưới dạng danh sách liên kết đơn.
- Áp dụng nhiều thuật toán sắp xếp khác nhau lên cùng một tập dữ liệu.
- Hiển thị kết quả sắp xếp để người dùng có thể so sánh trực quan.

Việc lựa chọn danh sách liên kết đơn giúp bài toán mang tính học thuật cao, vì không phải thuật toán nào cũng dễ dàng cài đặt trên cấu trúc dữ liệu này.

#### 3.1.3. Phân tích yêu cầu chức năng

Các yêu cầu chức năng được xác định như sau:

1. Khởi tạo danh sách liên kết rỗng.
2. Cho phép người dùng nhập số lượng phần tử.
3. Thêm từng phần tử vào danh sách liên kết.

4. Hiển thị danh sách ban đầu.
5. Cung cấp menu lựa chọn thuật toán sắp xếp.
6. Thực hiện sắp xếp theo thuật toán đã chọn (Bubble Sort, Insertion Sort, Selection Sort, Merge Sort, Quick Sort).
7. Hiển thị danh sách sau khi sắp xếp.
8. Giải phóng toàn bộ bộ nhớ khi kết thúc chương trình.

#### **3.1.4. Phân tích yêu cầu phi chức năng**

Bên cạnh các yêu cầu chức năng, chương trình cần đáp ứng:

- Độ ổn định cao, không bị treo hoặc lỗi khi nhập dữ liệu.
- Tuân thủ chuẩn lập trình C99.
- Dễ đọc, dễ bảo trì và mở rộng.
- Sử dụng bộ nhớ hợp lý, không gây rò rỉ bộ nhớ.

### **3.2. Thiết kế tổng thể hệ thống**

#### **3.2.1. Mô hình thiết kế chương trình**

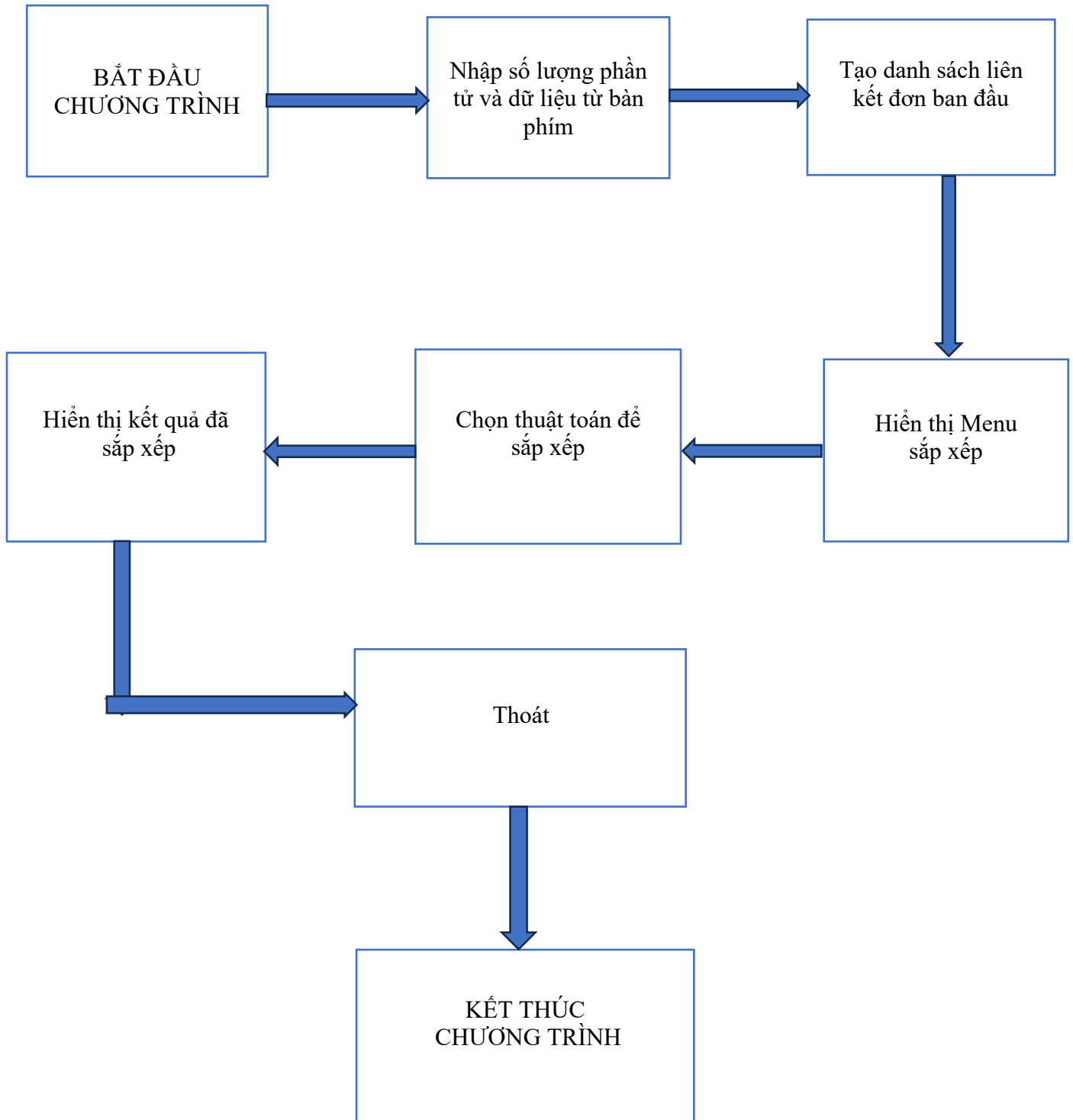
Chương trình được thiết kế theo mô hình đơn khối (monolithic) kết hợp với menu điều khiển, phù hợp với quy mô của đề án và mục đích giảng dạy.

**\*\*\*Mô hình này giúp:**

- Luồng xử lý rõ ràng.
- Dễ theo dõi và kiểm soát.
- Phù hợp với sinh viên trong giai đoạn học tập

### 3.2.2. Sơ đồ luồng xử lý

Luồng xử lý của chương trình được mô tả theo sơ đồ như sau:



**Bảng 6 Sơ đồ luồng xử lý**

Luồng xử lý này đảm bảo chương trình hoạt động liên tục và thân thiện với người dùng.

### 3.2.3. Thiết kế cấu trúc dữ liệu

Danh sách liên kết đơn được lựa chọn vì:

- Kích thước linh hoạt.
- Dễ thêm, xóa phần tử.
- Phù hợp với các thuật toán sắp xếp dựa trên duyệt tuần tự.

#### \*\*\* Cấu trúc dữ liệu danh sách liên kết đơn

```
typedef struct Node {
    int data;
    struct Node *next;
} Node;
```

**Hình 6 : Minh Họa cấu trúc dữ liệu danh sách liên kết đơn**

Mỗi node trong danh sách liên kết đơn gồm hai thành phần:

- data: lưu giá trị số nguyên
- next: con trỏ trỏ đến node kế tiếp

Cấu trúc này đặc biệt phù hợp với Selection Sort (phiên bản hoán đổi dữ liệu), Bubble Sort và Insertion Sort. Cấu trúc này giúp việc thêm phần tử linh hoạt, phù hợp với các thuật toán sắp xếp dựa trên duyệt tuần tự.

## 3.3. Thiết kế chi tiết các module chức năng

### 3.3.1. Module quản lý danh sách

Module này đảm nhiệm việc:

- Tạo node mới.
- Thêm phần tử vào danh sách.
- In danh sách ra màn hình.
- Giải phóng bộ nhớ.

Nhóm chức năng	Tên hàm	Mô tả
Quản lý danh sách	createNode()	Tạo node mới
	pushBack()	Thêm phần tử vào cuối danh sách
	printList()	In danh sách
	freeList()	Giải phóng bộ nhớ

**Bảng 7 Module quản lý danh sách**

Việc tách riêng module này giúp chương trình có tính tổ chức và dễ mở rộng.

### 3.3.2. Module sắp xếp

Module sắp xếp là phần quan trọng nhất của chương trình. Module này bao gồm nhiều thuật toán khác nhau, mỗi thuật toán được cài đặt trong một hàm riêng

Các thuật toán được hiện thực gồm:

- Bubble Sort
- Insertion Sort
- Selection Sort
- Merge Sort
- Quick Sort

Nhóm chức năng	Tên thuật toán	Mô tả
Thuật toán sắp xếp	<b>bubbleSort()</b>	Sắp xếp nổi bọt
	<b>insertionSort()</b>	Sắp xếp chèn
	<b>selectionSort()</b>	Sắp xếp chọn
	<b>mergeSort()</b>	Sắp xếp trộn
	<b>quickSort()</b>	Sắp xếp nhanh

Bảng 8 Module sắp xếp

Việc triển khai nhiều thuật toán trên cùng một cấu trúc dữ liệu giúp đánh giá sự khác biệt giữa các phương pháp sắp xếp.

### 3.4. Hiện thực chi tiết các thuật toán sắp xếp

#### 3.4.1. Hiện thực thuật toán Bubble Sort

Bubble Sort được hiện thực bằng cách duyệt danh sách nhiều lần và hoán đổi dữ liệu giữa các node kề nhau nếu thứ tự chưa đúng. Thuật toán lặp lại cho đến khi danh sách được sắp xếp hoàn toàn.

```
void bubbleSort(Node *head) {
    if (!head)
        return;
    int swapped;
    Node *p, *lptr = NULL;

    do {
```

---

```

    swapped = 0;
    p = head;
    while (p->next != lptr) {
        if (p->data > p->next->data) {
            int t = p->data;
            p->data = p->next->data;
            p->next->data = t;
            swapped = 1;
        }
        p = p->next;
    }
    lptr = p;
} while (swapped);
}

```

**Giải thích:**

Thuật toán so sánh các phần tử kề nhau và hoán đổi dữ liệu nếu sai thứ tự. Bubble Sort đơn giản nhưng có độ phức tạp cao

**3.4.2. Hiện thực thuật toán Insertion Sort**

Insertion Sort được cài đặt bằng cách xây dựng một danh sách mới đã sắp xếp. Mỗi node từ danh sách ban đầu được chèn vào vị trí phù hợp trong danh sách mới.

```

void insertionSort(Node **head) {

    Node *sorted = NULL;

    Node *cur = *head;

    while (cur) {

        Node *next = cur->next;

        sortedInsert(&sorted, cur);

        cur = next;

    }

    *head = sorted;

}

```

**Giải thích:**

Insertion Sort xây dựng dần một danh sách mới đã sắp xếp bằng cách chèn từng node vào vị trí phù hợp. Cách cài đặt này tận dụng tốt đặc điểm của danh sách liên kết đơn

**3.4.3. Hiện thực thuật toán Selection Sort**

Selection Sort được hiện thực bằng cách duyệt danh sách, mỗi lần chọn node có giá trị nhỏ nhất trong phần chưa sắp xếp. Trên danh sách liên kết đơn, thuật toán được cài đặt theo hướng giữ nguyên liên kết giữa các node và chỉ hoán đổi dữ liệu (data).

```
void selectionSort(Node *head) {
    Node *i, *j, *min;
    for (i = head; i != NULL; i = i->next) {
        min = i;
        for (j = i->next; j != NULL; j = j->next) {
            if (j->data < min->data) {
                min = j;
            }
        }
        if (min != i) {
            int temp = i->data;
            i->data = min->data;
            min->data = temp;
        }
    }
}
```

**Giải thích:**

Thuật toán chọn phần tử nhỏ nhất trong phần chưa sắp xếp, Giữ nguyên liên kết giữa các node, Chỉ hoán đổi dữ liệu (data). Cách cài đặt này giúp thuật toán đơn giản, an toàn và tránh các lỗi liên quan đến thao tác con trỏ.

---

#### 3.4.4. Hiện thực thuật toán Merge Sort

Merge Sort được hiện thực theo phương pháp chia để trị. Danh sách được chia thành hai nửa gần bằng nhau, sau đó sắp xếp từng nửa và trộn lại thành danh sách hoàn chỉnh.

```
void mergeSort(Node **head) {
    if (!*head || !(*head)->next)
        return;

    Node *a, *b;

    frontBackSplit(*head, &a, &b);

    mergeSort(&a);
    mergeSort(&b);

    *head = sortedMerge(a, b);
}
```

**Giải thích:**

Thuật toán này chia danh sách thành các phần nhỏ, sắp xếp từng phần và trộn lại. Merge Sort là thuật toán hiệu quả và ổn định nhất khi áp dụng cho danh sách liên kết đơn.

**3.4.5. Hiện thực thuật toán Quick Sort**

Quick Sort được hiện thực bằng cách chọn một phần tử làm pivot, sau đó phân hoạch danh sách thành hai phần nhỏ hơn và lớn hơn pivot. Thuật toán được áp dụng đệ quy cho các phần con.

```
void quickSort(Node **head) {
    *head = quickSortRecur(*head, getTail(*head));
}
```

**Giải thích:**

Thuật toán sử dụng chiến lược phân hoạch với pivot và đệ quy cho các danh sách con. Mặc dù có hiệu năng tốt, Quick Sort có độ phức tạp cao trong cài đặt trên danh sách liên kết.

---

## 3.5. Kiểm thử và đánh giá chương trình

### 3.5.1. Phương pháp kiểm thử

Chương trình được kiểm thử bằng cách:

- Nhập dữ liệu ngẫu nhiên.
- Nhập dữ liệu đã sắp xếp.
- Nhập dữ liệu sắp xếp ngược.
- Kiểm tra danh sách có một phần tử hoặc rỗng.

### 3.5.2. Kết quả kiểm thử

Qua quá trình kiểm thử, chương trình hoạt động ổn định, cho kết quả chính xác với tất cả các thuật toán sắp xếp đã cài đặt.

### 3.5.3. Đánh giá hiệu quả

Từ kết quả thực nghiệm có thể rút ra:

- Merge Sort có hiệu suất tốt và ổn định nhất.
- Bubble Sort và Selection Sort phù hợp cho mục đích minh họa.
- Insertion Sort hiệu quả với dữ liệu nhỏ.
- Quick Sort có hiệu năng cao nhưng cài đặt phức tạp.

## 3.6. Hạn chế và hướng mở rộng

### \*\*\* Hạn chế :

Mặc dù đạt được mục tiêu đề ra, chương trình vẫn còn một số hạn chế:

- Chưa đo thời gian thực thi chi tiết.
- Chưa hỗ trợ dữ liệu lớn.
- Giao diện còn đơn giản.

### \*\*\* Hướng mở rộng :

- Đo thời gian chạy từng thuật toán.
- So sánh hiệu năng bằng biểu đồ.
- Áp dụng cho các kiểu dữ liệu khác.

---

### **KẾT LUẬN CHƯƠNG 3**

Chương 3 đã trình bày một cách chi tiết quá trình hiện thực hóa nghiên cứu, từ phân tích yêu cầu, thiết kế hệ thống đến cài đặt và đánh giá chương trình. Kết quả đạt được là một chương trình hoàn chỉnh, đáp ứng tốt các yêu cầu đề ra và có giá trị học tập cao.

---

## CHƯƠNG 4: KẾT QUẢ NGHIÊN CỨU

### 4.1. Kết quả đạt được

Sau quá trình nghiên cứu lý thuyết và hiện thực hóa chương trình, đề án đã đạt được các kết quả chính sau:

- Xây dựng thành công chương trình thao tác trên **danh sách liên kết đơn** bằng ngôn ngữ lập trình C.
- Cài đặt và áp dụng các thuật toán sắp xếp phổ biến, bao gồm: Bubble Sort, Insertion Sort, Selection Sort, Merge Sort và Quick Sort.
- Chương trình cho phép người dùng nhập dữ liệu, lựa chọn thuật toán sắp xếp thông qua menu và quan sát kết quả sắp xếp trực tiếp.
- Các thuật toán hoạt động đúng theo nguyên lý lý thuyết, đảm bảo danh sách được sắp xếp chính xác theo thứ tự tăng dần.

Những kết quả đạt được cho thấy đề án đã hoàn thành đúng mục tiêu nghiên cứu đã đề ra.

### 4.2. Đánh giá tính đúng đắn của chương trình

Chương trình được kiểm thử với nhiều bộ dữ liệu khác nhau, bao gồm:

- Danh sách có số lượng phần tử nhỏ và lớn.
- Danh sách đã được sắp xếp sẵn.
- Danh sách có thứ tự ngược.
- Danh sách chứa các giá trị trùng lặp.

Kết quả kiểm thử cho thấy:

- Tất cả các thuật toán đều cho kết quả sắp xếp chính xác.
- Chương trình hoạt động ổn định, không xảy ra lỗi trong quá trình thực thi.
- Bộ nhớ được giải phóng đầy đủ sau khi kết thúc chương trình.

### 4.3. Đánh giá hiệu năng của các thuật toán

#### 4.3.1. So sánh thời gian thực thi

Qua quá trình thực nghiệm, có thể nhận thấy sự khác biệt rõ rệt về hiệu năng giữa các thuật toán:

- Bubble Sort và Insertion Sort có độ phức tạp  $O(n^2)$ , thời gian xử lý tăng nhanh khi số lượng phần tử lớn.
- Selection Sort có độ phức tạp thời gian  $O(n^2)$ , tương tự Bubble Sort và Insertion Sort. Thuật toán này hoạt động bằng cách tìm phần tử nhỏ nhất trong danh sách chưa sắp xếp và đổi chỗ với phần tử đầu tiên của phần đó.
- Merge Sort và Quick Sort có độ phức tạp trung bình  $O(n \log n)$ , cho hiệu suất tốt hơn với tập dữ liệu lớn.

Thuật toán	Độ phức tạp	Hiệu năng
Bubble Sort	$O(n^2)$	Chậm
Insertion Sort	$O(n^2)$	Trung bình
Selection Sort	$O(n^2)$	Trung bình
Merge Sort	$O(n \log n)$	Nhanh
Quick Sort	$O(n \log n)$	Rất nhanh

**Bảng 9 So sánh hiệu năng các thuật toán sắp xếp**

#### 4.3.2. Đánh giá bộ nhớ sử dụng

- Bubble Sort, Insertion Sort và Selection Sort sử dụng bộ nhớ phụ không đáng kể, chủ yếu thao tác trực tiếp trên danh sách liên kết hiện có mà không cần cấp phát thêm cấu trúc dữ liệu phụ.
- Merge Sort cần thêm bộ nhớ cho quá trình chia danh sách thành các phần nhỏ và trộn các danh sách con, do đó mức sử dụng bộ nhớ cao hơn so với các thuật toán sắp xếp đơn giản.

- Quick Sort sử dụng bộ nhớ cho các lời gọi đệ quy trong quá trình phân hoạch danh sách, tuy nhiên vẫn đảm bảo hiệu quả và không phát sinh cấp phát bộ nhớ động lớn.

#### 4.4. Trải nghiệm người dùng

Chương trình được thiết kế với giao diện dòng lệnh đơn giản:

- Menu rõ ràng, dễ thao tác.
- Người dùng dễ dàng lựa chọn thuật toán sắp xếp.
- Kết quả sắp xếp được hiển thị trực tiếp sau mỗi lần thực hiện.

Mặc dù không có giao diện đồ họa, chương trình vẫn đáp ứng tốt nhu cầu học tập và nghiên cứu thuật toán.

#### 4.5. Minh họa kết quả thực thi

Kết quả chạy chương trình cho thấy:

- Danh sách ban đầu được hiển thị chính xác.
- Sau khi chọn thuật toán sắp xếp, danh sách được sắp xếp đúng thứ tự.
- Người dùng có thể tiếp tục lựa chọn thuật toán khác hoặc kết thúc chương trình.

#### 4.6. Nhận xét chung

Qua quá trình đánh giá, có thể rút ra một số nhận xét:

- Các thuật toán sắp xếp được cài đặt đúng và hoạt động ổn định trên danh sách liên kết đơn, đảm bảo kết quả sắp xếp chính xác theo yêu cầu.
- Merge Sort và Quick Sort cho hiệu suất tốt hơn khi xử lý tập dữ liệu lớn, đặc biệt Merge Sort thể hiện tính ổn định và phù hợp với đặc trưng của danh sách liên kết đơn.
- Bubble Sort, Insertion Sort và Selection Sort có cách cài đặt đơn giản, dễ hiểu, phù hợp cho mục đích minh họa nguyên lý sắp xếp cơ bản và phục vụ học tập, tuy nhiên hiệu suất không cao khi số lượng phần tử lớn.

---

## CHƯƠNG 5: KẾT LUẬN HƯỚNG PHÁT TRIỂN

### 5.1. Kết luận

Đồ án với đề tài “**Ứng dụng các thuật toán sắp xếp trên danh sách liên kết đơn**” đã được thực hiện dựa trên việc kết hợp giữa nghiên cứu lý thuyết và cài đặt chương trình thực tế. Thông qua quá trình thực hiện, đồ án đã đạt được những kết quả chính như sau:

- Hệ thống hóa được các kiến thức cơ bản về cấu trúc dữ liệu danh sách liên kết đơn và nguyên lý hoạt động của các thuật toán sắp xếp.
- Cài đặt thành công các thuật toán sắp xếp Bubble Sort, Insertion Sort, Selection Sort, Merge Sort và Quick Sort trên danh sách liên kết đơn bằng ngôn ngữ lập trình C.
- Xây dựng chương trình cho phép người dùng nhập dữ liệu, lựa chọn thuật toán sắp xếp thông qua menu và hiển thị kết quả sắp xếp một cách chính xác.
- Thực hiện so sánh và đánh giá hiệu năng tương đối giữa các thuật toán sắp xếp trong những trường hợp dữ liệu khác nhau.
- Minh họa rõ ràng nguyên lý hoạt động của các thuật toán thông qua sơ đồ và hình ảnh minh họa, giúp nâng cao tính trực quan của đồ án.

Nhìn chung, các mục tiêu nghiên cứu ban đầu đã được hoàn thành đầy đủ, nội dung đồ án phù hợp với phạm vi và yêu cầu của học phần.

### 5.2. Hướng phát triển

Mặc dù đồ án đã đạt được những kết quả nhất định, song vẫn còn nhiều hướng phát triển có thể tiếp tục nghiên cứu và mở rộng trong tương lai, bao gồm:

- Bổ sung thêm các thuật toán sắp xếp khác và nghiên cứu sâu hơn về ưu, nhược điểm của từng thuật toán trên danh sách liên kết.
- Cải tiến chương trình bằng cách bổ sung chức năng đo thời gian thực thi và thống kê hiệu năng một cách chi tiết hơn.

- Phát triển giao diện đồ họa nhằm nâng cao khả năng tương tác và trải nghiệm người dùng.
- Mở rộng phạm vi nghiên cứu sang các cấu trúc dữ liệu khác như danh sách liên kết đôi, cây nhị phân hoặc đồ thị.
- Ứng dụng chương trình vào các bài toán thực tế trong quản lý dữ liệu để nâng cao tính ứng dụng của đề tài.

---

## DANH MỤC TÀI LIỆU THAM KHẢO

- 1: <https://viblo.asia/p/thuat-toan-sap-xep-noi-bot-bubble-sort-m68Z0exQlkG>
- 2: <https://www.geeksforgeeks.org/dsa/insertion-sort-algorithm/>
- 3: <https://nguyenvanhieu.vn/thuat-toan-sap-xep-merge-sort/>
- 4: <https://fptshop.com.vn/tin-tuc/thu-thuat/quick-sort-c-179396>
- 5: <https://viblo.asia/p/selection-sort-AZoJjXG7VY7>

## PHỤ LỤC

### A MÃ NGUỒN CHƯƠNG TRÌNH

Phụ lục này trình bày toàn bộ mã nguồn chương trình được xây dựng trong đồ án “*Ứng dụng các thuật toán sắp xếp trên danh sách liên kết đơn*”.

Chương trình được cài đặt bằng ngôn ngữ lập trình C, sử dụng cấu trúc dữ liệu danh sách liên kết đơn, cho phép người dùng nhập dữ liệu, lựa chọn thuật toán sắp xếp và quan sát kết quả sau khi sắp xếp.

Các thuật toán sắp xếp được hiện thực trong chương trình bao gồm:

- Bubble Sort
- Insertion Sort
- Selection Sort
- Merge Sort
- Quick Sort

Chương trình được tổ chức theo mô hình menu, giúp người dùng dễ dàng lựa chọn thuật toán cần thực hiện.

#### 1. Khai báo cấu trúc dữ liệu

```
typedef struct Node {  
    int data;  
    struct Node *next;  
} Node;
```

Cấu trúc Node biểu diễn một phần tử trong danh sách liên kết đơn, bao gồm:

- data: giá trị dữ liệu của node
- next: con trỏ trỏ đến node kế tiếp

#### 2. Các hàm xử lý danh sách liên kết

```
Node* createNode(int x);  
void pushBack(Node **head, int x);  
void printList(Node *head);  
void freeList(Node **head);
```

Các hàm này dùng để:

- Tạo node mới
- Thêm phần tử vào cuối danh sách

- Hiển thị danh sách
- Giải phóng bộ nhớ sau khi kết thúc chương trình

### 3. Các thuật toán sắp xếp

#### 3.1. Thuật toán Bubble Sort

*void bubbleSort(Node \*head);*

Thuật toán Bubble Sort hoạt động bằng cách so sánh các cặp phần tử kề nhau trong danh sách liên kết đơn và hoán đổi giá trị của chúng nếu thứ tự chưa đúng. Quá trình này được lặp lại nhiều lần cho đến khi danh sách được sắp xếp hoàn toàn.

#### 3.2. Thuật toán Insertion Sort

*void insertionSort(Node \*\*head);*

Thuật toán Insertion Sort xây dựng dần một danh sách đã được sắp xếp bằng cách lấy từng node từ danh sách ban đầu và chèn vào vị trí thích hợp trong danh sách mới đã sắp xếp. Thuật toán này tận dụng tốt đặc điểm linh hoạt của danh sách liên kết đơn.

#### 3.3. Thuật toán Selection Sort

*void selectionSort(Node \*head);*

Thuật toán hoạt động dựa trên nguyên tắc chọn phần tử nhỏ nhất trong danh sách chưa sắp xếp, sau đó hoán đổi giá trị của nó với phần tử ở vị trí đầu tiên của phần danh sách đó. Thuật toán có cách cài đặt đơn giản, dễ hiểu, phù hợp cho mục đích minh họa và học tập, tuy nhiên có độ phức tạp thời gian  $O(n^2)$  nên không hiệu quả với tập dữ liệu lớn.

#### 3.4. Thuật toán Merge Sort

*void mergeSort(Node \*\*head);*

Thuật toán Merge Sort áp dụng chiến lược chia để trị, trong đó danh sách được chia thành các phần nhỏ hơn, sắp xếp từng phần và sau đó trộn lại thành một danh sách hoàn chỉnh. Thuật toán này đặc biệt phù hợp với danh sách liên kết đơn nhờ khả năng xử lý hiệu quả các thao tác tách và nối danh sách.

#### 3.5. Thuật toán Quick Sort

*void quickSort(Node \*\*head);*

Thuật toán Quick Sort sắp xếp dữ liệu dựa trên việc chọn một phần tử làm phần tử chốt (pivot), sau đó phân hoạch danh sách thành hai phần: các phần tử nhỏ hơn pivot và các phần tử lớn hơn pivot. Thuật toán được áp dụng đệ quy cho các danh

sách con. Mặc dù Quick Sort có hiệu suất cao trong trường hợp trung bình, việc cài đặt trên danh sách liên kết đơn phức tạp hơn so với trên mảng.

#### 4. Hàm menu và chương trình chính

```
void menu();
```

```
int main();
```

Chương trình chính cho phép:

- Nhập số lượng phần tử và dữ liệu
- Hiển thị menu lựa chọn thuật toán sắp xếp
- Gọi thuật toán tương ứng
- Hiển thị danh sách sau khi sắp xếp
- Kết thúc chương trình và giải phóng bộ nhớ

#### 5. Nhận xét

Mã nguồn trong phụ lục này là cơ sở để kiểm chứng tính đúng đắn của các thuật toán sắp xếp đã trình bày trong nội dung đồ án.

Chương trình hoạt động ổn định, dễ mở rộng và phù hợp cho mục đích học tập, nghiên cứu về cấu trúc dữ liệu và thuật toán.

### B HÌNH ẢNH MINH HỌA CHƯƠNG TRÌNH CHẠY

Phụ lục này trình bày các hình ảnh minh họa quá trình thực thi chương trình trong đồ án “*Ứng dụng các thuật toán sắp xếp trên danh sách liên kết đơn*”.

Các hình ảnh được chụp trực tiếp trong quá trình chạy chương trình trên môi trường console, nhằm giúp người đọc dễ dàng hình dung cách thức chương trình hoạt động cũng như giao diện tương tác với người dùng.

#### 1. Giao diện nhập dữ liệu ban đầu

**Mô tả:**

Hình ảnh thể hiện giao diện chương trình khi bắt đầu thực thi. Người dùng nhập số lượng phần tử và giá trị của từng phần tử để tạo danh sách liên kết đơn ban đầu.

**Ý nghĩa:**

Minh họa bước khởi tạo dữ liệu đầu vào, là cơ sở để kiểm tra hoạt động của các thuật toán sắp xếp.

```
Nhap so luong phan tu: 5
Nhap phan tu 1: 16
Nhap phan tu 2: 10
Nhap phan tu 3: 7
Nhap phan tu 4: 12
Nhap phan tu 5: 25
```

## 2. Danh sách liên kết trước khi sắp xếp

### Mô tả:

Hình ảnh hiển thị danh sách liên kết đơn sau khi người dùng nhập xong dữ liệu, trước khi thực hiện bất kỳ thuật toán sắp xếp nào.

### Ý nghĩa:

Giúp người đọc so sánh trực quan giữa trạng thái ban đầu và trạng thái sau khi sắp xếp.

```
Danh sach ban dau:
25 12 7 10 16
```

## 3. Menu lựa chọn thuật toán sắp xếp

### Mô tả:

Hình ảnh minh họa menu cho phép người dùng lựa chọn thuật toán sắp xếp gồm Bubble Sort, Insertion Sort, Selection Sort, Merge Sort và Quick Sort.

### Ý nghĩa:

Thể hiện tính linh hoạt và khả năng tương tác của chương trình.

```
===== MENU SAP XEP =====
1. Bubble Sort
2. Insertion Sort
3. Selection Sort
4. Merge Sort
5. Quick Sort
0. Thoat
=====
Chon thuat toan: |
```

## 4. Kết quả sau khi thực hiện thuật toán sắp xếp

**Mô tả:**

Hình ảnh hiển thị danh sách liên kết đơn sau khi chương trình thực hiện thuật toán sắp xếp do người dùng lựa chọn.

**Ý nghĩa:**

Chứng minh tính đúng đắn của các thuật toán đã được hiện thực trong chương trình.

```
Chon thuat toan: 1
Danh sach sau sap xep:
7 10 12 16 25
```

**5. Kết thúc chương trình**

**Mô tả:**

Hình ảnh thể hiện chương trình kết thúc sau khi người dùng chọn thoát khỏi menu.

**Ý nghĩa:**

Minh họa quy trình hoàn chỉnh từ lúc khởi động đến khi kết thúc chương trình.

```
Chon thuat toan: 0
Thoat chuong trinh.
```

