

**BỘ GIÁO DỤC VÀ ĐÀO TẠO  
TRƯỜNG ĐẠI HỌC SƯ PHẠM THÀNH PHỐ HỒ CHÍ MINH  
KHOA CÔNG NGHỆ THÔNG TIN**

~~~~~\*~~~~~



**TRƯỜNG ĐẠI HỌC  
Sư phạm Thành phố Hồ Chí Minh**  
HCMC University of Education



# **ĐỒ ÁN CUỐI KỲ**

## **ĐỀ TÀI**

## **XÂY DỰNG CHƯƠNG TRÌNH MÔ PHỎNG CÁC THUẬT TOÁN SẮP XẾP CƠ BẢN**

**Học phần: 2111COMP101904 – LẬP TRÌNH TRÊN WINDOWS**

|                       |   |                       |               |
|-----------------------|---|-----------------------|---------------|
| Sinh viên thực hiện 1 | : | Nguyễn Hữu Minh Dương | 46.01.104.039 |
| Sinh viên thực hiện 2 | : | Thái Khánh Ngọc       | 46.01.104.120 |
| Sinh viên thực hiện 3 | : | Phước Công Nguyên     | 46.01.104.125 |
| Sinh viên thực hiện 4 | : | Huỳnh Trần Như Quỳnh  | 46.01.104.153 |

**TP HỒ CHÍ MINH – 1/2021**

BỘ GIÁO DỤC VÀ ĐÀO TẠO  
TRƯỜNG ĐẠI HỌC SƯ PHẠM THÀNH PHỐ HỒ CHÍ MINH  
KHOA CÔNG NGHỆ THÔNG TIN

~~~~~\*~~~~~



TRƯỜNG ĐẠI HỌC  
Sư phạm Thành phố Hồ Chí Minh  
HCMC University of Education



## ĐỒ ÁN CUỐI KỲ

### ĐỀ TÀI

### XÂY DỰNG CHƯƠNG TRÌNH MÔ PHỎNG CÁC THUẬT TOÁN SẮP XẾP CƠ BẢN

Học phần: 2111COMP101904 – LẬP TRÌNH TRÊN WINDOWS

Giảng viên hướng dẫn: ThS. Võ Tiến An

TP HỒ CHÍ MINH – 1/2021

# LỜI CAM ĐOAN

Chúng tôi xin cam đoan rằng đồ án cuối kỳ với đề tài “**Xây dựng chương trình mô phỏng các thuật toán sắp xếp cơ bản**” được tiến hành một cách công khai, minh bạch. Mọi thứ được dựa trên sự cố gắng cũng như sự nỗ lực cùng với sự hợp tác của các thành viên trong nhóm.

Các thông số và kết quả nghiên cứu được đưa ra trong đồ án là trung thực và không sử dụng kết quả của bất kỳ đề tài nghiên cứu nào tương tự. Nếu như phát hiện rằng có sự sao chép kết quả nghiên cứu đề những đề tài khác chúng tôi xin chịu hoàn toàn trách nhiệm.

# MỤC LỤC

|   |    |
|---|----|
| CHƯƠNG 1. CƠ SỞ LÝ THUYẾT .....                 | 1  |
| 1.1. Lý thuyết chung về thuật toán sắp xếp..... | 1  |
| 1.2. Các thuật toán sắp xếp .....               | 1  |
| 1.2.1. Sắp xếp chèn (Insertion Sort).....       | 1  |
| 1.2.2. Sắp xếp chọn (Selection Sort).....       | 3  |
| 1.2.3. Sắp xếp nổi bọt (Bubble Sort).....       | 4  |
| 1.2.4. Sắp xếp trộn (Merge Sort).....           | 5  |
| CHƯƠNG 2. CHƯƠNG TRÌNH.....                     | 8  |
| 2.1. Giới thiệu chương trình.....               | 8  |
| 2.2. Các thành phần trong chương trình.....     | 8  |
| 2.3. Mô tả chương trình.....                    | 9  |
| 2.4. Hàm thuật toán trong chương trình.....     | 12 |
| CHƯƠNG 3. KẾT LUẬN .....                        | 24 |
| 4.1. Kết quả đạt được .....                     | 24 |
| 4.2. Hạn chế .....                              | 24 |
| 4.3. Hướng phát triển .....                     | 24 |
| 4.4. Phân công công việc .....                  | 24 |

# DANH MỤC KÝ HIỆU VÀ CHỮ VIẾT TẮT

**DANH MỤC CÁC BẢNG BIỂU**

Bảng 2.1. Các thành phần trong chương trình .....9

Bảng 3.1. Bảng phân công công việc.....25

# DANH MỤC CÁC HÌNH VẼ, BIỂU ĐỒ

|  |    |
|--|----|
| Hình 1.1. Ví dụ minh hoạ 1 cho thuật toán sắp xếp chèn .....   | 2  |
| Hình 1.2. Ví dụ minh hoạ 2 cho thuật toán sắp xếp chèn .....   | 2  |
| Hình 1.3. Ví dụ minh hoạ 1 cho thuật toán sắp xếp chọn .....   | 3  |
| Hình 1.4. Ví dụ minh hoạ 2 cho thuật toán sắp xếp chọn .....   | 4  |
| Hình 1.5. Ví dụ minh hoạ 1 cho thuật toán sắp xếp nổi bọt..... | 5  |
| Hình 1.6. Ví dụ minh hoạ 2 cho thuật toán sắp xếp nổi bọt..... | 5  |
| Hình 1.7. Ví dụ minh hoạ 1 cho thuật toán sắp xếp trộn .....   | 6  |
| Hình 1.8. Ví dụ minh hoạ 2 cho thuật toán sắp xếp trộn .....   | 7  |
|  |    |
| Hình 2.1. Giao diện chính của chương trình .....               | 8  |
| Hình 2.2. Nhập dữ liệu .....                                   | 9  |
| Hình 2.3. Hiển thị dữ liệu .....                               | 10 |
| Hình 2.4. Lỗi nhập dữ liệu .....                               | 10 |
| Hình 2.5. Chọn thuật toán sắp xếp .....                        | 11 |
| Hình 2.6. Chọn cách sắp xếp.....                               | 11 |
| Hình 2.7. Xoá dữ liệu .....                                    | 12 |
| Hình 2.8. Lỗi xoá dữ liệu .....                                | 12 |

# MỞ ĐẦU

## 1. Lý do chọn đề tài

Trong thời đại công nghệ thông tin như hiện nay, nhờ công nghệ mới mà lượng dữ liệu khổng lồ được tạo ra. Vì vậy việc quản lý có thể rất khó khăn nếu những dữ liệu đó không được sắp xếp theo một trật tự nhất định. Chính nhu cầu muốn sắp xếp lại dữ liệu, thuật toán sắp xếp đã được tạo ra.

Thuật toán sắp xếp có rất nhiều loại. Mỗi một loại thuật toán đều có một độ phức tạp nhất định, đòi hỏi chúng ta phải tìm hiểu mới hiểu được rõ ràng về chức năng, cơ chế và những thứ liên quan đến thuật toán. Dẫn đến một khó khăn là mất thời gian để chúng ta làm công việc tìm hiểu.

Chính vì điều đó mà các chương trình mô phỏng thuật toán sắp xếp được chú trọng. Nhờ mô phỏng thuật toán, chúng ta có thể hiểu dễ dàng được cách các thuật toán sắp xếp chạy như thế nào. Và việc dạy học và học của chúng ta được trở nên đơn giản và mất ít thời gian hơn việc tự tìm hiểu bằng những lý thuyết.

Vì những lý do trên, chúng tôi đã chọn đề tài **“Xây dựng chương trình mô phỏng thuật toán sắp xếp cơ bản”**.

## 2. Nhiệm vụ nghiên cứu

- Nghiên cứu tổng quan về các thuật toán sắp xếp cơ bản.
- Nghiên cứu về cách xây dựng mô phỏng.
- Áp dụng kết quả nghiên cứu xây dựng chương trình mô phỏng thuật toán sắp xếp cơ bản.

## 3. Phạm vi nghiên cứu

- Phạm vi thời gian: bắt đầu nghiên cứu từ ngày 15/11/2021, kết thúc ngày 06/01/2022.
- Phạm vi nội dung: Các thuật toán sắp xếp phổ biến như sắp xếp chèn, sắp xếp trộn, sắp xếp chọn và sắp xếp nổi bọt.

## 4. Phương pháp nghiên cứu

- Nghiên cứu tổng quan về cách mô phỏng thuật toán.



- Hướng đến kỹ thuật lập trình, trọng tâm là ngôn ngữ lập trình C# / C++.
- Áp dụng kết quả nghiên cứu làm một video demo hướng dẫn sử dụng chương trình mô phỏng thuật toán sắp xếp.

## **5. Cấu trúc của đề tài**

Đề tài gồm: Mở đầu, 3 chương và danh mục tài liệu tham khảo.

# CHƯƠNG 1. CƠ SỞ LÝ THUYẾT

## 1.1. Lý thuyết chung về thuật toán sắp xếp

*Sắp xếp* là quá trình trình bày lại các phần tử trong một tập hợp theo một trình tự xác định nào đó nhằm mục đích tìm kiếm và quản lý các phần tử trong tập hợp đó dễ dàng và nhanh chóng.

Ví dụ cho việc sắp xếp đó chính là các ứng dụng quản lý danh bạ điện thoại thì có sắp xếp theo tên hoặc số. Quản lý học sinh thì có sắp xếp theo mã học sinh, theo lớp. Ngoài ra còn các ví dụ khác như quản lý thư viện, quản lý nhân sự, ... đều có mặt của công cụ sắp xếp.

Như chúng ta đã thấy, có rất nhiều thứ cần phải được sắp xếp theo một trình tự nhất định. Điển hình như trong lúc học tập, cụ thể là học lập trình, đề bài yêu cầu sắp xếp dãy số theo một trình tự nhất định bằng những kỹ thuật sắp xếp được nghiên cứu kỹ lưỡng.

Trong khoa học máy tính và toán học, *thuật toán sắp xếp* là một thuật toán sắp xếp các phần tử của một danh sách (hoặc mảng) theo một thứ tự cụ thể (tăng hoặc giảm). Và để thuận tiện cho việc học tập và nghiên cứu, người ta thường kí hiệu các phần tử đã sắp xếp là các số.

## 1.2. Các thuật toán sắp xếp

### 1.2.1. Sắp xếp chèn (*Insertion Sort*)

**Giải thuật sắp xếp chèn** là một giải thuật sắp xếp dựa trên việc so sánh in-place.

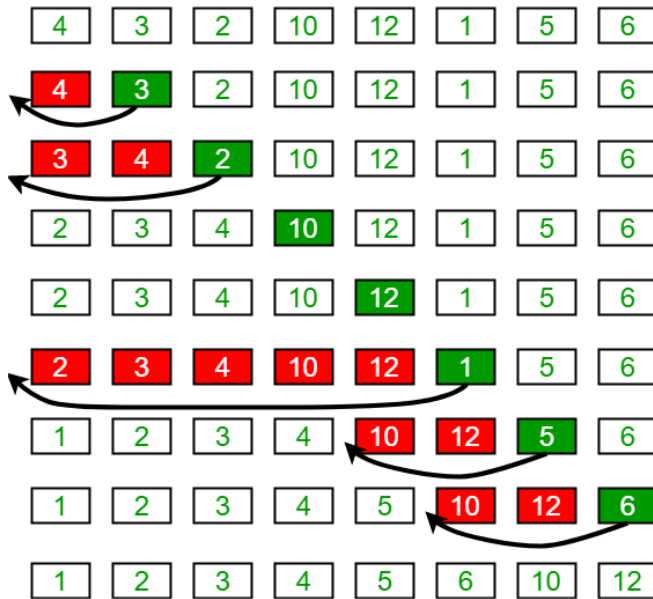
\**In-place* là không yêu cầu thêm bất kỳ bộ nhớ phụ và việc sắp xếp được tiến hành trong chính phần bộ nhớ đã khai báo trước đó.

**Ý tưởng:** Lấy ý tưởng từ việc “chơi bài”, dựa theo cách người chơi chèn thêm một quân bài mới vào bộ bài đã được sắp xếp trên tay.

**Thuật toán:**

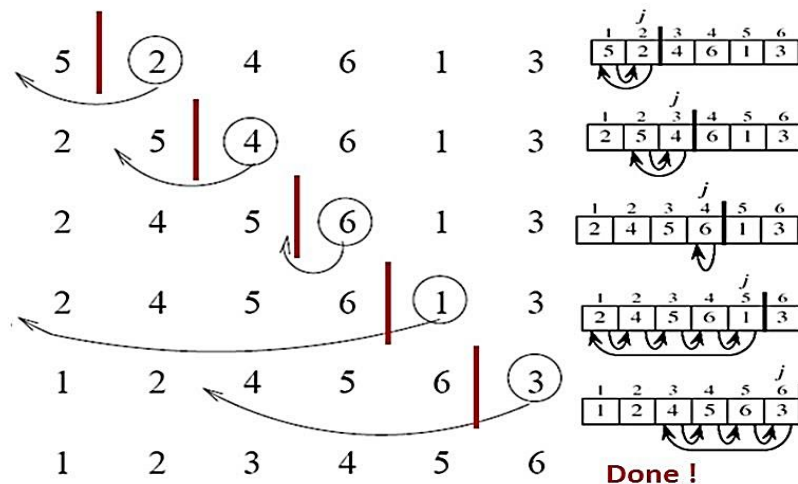
- Tại bước  $k = 1, 2, \dots, n$  đưa phần tử thứ  $k$  trong mảng đã cho vào đúng vị trí trong dãy gồm  $k$  phần tử đầu tiên.
- Kết quả là sau bước thứ  $k$ , sẽ có  $k$  phần tử đầu tiên được sắp xếp theo thứ tự.

**Ví dụ 1:** Sắp xếp dãy số (4, 3, 2, 10, 12, 1, 5, 6) bằng thuật toán Insertion Sort.



Hình 1.1. Ví dụ minh họa 1 cho thuật toán sắp xếp chèn

**Ví dụ 2:** Sắp xếp dãy số (5, 2, 4, 6, 1, 3) bằng thuật toán Insertion Sort.



Hình 1.2. Ví dụ minh họa 2 cho thuật toán sắp xếp chèn

**Đánh giá:**

- Best Case: 0 hoán đổi,  $n - 1$  so sánh (khi dãy đầu vào là đã được sắp)

- Worst Case:  $n^2/2$  hoán đổi và so sánh (khi dãy đầu vào có thứ tự ngược lại với thứ tự cần sắp xếp)
- Average Case:  $n^2/4$  hoán đổi và so sánh.

### 1.2.2. Sắp xếp chọn (Selection Sort)

**Giải thuật sắp xếp chọn** là một giải thuật sắp xếp trong đó danh sách được chia làm hai phần, phần được sắp xếp (sorted list) ở bên trái và phần chưa được sắp xếp (unsorted list) ở bên phải. Ban đầu, phần được sắp xếp là trống và phần chưa được sắp xếp là toàn bộ danh sách ban đầu.

**Ý tưởng:** Tìm từng phần tử cho mỗi vị trí của mảng hoán vị A' cần tìm.

**Thuật toán:**

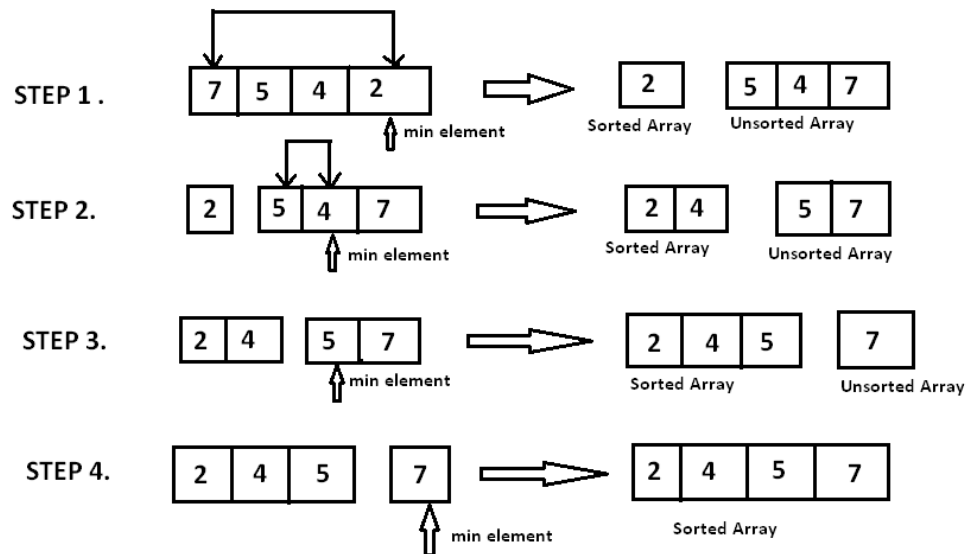
- Tìm phần tử nhỏ nhất đưa vào vị trí 1
- Tìm phần tử nhỏ tiếp theo đưa vào vị trí 2
- Tìm phần tử nhỏ tiếp theo đưa vào vị trí 3
- ...

**Ví dụ 1:** Sắp xếp dãy số (5, 6, 2, 2, 10, 12, 9, 10, 9, 3) bằng thuật toán Selection Sort.

| Khóa<br>Bước | a[1] | a[2] | a[3] | a[4] | a[5] | a[6] | a[7] | a[8] | a[9] | a[10] |
|--------------|------|------|------|------|------|------|------|------|------|-------|
| Ban đầu      | 5    | 6    | 2    | 2    | 10   | 12   | 9    | 10   | 9    | 3     |
| Bước 1       | 2    | 6    | 5    | 2    | 10   | 12   | 9    | 10   | 9    | 3     |
| Bước 2       |      | 2    | 5    | 6    | 10   | 12   | 9    | 10   | 9    | 3     |
| Bước 3       |      |      | 3    | 6    | 10   | 12   | 9    | 10   | 9    | 5     |
| Bước 4       |      |      |      | 5    | 10   | 12   | 9    | 10   | 9    | 6     |
| Bước 5       |      |      |      |      | 6    | 12   | 9    | 10   | 9    | 10    |
| Bước 6       |      |      |      |      |      | 9    | 12   | 10   | 9    | 10    |
| Bước 7       |      |      |      |      |      |      | 9    | 10   | 12   | 10    |
| Bước 8       |      |      |      |      |      |      |      | 10   | 12   | 10    |
| Bước 9       |      |      |      |      |      |      |      |      | 10   | 12    |
| Kết quả      | 2    | 2    | 3    | 5    | 6    | 9    | 9    | 10   | 10   | 12    |

Hình 1.3. Ví dụ minh họa 1 cho thuật toán sắp xếp chọn

**Ví dụ 2:** Sắp xếp dãy số (7, 5, 4, 2) bằng thuật toán Selection Sort.



Hình 1.4. Ví dụ minh họa 2 cho thuật toán sắp xếp chọn

**Đánh giá:**

- Best case: 0 đổi chỗ ( $n - 1$  như trong đoạn mã),  $n_2/2$  so sánh.
- Worst case:  $n - 1$  đổi chỗ và  $n^2/2$  so sánh.
- Average case:  $O(n)$  đổi chỗ và  $n^2/2$  so sánh.

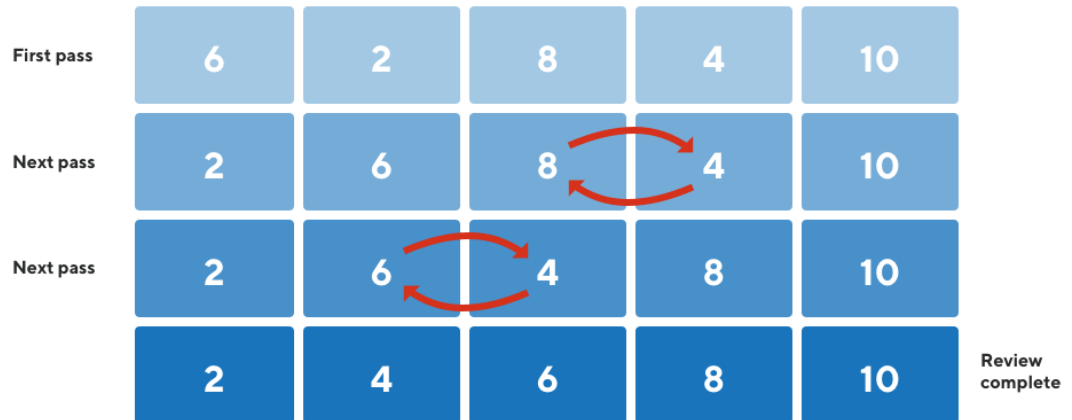
**1.2.3. Sắp xếp nổi bọt (Bubble Sort)**

**Thuật toán sắp xếp nổi bọt** là thuật toán đẩy phần tử lớn nhất xuống cuối dãy, đồng thời những phần tử nhỏ hơn sẽ dịch chuyển dần về đầu dãy.

**Ý tưởng:** Lấy ý tưởng từ sự “nổi bọt”, những phần tử nhẹ hơn sẽ nổi lên trên và ngược lại, những phần tử lớn hơn sẽ chìm xuống dưới.

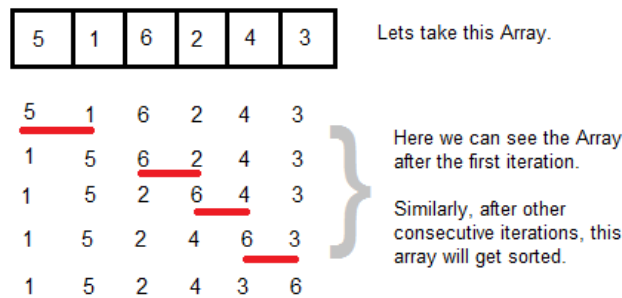
**Thuật toán:** Duyệt mảng từ phần tử đầu tiên. Ta sẽ so sánh mỗi phần tử với phần tử liền trước nó, nếu chúng đứng sai vị trí, ta sẽ đổi chỗ chúng cho nhau. Quá trình này sẽ được dừng nếu gặp lần duyệt từ đầu dãy đến cuối dãy mà không phải thực hiện đổi chỗ bất kỳ 2 phần tử nào (tức là tất cả các phần tử đã được sắp xếp đúng vị trí).

**Ví dụ 1:** Sắp xếp dãy số (6, 2, 8, 4, 10) bằng thuật toán Bubble Sort.



Hình 1.5. Ví dụ minh họa 1 cho thuật toán sắp xếp nổi bọt

**Ví dụ 2:** Sắp xếp dãy số (5, 1, 6, 2, 4, 3) bằng thuật toán Bubble Sort.



Hình 1.6. Ví dụ minh họa 2 cho thuật toán sắp xếp nổi bọt

### Đánh giá:

Tuy đơn giản nhưng Bubble Sort là thuật toán kém hiệu quả nhất so với Insertion Sort và Selection Sort.

- Best case: 0 đổi chỗ,  $n^2/2$  so sánh.
- Worst case:  $n^2/2$  đổi chỗ và so sánh.
- Average case:  $n^2/4$  đổi chỗ và  $n^2/2$  so sánh.

### 1.2.4. Sắp xếp trộn (Merge Sort)

**Thuật toán sắp xếp trộn** là một thuật toán sắp xếp thuộc loại sắp xếp nhanh trong khoa học máy tính. Cụ thể với bài toán sắp xếp, nó sẽ chia nhỏ danh sách cần sắp xếp thành từng phần tử rồi sau đó hòa nhập theo phương pháp trộn tự nhiên thành dãy có thứ tự.

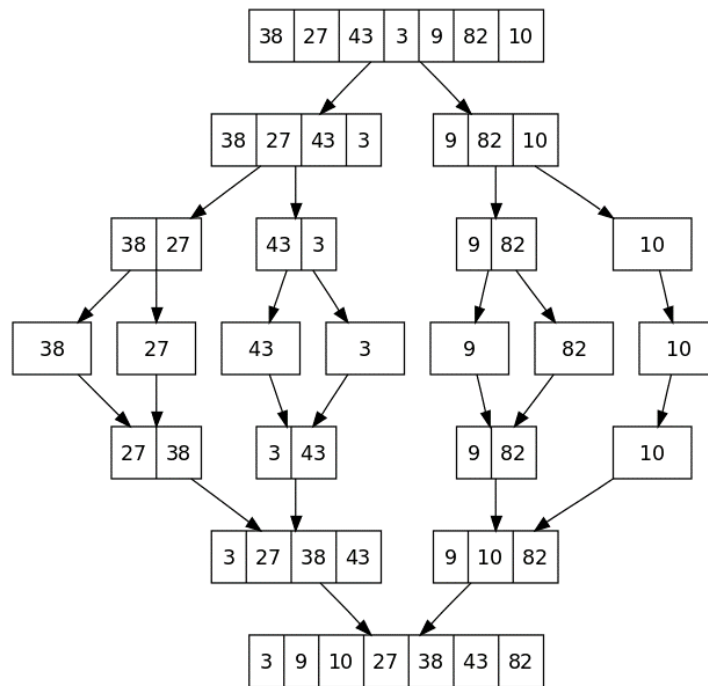
**Ý tưởng:** Lấy ý tưởng từ việc “chia để trị”, từ đó chia nhỏ bài toán thành các bài toán nhỏ hơn, sau đó giải quyết chúng. Thuật toán này giúp xử lý dữ liệu lớn một cách tốt hơn, tối ưu về mặt thời gian.

**Thuật toán:** Giả sử có hai dãy đã được sắp xếp  $L[1..n_1]$  và  $R[1..n_2]$ . Ta có thể trộn chúng lại thành một dãy mới  $M[1..n_1 + n_2]$  được sắp xếp theo cách sau:

- So sánh hai phần tử đứng đầu của hai dãy, lấy phần tử nhỏ hơn cho vào dãy mới. Tiếp tục như vậy cho tới khi một trong hai dãy rỗng.
- Khi một trong hai dãy rỗng, ta lấy phần còn lại của dãy kia cho vào cuối dãy mới.

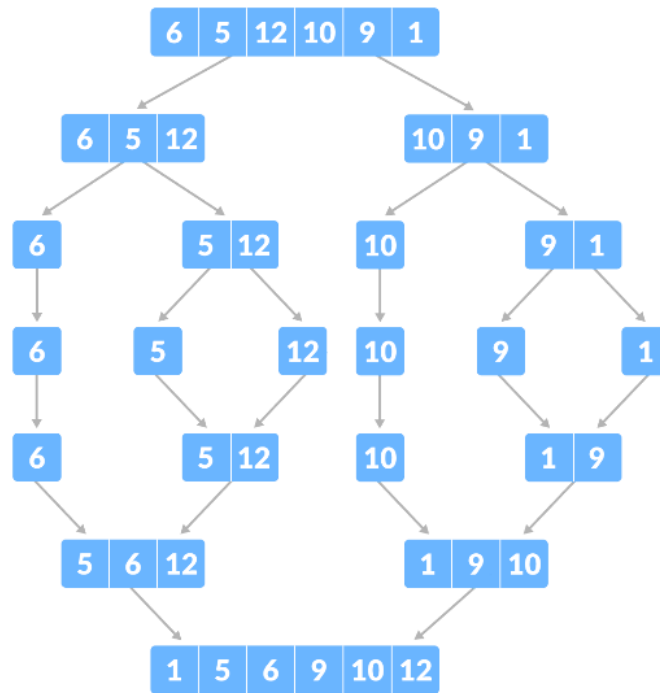
Khi đó, ta sẽ thu được dãy cần tìm.

**Ví dụ 1:** Cho danh sách (38, 27, 43, 3, 9, 82, 10). Sắp xếp chúng bằng thuật toán Merge Sort.



Hình 1.7. Ví dụ minh họa 1 cho thuật toán sắp xếp trộn

**Ví dụ 2:** Sắp xếp dãy số (6, 5, 12, 10, 9, 1) theo thứ tự từ nhỏ đến lớn bằng thuật toán Merge Sort.



Hình 1.8. Ví dụ minh họa 2 cho thuật toán sắp xếp trộn

**Đánh giá:**  $O(n \times \log n)$ .



## CHƯƠNG 2. CHƯƠNG TRÌNH

### 2.1. Giới thiệu chương trình

Chương trình này minh họa trực quan các thuật toán sắp xếp cơ bản.

Ngôn ngữ chính được sử dụng để viết chương trình là ngôn ngữ C#.

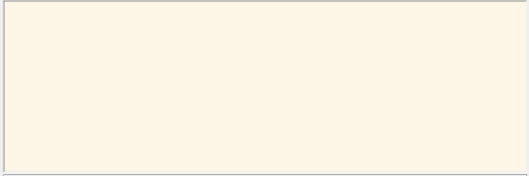

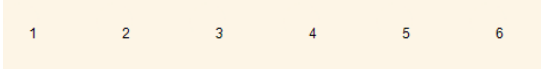
### 2.2. Các thành phần trong chương trình

The screenshot shows the main interface of the program. It consists of three main sections at the bottom, each with a title and several controls:

- NHẬP DỮ LIỆU (Input Data):** Contains two input fields. The first field has a label "(Tối đa 15 phần tử)" (Maximum 15 elements). Below the first field is the label "XÓA DỮ LIỆU" (Delete Data). Each input field has an "Input" button and a "Delete" button.
- CHỌN THUẬT TOÁN CẦN THỰC HIỆN (Select Algorithm to Perform):** Contains four radio buttons for selecting a sorting algorithm: "Selection sort", "Insertion sort", "Bubble sort", and "Merge sort".
- BẢNG ĐIỀU KHIỂN (Control Panel):** Contains two checkboxes: "Sắp xếp tăng dần" (Sort ascending) and "Sắp xếp giảm dần" (Sort descending). There is a "Go" button to the right of these checkboxes.

Hình 2.1. Giao diện chính của chương trình

| NÚT  | CHỨC NĂNG   |
|--|---|
| <div> <p><b>NHẬP DỮ LIỆU</b></p> <p><input type="text"/></p> <p>(Tối đa 15 phần tử)</p> <p><b>XÓA DỮ LIỆU</b></p> <p><input type="text"/></p> </div>   | <p>Nhập dữ liệu:</p> <ul style="list-style-type: none"> <li>• Input: Nhập dữ liệu</li> <li>• Paint: Vẽ dữ liệu lên màn hình chính</li> <li>• Delete: Xoá dữ liệu</li> </ul> |
| <p><b>CHỌN THUẬT TOÁN CẦN THỰC HIỆN</b></p> <p><input type="checkbox"/> Selection sort</p> <p><input type="checkbox"/> Insertion sort</p> <p><input type="checkbox"/> Bubble sort</p> <p><input type="checkbox"/> Merge sort</p> | <p>Chọn thuật toán:</p> <ul style="list-style-type: none"> <li>• Sắp xếp chọn</li> <li>• Sắp xếp chèn</li> <li>• Sắp xếp nổi bọt</li> <li>• Sắp xếp trộn</li> </ul>         |

|   |   |
|---|---|
| <div> <div><b>BẢNG ĐIỀU KHIỂN</b></div> <div> <input type="checkbox"/> Sắp xếp tăng dần         <input type="checkbox"/> Sắp xếp giảm dần         <div>Go</div> </div> </div> | <p>Chọn cách sắp xếp:</p> <ul style="list-style-type: none"> <li>Sắp xếp tăng dần</li> <li>Sắp xếp giảm dần</li> </ul> <p>Go: Tiến hành chạy chương trình</p> |
|    | <p>Màn hình hiển thị dữ liệu</p>  |
|    | <p>Các phần tử trong dãy dữ liệu</p>  |
|    | <p>Số thứ tự các phần tử</p>  |

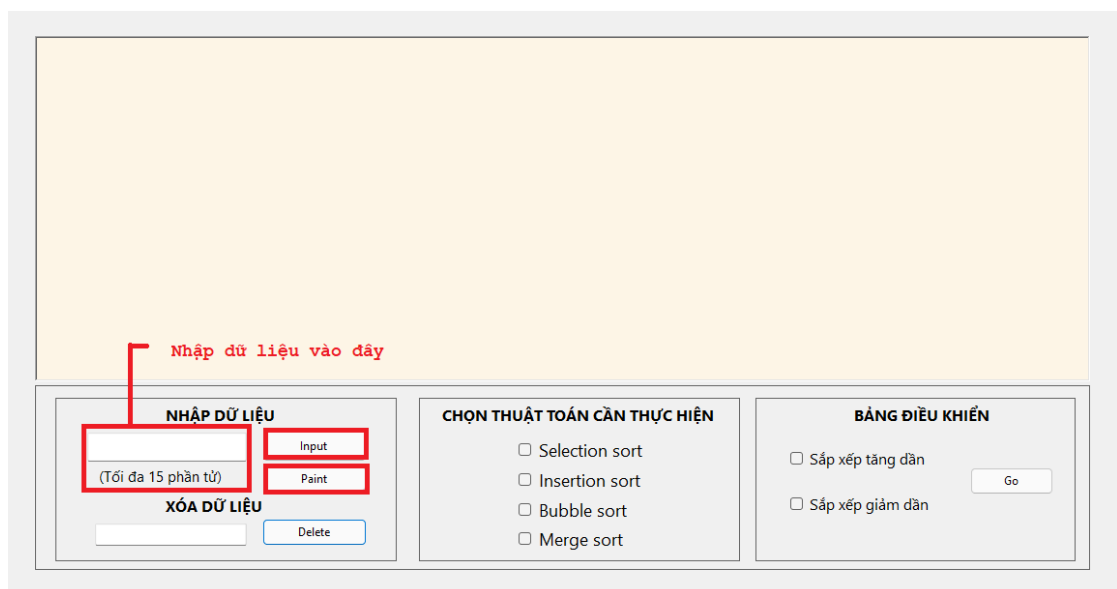
Bảng 2.1. Các thành phần trong chương trình

### 2.3. Mô tả chương trình

Để chương trình hoạt động, bước đầu tiên, chúng ta sẽ nhập dữ liệu vào ô dữ liệu.

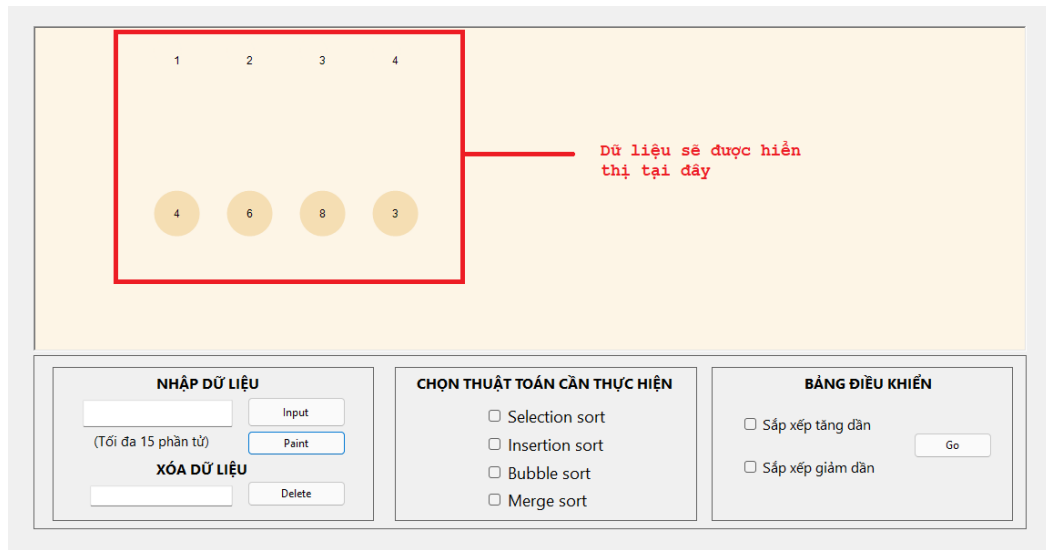
Ví dụ: Dãy cần nhập là (4, 6, 8, 3) thì chúng ta sẽ nhập từ số của dãy.

Nhập 4 từ bàn phím, sau đó chọn **Input**. Tiếp tục nhập các số còn lại cho đến khi hết dãy.



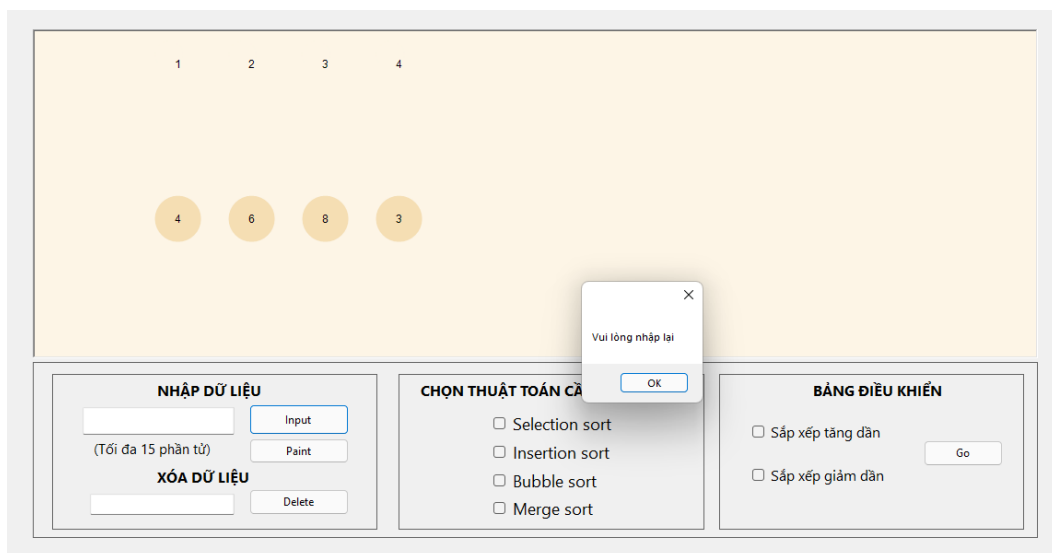
Hình 2.2. Nhập dữ liệu

Sau khi nhập dữ liệu xong, chọn **Paint** để dãy số được hiển thị lên màn hình chính.



Hình 2.3. Hiển thị dữ liệu

Lưu ý: Dữ liệu ban đầu phải nhập từ bàn phím và nhập từng số một, dữ liệu chỉ là số, không bao gồm những chữ cái và kí hiệu khác. Nếu như nhập sai, hệ thống sẽ thông báo “**Vui lòng nhập lại**”.



Hình 2.4. Lỗi nhập dữ liệu

Bước tiếp theo là chọn thuật toán sắp xếp. Chọn thuật toán bằng cách tick vào ô trống trước tên thuật toán.

The interface consists of three main sections at the bottom:

- NHẬP DỮ LIỆU (Input Data):** Includes an 'Input' field with a 'Paint' button and an 'XÓA DỮ LIỆU (Delete Data)' section with a 'Delete' button.
- CHỌN THUẬT TOÁN CẦN THỰC HIỆN (Select algorithm to perform):** Contains four checkboxes: 'Selection sort', 'Insertion sort', 'Bubble sort', and 'Merge sort'. The 'Selection sort' checkbox is highlighted with a red box.
- BẢNG ĐIỀU KHIỂN (Control panel):** Contains two checkboxes: 'Sắp xếp tăng dần (Sort increasing)' and 'Sắp xếp giảm dần (Sort decreasing)', and a 'Go' button.

A red annotation 'tick vào để chọn thuật toán' points to the 'Selection sort' checkbox.

Hình 2.5. Chọn thuật toán sắp xếp

Lưu ý: Chỉ chọn được **một thuật toán duy nhất** cho một lần thực hiện.

Sau khi chọn được thuật toán, bước tiếp theo sẽ chọn trình tự sắp xếp bằng cách tick vào ô trống trước cách sắp xếp. Cuối cùng là chọn vào **Go** để dãy số được sắp xếp.

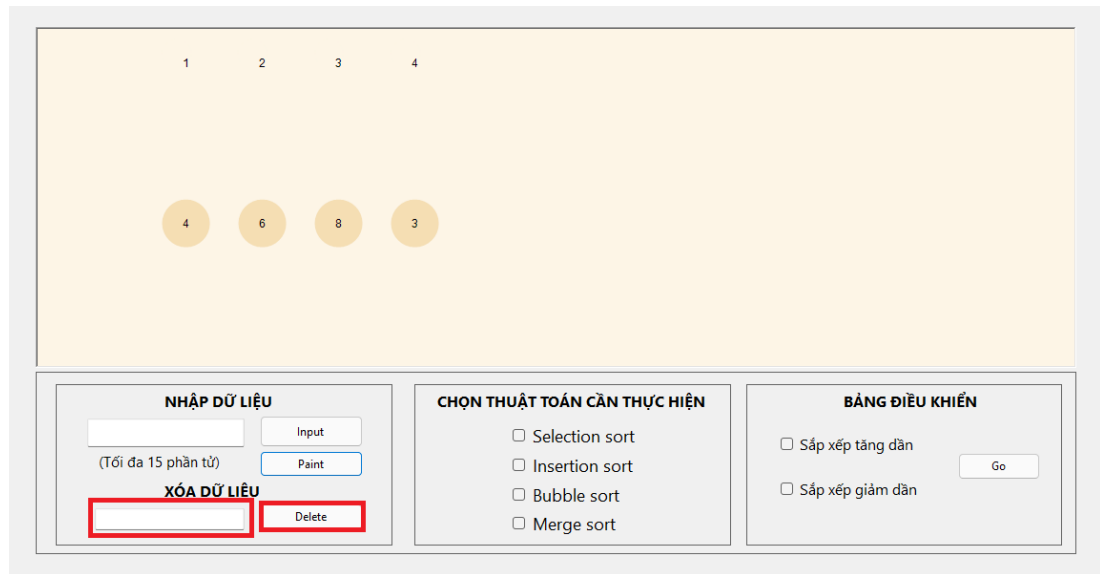
The interface is the same as in Figure 2.5, but with the following changes:

- The 'Selection sort' checkbox in the 'CHỌN THUẬT TOÁN' panel is no longer highlighted.
- The 'Sắp xếp tăng dần' checkbox in the 'BẢNG ĐIỀU KHIỂN' panel is highlighted with a red box.
- A red annotation 'tick vào để chọn cách sắp xếp' points to the 'Sắp xếp tăng dần' checkbox.

Hình 2.6. Chọn cách sắp xếp

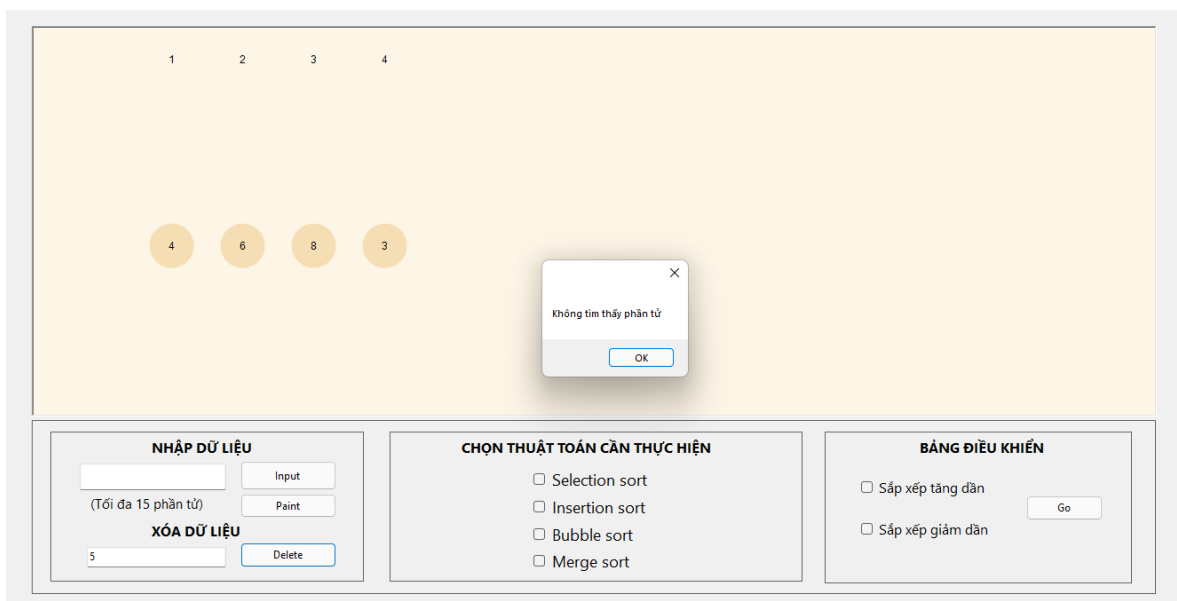
Lưu ý: Chỉ chọn được **một cách sắp xếp duy nhất** cho một lần sắp xếp.

Để xoá dữ liệu, chúng ta nhập số thứ tự của dữ liệu vào ô dữ liệu, sau đó chọn **Delete** để xoá dữ liệu.



Hình 2.7. Xóa dữ liệu

Ví dụ như dãy có 4 phần tử, mà chúng ta muốn xóa phần tử thứ 5 thì chương trình sẽ thông báo “**Không tìm thấy phần tử**”. Đây là lỗi xóa dữ liệu.



Hình 2.8. Lỗi xóa dữ liệu

## 2.4. Hàm thuật toán trong chương trình

- Sắp xếp tăng dần của thuật toán Selection Sort

```
public void ssmin()
{
    var g = panel1.CreateGraphics();
    Brush cbrush = new SolidBrush(Color.Wheat);
    Brush cbrushselect = new SolidBrush(Color.LightGreen);
    Brush cbrushend = new SolidBrush(Color.Aqua);
```

```

/* Brush cbrushnode = new SolidBrush(Color.Yellow);
Brush cbrushgray = new SolidBrush(Color.Gray);
Pen pennode = new Pen(Color.Black, 3.0F);
Pen penback = new Pen(Color.Gray,3.0F);*/
Size sizeCircle = new Size(50, 50);
adddata();
for (int i=0;i<ls.Count;i++)
{
    int u = run[i].ts;
    Point pt1 = new Point(run[i].x, run[i].y);
    Rectangle rec1 = new Rectangle(pt1, sizeCircle);
    DrawNode(g, cbrushselect, rec1, pt1, run[i].ts.ToString(), 50);
    // movenode(g, cbrushnode, run[i], pennode, 10,i);
    for (int j = i + 1; j < ls.Count; j++)
    {
        // movenode(g, cbrushnode, run[j], pennode, panel1.Height -
60,j);

        Point pt2 = new Point(run[j].x, run[j].y);
        Rectangle rec2 = new Rectangle(pt2, sizeCircle);
        DrawNode(g, cbrushselect, rec2, pt2, run[j].ts.ToString(), 50);
        Thread.Sleep(100);
        int v = run[j].ts;
        if (u > v)
        {
            move(run[i], run[j]);
            swappointer(i, j);
            pt1 = new Point(run[i].x, run[i].y);
            rec1 = new Rectangle(pt1, sizeCircle);
            DrawNode(g, cbrushselect, rec1, pt1, run[i].ts.ToString(),
50);

            pt2 = new Point(run[j].x, run[j].y);
            rec2 = new Rectangle(pt2, sizeCircle);
            DrawNode(g, cbrush, rec2, pt2, run[j].ts.ToString(), 50);
            Thread.Sleep(100);
            u = run[i].ts;
        }
        else
        {
            pt2 = new Point(run[j].x, run[j].y);
            rec2 = new Rectangle(pt2, sizeCircle);
            DrawNode(g, cbrush, rec2, pt2, run[j].ts.ToString(), 50);
            Thread.Sleep(100);
        }
        // movenode(g, cbrushgray, run[i], penback, (panel1.Height -
60), j);
    }
    if (i==ls.Count-1)
    {
        var pt2 = new Point(run[ls.Count - 1].x, run[ls.Count - 1].y);
        Rectangle rec2 = new Rectangle(pt2, sizeCircle);
        DrawNode(g, cbrushend, rec2, pt2, run[ls.Count -
1].ts.ToString(), 50);
    }
    else
    {
        pt1 = new Point(run[i].x, run[i].y);
        rec1 = new Rectangle(pt1, sizeCircle);
        DrawNode(g, cbrushend, rec1, pt1, run[i].ts.ToString(), 50);
        Thread.Sleep(100);
    }
    // movenode(g, cbrushgray, run[i], penback, 10, i);
}
}

```

– Sắp xếp giảm dần của thuật toán Selection Sort.

```

public void ssmax()
{
    var g = panel1.CreateGraphics();
    Brush cbrush = new SolidBrush(Color.Wheat);
    Brush cbrushselect = new SolidBrush(Color.LightGreen);
    Brush cbrushend = new SolidBrush(Color.Aqua);
    Brush cbrushnode = new SolidBrush(Color.Yellow);
    Brush cbrushgray = new SolidBrush(Color.OldLace);
    Pen pennode = new Pen(Color.Black, 3.0F);
    Pen penback = new Pen(Color.Gray, 3.0F);
    Size sizeCircle = new Size(50, 50);
    adddata();
    for (int i = 0; i < ls.Count; i++)
    {
        int u = run[i].ts;
        Point pt1 = new Point(run[i].x, run[i].y);
        Rectangle rec1 = new Rectangle(pt1, sizeCircle);
        DrawNode(g, cbrushselect, rec1, pt1, run[i].ts.ToString(), 50);
        // movenode(g, cbrushnode, run[i], pennode, 10,i);
        for (int j = i + 1; j < ls.Count; j++)
        {
            // movenode(g, cbrushnode, run[j], pennode, panel1.Height -
60,j);

            Point pt2 = new Point(run[j].x, run[j].y);
            Rectangle rec2 = new Rectangle(pt2, sizeCircle);
            DrawNode(g, cbrushselect, rec2, pt2, run[j].ts.ToString(), 50);
            Thread.Sleep(100);
            int v = run[j].ts;
            if (u < v)
            {
                move(run[i], run[j]);
                swappointer(i, j);
                pt1 = new Point(run[i].x, run[i].y);
                rec1 = new Rectangle(pt1, sizeCircle);
                DrawNode(g, cbrushselect, rec1, pt1, run[i].ts.ToString(),
50);

                pt2 = new Point(run[j].x, run[j].y);
                rec2 = new Rectangle(pt2, sizeCircle);
                DrawNode(g, cbrush, rec2, pt2, run[j].ts.ToString(), 50);
                Thread.Sleep(100);
                u = run[i].ts;
            }
            else
            {
                pt2 = new Point(run[j].x, run[j].y);
                rec2 = new Rectangle(pt2, sizeCircle);
                DrawNode(g, cbrush, rec2, pt2, run[j].ts.ToString(), 50);
                Thread.Sleep(100);
            }
            // movenode(g, cbrushgray, run[i], penback, (panel1.Height -
60), j);
        }
        if (i == ls.Count - 1)
        {
            var pt2 = new Point(run[ls.Count - 1].x, run[ls.Count - 1].y);
            Rectangle rec2 = new Rectangle(pt2, sizeCircle);
            DrawNode(g, cbrushend, rec2, pt2, run[ls.Count -
1].ts.ToString(), 50);
        }
        else
        {
            pt1 = new Point(run[i].x, run[i].y);
            rec1 = new Rectangle(pt1, sizeCircle);

```

```

        DrawNode(g, cbrushend, rec1, pt1, run[i].ts.ToString(), 50);
        Thread.Sleep(100);
    }
    // movenode(g, cbrushgray, run[i], penback, 10, i);
}
}

```

- Sắp xếp tăng dần của thuật toán Insertion Sort.

```

public void ismin()
{
    var g = panel1.CreateGraphics();
    Brush cbrush = new SolidBrush(Color.Wheat);
    Brush cbrushselect = new SolidBrush(Color.LightGreen);
    Brush cbrushend = new SolidBrush(Color.Aqua);
    Brush cbrushnode = new SolidBrush(Color.Yellow);
    Brush cbrushgray = new SolidBrush(Color.OldLace);
    Pen pennode = new Pen(Color.Black, 3.0F);
    Pen penback = new Pen(Color.Gray, 3.0F);
    Size sizeCircle = new Size(50, 50);
    int x, y;
    adddata();
    for (int i = 1; i < ls.Count; i++)
    {
        int u = run[i].ts; // 0
        x = run[i].x;
        y = run[i].y - 80;
        // MessageBox.Show(run[i].y.ToString());
        Point pt1 = new Point(pointers[i].x, pointers[i].y);
        Rectangle rec1 = new Rectangle(pt1, sizeCircle);
        DrawNode(g, cbrushselect, rec1, pt1, run[i].ts.ToString(), 50);
        Thread.Sleep(100);
        Nodesimpleup(run[i]);
        // MessageBox.Show(run[i].y.ToString());
        run[i].y += 80;
        int j = i - 1;
        Point pt2 = new Point(run[j].x, run[j].y);
        Rectangle rec2 = new Rectangle(pt2, sizeCircle);
        //DrawNode(g, cbrushselect, rec2, pt2, run[j].ts.ToString(), 50);
        Thread.Sleep(300);
        while (run[j].ts > u)
        {
            pt2 = new Point(run[j].x, run[j].y);
            rec2 = new Rectangle(pt2, sizeCircle);
            DrawNode(g, cbrushselect, rec2, pt2, run[j].ts.ToString(), 50);
            Thread.Sleep(100);
            j--;
            if (j == -1)
                break;
            Thread.Sleep(100);
        }
        if (j == i - 1)
        {
            run[i].y -= 80;
            Nodesimpledown(run[i]);
            continue;
        }
        j++;
        for (int k = i - 1; k >= j; k--)
        {
            int x1 = run[k].x;
            int y1 = run[k].y;
            int tmp = run[k].ts;
            Nodesimpleright(x1, y1, tmp, run[k + 1].x);
        }
    }
}

```



```

        swapnode(k, k + 1);
    }
    Nodesimpleleftdown(run[j], x, y);
    for (int k = j; k <= i; k++)
    {
        pt1 = new Point(run[k].x, run[k].y);
        rec1 = new Rectangle(pt1, sizeCircle);
        DrawNode(g, cbrush, rec1, pt1, run[k].ts.ToString(), 50);
        Thread.Sleep(100);
    }
}

```

– Sắp xếp giảm dần của thuật toán Insertion Sort.

```

public void ismax()
{
    var g = panel1.CreateGraphics();
    Brush cbrush = new SolidBrush(Color.Wheat);
    Brush cbrushselect = new SolidBrush(Color.LightGreen);
    Brush cbrushend = new SolidBrush(Color.Aqua);
    Brush cbrushnode = new SolidBrush(Color.Yellow);
    Brush cbrushgray = new SolidBrush(Color.OldLace);
    Pen pennode = new Pen(Color.Black, 3.0F);
    Pen penback = new Pen(Color.Gray, 3.0F);
    Size sizeCircle = new Size(50, 50);
    int x, y;
    adddata();
    for (int i = 1; i < ls.Count; i++)
    {
        int u = run[i].ts; // 0
        x = run[i].x;
        y = run[i].y - 80;
        // MessageBox.Show(run[i].y.ToString());
        Point pt1 = new Point(pointers[i].x, pointers[i].y);
        Rectangle rec1 = new Rectangle(pt1, sizeCircle);
        DrawNode(g, cbrushselect, rec1, pt1, run[i].ts.ToString(), 50);
        Thread.Sleep(100);
        Nodesimpleup(run[i]);
        // MessageBox.Show(run[i].y.ToString());
        run[i].y += 80;
        int j = i - 1;
        Point pt2 = new Point(run[j].x, run[j].y);
        Rectangle rec2 = new Rectangle(pt2, sizeCircle);
        //DrawNode(g, cbrushselect, rec2, pt2, run[j].ts.ToString(), 50);
        Thread.Sleep(300);
        while (run[j].ts < u)
        {
            pt2 = new Point(run[j].x, run[j].y);
            rec2 = new Rectangle(pt2, sizeCircle);
            DrawNode(g, cbrushselect, rec2, pt2, run[j].ts.ToString(), 50);
            Thread.Sleep(100);
            j--;
            if (j == -1)
                break;
            Thread.Sleep(100);
        }
        if (j == i - 1)
        {
            run[i].y -= 80;
            Nodesimpledown(run[i]);
            continue;
        }
        j++;
        for (int k = i - 1; k >= j; k--)
    }
}

```

```

    {
        int x1 = run[k].x;
        int y1 = run[k].y;
        int tmp = run[k].ts;
        Nodesimpleright(x1, y1, tmp, run[k + 1].x);
        swapnode(k, k + 1);
    }
    Nodesimpleleftdown(run[j], x, y);
    for (int k = j; k <= i; k++)
    {
        pt1 = new Point(run[k].x, run[k].y);
        rec1 = new Rectangle(pt1, sizeCircle);
        DrawNode(g, cbrush, rec1, pt1, run[k].ts.ToString(), 50);
        Thread.Sleep(100);
    }
}
}

```

- Sắp xếp giảm dần của thuật toán Bubble Sort.

```

public void bbmax()
{
    var g = panel1.CreateGraphics();
    Brush cbrush = new SolidBrush(Color.Wheat);
    Brush cbrushselect = new SolidBrush(Color.LightGreen);
    Brush cbrushend = new SolidBrush(Color.Aqua);
    Brush cbrushnode = new SolidBrush(Color.Yellow);
    Brush cbrushgray = new SolidBrush(Color.OldLace);
    Pen pennode = new Pen(Color.Black, 3.0F);
    Pen penback = new Pen(Color.Gray, 3.0F);
    Size sizeCircle = new Size(50, 50);
    adddata();
    for (int i=0;i<ls.Count;i++)
    {
        for (int j=ls.Count-1;j>i;j--)
        {
            Point pt1 = new Point(run[j].x, run[j].y);
            Point pt2 = new Point(run[j - 1].x, run[j - 1].y);
            Rectangle rec1 = new Rectangle(pt1, sizeCircle);
            Rectangle rec2 = new Rectangle(pt2, sizeCircle);
            DrawNode(g, cbrushselect, rec1, pt1, run[j].ts.ToString(), 50);
            DrawNode(g, cbrushselect, rec2, pt2, run[j - 1].ts.ToString(),
50);

            Thread.Sleep(100);
            if (run[j].ts>run[j-1].ts)
            {
                move(run[j-1], run[j]);
                swappointer(j - 1, j);
            }
            pt1 = new Point(run[j].x, run[j].y);
            pt2 = new Point(run[j - 1].x, run[j - 1].y);
            rec1 = new Rectangle(pt1, sizeCircle);
            rec2 = new Rectangle(pt2, sizeCircle);
            DrawNode(g, cbrush, rec1, pt1, run[j].ts.ToString(), 50);
            DrawNode(g, cbrush, rec2, pt2, run[j - 1].ts.ToString(), 50);
        }
        Point pt = new Point(run[i].x, run[i].y);
        Rectangle rec = new Rectangle(pt, sizeCircle);
        DrawNode(g, cbrushend, rec, pt, run[i].ts.ToString(), 50);
        Thread.Sleep(100);
    }
}
}

```

- Sắp xếp tăng dần của thuật toán Bubble Sort.

```

public void bbmin()
{
    var g = panel1.CreateGraphics();
    Brush cbrush = new SolidBrush(Color.Wheat);
    Brush cbrushselect = new SolidBrush(Color.LightGreen);
    Brush cbrushhend = new SolidBrush(Color.Aqua);
    Brush cbrushnode = new SolidBrush(Color.Yellow);
    Brush cbrushgray = new SolidBrush(Color.OldLace);
    Pen pennode = new Pen(Color.Black, 3.0F);
    Pen penback = new Pen(Color.Gray, 3.0F);
    Size sizeCircle = new Size(50, 50);
    adddata();
    for (int i = 0; i < ls.Count; i++)
    {
        for (int j = ls.Count - 1; j > i; j--)
        {
            Point pt1 = new Point(run[j].x, run[j].y);
            Point pt2 = new Point(run[j - 1].x, run[j - 1].y);
            Rectangle rec1 = new Rectangle(pt1, sizeCircle);
            Rectangle rec2 = new Rectangle(pt2, sizeCircle);
            DrawNode(g, cbrushselect, rec1, pt1, run[j].ts.ToString(), 50);
            DrawNode(g, cbrushselect, rec2, pt2, run[j - 1].ts.ToString(),
50);

            Thread.Sleep(100);
            if (run[j].ts < run[j - 1].ts)
            {
                move(run[j - 1], run[j]);
                swappointer(j - 1, j);
            }
            pt1 = new Point(run[j].x, run[j].y);
            pt2 = new Point(run[j - 1].x, run[j - 1].y);
            rec1 = new Rectangle(pt1, sizeCircle);
            rec2 = new Rectangle(pt2, sizeCircle);
            DrawNode(g, cbrush, rec1, pt1, run[j].ts.ToString(), 50);
            DrawNode(g, cbrush, rec2, pt2, run[j - 1].ts.ToString(), 50);
        }
        Point pt = new Point(run[i].x, run[i].y);
        Rectangle rec = new Rectangle(pt, sizeCircle);
        DrawNode(g, cbrushhend, rec, pt, run[i].ts.ToString(), 50);
        Thread.Sleep(100);
    }
}

```

– Sắp xếp giảm dần của thuật toán Merge Sort.

```

public void mgmax(int l, int r, int k)
{
    var g = panel1.CreateGraphics();
    Brush cbrush = new SolidBrush(Color.Wheat);
    Brush cbrushselect = new SolidBrush(Color.LightGreen);
    Brush cbrushhend = new SolidBrush(Color.Aqua);
    Brush cbrushnode = new SolidBrush(Color.Yellow);
    Brush cbrushback = new SolidBrush(Color.OldLace);
    Pen pennode = new Pen(Color.Black, 3.0F);
    Pen penback = new Pen(Color.Gray, 3.0F);
    Size sizeCircle = new Size(50, 50);
    Size sizeCircle1 = new Size(70, 70);
    int cnt;
    for (int i = l; i <= r; i++)
    {
        Thread.Sleep(100);
        Point pt = new Point(run[i].x, k);
        pt.X -= 5;
        pt.Y -= 5;
        Rectangle rec = new Rectangle(pt, sizeCircle1);
    }
}

```

```

        deletedraw(g, cbrushback, rec, pt);
        nodeup(k, run[i]);
    }
    if (r-l<=1)
    {
        if (r-l==1)
        {
            if (run[l].ts < run[r].ts)
                swapnode(l, r);
        }
        /* for (int i = l; i <= r; i++)
        {
            pointer tmp = run[i];
            for (int j = i + 1; j <= r; j++)
            {
                if (run[i].ts < run[j].ts)
                    swapnode(i, j);
            }
        } */
        cnt = k - 60;
        while (cnt <= k)
        {
            for (int i = l; i <= r; i++)
            {
                Point pt = new Point(run[i].x, cnt);
                Rectangle rec = new Rectangle(pt, sizeCircle);
                DrawNode(g, cbrushend, rec, pt, run[i].ts.ToString(), 50);
            }
            Thread.Sleep(100);
            if (cnt == k)
                break;
            for (int i = l; i <= r; i++)
            {
                Point pt = new Point(run[i].x, cnt);
                pt.X -= 5;
                pt.Y -= 5;
                Rectangle rec = new Rectangle(pt, sizeCircle1);
                deletedraw(g, cbrushback, rec, pt);
            }
            cnt += 20;
        }
        return;
    }
    int mid = (l + r) / 2;
    mgmax(l, mid, k - 60);
    mgmax(mid+1, r, k - 60);
    cnt = k - 60;
    int left = l;
    int right = mid+1;
    List<int> check = new List<int>();
    for (int i = l; i <= r; i++)
    {
        check.Add(run[i].ts);
    }
    List<int> kq = new List<int>();
    while (true)
    {
        if (left==mid+1)
        {
            for (int i = right; i <= r; i++)
                kq.Add(run[i].ts);
            break;
        }
        if (right==r+1)
        {

```

```

        for (int i = left; i <= mid; i++)
            kq.Add(run[i].ts);
        break;
    }
    if (run[left].ts < run[right].ts)
    {
        kq.Add(run[right].ts);
        right++;
    }
    else
    {
        kq.Add(run[left].ts);
        left++;
    }
}
int dem = 0;
for (int i = l; i <= r; i++)
{
    run[i].ts = kq[dem];
    dem++;
}
while (cnt <= k)
{
    for (int i = l; i <= r; i++)
    {
        Point pt = new Point(run[i].x, cnt);
        Rectangle rec = new Rectangle(pt, sizeCircle);
        DrawNode(g, cbrushend, rec, pt, run[i].ts.ToString(), 50);
    }
    Thread.Sleep(100);
    if (cnt == k)
        break;
    for (int i = l; i <= r; i++)
    {
        Point pt = new Point(run[i].x, cnt);
        pt.X -= 5;
        pt.Y -= 5;
        Rectangle rec = new Rectangle(pt, sizeCircle1);
        deletedraw(g, cbrushback, rec, pt);
    }
    cnt += 20;
}
}

```

- Sắp xếp tăng dần của thuật toán Merge Sort.

```

public void mgmin(int l, int r, int k)
{
    var g = panel1.CreateGraphics();
    Brush cbrush = new SolidBrush(Color.Wheat);
    Brush cbrushselect = new SolidBrush(Color.LightGreen);
    Brush cbrushend = new SolidBrush(Color.Aqua);
    Brush cbrushnode = new SolidBrush(Color.Yellow);
    Brush cbrushback = new SolidBrush(Color.OldLace);
    Pen pennode = new Pen(Color.Black, 3.0F);
    Pen penback = new Pen(Color.Gray, 3.0F);
    Size sizeCircle = new Size(50, 50);
    Size sizeCircle1 = new Size(70, 70);
    int cnt;
    for (int i = l; i <= r; i++)
    {
        Thread.Sleep(100);
        Point pt = new Point(run[i].x, k);
        pt.X -= 5;
        pt.Y -= 5;
    }
}

```

```

        Rectangle rec = new Rectangle(pt, sizeCircle1);
        deletedraw(g, cbrushback, rec, pt);
        nodeup(k, run[i]);
    }
    if (r - l <= 1)
    {
        /*for (int i = l; i <= r; i++)
        {
            pointer tmp = run[i];
            for (int j = i + 1; j <= r; j++)
            {
                if (run[i].ts > run[j].ts)
                    swapnode(i, j);
            }
        }*/
        if (r - l == 1)
        {
            if (run[l].ts > run[r].ts)
                swapnode(l, r);
        }
        cnt = k - 60;
        while (cnt <= k)
        {
            for (int i = l; i <= r; i++)
            {
                Point pt = new Point(run[i].x, cnt);
                Rectangle rec = new Rectangle(pt, sizeCircle);
                DrawNode(g, cbrushend, rec, pt, run[i].ts.ToString(), 50);
            }
            Thread.Sleep(100);
            if (cnt == k)
                break;
            for (int i = l; i <= r; i++)
            {
                Point pt = new Point(run[i].x, cnt);
                pt.X -= 5;
                pt.Y -= 5;
                Rectangle rec = new Rectangle(pt, sizeCircle1);
                deletedraw(g, cbrushback, rec, pt);
            }
            cnt += 20;
        }
        return;
    }
    int mid = (l + r) / 2;
    mgmin(l, mid, k - 60);
    mgmin(mid + 1, r, k - 60);
    cnt = k - 60;
    int left = l;
    int right = mid + 1;
    List<int> check = new List<int>();
    for (int i = l; i <= r; i++)
    {
        check.Add(run[i].ts);
    }
    List<int> kq = new List<int>();
    while (true)
    {
        if (left == mid + 1)
        {
            for (int i = right; i <= r; i++)
                kq.Add(run[i].ts);
            break;
        }
        if (right == r + 1)

```

```

    {
        for (int i = left; i <= mid; i++)
            kq.Add(run[i].ts);
        break;
    }
    if (run[left].ts > run[right].ts)
    {
        kq.Add(run[right].ts);
        right++;
    }
    else
    {
        kq.Add(run[left].ts);
        left++;
    }
}
int dem = 0;
for (int i = l; i <= r; i++)
{
    run[i].ts = kq[dem];
    dem++;
}
/*for (int i = l; i <= r; i++)
{
    pointer tmp = run[i];
    for (int j = i + 1; j <= r; j++)
    {
        if (run[i].ts > run[j].ts)
            swapnode(i, j);
    }
}*/
while (cnt <= k)
{
    for (int i = l; i <= r; i++)
    {
        Point pt = new Point(run[i].x, cnt);
        Rectangle rec = new Rectangle(pt, sizeCircle);
        DrawNode(g, cbrushend, rec, pt, run[i].ts.ToString(), 50);
    }
    Thread.Sleep(100);
    if (cnt == k)
        break;
    for (int i = l; i <= r; i++)
    {
        Point pt = new Point(run[i].x, cnt);
        pt.X -= 5;
        pt.Y -= 5;
        Rectangle rec = new Rectangle(pt, sizeCircle1);
        deletedraw(g, cbrushback, rec, pt);
    }
    cnt += 20;
}
for (int i = l; i <= r; i++)
{
    pointer tmp = run[i];
    for (int j = i + 1; j <= r; j++)
    {
        if (tmp.ts > run[j].ts)
        {
            swapnode(i, j);
        }
    }
}
for (int i = l; i <= r; i++)
{
    Point pt = new Point(run[i].x, run[i].y + k);

```

```
        Rectangle rec = new Rectangle(pt, sizeCircle);  
    }  
}
```



## CHƯƠNG 3. KẾT LUẬN

### 4.1. Kết quả đạt được

- Hoàn thành chương trình mô phỏng thuật toán sắp xếp như mục tiêu ban đầu có thể hiện node giữa màn hình, mô phỏng thành công các thuật toán sắp xếp cơ bản.
- Bước đầu nắm bắt được những kỹ thuật lập trình của C#, Winform như: timer, backgroundworker, ...
- Kỹ năng làm việc nhóm của các thành viên được cải thiện.

### 4.2. Hạn chế

- Giao diện chương trình khá đơn giản.
- Số lượng phần tử còn hạn chế (chỉ 15 phần tử).
- Số lượng thuật toán ít (chỉ 4 thuật toán).
- Một số thao tác chưa được tối ưu.

### 4.3. Hướng phát triển

- Cải thiện thêm phần giao diện thân thiện với người dùng hơn.
- Tăng thêm số lượng phần tử.
- Thêm nhiều cách nhập dữ liệu hơn: nhập từ file, nhập nguyên dãy, ...
- Thêm nhiều chức năng như so sánh độ phức tạp từ hai thuật toán trở lên.
- Tối ưu hoá tốc độ xử lý, hiệu suất chương trình.
- Thêm nhiều thuật toán cho người dùng lựa chọn.
- Thêm nhiều chức năng, nhiều lựa chọn hơn cho người dùng.

### 4.4. Phân công công việc

| STT | Công việc   | Phân công | Tiến độ |
|-----|---|-----------|---------|
| 1   | Lên ý tưởng, viết code chính, nghiên cứu thuật toán, chỉnh sửa lỗi  | Dương     | 100%    |
| 2   | Viết báo cáo, góp ý thiết kế, nghiên cứu thuật toán                 | Nguyên    | 100%    |
| 3   | Làm video demo chương trình, lên ý tưởng hoàn thiện chương trình    | Quỳnh     | 100%    |
| 4   | Lên ý tưởng, đóng góp ý kiến hoàn thiện chương trình, báo cáo đồ án | Ngọc      | 100%    |

*Bảng 3.1. Bảng phân công công việc*

# TÀI LIỆU THAM KHẢO

1. Các thuật toán sắp xếp cơ bản  
<https://viblo.asia/p/cac-thuat-toan-sap-xep-co-ban-Eb85ooNO52G>
2. Các thuật toán sắp xếp trong C++  
<https://codelearn.io/sharing/cac-thuat-toan-sap-xep-trong-cpp>
3. Thuật toán sắp xếp  
<https://vnoi.info/wiki/algo/basic/sorting.md>