**Mục lục**

[Lời nói đầu 4](#_Toc27152177)

[Chương I: Giới thiệu và phát triển đề tài 5](#_Toc27152178)

[1. Nêu vấn đề 5](#_Toc27152179)

[2. Phạm vi nghiên cứu của đề tài 5](#_Toc27152180)

[3. Mục tiêu 5](#_Toc27152181)

[4. Phương pháp nghiên cứu 5](#_Toc27152182)

C[hương II: Trình bày nội dung về đề tài báo cáo 6](#_Toc27152183)

[1. Mô tả project 6](#_Toc27152184)

[2. Các lý thuyết về danh sách liên kết và thuật toán Merge sort 6](#_Toc27152185)

[2.1. Danh sách liên kết 6](#_Toc27152186)

[2.2. Thuật toán Merge sort 8](#_Toc27152187)

[2.2.1. Ý tưởng thuật toán Merge sort 8](#_Toc27152188)

[2.2.2. Cách hàm merge hoạt động khi gộp hai mảng con 9](#_Toc27152189)

[3. Mô tả quá trình làm 10](#_Toc27152190)

[3.1. Thuật toán sắp xếp Mergesort 10](#_Toc27152191)

[3.2. Các bước thực hiện 11](#_Toc27152192)

[3.3. Code mẫu cho thuật toán Merge sort 11](#_Toc27152193)

[4. Cài đặt và kiểm thử 13](#_Toc27152194)

[Chương III: Phân công công việc 17](#_Toc27152195)

[Chương IV: Đánh giá kết quả nghiên cứu và kết luận 18](#_Toc27152196)

[1. Đánh giá mức độ hoàn thành 18](#_Toc27152197)

[2. Các khó khăn gặp phải 18](#_Toc27152198)

[3. Hướng phát triển 19](#_Toc27152199)

[Tài liệu tham khảo 20](#_Toc27152200)

**Danh mục các hình**

[Hình 1. Hình mô tả cho một Node trong danh sách liên kết đơn 7](#_Toc25377278)

[Hình 2. Hình mô phỏng một danh sách liên kết đơn đầy đủ 7](#_Toc25377279)

[Hình 3. Hình minh họa cho thuật toán Mergesort 9](#_Toc25377280)

[Hình 4. Hàm thuật toán Mergesort 12](#_Toc25377284)

[Hình 5. Hàm tách danh sách L thành hai danh sách L1, L2 12](#_Toc25377285)

[Hình 6. Hàm gộp hai danh sách L1, L2 13](#_Toc25377286)

**Danh mục các bảng**

[Bảng 1. Bảng so sánh mảng và danh sách liên kết](#_Toc25377383). 7

[Bảng 2. Bảng so sánh khi Mergesort với mảng và với danh sách liên kết](#_Toc25377384) 11

[Bảng 3. Bảng ưu điểm và khuyết điểm của thuật toán Mergsort](#_Toc25377385) 18

# **Lời nói đầu**

Trong thời đại Công nghệ thông tin, Cấu trúc dữ liệu là nền tảng trong mọi tổ chức. Cấu trúc dữ liệu được biểu diễn dưới nhiều khía cạnh. Cấu trúc dữ liệu và giải thuật là một môn học cơ sở trong chương trình đào tạo trang bị cho mọi sinh viên những kiến thức cơ bản về cấu trúc và dữ liệu trong thiết kế và cài đặt phần mềm.

Sắp xếp (Sort) là một quá trình biến đổi danh sách các đối tượng thành một đối tượng thỏa mãn một thứ tự nào đó. Sắp xếp đóng vai trò rất quan trọng trong việc tìm kiếm dữ liệu. Chẳng hạn chúng ta thử hình dung xem một cuốn từ điển nếu các từ không được sắp xếp theo một thứ tự nhất định mà người ta vẫn thường làm sẽ khó khăn thế nào trong việc chúng ta tra cứu các từ. Hay là nếu như danh sách các sinh viên không được sắp xếp theo thứ tự bảng chữ cái sẽ rất khó khăn cho giáo viên nếu muốn tìm kiếm một học sinh nào đó. Trong lĩnh vực kinh tế việc sắp xếp ngày càng quan trọng hơn. Hiện nay đã có nhiều giải thuật sắp xếp được xây dựng, mức độ hiệu quả của từng giải thuật còn phụ thuộc vào tính chất của cấu trúc dữ liệu cụ thể mà nó tác động đến mà ta lựa chọ phương pháp sắp xếp sao cho phù hợp. Trong khoa học máy tính và trong toán học, một thuật toán sắp xếp là một thuật toán sắp xếp các phần tử của một danh sách (hoặc mảng) theo thứ tự tăng dần hoặc giảm dần. Hầu hết bài toán đều có nhiều cách khác nhau để giải quyết chúng.

Nhằm chuẩn bị cho đồ án cuối kì cũng như giải quyết các bài toán sắp xếp, em làm báo cáo về sắp xếp danh sách liên kết bằng thuật toán Merge sort. Mục tiêu của báo cáo này nhằm giúp các bạn cũng như em hiểu rõ hơn về danh sách liên kết và các thuật toán sắp xếp mà trọng tâm là thuật toán sắp xếp Merge sort.

Mặc dù đã cố gắng nhiều trong quá trình làm nhưng chắc chắn báo cáo vẫn còn nhiều thiếu sót và hạn chế. Rất mong nhận được sự góp ý từ thầy và các bạn để cho báo cáo của em được hoàn thiện hơn.

Hồ Chí Minh, ngày 14 tháng 12 năm 2019

Người thực hiện

Nguyễn Đức Tính

# **Chương I: Giới thiệu và phát triển đề tài**

1. **Nêu vấn đề:**

Quá trình sắp xếp là quá trình bố trí lại các phần tử của một tập đối tượng điển hình như một tập dãy số nào đó, một dãy chữ theo thứ tự của từ điển,… nhằm sắp xếp theo thứ tự nhất định (tăng dần hay giảm dần).

Bài toán sắp xếp thường được ứng dụng nhiều nhất trong các ứng dụng tin học với những yêu cầu, mục đích khác nhau thì sắp xếp dữ liệu trong máy tính theo các cách và các bước khác nhau. Nói chung dữ liệu có thể xuất hiện dưới nhiều dạng khác nhau và có thể lưu trữ một khối lượng đáng kể, nên việc xây dựng các giải thuật sắp xếp cho phép tìm kiếm nhanh sẽ có ý nghĩa rất lớn. Từ các vấn đề nêu trên giúp em hiểu rõ mục đích đề tài là: Sắp xếp các dãy số theo một trật tự tùy theo các yêu cầu của người muốn sắp xếp, sắp xếp theo thứ tự để giúp cho việc tìm kiếm dễ dàng hơn, từ đó có thể vận dụng phương pháp trong việc sắp xếp theo yêu cầu của bài toán đặt ra một cách hiệu quả nhất.

1. **Phạm vi nghiên cứu của đề tài:**

Tìm hiểu và vận dụng các lý thuyết cơ bản về danh sách liên kết và thuật toán Merge sort đề cài đặt chương trình, cho phép sắp xếp một dãy số đã cho tùy ý thành một dãy số có thứ tự hoặc sắp xếp một danh sách các sinh viên tăng dần theo họ tên.

1. **Mục tiêu:**

Mô tả quá trình thực hiện phương pháp sắp xếp.

Phân tích, đánh giá độ phức tạp của giải thuật.

Chạy thành công chương trình cài đặt.

1. **Phương pháp nghiên cứu:**

Nêu ý tưởng thuật toán.

Tìm hiểu giải thuật.

Mô phỏng thuật toán.

Chạy file cài đặt.

# **Chương II: Trình bày nội dung về đề tài báo cáo**

1. **Mô tả project:**

Project cần xây dựng là một danh sách liên kết được sắp xếp bằng thuật toán Merge sort. Trong bài báo cáo này là danh sách các sinh viên được sắp xếp tăng dần.

Phần mềm được dùng để sắp xếp một danh sách liên kết được nhập từ bàn phím theo một trình tự nhất định (tăng dần hay giảm dần tùy theo yêu cầu).

Cấu trúc dữ liệu và giải thuật được sử dụng trong đề tài liên quan đến các kĩ thuật về danh sách liên kết và các thuật toán sắp xếp.

1. **Các lý thuyết về danh sách liên kết và thuật toán Merge sort:**
   1. **Danh sách liên kết:**

Về bản chất, danh sách liên kết có chức năng như một mảng, có thể thêm và xóa các phần tử ở bất kỳ vị trí nào khi cần thiết. Một số sự khác nhau giữa danh sách liên kết và mảng:

|  |  |  |
| --- | --- | --- |
| **Nội dung** | **Mảng** | **Danh sách liên kết** |
| Kích thước | * Kích thước cố định * Cần chỉ rõ kích thước trong khi khai báo | * *Kích thước thay đổi trong quá trình thêm/ xóa phần tử* * *Kích thước tối đa phụ thuộc vào bộ nhớ* |
| Cấp phát bộ nhớ | * Tĩnh: Bộ nhớ được cấp phát trong quá trình biên dịch | * *Động: Bộ nhớ được cấp phát trong quá trình chạy* |
| Thứ tự & sắp xếp | * Được lưu trữ trên một dãy ô nhớ liên tục | * *Được lưu trữ trên các ô nhớ ngẫu nhiên* |
| Truy cập | * *Truy cập tới phần tử ngẫu nhiên trực tiếp bằng cách sử dụng chỉ số mảng: O(1)* | * Truy cập tới phần tử ngẫu nhiên cần phải duyệt từ đầu/cuối đến phần tử đó: O(n) |
| Tìm kiếm | 1. *Tìm kiếm tuyến tính hoặc tìm kiếm nhị phân* | * Chỉ có thể tìm kiếm tuyến tính |

Bảng 1. Bảng so sánh mảng và danh sách liên kết

**Lưu ý:** Ở bảng phía trên, các phần in nghiêng màu đỏ thể hiện đó là ưu điểm so với đối thủ còn lại.

Danh sách liên kết đơn là một tập hợp các Node được phân bố động, được sắp xếp theo cách sao cho mỗi Node chứa “*một giá trị”(Data)* và “*một con trỏ”(Next).*Con trỏ sẽ trỏ đến phần tử kế tiếp của danh sách liên kết đó. Nếu con trỏ mà trỏ tới NULL, nghĩa là đó là phần tử cuối cùng của linked list.



Hình 1. Hình mô tả cho một Node trong danh sách liên kết đơn (3)



Hình 2. Hình mô phỏng một danh sách liên kết đơn đầy đủ (3)

* 1. **Thuật toán Merge sort:**
     1. **Ý tưởng thuật toán Merge sort:**

Giống như Quick sort, Merge sort là một thuật toán chia để trị. Thuật toán này chia mảng cần sắp xếp thành 2 nửa. Tiếp tục lặp lại việc này ở các nửa mảng đã chia. Sau cùng gộp các nửa đó thành mảng đã sắp xếp. Hàm merge() được sử dụng để gộp hai nửa mảng. Hàm merge(arr, l, m, r) là tiến trình quan trọng nhất sẽ gộp hai nửa mảng thành 1 mảng sắp xếp, các nửa mảng là arr[l…m] và arr[m+1…r] sau khi gộp sẽ thành một mảng duy nhất đã sắp xếp.

Hãy xem ý tưởng triển khai code dưới đây để hiểu hơn:

mergeSort(arr[], l,  r)

If r > l

     1. Tìm chỉ số nằm giữa mảng để chia mảng thành 2 nửa:

             middle m = (l+r)/2

     2. Gọi đệ quy hàm mergeSort cho nửa đầu tiên:

             mergeSort(arr, l, m)

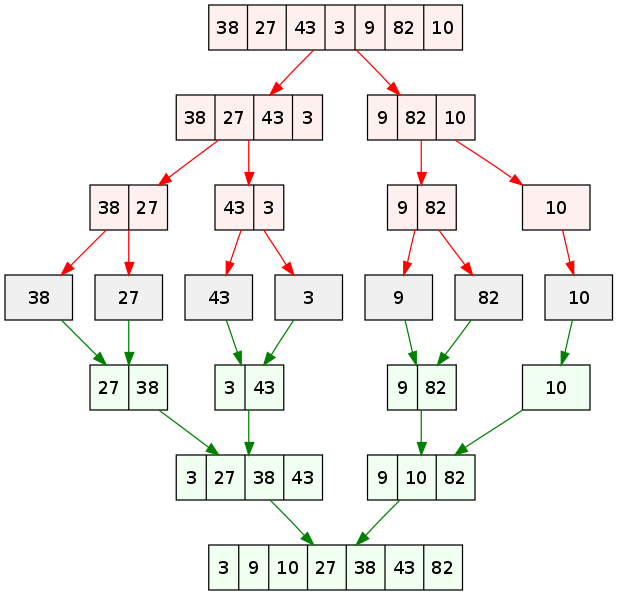
     3. Gọi đệ quy hàm mergeSort cho nửa thứ hai:

             mergeSort(arr, m+1, r)

     4. Gộp 2 nửa mảng đã sắp xếp ở (2) và (3):

             merge(arr, l, m, r)

Hình ảnh dưới đây từ wikipedia sẽ hiển thị toàn bộ sơ đồ tiến trình của thuật toán merge sort cho mảng {38, 27, 43, 3, 9, 82, 10}. Nếu nhìn kỹ hơn vào sơ đồ này, chúng ta có thể thấy mảng ban đầu được lặp lại hành động chia cho tới khi kích thước các mảng sau chia là 1. Khi kích thước các mảng con là 1, tiến trình gộp sẽ bắt đầu thực hiện gộp lại các mảng này cho tới khi hoàn thành và chỉ còn một mảng đã sắp xếp.



Hình 3. Hình minh họa cho thuật toán Mergesort (3)

* + 1. **Cách hàm merge hoạt động khi gộp hai mảng con:**

Với trường hợp khi 2 mảng con chỉ có 1 phần tử, ta chỉ việc xem phần tử nào nhỏ hơn và đẩy lên đầu, phần tử còn lại đặt phía sau. Do vậy, các mảng con cần gộp lại có tính chất luôn được sắp tăng dần.

Giả sử ta có 2 mảng con lần lượt là:

arr1 = [1 9 10 10] , n1 = 4 // chiều dài của mảng con

arr2 = [3 5 7 9], n2 = 4

sort\_arr = [] // Mảng lưu lại tiến trình gộp

Khởi tạo i = 0, j = 0 tương ứng là chỉ số bắt đầu của arr1 và arr2

Xét thấy arr1[i] < arr2[j] => chèn arr1[i] vào cuối mảng sort\_arr, tăng i lên 1 đơn vị

=> sort\_arr = [1], i = 1

Xét thấy arr1[i] > arr2[j] => chèn arr2[j] vào cuối mảng sort\_arr, tăng j lên 1 đơn vị

=> sort\_arr = [1, 3], i = 1, j = 1

Xét thấy arr1[i] > arr2[j] => chèn arr2[j] vào cuối mảng sort\_arr, tăng j lên 1 đơn vị

=> sort\_arr = [1, 3, 5], i = 1, j = 2

Xét thấy arr1[i] > arr2[j] => chèn arr2[j] vào cuối mảng sort\_arr, tăng j lên 1 đơn vị

=> sort\_arr = [1, 3, 5, 7], i = 1, j = 3

Xét thấy arr1[i] = arr2[j] => chèn arr1[i] hoặc arr2[j] vào cuối mảng sort\_arr

Giả sử chọn arr1, tăng i lên 1 đơn vị

=> sort\_arr = [1, 3, 5, 7, 9], i = 2, j = 3

Xét thấy arr1[i] > arr2[j] => chèn arr2[j] vào cuối mảng sort\_arr, tăng j lên 1 đơn vị

=> sort\_arr = [1, 3, 5, 7, 9, 9], i = 1, j = 4

Do j >= n2, tiếp tục tăng i chừng nào i < n1 thi thêm phần tử ở arr1[i] vào sort\_arr.

Sau cùng ta được mảng đã sắp xếp là sort\_arr = [1, 3, 5, 7, 9, 9, 10, 10]

1. **Mô tả quá trình làm:**
   1. **Thuật toán sắp xếp Mergesort:**

Như ở trên chúng ta đã tìm hiểu về thuật toán Merge sort với mảng. Bây giờ chúng ta chỉ cần phát triển lên để thuật toán Merge sort có thể làm việc được với danh sách liên kết.

Đầu tiên chúng ta cần phân biệt được sự khác nhau khi thuật toán Mergesort làm việc với mảng và khi làm việc với danh sách liên kết:

|  |  |
| --- | --- |
| Mergesort với mảng | Mergesort với danh sách liên kết |
| Phần tử mid dễ dàng tìm kiếm được bằng cách: mid = (left + right) / 2 hay  mid = left + (right – left) / 2 với trường hợp right, left lớn để tránh tràn số. | Phần tử mid tìm được khó khăn hơn. Phải duyệt từ đầu danh sách bằng hai con trỏ với con trỏ thứ nhất tăng hai lần và con trỏ thứ hai tăng một lần sau mỗi lần lặp. Và khi con trỏ thứ nhất trỏ tới cuối danh sách thì ta tìm được phần tử mid là phần tử mà con trỏ thứ hai trỏ tới. |
| Các giá trị được lưu trữ trên các ô nhớ liên tục nên việc gộp hai mảng con thành mảng sắp xếp hơi khó khăn. | Các giá trị được lưu trữ trên các ô nhớ ngẫu nhiên nên việc gộp hai danh sách con thành danh sách sắp xếp dễ dàng hơn. |

Bảng 2. Bảng so sánh khi Mergesort với mảng và với danh sách liên kết

* 1. **Các bước thực hiện:**

Bước 1: Ban đầu ta tạo ra hai danh sách rỗng là L1 và L2.

Bước 2: Tiếp theo ta tách danh sách L ban đầu thành hai danh sách con L1 và L2 vừa tạo bằng cách:

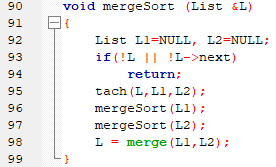
+ L1 là danh sách từ phần tử đầu đến phần tử mid của L.

+ L2 là danh sách từ phần tử mid của L đến phần tử cuối.

Bước 3: Sau đó ta thực hiện đệ quy bước 2 cho đến khi hai danh sách con chỉ gồm một phần tử duy nhất.

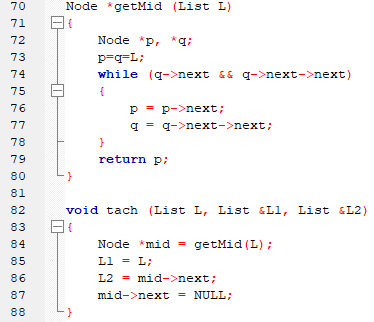
Bước 4: Cuối cùng ta thực hiện gộp hai danh sách L1 và L2 thành danh sách L theo thứ tự tăng dần hay giảm dần tùy theo yêu cầu đề bài.

* 1. **Code mẫu cho thuật toán Merge sort:**



Hình 4. Hàm thuật toán Mergesort

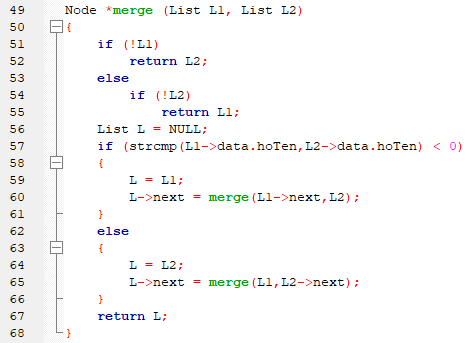
Đầu tiên chúng ta tạo ra hai danh sách rỗng L1, L2. Sau đó chúng ta tách danh sách L ban đầu thành hai danh sách vừa tạo. Sau đó chúng ta tiếp tục tách danh sách L1 thành hai danh sách nữa bằng cách đệ quy hàm mergeSort với L1 cho đến khi danh sách chỉ gồm 1 phần tử. Thực hiện tương tự với danh sách L2. Khi đã được các danh sách chỉ gồm 1 phần tử chúng ta bắt đầu thực hiện việc thêm các phần tử đó theo thứ tự tăng dần.



Hình 5. Hàm tách danh sách L thành hai danh sách L1, L2

Ở hàm tách chúng ta gán L2 = mid->next trước mid->next = NULL để có thể lấy được danh sách L1 là từ đầu danh sách cho đến phần tử mid và danh sách L2 từ phần từ mid về sau. Không đổi được hai vị trí này vì nếu gán mid->next = NULL trước sẽ không lấy được phần sau của danh sách liên kết nữa.

Ở hàm getMid ta tạo ra hai Node p,q và cho nó chạy từ đầu danh sách với Node p tăng lên 1 lần và Node q tăng lên 2 lần sau mỗi lần lặp. Như vậy khi Node q đến cuối danh sách thì Node p sẽ chỉ đến phần tử mid của danh sách liên kết.



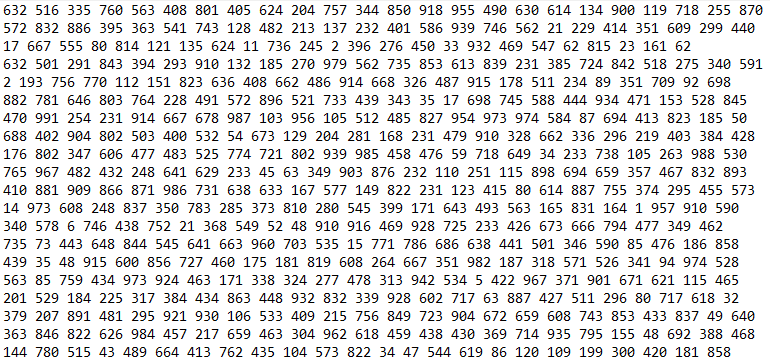
Hình 6. Hàm gộp hai danh sách L1, L2

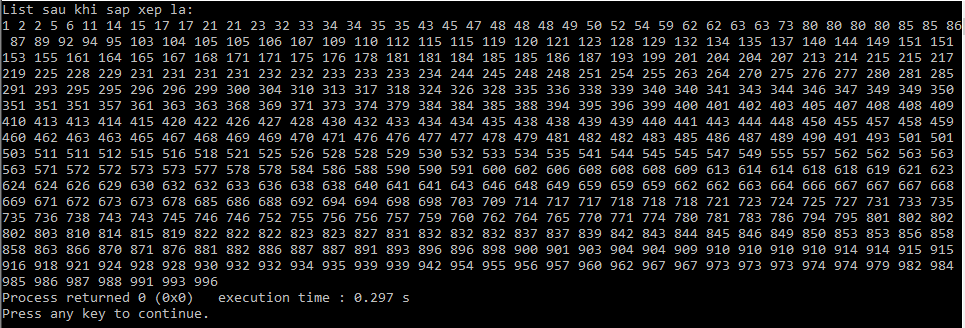
Ở hàm merge, đầu tiên chúng ta sẽ tạo ra một danh sách L rỗng để lưu trữ các phần tử được sắp xếp theo thứ tự tăng dần. Sau đó, chúng ta so sánh lần lượt các giá trị trong hai danh sách L1, L2 và thêm vào danh sách L phần tử có giá trị nhỏ hơn. Tiếp theo chúng ta cho con trỏ next của L trỏ tới phần tử có giá trị nhỏ tiếp theo bằng việc gọi đệ quy lại hàm merge một lần nữa. Tiếp tục thực hiện đệ quy như thế cho đến khi một trong hai danh sách L1, L2 đã hết và phần tử cuối cùng của một trong hai danh sách sẽ là phần tử cuối cùng của danh sách L.

1. **Cài đặt và kiểm thử:**

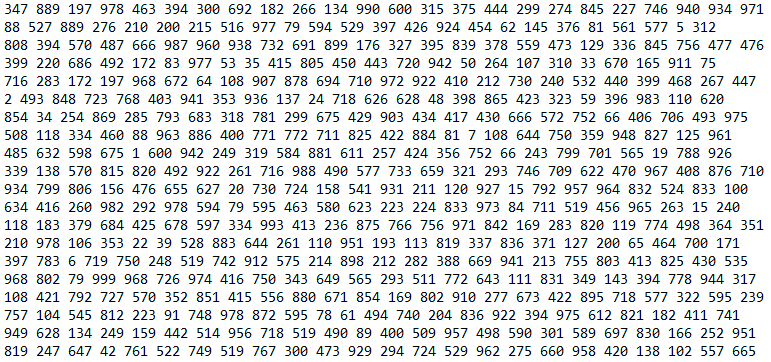
Và sau khi đã hoàn thành các code về sắp xếp danh sách liên kết bằng thuật toán Mergesort thì sau đây là một số trường hợp kiểm thử mà em đã chạy trong quá trình thực hiện:

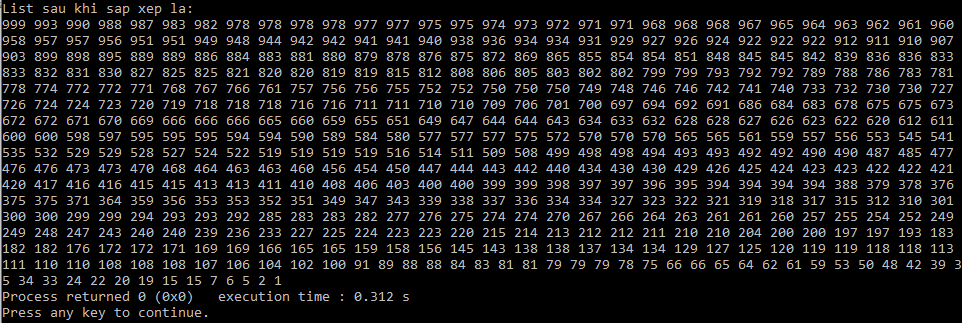
Trường hợp 1: Sắp xếp danh sách số nguyên ngẫu nhiên tăng dần theo giá trị.



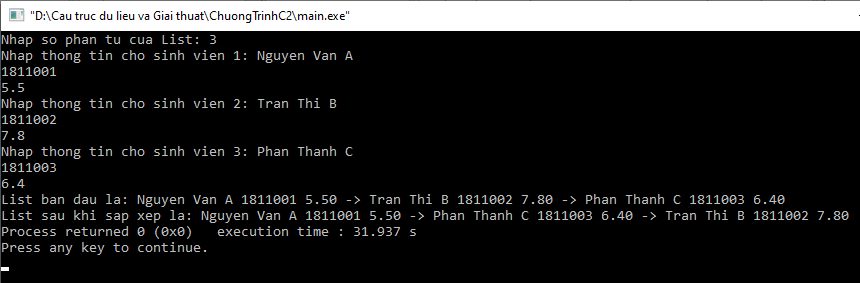


Trường hợp 2: Sắp xếp danh sách số nguyên ngẫu nhiên giảm dần theo giá trị.

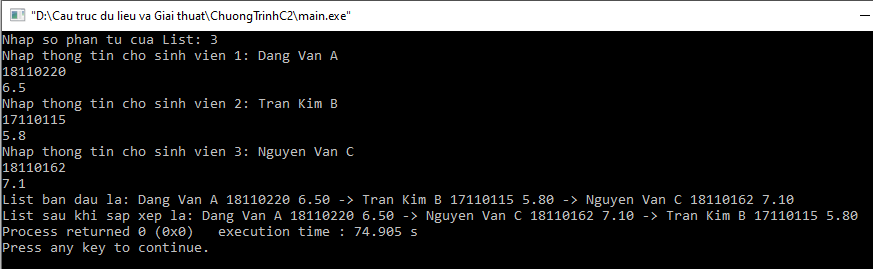




Trường hợp 3: Sắp xếp danh sách sinh viên tăng dần theo điểm trung bình.



Trường hợp 4: Sắp xếp danh sách sinh viên giảm dần theo MSSV.



# **Chương III: Phân công công việc**

# 

# **Chương IV: Đánh giá kết quả nghiên cứu và kết luận**

* + 1. **Đánh giá mức độ hoàn thành:**

Sau khi thực hiện, em nhận thấy đã hoàn thành được cơ bản các yêu cầu của đồ án được giao. Trong đồ án này, em đã tìm hiểu được phương pháp sắp xếp Merge sortvà cũng ôn lại được các kiến thức về danh sách liên kết. Từ đó có thể triển khai được thuật toán sắp xếp Merge sort vào danh sách liên kết.

Qua đó em đã rút ra được những ưu điểm và khuyết điểm của thuật toán Mergesort như sau:

|  |  |
| --- | --- |
| ***Ưu điểm*** | ***Khuyết điểm*** |
| Sắp xếp nhanh hơn so với các thuật toán cơ bản (Insertion sort, Interchange sort, Selection sort). | Thuật toán Merge sort là một thuât toán khó cài đặt. |
| Sắp xếp nhanh hơn Quick sort trong một số trường hợp. | Không nhận dạng được danh sách đã được sắp xếp. |

Bảng 3. Bảng ưu điểm và khuyết điểm của thuật toán Mergesort

Tuy nhiên thì em vẫn chưa thật sự vận dụng được hết các ưu điểm của thuật toán sắp xép Merge sort một cách tối ưu nhất, chỉ là sắp xếp danh sách các số hay danh sách các sinh viên tăng dần hoặc giảm dần.

* + 1. **Các khó khăn gặp phải:**

Trong quá trình làm, em đã gặp không ít những khó khăn. Có lẽ vì đây là lần đầu thực hiện một đồ án cuối kì và chỉ có một mình làm nên hầu như em đã tự tìm hiểu các tài liệu về thuật toán sắp xếp Merge sort. Bên cạnh đó thì áp lực về thời gian và công việc cũng là những khó khăn mà em gặp phải. Mặc dù ban đầu gặp khó khăn nhưng dần dần em đã có thể tự vượt qua được các khó khăn đó để có thể hoàn thành được đồ án bằng tất cả khả năng của mình.

* + 1. **Hướng phát triển:**

Trong thời gian tiếp theo, em sẽ cố gắng cải thiện để có thể ứng dụng được thuật toán Merge sort một cách tối ưu nhất. Mong muốn của em là có thể ứng dụng thuật toán Merge sort này để có thể sắp xếp một ứng dụng như là từ điển theo thứ tự các chữ cái giống như cuốn từ điển thông thường để người dùng có thể dễ dàng tra cứu từ một cách nhanh hơn hoặc là sắp xếp một danh sách vài chục ngàn học sinh theo thứ tự điểm từ trên xuống.

# 

# **Tài liệu tham khảo**

*1. https://www.geeksforgeeks.org/merge-sort/*

*2.* [*https://www.stdio.vn/articles/merge-sort-184*](https://www.stdio.vn/articles/merge-sort-184)

*3. https://nguyenvanhieu.vn/thuat-toan-sap-xep-merge-sort/*