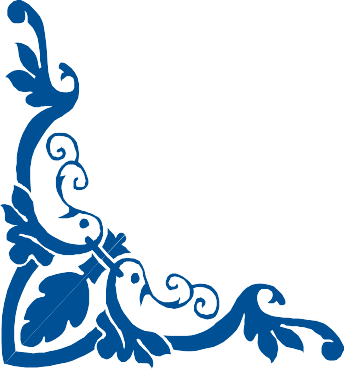
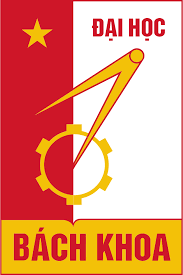
# ĐẠI HỌC BÁCH KHOA HÀ NỘI TRƯỜNG ĐIỆN-ĐIỆN TỬ



**BÁO** **CÁO** **MÔN** **CẤU** **TRÚC** **DỮ** **LIỆU** **VÀ** **GIẢI** **THUẬT**

**ĐỀ** **TÀI:**

**Merge sort algorithm for a singly linked list**

*Sinh* *viên* *thực* *hiện:* Nguyễn Mạnh Quân 20224453

*Giảng* *viên:* Tạ Thị Kim Huệ

## Hà Nội-2025

**Mục Lục**

1. [Giới thiệu 2](#_TOC_250012)
2. Tổng quan về đề tài 2
3. [Định nghĩa bài toán 2](#_TOC_250011)
4. Phương pháp lựa chọn 4
5. [Cấu trúc dữ liệu 4](#_TOC_250010)
6. [Giải thuật 8](#_TOC_250009)
7. [Cài đặt chương trình 11](#_TOC_250008)
   1. Dữ liệu đầu vào [11](#_TOC_250007)
   2. [Kết quả chương trình 11](#_TOC_250006)
8. [Kết luận 18](#_TOC_250002)
   1. Đánh giá về mức độ hoàn thành 18
   2. Bài học rút ra 19
   3. Khó khăn khi học tập môn học 19
9. [Tài liệu tham khảo 20](#_TOC_250001)
10. [Lời cảm ơn 20](#_TOC_250000)

1

# Giới thiệu

# Giới thiệu tổng quan về đề tài

# Đề tài " Merge sort algorithm for a singly linked list " tập trung vào việc áp dụng thuật toán sắp xếp Merge Sort để tổ chức lại các phần tử trong một danh sách liên kết đơn.

# Merge Sort là một thuật toán sắp xếp chia để trị, nổi bật với khả năng xử lý hiệu quả các tập dữ liệu lớn với độ phức tạp thời gian O(nlog(n)). Chương trình sẽ sử dụng danh sách liên kết đơn, một cấu trúc dữ liệu động, để thao tác với các phần tử mà không cần phải sử dụng bộ nhớ liên tục như trong mảng.

# Đề tài này không chỉ giúp chúng ta hiểu rõ cách thức hoạt động của Merge Sort mà còn mở rộng khả năng làm việc với các cấu trúc dữ liệu phức tạp, từ đó áp dụng các kiến thức về thuật toán và cấu trúc dữ liệu vào việc giải quyết các bài toán thực tế trong lập trình.

* 1. **Lý do lựa chọn đề tài**
     + Để đáp ứng yêu cầu môn học cũng như mong muốn tìm hiểu, mở rộng kiến thức về ngôn ngữ lập trình nói chung và môn cấu trúc dữ liệu và giải thuật nói riêng, em đã lựa chọn đề tài “**Merge sort algorithm for a singly linked list”**hướng với những tiêu chí:
       - Đây là một chủ đề có tính thực tế cao
       - Thuật toán Merge Sort là một trong những thuật toán sắp xếp cơ bản và hiệu quả.
       - Áp dụng được kiến thức đã học.

## Định nghĩa bài toán

* + - Input: Một danh sách liên kết đơn, trong đó:
      * Mỗi phần tử có giá trị kiểu số nguyên (int).
      * Các giá trị trong danh sách có thể không được sắp xếp.
    - Output: Một danh sách liên kết đơn đã được sắp xếp theo thứ tự tăng dần.

# Phương pháp lực chọn

## Cấu trúc dữ liệu

### Linked List

Linked list là một cấu trúc tuần tự bao gồm một chuỗi các item theo thứ tự tuyến tính được liên kết với nhau. Do đó, ta chỉ có thể truy cập tuần tự vào linked list, không thể thực hiện truy cập ngẫu nhiên. Linked list là cung cấp cho chúng ta một cấu trúc dữ liệu đơn giản và linh hoạt cho các tập hợp động:

* Các phần tử trong linked list được gọi là các **node**.
* Mỗi node sẽ chưa một **key** và một con trỏ trỏ tới node kế tiếp của nó, được gọi là **next**
* Thuộc tính tên là **head** trỏ tới phần tử đầu tiên của linked list.
* Phần tử cuối cùng của linked list có tên là **tail**

**A diagram of a key

Description automatically generated**

Một số loại linked list có thể kể tới bao gồm:

* Singly linked list: Duyệt qua các phẩn tử chỉ có thể thực hiện theo chiều hướng về phía trước.
* Doubly linked list: Duyệt qua các phẩn tử có thể thực hiện theo cả chiều tiến và lùi. Các node sẽ bao gồm thêm một con trỏ được gọi là **pre**, trỏ tới node trước đó.
* Circular linked list: Là một doubly linked list đặc biệt, khi mà con trỏ prev của head trỏ tới tail và con trỏ next của tail trỏ tới head.

Các phép toán trên linked list:

* Tìm kiếm: Tìm phần tử đầu tiên với key là **k** trong một linked list được cho trước được thực hiện đơn giản bằng một quá trình duyệt tuần tự và trả về con trỏ trỏ tới phần tử đó.
* Thêm: Để thêm một key vào một linked list có sẵn, ta có thể thực hiện theo 3 cách: thêm vào đầu list, thêm vào giữa list hoặc thêm vào cuối của list.
* Xoá: Xoá một phần tử **x** khỏi một linked list cho trước. Ta không thể xoá một node với chỉ một bước. Việc xoá một node có thể thực hiện theo 3 cách: xoá từ đầu danh sách, xoá từ giữa danh sách hoặc xoá từ cuối danh sách.

Ưu điểm:

* Tiết kiếm bộ nhớ và cấp phát động: Không như array cần 1 lượng chỉ định ô nhớ trên bộ nhớ ngay khi khỏi tạo. Linked list chỉ sử dụng bộ nhớ để lưu trữ khi dữ liệu thực sự được lưu vào linked list.
* Nó còn có thể lưu các phần tử ở bất cứ đâu được phép trên bộ nhớ mà không cần các ô nhớ liền kề nhau như array
* Quick insertion (Thêm rất nhanh với complexity chỉ là O(1))
* Quick deletion (Xóa nhanh)

## Giải thuật

**Merge Sort** là một trong những thuật toán sắp xếp theo phương pháp chia để trị (divide and conquer). Thuật toán hoạt động bằng cách chia nhỏ danh sách ban đầu thành hai phần bằng nhau, tiếp tục chia nhỏ cho đến khi chỉ còn các phần tử đơn lẻ, sau đó tiến hành hợp nhất (merge) các phần tử theo thứ tự đã sắp xếp. Merge Sort có độ phức tạp thời gian ổn định là , ngay cả trong trường hợp xấu nhất.

Trong các cấu trúc dữ liệu như mảng (array), Merge Sort thường được áp dụng hiệu quả, nhưng đặc biệt trong Linked List, Merge Sort trở nên nổi bật do các tính chất của danh sách liên kết:

1. **Không cần truy cập ngẫu nhiên:** Với Linked List, việc truy cập phần tử không thuận tiện như với mảng, nhưng Merge Sort chỉ cần thao tác tuần tự. Điều này phù hợp với đặc tính của Linked List.
2. **Tiết kiệm không gian:** Khi áp dụng Merge Sort trên Linked List, việc chia danh sách thành các phần nhỏ hơn và hợp nhất có thể thực hiện trực tiếp bằng cách thay đổi con trỏ thay vì sao chép dữ liệu.
3. **Ứng dụng trong xử lý dữ liệu lớn:** Do không cần sử dụng thêm không gian ngoài để lưu các mảng con, Merge Sort rất hữu ích khi sắp xếp dữ liệu trên các danh sách lớn hoặc khi bộ nhớ có hạn.

# A diagram of a number Description automatically generated

# Cài đặt chương trình

* **Chi tiết chương trình cài đặt sẽ được trình bày trong đường link sau:** https://github.com/firstyearstudent/Project\_DSA.git
* **Kết** **quả** **chạy chương trình**

**A screenshot of a computer program

Description automatically generated**

# Kết luận

* 1. **Đánh** **giá** **về** **mức** **độ** **hoàn** **thành**
* **Ưu điểm:**
  + **Thuật toán Merge Sort đã được triển khai thành công:** Chương trình sử dụng thuật toán Merge Sort để sắp xếp danh sách liên kết đơn, giúp sắp xếp hiệu quả với độ phức tạp thời gian O(n log n), thích hợp cho dữ liệu lớn.
  + **Sử dụng cấu trúc dữ liệu danh sách liên kết đơn:** Cấu trúc này giúp quản lý bộ nhớ động, cho phép thêm các phần tử vào cuối danh sách một cách dễ dàng và linh hoạt.
  + **Mã nguồn rõ ràng, dễ hiểu:** Chương trình chia rõ các phần như thêm phần tử vào danh sách, in danh sách, chia danh sách, và thuật toán Merge Sort, giúp người đọc dễ dàng theo dõi và hiểu từng bước thực hiện.
  + **Khả năng làm việc với danh sách có số lượng phần tử lớn:** Do Merge Sort có độ phức tạp O(nlog(n)), chương trình có thể xử lý hiệu quả với dữ liệu lớn.
  + Đã ứng dụng được kiến thức đã học của môn học, kết hợp với kiến thức tự tìm hiểu để làm sản phẩm
* **Nhược điểm:**
  + **Không có kiểm tra lỗi đầu vào:** Nếu người dùng nhập số lượng phần tử không hợp lệ (ví dụ, nhập số âm hoặc không phải số nguyên), chương trình chỉ dừng mà không có phản hồi rõ ràng về lỗi.
  + **Mức độ tối ưu về bộ nhớ chưa cao:** Chương trình tạo ra nhiều node mới trong quá trình chia nhỏ danh sách và hợp nhất, điều này có thể tiêu tốn nhiều bộ nhớ, đặc biệt khi xử lý với dữ liệu lớn.
* **Khả năng tối ưu hóa và nâng cấp:**
  + **Tối ưu về bộ nhớ:** Một cách để tối ưu hóa bộ nhớ là áp dụng thuật toán Merge Sort tại chỗ (in-place). Điều này sẽ tránh việc tạo ra các node mới khi thực hiện sắp xếp.
  + **Kiểm tra lỗi đầu vào:** Chương trình có thể được cải tiến để kiểm tra các giá trị đầu vào của người dùng, đảm bảo tính đúng đắn của dữ liệu trước khi tiến hành xử lý.
  + **Tối ưu tốc độ:** Mặc dù thuật toán Merge Sort có độ phức tạp O(n log n), nhưng có thể tối ưu thêm cách thức chia nhỏ và hợp nhất danh sách sao cho hiệu quả hơn, ví dụ như giảm thiểu việc di chuyển con trỏ qua lại nhiều lần.
  1. **Bài** **học** **rút** **ra**
* Môn học Cấu trúc dữ liệu và giải thuật vô cùng quan trong, là phần kiến thức không thể thiếu đối với sinh viên ngành Điện tử viễn thông.
* Học đi đôi với hành, em cần sử dụng kiến thức đã học để áp

dụng vào làm project để nắm vững kiến thức và tìm ra được những điều mới mẻ.

* Chọn đề tài vừa trong khả năng của bản thân, vừa có sự thử thách nhất định.
* Sau khi hoàn thành, cần phải suy nghĩ thêm để phát triển, khắc phục những hạn chế.
  1. **Khó** **khăn** **khi** **học** **tập** **môn** **học**
* Để học tốt môn Cấu trúc dữ liệu và giải thuật, em cần phải nắm vững kiến thức môn Kỹ thuật lập trình C/C++.
* Cần hiểu bản chất khi học.
* Lượng kiến thức lớn, cần phải đầu tư nhiều thời gian học và thực hành để nắm vững kiến thức.

# Lời cảm ơn

* Cấu trúc dữ liệu và giải thuật là môn học quan trọng đối với sinh viên những ngành liên quan CNTT nói chung và ngành Điện tử-Viễn thông nói riêng. Chúng em đã hiểu rằng ngôn ngữ lập trình có thể thay đổi nhưng cốt lõi là cấu trúc giữ liệu và giải thuật sẽ giúp em học được ngôn ngữ mới nhanh chóng.
* Em xin chân thành cảm ơn sự chỉ dạy nhiệt tình và những ví dụ minh

họa của cô giúp em học tập.