TRƯỜNG ĐẠI HỌC HỌC VĂN LANG

**KHOA CÔNG NGHỆ THÔNG TIN**



**BÁO CÁO ĐỒ ÁN MÔN HỌC**

**LẬP TRÌNH TÍNH TOÁN SONG SONG**

NGÀNH: CÔNG NGHỆ THÔNG TIN

***Đề tài:***

**TÌM HIỂU VÀ CÀI ĐẶT ỨNG DỤNG SONG SONG HOÁ THUẬT TOÁN SẮP XẾP NỔI BỌT (BUBBLE\_SORT ALGORITHM)**

**SVTH: Trần Hữu Luân**

**MSSV: 2274802010520**

**GVHD: TRẦN NGỌC VIỆT**

THÀNH PHỐ HỒ CHÍ MINH – NĂM 2025

LỜI CẢM ƠN  
-------------

Lời đầu tiên, em xin gửi lời cảm ơn chân thành đến thầy **Trần Ngọc Việt**. Trong quá trình học tập và tìm hiểu bộ môn Lập trình tính toán song song, em đã nhận được sự quan tâm, giúp đỡ và hướng dẫn tận tình, tâm huyết của thầy. Thầy đã truyền đạt cho chúng em hiểu rõ hơn về các thuật toán tối ưu hiệu suất tính toán và ứng dụng thực tiễn của chúng.

Nhóm thực hiện báo cáo

MỤC LỤC

[CHƯƠNG 1. CƠ SỞ LÝ THUYẾT – SONG SONG HOÁ THUẬT TOÁN SẮP XẾP NỔI BỌT (BUBBLE SORT ALGORITHM) 1](#_Toc139835835)

[1.1. Lý do chọn đề tài 1](#_Toc139835836)

[1.2. Tổng quan về song song hoá sắp xếp nổi bọt (Bubble Sort Algorithm) 1](#_Toc139835837)

[A. Giới thiệu về Bubble Sort 1.](#_Toc139835838)

[B. Song song hoá Bubble Sort 1.](#_Toc139835839)

[C. Các phương pháp song song hoá Bubble Sort 2.](#_Toc139835840)

[D. Ứng dụng thuật toán 2.](#_Toc139835841)

[1.3 Kết luận 3](#_Toc139835842)

[CHƯƠNG 2. ÁP DỤNG THUẬT TOÁN SONG SONG HOÁ NỔI BỌT TRONG SẮP XẾP ĐIỂM SỐ SINH VIÊN 4](#_Toc139835843)

[2.1 Phát biểu bài toán và mô tả dữ liệu: 4](#_Toc139835844)

[2.2 Minh hoạ 4](#_Toc139835844)

[2.3 So sánh sắp xếp nổi bọt tuần tự và song song hoá sắp xếp nổi bọt 4](#_Toc139835845)

[2.3.1 Sắp xếp nổi bọt tuần tự 4](#_Toc139835845)

[2.3.2 Áp dụng thuật toán song song hoá sắp xếp nổi bọt 4](#_Toc139835845)

[2.4 Kết luận 5](#_Toc139835846)

[CHƯƠNG 3. KẾT LUẬN 9](#_Toc139835848)

[TÀI LIỆU THAM KHẢO 11](#_Toc139835849)

# CHƯƠNG 1. CƠ SỞ LÝ THUYẾT – SONG SONG HOÁ THUẬT TOÁN SẮP XẾP NỔI BỌT (BUBBLE\_SORT ALGORITHM)

## 1.1. Lý do chọn đề tài

Trong thời đại công nghệ số phát triển mạnh mẽ, việc tối ưu hóa hiệu suất tính toán ngày càng trở nên quan trọng. Thuật toán sắp xếp nổi bọt (Bubble Sort) là một trong những thuật toán sắp xếp đơn giản nhất, thường được sử dụng để minh họa các nguyên tắc cơ bản của thuật toán sắp xếp. Tuy nhiên, Bubble Sort có hiệu suất thấp khi xử lý dữ liệu lớn. Do đó, việc song song hóa thuật toán này giúp cải thiện tốc độ thực thi đáng kể. Mục tiêu của đề tài này là tìm hiểu và cài đặt thuật toán Bubble Sort theo hướng song song hóa, từ đó so sánh hiệu suất giữa thuật toán sắp xếp tuần tự và thuật toán sắp xếp song song trên các nền tảng khác nhau. Việc nghiên cứu và áp dụng song song hóa không chỉ giúp nâng cao tốc độ xử lý mà còn mở rộng khả năng ứng dụng trong các hệ thống xử lý dữ liệu lớn.

## 1.2. Tổng quan về thuật toán song song hoá sắp xếp nổi bọt (Bubble Sort Algorithm)

### Giới thiệu về Bubble Sort

### Thuật toán sắp xếp nổi bọt là một thuật toán sắp xếp đơn giản, hoạt động bằng cách liên tục hoán đổi các phần tử liền kề nếu chúng không theo đúng thứ tự mong muốn. Thuật toán này lặp lại cho đến khi mảng được sắp xếp hoàn toàn.

**Độ phức tạp thời gian:**

* **Trường hợp xấu nhất và trung bình:** O(n²)
* **Trường hợp tốt nhất (danh sách đã được sắp xếp):** O(n)

Mặc dù thuật toán dễ hiểu và dễ triển khai, nhưng hiệu suất thấp với dữ liệu lớn khiến nó ít được sử dụng trong thực tế.

### Song song hoá Bubble Sort

### Song song hóa (parallelization) là quá trình chia nhỏ một thuật toán thành nhiều phần có thể chạy đồng thời trên nhiều luồng (threads) hoặc nhiều bộ xử lý (processors), giúp tăng tốc độ xử lý.

Thuật toán Bubble Sort có thể được song song hóa theo hai cách chính:

* Song song hóa dựa trên luồng (Multithreading - Shared Memory Model)
* Song song hóa trên nhiều bộ xử lý (Distributed Parallelism - Distributed Memory Model)

### Các phương pháp song song hoá Bubble Sort

* Phương pháp sử dụng OpenMP: Trong C/C++, OpenMP có thể được dùng để tạo các luồng xử lý đồng thời. Chia vòng lặp thành các phần chạy trên nhiều luồng.
* Phương pháp sử dụng MPI: Trong môi trường phân tán (cluster), MPI giúp mỗi bộ xử lý đảm nhiệm một phần của mảng. Kết quả được trao đổi giữa các bộ xử lý để đảm bảo danh sách được sắp xếp hoàn chỉnh.
* Phương pháp Even-Odd Transposition Sort (Sắp xếp hoán vị chẵn-lẻ)**:** Đây là một biến thể của Bubble Sort có thể song song hóa hiệu quả. Thuật toán thực hiện hai bước (bước chẵn và bước lẻ). Hai bước này có thể chạy song song, giúp cải thiện tốc độ so với Bubble Sort truyền thống.

1. *Ứng dụng thuật toán*

* Xử lý dữ liệu lớn (Big Data Processing):**Parallel Bubble Sort** có thể được sử dụng để tối ưu hóa dữ liệu nhỏ trong các bước trung gian trước khi chuyển sang các thuật toán mạnh hơn. Một số hệ thống phân tán (Distributed Systems) có thể tận dụng song song hóa để xử lý các tập dữ liệu nhỏ song song trên nhiều máy chủ.
* Hệ thống xử lý tín hiệu và âm thanh (Signal & Audio Processing)*:* rong các hệ thống xử lý âm thanh, tín hiệu số thường cần sắp xếp các mẫu dữ liệu theo thứ tự nhất định (ví dụ: loại bỏ nhiễu). Song song hóa thuật toán giúp xử lý nhanh hơn, đặc biệt khi sử dụng GPU hoặc CPU đa nhân.
* Tăng tốc xử lý đồ họa và hình ảnh (Graphics & Image Processing): Trong các ứng dụng **rendering đồ họa, xử lý ảnh**, sắp xếp pixel theo độ sáng hoặc màu sắc là một thao tác quan trọng. Khi song song hóa, thuật toán có thể chạy trên **GPU (CUDA/OpenCL)** để tăng tốc quá trình xử lý hình ảnh.

## 

## 1.3 Kết luận

Thuật toán **sắp xếp nổi bọt (Bubble Sort)**là một thuật toán đơn giản nhưng có độ phức tạp cao **O(n²),** khiến nó trở nên kém hiệu quả khi làm việc với dữ liệu lớn. Tuy nhiên, với sự phát triển của **lập trình song song**, việc **song song hóa Bubble Sort**có thể giúp cải thiện tốc độ xử lý bằng cách tận dụng sức mạnh của **đa luồng (multithreading)** hoặc **đa bộ xử lý (multiprocessing).** Trong nghiên cứu này, chúng tôi đã phân tích các phương pháp song song hóa thuật toán Bubble Sort, bao gồm **sử dụng OpenMP, MPI và phương pháp Even-Odd Transposition Sort,** giúp giảm đáng kể thời gian thực thi so với phiên bản tuần tự. Mặc dù song song hóa không làm thay đổi bản chất độ phức tạp thuật toán, nhưng nó giúp tối ưu hóa hiệu suất trên các hệ thống hiện đại, đặc biệt trong các bài toán có quy mô nhỏ và trung bình. Kết quả nghiên cứu này mở ra hướng phát triển các thuật toán song song hóa khác, giúp cải thiện hiệu suất xử lý dữ liệu trong nhiều lĩnh vực như **xử lý dữ liệu lớn, trí tuệ nhân tạo, và hệ thống thời gian thực.**

# CHƯƠNG 2. ÁP DỤNG THUẬT TOÁN SONG SONG HOÁ NỔI BỌT TRONG SẮP XẾP ĐIỂM SỐ SINH VIÊN

## 

## 2.1 Phát biểu bài toán:

## Trong thực tế, việc sắp xếp điểm số sinh viên là một tác vụ quan trọng trong quản lý giáo dục, giúp đánh giá thành tích học tập và hỗ trợ công tác xếp hạng. Khi số lượng sinh viên ngày càng lớn, các thuật toán sắp xếp truyền thống có thể gặp phải vấn đề về hiệu suất. Bài toán đặt ra là xây dựng một phương pháp sắp xếp điểm số sinh viên bằng thuật toán **Bubble Sort**theo hướng **song song hóa**để tăng tốc độ xử lý. Thuật toán sẽ được triển khai trên một hệ thống đa luồng nhằm cải thiện hiệu suất so với thuật toán tuần tự. Dữ liệu đầu vào bao gồm danh sách điểm số của sinh viên trong một kỳ thi hoặc một học phần. Kết quả đầu ra là danh sách điểm số đã được sắp xếp theo thứ tự tăng dần hoặc giảm dần, tùy theo yêu cầu. Bằng cách áp dụng **song song hóa**, thuật toán có thể chia nhỏ công việc và xử lý đồng thời trên nhiều luồng, giúp giảm thời gian thực thi và tối ưu hóa hiệu suất khi xử lý dữ liệu lớn.

2.2. Minh hoạ

Ta có một mảng [5, 3, 8, 4, 2, 7, 1, 6].

B1: Chia thành 2 mảng: mảng 1 là [1 , 5, 3 , 8 , 4] và mảng 2 là [2, 2, 7, 1,6]

B2: Sắp xếp từng phần bằng Bubble Sort:

Ta có mảng 1: [5, 3, 8, 4]

1. **Lần lặp 1:**
   * So sánh 5 và 3, đổi chỗ → [3, 5, 8, 4]
   * So sánh 5 và 8, đúng thứ tự → [3, 5, 8, 4]
   * So sánh 8 và 4, đổi chỗ → [3, 5, 4, 8]
2. **Lần lặp 2:**
   * So sánh 3 và 5, đúng thứ tự → [3, 5, 4, 8]
   * So sánh 5 và 4, đổi chỗ → [3, 4, 5, 8]
   * So sánh 5 và 8, đúng thứ tự → [3, 4, 5, 8]
3. **Lần lặp 3:**
   * So sánh 3 và 4, đúng thứ tự → [3, 4, 5, 8]
   * So sánh 4 và 5, đúng thứ tự → [3, 4, 5, 8]
   * So sánh 5 và 8, đúng thứ tự → [3, 4, 5, 8]

**Kết quả sắp xếp mảng 1:** [3, 4, 5, 8]

Ta có mảng 2: [2, 7, 1, 6] → [2, 1, 7, 6] → [1, 2, 6, 7]

1. **Lần lặp 1:**
   * So sánh 2 và 7, đúng thứ tự → [2, 7, 1, 6]
   * So sánh 7 và 1, đổi chỗ → [2, 1, 7, 6]
   * So sánh 7 và 6, đổi chỗ → [2, 1, 6, 7]
2. **Lần lặp 2:**
   * So sánh 2 và 1, đổi chỗ → [1, 2, 6, 7]
   * So sánh 2 và 6, đúng thứ tự → [1, 2, 6, 7]
   * So sánh 6 và 7, đúng thứ tự → [1, 2, 6, 7]
3. **Lần lặp 3:**
   * So sánh 1 và 2, đúng thứ tự → [1, 2, 6, 7]
   * So sánh 2 và 6, đúng thứ tự → [1, 2, 6, 7]
   * So sánh 6 và 7, đúng thứ tự → [1, 2, 6, 7]

💡 **Kết quả sắp xếp mảng 2:** [1, 2, 6, 7]

B3: Gộp 2 mảng đã được sắp xếp, ta có: [1, 2, 3, 4, 5, 6, 7, 8]

2.3. So sánh sắp xếp nổi bọt tuần tự và song song hoá nổi bọt

2.3.1. Sắp xếp nổi bọt tuần tự

import pandas as pd

def bubble\_sort(arr):

""" Thuật toán Bubble Sort cơ bản (tuần tự) """

n = len(arr)

for i in range(n):

for j in range(0, n - i - 1):

if arr[j][2] > arr[j + 1][2]: # Sắp xếp theo cột Marks (cột thứ 3)

arr[j], arr[j + 1] = arr[j + 1], arr[j]

return arr

if \_\_name\_\_ == "\_\_main\_\_":

file\_path = "/Users/macbook/Desktop/Hoc tap/Lập trình tính toán song song/Student\_Marks.csv"

try:

df = pd.read\_csv(file\_path)

data\_list = df.values.tolist()

# Thực hiện sắp xếp tuần tự

sorted\_result = bubble\_sort(data\_list)

# Chuyển kết quả thành DataFrame

sorted\_df = pd.DataFrame(sorted\_result, columns=df.columns)

# Lưu kết quả vào file CSV

sorted\_df.to\_csv("/Users/macbook/Desktop/Hoc tap/Lập trình tính toán song song/Sorted\_Student\_Marks\_Sequential.csv", index=False)

print("Sắp xếp hoàn tất! Kết quả được lưu vào Sorted\_Student\_Marks\_Sequential.csv")

except FileNotFoundError:

print(f"Không tìm thấy file: {file\_path}")

A screenshot of a computer

AI-generated content may be incorrect.

## 

## 2.3.2. Áp dụng thuật toán song song hoá sắp xếp nổi bọt

import pandas as pd

import multiprocessing

def bubble\_sort(arr):

""" Thuật toán Bubble Sort cơ bản (sắp xếp tăng dần theo cột Marks) """

n = len(arr)

for i in range(n):

for j in range(0, n - i - 1):

if arr[j][2] > arr[j + 1][2]:

arr[j], arr[j + 1] = arr[j + 1], arr[j]

return arr

def process\_sort(chunk):

""" Hàm thực thi Bubble Sort trên một phần dữ liệu """

return bubble\_sort(chunk)

def merge\_sorted\_chunks(chunks):

""" Hợp nhất các phần dữ liệu đã sắp xếp """

merged = []

for chunk in chunks:

merged.extend(chunk)

return sorted(merged, key=lambda x: x[2]) # Sắp xếp lại toàn bộ dữ liệu theo Marks

def parallel\_bubble\_sort(data, num\_processes=4):

""" Sắp xếp song song bằng Multiprocessing """

size = len(data)

chunk\_size = size // num\_processes

chunks = [data[i \* chunk\_size:(i + 1) \* chunk\_size] for i in range(num\_processes - 1)]

chunks.append(data[(num\_processes - 1) \* chunk\_size:])

with multiprocessing.Pool(processes=num\_processes) as pool:

sorted\_chunks = pool.map(process\_sort, chunks)

# Hợp nhất kết quả đã sắp xếp

sorted\_data = merge\_sorted\_chunks(sorted\_chunks)

return sorted\_data

if \_\_name\_\_ == "\_\_main\_\_":

file\_path = "Student\_Marks.csv"

output\_path = "Sorted\_Parallel.csv"

try:

df = pd.read\_csv(file\_path)

data\_list = df.values.tolist()

# Thực hiện sắp xếp song song bằng Multiprocessing

sorted\_result = parallel\_bubble\_sort(data\_list, num\_processes=4)

# Chuyển kết quả thành DataFrame

sorted\_df = pd.DataFrame(sorted\_result, columns=df.columns)

# Lưu kết quả vào file CSV

sorted\_df.to\_csv(output\_path, index=False)

print(f"Sắp xếp hoàn tất! Kết quả được lưu vào {output\_path}")

except FileNotFoundError:

print(f"Không tìm thấy file: {file\_path}")

## **A screenshot of a computer AI-generated content may be incorrect.**

Mô hình thuật toán song song hoá sắp xếp nổi bọt để sắp xếp điểm số sinh viên:

A hand holding a graph

AI-generated content may be incorrect.

Mô tả: Mảng gồm 4 thành phần là P0 ,P1 ,P2, P3 ( mỗi mảng gồm 25 phần tử chưa được sắp xếp)

P1 truyền thông cho P0 , sau đó hợp các phần tử của P0 và P1 lại đã được sắp xếp

P3 truyền thông cho P2, sau đố hợp các phần tử của P2 và P3 lại đã được sắp xếp

Mảng đã được sắp xếp của P2 và P3 truyền thông cho mảng đã được sắp xếp của P0 và P1

Sau đó cho ra mảng A đã được sắp xếp theo thứ tự tăng dần của 4 thành phần (P0, P1, P2, P3). Ta được

* + Mảng A đã được sắp xếp 100 phần tử từ bé đến lớn theo điểm số của sinh viên.

## 2.4 Kết luận

Việc song song hóa thuật toán Bubble Sort bằng **multiprocessing** giúp **cải thiện hiệu suất** khi xử lý dữ liệu lớn. Cách tiếp cận này chia dữ liệu thành nhiều phần nhỏ, thực hiện sắp xếp đồng thời trên từng phần, sau đó hợp nhất kết quả để tạo ra danh sách đã sắp xếp hoàn chỉnh. So với phương pháp tuần tự, cách tiếp cận song song có thể **giảm đáng kể thời gian thực thi**, đặc biệt khi hệ thống có nhiều **CPU cores** để tận dụng. Tuy nhiên, việc hợp nhất các phần dữ liệu đã sắp xếp vẫn yêu cầu sắp xếp lại toàn bộ danh sách, điều này có thể làm tăng chi phí tính toán. Do đó, cần xem xét tối ưu hóa quá trình hợp nhất để đảm bảo hiệu suất cao hơn khi xử lý dữ liệu điểm số của sinh viên với số lượng lớn.

# 

# CHƯƠNG 3. KẾT LUẬN

Thuật toán Bubble Sort đã được triển khai theo hướng song song để cải thiện hiệu suất xử lý dữ liệu. Trong quá trình thực hiện, chúng em nhận thấy rằng việc sử dụng **multiprocessing** giúp chia nhỏ dữ liệu và thực hiện sắp xếp đồng thời trên nhiều luồng, từ đó **giảm đáng kể thời gian thực thi** so với thuật toán tuần tự. Cách tiếp cận này đặc biệt hữu ích khi làm việc với tập dữ liệu lớn và hệ thống có nhiều **CPU cores** để tận dụng khả năng xử lý song song. Tuy nhiên, một trong những hạn chế của phương pháp này là **quá trình hợp nhất dữ liệu sau khi sắp xếp**. Khi các phần dữ liệu đã sắp xếp riêng lẻ được kết hợp lại, cần thực hiện sắp xếp lại toàn bộ để đảm bảo thứ tự chính xác, điều này có thể làm **tăng chi phí tính toán**. Ngoài ra, hiệu suất của thuật toán còn phụ thuộc vào **kích thước dữ liệu đầu vào**và **số lượng tiến trình** được sử dụng. Nếu số tiến trình không được tối ưu, có thể dẫn đến **tình trạng quá tải tài nguyên hoặc giảm hiệu suất** do chi phí quản lý luồng cao hơn lợi ích thu được từ song song hóa. Dù vẫn tồn tại một số thách thức, nhưng kết quả thực nghiệm cho thấy rằng **việc song song hóa Bubble Sort có thể giúp tối ưu hóa thời gian xử lý**trong những trường hợp dữ liệu lớn, đồng thời vẫn giữ nguyên tính đơn giản của thuật toán. Điều này cho thấy tiềm năng áp dụng trong các hệ thống yêu cầu **xử lý nhanh dữ liệu theo thời gian thực**, chẳng hạn như trong lĩnh vực tài chính, nơi các quyết định cần được đưa ra kịp thời dựa trên khối lượng lớn thông tin.

# TÀI LIỆU THAM KHẢO

1. Giải thuật sắp xếp nổi bọt (Bubble Sort)

<https://vietjack.com/cau-truc-du-lieu-va-giai-thuat/giai-thuat-sap-xep-noi-bot.jsp> Năm 2019.

1. Parallel Bubble Sort: <https://iq.opengenus.org/parallel-bubble-sort/> Ngày 21/1/2025
2. Bubble Sort Time Complexity and Algorithm Explained <https://builtin.com/data-science/bubble-sort-time-complexity>