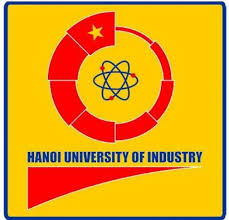
**BỘ CÔNG THƯƠNG**

**TRƯỜNG ĐẠI HỌC CÔNG NGHIỆP HÀ NỘI**

**KHOA: CÔNG NGHỆ THÔNG TIN**



**BÀI TẬP LỚN MÔN**

**TÍNH TOÁN SONG SONG VÀ PHÂN TÁN**

**ĐỀ TÀI: TÌM HIỂU THUẬT TOÁN SẮP XẾP SONG SONG( THUẬT TOÁN SẮP XẾP MERGE SORT)**

**NHÓM: 9**

|  |  |
| --- | --- |
| **Giáo viên hướng dẫn:** | **Ths. Nguyễn Thị Lan Anh** |
| **Sinh viên thực hiện:** | **Đỗ Văn Sơn** |
|  | **Phạm Hồng Sơn** |
|  | **Nguyễn Thị Quỳnh** |
|  |  |

**Hà Nội, 10/2018**

**Mục lục**

[CHƯƠNG 1: TỔNG QUAN VỀ TÍNH TOÁN SONG SONG 3](#_Toc528551205)

[**1.** **Tính toán song song** 3](#_Toc528551206)

[**2.** **Nguyên lý thiết kế tính toán song song** 4](#_Toc528551207)

[*2.1.* *Nguyên lý lập lịch* 4](#_Toc528551208)

[*2.2.* *Nguyên lý hình ống* 4](#_Toc528551209)

[*2.3.* *Nguyên lý chia để trị* 4](#_Toc528551210)

[*2.4.* *Nguyên lý đồ thị phụ thuộc dữ liệu* 4](#_Toc528551211)

[Phân tích mối quan hệ dữ liệu trong tính toán để xây dựng đồ thị phụ thuộc dữ liệu và dựa vào đó để xây dựng thuật toán song song. 5](#_Toc528551212)

[*2.5.* *Nguyên lý điều khiển tương tranh* 5](#_Toc528551213)

[Nếu hai tiến trình cùng muốn truy cập vào cùng một mục dữ liệu chia sẻ thì chúng phải tương tranh với nhau, nghĩa là chúng có thể cản trở lẫn nhau. 5](#_Toc528551214)

[**3.** **Mục đích của tính toán song song** 5](#_Toc528551215)

[**4.** **Ưu điểm của tính toán song song** 5](#_Toc528551216)

[CHƯƠNG 2: THUẬT TOÁN MERGE SORT 6](#_Toc528551217)

[1. **Tìm hiểu về thuật toán Merge sort** 6](#_Toc528551218)

[CHƯƠNG 3: TÍNH TOÁN TUẦN TỰ THUẬT TOÁN MERGE SORT 9](#_Toc528551219)

[**1.** **Khái niệm tính toán tuần tự** 9](#_Toc528551220)

[2. Mô tả bài toán 9](#_Toc528551221)

[CHƯƠNG 4: TÍNH TOÁN SONG SONG THUẬT TOÁN MERGE SORT 14](#_Toc528551222)

[**1.** **Mô tả bài toán** 15](#_Toc528551223)

[**2.** **Demo song song** 15](#_Toc528551224)

[**Kết luận** 19](#_Toc528551225)

[**Tài liệu tham khảo** 21](#_Toc528551226)

[1. Giáo trình Xử lý song song và phân tán 21](#_Toc528551227)

[1. https://www.stdio.vn/articles/merge-sort-184 21](#_Toc528551228)

[2. https://parcomp.wordpress.com/2017/02/26/parallel-merge-sort/ 21](#_Toc528551229)

**Lời mở đầu**

Trong những năm gần đây, mặc dù nền Công nghệ thông tin của thế giới ngày một phát triển. Tốc độ xử lý máy tính ngày càng tăng lên. Tuy nhiên, chúng ta cũng gặp phải khó khăn trong một số bài toán có dữ liệu đầu vào lớn (bài toán dự báo thời tiết, dự báo động đất,...). Với dữ liệu đầu vào là một con số rất lớn, dù máy tính có tốc độ lơn, bộ nhớ nhiều vấp phải yêu cầu giải quyết bài toán trong thời gian chấp nhận được.

Trong nhiều năm qua, các nhà khoa học đã nghĩ ra biện pháp giải quyết hiệu quả đó là chia nhỏ bài toán ra thành nhiều bài toán. Việc giải quyết các bài toán nhỏ được tiến hành đồng thời với nhiều máy tính. Kết quả của bài toán lớn được giải quyết khi tất cả các bài toán nhỏ đã được làm.

Các máy tính tiến hành xử lý song song được kết nối với nhau thành các cụm tính toán tốc độ cao.

# 

# CHƯƠNG 1: TỔNG QUAN VỀ TÍNH TOÁN SONG SONG

1. **Tính toán song song**

Hiện nay, để giải quyết các bài toán lớn người ta thường nghĩ đến việc sử dụng các siêu máy tính hoặc việc kết hợp nhiều máy tính với nhau để tính toán. Tuy nhiên, với phương pháp lập trình cổ điển thì không thể nào phát triển được chương trình có thể tận dụng được sức mạnh của các hệ thống đó. Đó chính là lý do lập trình song song ra đời.

Lập trình song song là một công việc rất phức tạp so với lập trình tuần tự thông thường, người phát triển phải thực hiện một quá trình song song có khả năng tận dụng tối đa sức mạnh của hệ thống.

Tính toán song song là quá trình tính toán gồm những tiến trình được kích hoạt đồng thời và cùng tham gia tính toán giải quyết một công việc, và nói chung phải thực hiện trên các hệ thống đa bộ xử lý.

1. **Nguyên lý thiết kế tính toán song song**
   1. *Nguyên lý lập lịch*

Giảm tối thiểu các bộ xử lý sử dụng trong thuật toán sao cho thời gian tính taosn là không tang (xét theo khía cạnh độ phức tạp). Nghĩa là, nếu độ phức tạp tính toán của thuật toán là O(f(n)) thì thời gian thực hiện của chương trình có thể tăng khi số bộ xử lý giảm, và thời gian tính toán tổng thể tăng lên một hằng số nào đó – nhưng vẫn là O(f(n)).

* 1. *Nguyên lý hình ống*

Nguyên lý này được áp dụng khi bài toán xuất hiện một dây các thao tác {T1, T2,…,Tn}, tỏng đó Ti+1 thực hiện sau khi Ti kết thúc.

* 1. *Nguyên lý chia để trị*

Tức là chia bài toán thành những phần nhỏ hơn tương đối độc lập với nhau và giải quyết chúng một cách song song. Tạo ra một mô hình cây phân cấp để phân cấp quá trình truyền thông và tính toán.

+ Tăng tính song song so với mô hình trước

+ Thời gian chạy giảm từ O(n) xuống O(logn)

* 1. *Nguyên lý đồ thị phụ thuộc dữ liệu*

Phân tích mối quan hệ dữ liệu trong tính toán để xây dựng đồ thị phụ thuộc dữ liệu và dựa vào đó để xây dựng thuật toán song song.

* 1. *Nguyên lý điều khiển tương tranh*

Nếu hai tiến trình cùng muốn truy cập vào cùng một mục dữ liệu chia sẻ thì chúng phải tương tranh với nhau, nghĩa là chúng có thể cản trở lẫn nhau.

1. **Mục đích của tính toán song song**

* Nâng cao hiệu năng tính toán
* Tính toán các vấn đề khoa học rất phức tạp trong thế giới thực
* Khí quyển, trái đất, môi trường
* Vật lý ứng dụng, hạt nhân, hạt, áp suất cao, sự nóng chảy, lượng tử ánh sang
* Sinh học, công nghệ sinh học, di truyền học
* Hóa học, khoa học phân tử
* Địa chất, địa chấn học
* Công nghệ vũ trụ
* Kỹ thuật điện, thiết kế mạch, vi điện tử
* Khoa học máy tính, toán học

1. **Ưu điểm của tính toán song song**

* Tiết kiệm thời gian hoặc tài chính:
* Khi chi phí nhiều tài nguyên cho một công việc nào đó sẽ rút ngắn được thời gian hoàn thành công việc đó, và tiết kiệm các chi phí tiềm năng
* Phần cứng song song có thể được xây dựng với giá thành thấp và dễ mua
* Giải quyết được các vấn đề lớn hơn phức tạp hơn:
* Nhiều vấn đề quá lớn quá phức tạp đến nỗi nó không thực tế hoặc không có khả năng giải quyết chúng trên máy tính đơn, đặc biệt là bộ nhớ máy tính lại bị hạn chế. Chẳng hạn các vấn đề:
* Dự báo thời tiết, bão, động đất, sóng thần, mô hình sinh thái
* Máy dò tìm Web/CSDL để xử lý hàng triệu giao dịch trên một giây
* Cung cấp tính đồng thời:
* Tài nguyên của một máy tính đơn nguyên chỉ có khả năng thực hiện một lệnh tại một thời điểm
* Tài nguyên của máy tính đa bộ xử lý có thể thực hiện nhiều lệnh đồng thời
* Sử dụng tài nguyên phi cục bộ:

* Sử dụng tài nguyên tính toán trên một mạng diện rộng hoặc thậm chí internet khi tài nguyên nơi tính toán tài nguyên đang khan hiếm, không đủ

# 

# CHƯƠNG 2: THUẬT TOÁN MERGE SORT

1. **Tìm hiểu về thuật toán Merge sort**

Trong khoa học máy tính , sắp xếp trộn (Merge Sort) là một thuật toán sắp xếp các danh sách (hoặc bất kỳ cấu trúc dữ liệu nào có thể truy cập tuần tự) theo một trật tự nào đó. Nó được xếp vào thể loại sắp xếp so sánh.

Thuật toán này là một ví dụ tương đối điển hình của lối thuật toán chia để trị do John von Neumann đưa ra lần đầu năm 1945. Một thuật toán chi tiết được Goldstine và Neumann đưa ra vào năm 1948.

Chúng ta sẽ chia mảng lớn thành những mảng con nhỏ hơn bằng cách chia đôi mảng lớn và chúng ta tiếp tục chia đôi các mảng con cho tới khi mảng con nhỏ nhất chỉ còn 1 phần tử.

Sau đó chúng ta sẽ tiến hành **trộn** 2 mảng con có cùng mảng cơ sở (khi chúng ta chia đôi mảng lớn thành 2 mảng con thì mảng lớn đó chúng ta gọi là mảng cơ sở của 2 mảng con đó) khi so sánh chúng sẽ vừa sắp xếp vừa ghép 2 mảng con đó lại thành mảng cơ sở, chúng ta tiếp tục trộn các mảng con lại đến khi còn lại mảng duy nhất thì đó là mảng đã được sắp xếp.

Giả sử chúng ta có mảng sau:

Giải thuật sắp xếp trộn (Merge Sort)

* Đầu tiên, giải thuật sắp xếp trộn chia toàn bộ mảng thành hai nửa. Tiến trình chia này tiếp tục diễn ra cho đến khi không còn chia được nữa và chúng ta thu được các giá trị tương ứng biểu diễn các phần tử trong mảng. Trong hình dưới, đầu tiên chúng ta chia mảng kích cỡ 8 thành hai mảng kích cỡ 4.

Giải thuật sắp xếp trộn (Merge Sort)

* Tiến trình chia này không làm thay đổi thứ tự các phần tử trong mảng ban đầu. Bây giờ chúng ta tiếp tục chia các mảng này thành 2 nửa.

Giải thuật sắp xếp trộn (Merge Sort)

* Tiến hành chia tiếp cho tới khi không còn chia được nữa.

Giải thuật sắp xếp trộn (Merge Sort)

* Bây giờ chúng ta tổ hợp chúng theo như đúng cách thức mà chúng được chia ra.
* Đầu tiên chúng ta so sánh hai phần tử trong mỗi list và sau đó tổ hợp chúng vào trong một list khác theo cách thức đã được sắp xếp. Ví dụ, 14 và 33 là trong các vị trí đã được sắp xếp. Chúng ta so sánh 27 và 10 và trong list khác chúng ta đặt 10 ở đầu và sau đó là 27. Tương tự, chúng ta thay đổi vị trí của 19 và 35. 42 và 44 được đặt tương ứng.

Giải thuật sắp xếp trộn (Merge Sort)

* Vòng lặp tiếp theo là để kết hợp từng cặp list một ở trên. Chúng ta so sánh các giá trị và sau đó hợp nhất chúng lại vào trong một list chứa 4 giá trị, và 4 giá trị này đều đã được sắp thứ tự.

Giải thuật sắp xếp trộn (Merge Sort)

* Sau bước kết hợp cuối cùng, danh sách sẽ trông giống như sau:

Giải thuật sắp xếp trộn (Merge Sort)

# 

# CHƯƠNG 3: TÍNH TOÁN TUẦN TỰ THUẬT TOÁN MERGE SORT

1. **Khái niệm tính toán tuần tự**

Một công việc được chia thành một dãy rời rạc các chỉ thị mà máy tính có thể hiểu được để thực hiện.

Máy tính thực hiện các chỉ thị theo thứ tự mà chúng xuất hiện

Chỉ có duy nhất một chỉ thị được thực hiện tại một thời điểm

## 2. Mô tả bài toán

Cho 1 danh sách A chưa sắp xếp có q phần tử

Thuật toán:

**Bước 1:** Chia mảng thành 2 mảng nhỏ m[i] và n[j]

**Bước 2:** Thực hiện với mảng m trước quay lại bước 1 cho đến khi mỗi mảng chỉ có một phần tử

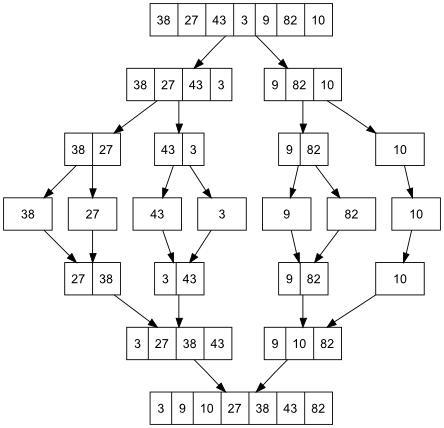
**Bước 3:** Tại mỗi bước nếu cả hai chỉ số (i<m và j<n) ta chọn min(A[i], B[j]) và lưu nó vào trong C[k]. Chuyển sang Bước 4.

**Bước 4:** Tăng giá trị k lên 1 và quay về Bước 2.

**Bước 5:** Sao chép tất cả các giá trị còn lại từ các danh sách mà chỉ số còn vi phạm (tức i<m hoặc j<n) vào trong mảng C.

**2.3. Demo tuần tự**

Ví dụ: Sắp xếp mảng a[ 38, 27, 43, 3, 9, 82, 10]



#include "stdafx.h"

#include <stdio.h>

#include <stdlib.h>

#include <omp.h>

#include <iostream>

#include <iomanip>

#include <time.h>

#include <conio.h>

#include <Windows.h>

int n;

void simple\_merge(int a[], int low, int mid, int high);

void merge\_sort(int a[], int low, int high);

int main()

{

int a[10], n = 10;

double start\_time, end\_time, val;

//printf\_s("\n nhap kich thuoc cua phan tu mang:");

// scanf\_s("%d", &n);

for (int i = 0; i < n; i++)

{

a[i] = rand() % 10 + 1;

//printf\_s("Nhap a[%d] = ", i);

//scanf\_s("%d", &a[i]);

}

printf\_s("\n mang truoc khi sap xep la:\n");

printf\_s("\n");

for (int i = 0; i < n; i++)

printf\_s("%d\t", a[i]);

#pragma omp parallel for num\_threads(2)

for (int i = 0; i < n; i++)

{

printf\_s("\nLuong so %d, Thuc thi so %d...", omp\_get\_thread\_num(), a[i]);

}

start\_time = omp\_get\_wtime();

merge\_sort(a, 0, n);

end\_time = omp\_get\_wtime();

val = (end\_time - start\_time);

printf\_s("\n Mang sau khi sap xep song song : \n");

for (int i = 1; i <= n; i++)

printf\_s("%d\t", a[i]);

printf\_s("\n thoi gian sap xep la %f \n", val);

system("pause");

return 0;

}

void simple\_merge(int a[], int low, int mid, int high)

{

int \*temp; // Khoi tao mang tam de sap xep

int i = low; // Vi tri phan tu dau tien cua mang con ben trai

int j = mid + 1; // Vi tri phan tu dau tien cua mang con ben phai

temp = new int[high - low + 1]; // Khoi tao so luong phan tu cua mang tam

for (int k = 0; k <= high - low; k++)

{

// Kiem phan tu cua mang con ben trai va ben phai

if (a[i] < a[j])

{

// Neu a[i] < a[j] thi copy phan tu ben trai vao mang tam

temp[k] = a[i];

i++; // Vi tri phan tu tiep theo cua mang

}

else // Nguoc lai copy phan tu cua mang con ben phai vao mang tam

{

temp[k] = a[j];

j++; // Vi tri phan tu tiep theo cua mang

}

// Kiem tra mang con ben trai con phan tu nao khong

if (i == mid + 1)

{

// Nguoc lai dua cac phan tu con lai cua mang con ben phai vao mang tam

while (j <= high)

{

k++;

temp[k] = a[j];

j++;

}

break;

}

// Kiem tra mang con ben phai con phan tu nao khong

if (j == high + 1)

{

// Nguoc lai dua cac phan tu con lai cua mang con ben phai vao mang tam

while (i <= mid)

{

k++;

temp[k] = a[i];

i++;

}

break;

}

}

for (int k = 0; k <= high - low; k++) // Chep mang tam vao mang chinh

a[low + k] = temp[k];

delete temp;

}

void merge\_sort(int a[], int low, int high)

{

int mid;

if (low<high)

{

mid = (low + high) / 2;

#pragma omp parallel sections

{

#pragma omp section

merge\_sort(a, low, mid);

#pragma omp section

merge\_sort(a, mid + 1, high);

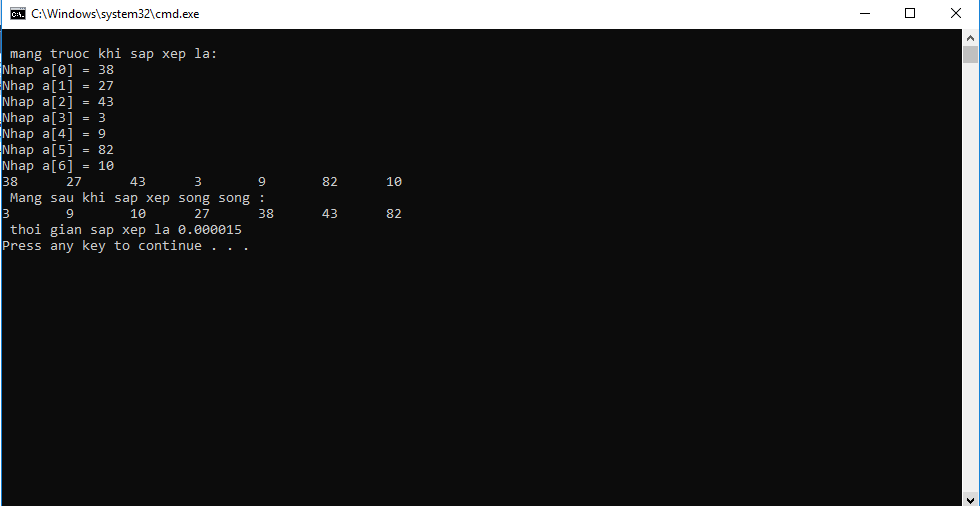
}

simple\_merge(a, low, mid, high);

}

}

* Kết quả:



# 

# CHƯƠNG 4: TÍNH TOÁN SONG SONG THUẬT TOÁN MERGE SORT

1. **Mô tả bài toán**

Cho 1 danh sách A chưa sắp xếp có q phần tử. Yêu cầu tính toán song song thực hiện trên hai bộ xử lý.

Thuật toán:

Việc tính toán song song sẽ tương tự như tính toán tuần tự nhưng tại bước 2 sau khi chia thành 2 mảng. Mỗi mảng sẽ được phân cho mỗi bộ xử lý và công việc tại mỗi bộ xử lý sẽ giống như tính toán tuần tự.

**Bước 1:** Chia mảng thành 2 mảng nhỏ m[i] và n[j]

**Bước 2:** Thực hiện với mảng m trước quay lại bước 1 cho đến khi mỗi mảng chỉ có một phần tử

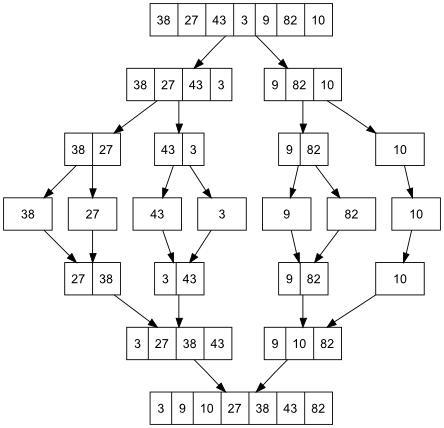
**Bước 3:** Tại mỗi bước nếu cả hai chỉ số (i<m và j<n) ta chọn min(A[i], B[j]) và lưu nó vào trong C[k]. Chuyển sang Bước 4.

**Bước 4:** Tăng giá trị k lên 1 và quay về Bước 2.

**Bước 5:** Sao chép tất cả các giá trị còn lại từ các danh sách mà chỉ số còn vi phạm (tức i<m hoặc j<n) vào trong mảng C.

1. **Demo song song**

Ví dụ: Sắp xếp mảng a[ 38, 27, 43, 3, 9, 82, 10]



//#include "stdafx.h"

#include <stdio.h>

#include <stdlib.h>

#include <omp.h>

#include <iostream>

#include <iomanip>

#include <time.h>

#include <conio.h>

#include <Windows.h>

int n;

void simple\_merge(int a[], int low, int mid, int high);

void merge\_sort(int a[], int low, int high);

int main()

{

int a[10], n = 10;

double start\_time, end\_time, val;

//printf\_s("\n nhap kich thuoc cua phan tu mang:");

// scanf\_s("%d", &n);

for (int i = 0; i < n; i++)

{

a[i] = rand() % 10 + 1;

//printf\_s("Nhap a[%d] = ", i);

//scanf\_s("%d", &a[i]);

}

printf\_s("\n mang truoc khi sap xep la:\n");

printf\_s("\n");

for (int i = 0; i < n; i++)

printf\_s("%d\t", a[i]);

start\_time = omp\_get\_wtime();

merge\_sort(a, 0, n);

end\_time = omp\_get\_wtime();

val = (end\_time - start\_time);

printf\_s("\n Mang sau khi sap xep song song : \n");

for (int i = 1; i <= n; i++)

printf\_s("%d\t", a[i]);

printf\_s("\n thoi gian sap xep la %f \n", val);

system("pause");

return 0;

}

void simple\_merge(int a[], int low, int mid, int high)

{

int \*temp; // Khoi tao mang tam de sap xep

int i = low; // Vi tri phan tu dau tien cua mang con ben trai

int j = mid + 1; // Vi tri phan tu dau tien cua mang con ben phai

temp = new int[high - low + 1]; // Khoi tao so luong phan tu cua mang tam

for (int k = 0; k <= high - low; k++)

{

// Kiem phan tu cua mang con ben trai va ben phai

if (a[i] < a[j])

{

// Neu a[i] < a[j] thi copy phan tu ben trai vao mang tam

temp[k] = a[i];

i++; // Vi tri phan tu tiep theo cua mang

}

else // Nguoc lai copy phan tu cua mang con ben phai vao mang tam

{

temp[k] = a[j];

j++; // Vi tri phan tu tiep theo cua mang

}

// Kiem tra mang con ben trai con phan tu nao khong

if (i == mid + 1)

{

// Nguoc lai dua cac phan tu con lai cua mang con ben phai vao mang tam

while (j <= high)

{

k++;

temp[k] = a[j];

j++;

}

break;

}

// Kiem tra mang con ben phai con phan tu nao khong

if (j == high + 1)

{

// Nguoc lai dua cac phan tu con lai cua mang con ben phai vao mang tam

while (i <= mid)

{

k++;

temp[k] = a[i];

i++;

}

break;

}

}

for (int k = 0; k <= high - low; k++) // Chep mang tam vao mang chinh

a[low + k] = temp[k];

delete temp;

}

void merge\_sort(int a[], int low, int high)

{

int mid;

if (low<high)

{

mid = (low + high) / 2;

merge\_sort(a, low, mid);

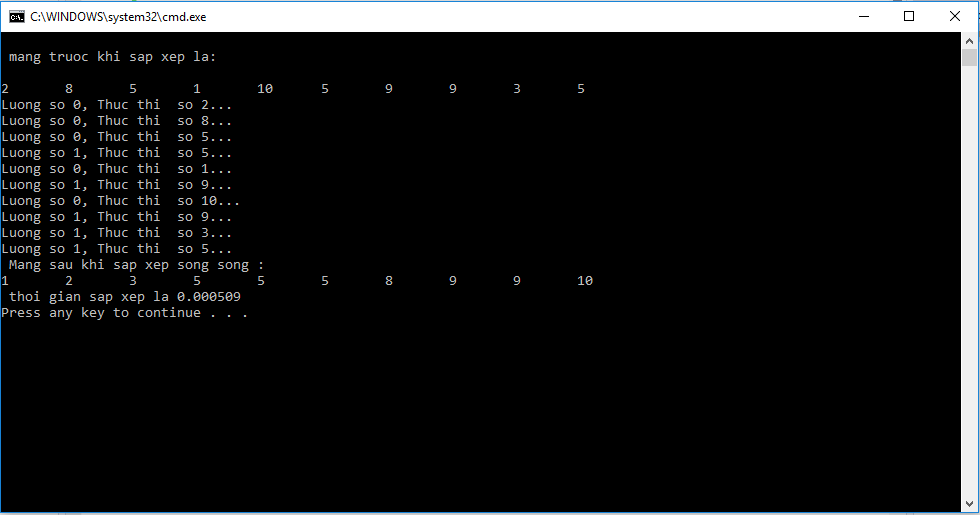
merge\_sort(a, mid + 1, high);

simple\_merge(a, low, mid, high);

}

}

* Kết quả:



* Dựa vào kết quả trên, ta có thể thấy sự tối ưu của giải thuật song song để giải quyết các bài toán lớn. Với sự tham gia của nhiều bộ xử lý, kết hợp với giải thuật tính toán song song tối ưu sẽ mang lại hiệu quả tính toán cao nhất.

**Kết luận**

Bộ môn giải thuật và tính toán song song trong Công nghệ thông tin là một chuyên ngành đang phát triển mạnh mẽ trong nhiều năm gần đây. Bằng sự hiệu quả của tính toán song song đã giúp cho chúng ta giải quyết một cách thành công các bài toán khó trong nhiều ngành khoa học.

Khi tăng tốc độ cho CPU ngày càng khó khăn, thì việc xử lý song song có nhiều lợi thế: Tận dụng được tốc độ của CPU, tận dụng thời gian để thực hiện bài toán thực tế (giải quyết cùng một lúc nhiều bài toán). Xử lý song song có các lợi thế: Thứ nhất: bản chất các luồng điều khiển độc lập nhau về lý luận. Thứ hai, song hành đem đến sự cải thiện hiệu suất, hoặc nhờ tận dụng nhiều CPU hoặc thời gian “chết” của các tiến trình trong ứng dụng.

Bên cạnh những lợi thế trên, thì xử lý song song còn có những nhược điểm sau: Trước hết, không phải tất cả ứng dụng đều có thể thực hiện song song. Và trở ngại lớn nhất là việc lập trình song song khó hơn nhiều so với lập trình tuần tự quen thuộc trước đây. Đa số lập trình viên hiện nay không am hiểu kỹ thuật song song, cũng như cách đây 15 năm đa số lập trình viên không am hiểu kỹ thuật hướng đối tượng. Tuy nhiên, mọi thứ đều có thể học được.

**Tài liệu tham khảo**

1. Giáo trình Xử lý song song và phân tán

1. <https://www.stdio.vn/articles/merge-sort-184>
2. <https://parcomp.wordpress.com/2017/02/26/parallel-merge-sort/>