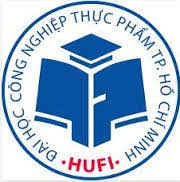


**BỘ CÔNG THƯƠNG**

**TRƯỜNG ĐẠI HỌC CÔNG NGHIỆP THỰC PHẨM TP.HCM**

**KHOA CÔNG NGHỆ THÔNG TIN**

🙜🙢🙠🙞



**ĐỒ ÁN XỬ LÝ TÍNH TOÁN SONG SONG**

**ĐỀ TÀI: Tìm hiểu thuật toán sắp xếp theo hạng, thuật toán so sánh và đổi chỗ**

Sinh viên thực hiện:

1. 2001170118 – *Mai Thành Nhân*

2. 2001170181 *– Phạm Đức Thông*

3. 2001170202 – *Khưu Minh Triết*

1. Thuật toán sắp xếp theo hạng (rank sort):

* Xét một dãy hỗn độn gồm n phần tử a*1*, a*2*, …, a*n*.
* Vị trí của một phần tử ai trong dãy các phần tử đã sắp xếp được xác định bởi số các phần tử bé hơn nó.
* Ví dụ: có ci phần tử bé hơn ai thì phần tử ai có số thứ tự là ci + 1.
* Nếu có hai hay nhiều phần tử bằng nhau thì thứ tự sắp xếp dựa trên chỉ số của các phần tử đó. Phần tử có chỉ số lớn hơn được xem là có giá trị lớn hơn.
* Ví dụ: giả sử ta có hai số ai = aj, nếu i > j thì ta xem như ai > aj.
* Dưới đây là đoạn mã chương trình tuần tự:

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

{

x = 0;

for (j = 0; j < n; j++)

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

x++;

b[x] = a[i];

}

+ Đoạn chương trình trên có 2 vòng lặp for chạy từ 0 -> n - 1, theo quy tắc nhân thì đoạn chương trình trên có độ phức tạp là: O(n) \* O(n) = O(n2).

* Đối với đoạn mã chương trình song song:

Giả sử chúng ta có n bộ xử lý (processor) và mỗi processor được cấp 1 số trong dãy cho trước. Dưới đây là đoạn mã chượng trình song song sử dụng n processor:

forall (i = 0; i < n; i++)

{

x = 0;

for (j = 0; j < n; j++)

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

x++;

b[x] = a[i];

}

+ Ta thấy sự khác biệt giữa đoạn chương trình tuần tự và song song nằm ở vòng for đầu tiên, ở đây chúng ta có forall là vòng lặp biểu thị cho tính toán song song. Ở đây độ phức tạp của thuật toán chỉ là O(n) vì forall thực hiện tính toán song song nên độ phức tạp của nó chỉ là O(1), riêng vòng lặp for với j chạy từ 0 -> n – 1 nên độ phức tạp là O(n). Theo quy tắc nhân, độ phức tạp của thuật toán trên là O(1) \* O(n) = O(n).

* Thuật toán sử dụng mô hình PRAM kiểu CRCW (đọc đồng thời, ghi đồng thời) được mô tả như sau:
* Thuật toán sử dụng n2 processor phân bố thành mảng 2 chiều P*i, j* (i là hàng, j là cột).
* Thuật toán còn sử dụng 2 mảng 1 chiều là A[1 . . *n*] và C[1 . . *n*]:
* Mảng A[1 . . *n*] dùng để lưu kết quả trên bộ nhớ dùng chung của dãy các phần tử sẽ được sắp xếp.
* Mảng C[1 . . *n*] dùng để lưu số các phần tử bé hơn cho mỗi một phần tử trong mảng A[1 . . *n*], tức là thứ hạng của các số trong mảng A[1 . . *n*].
* Thuật toán trên được chia thành 2 bước:

1. Mỗi hàng processors thứ i tính giá trị C[i] là số các phần tử bé hơn A[i]. Mỗi processor Pi, j so sánh A[i] và A[j], rồi cập nhật giá trị của C[i].
2. Processor đầu tiên trong mỗi hàng Pi, 1, đưa A[i] vào vị trí của nó trong dãy được sắp xếp, (vị trí C[i] + 1).

* Dưới đây là đoạn mã thể hiện 2 bước trên:

+ Bước 1:

forall Pi,j, where 1 ≤ i, j ≤ n do in parallel

if A[i] > A[j] or (A[i] = A[j] and i > j) then

C[i] ← 1

else

C[i] ← 0

endif

endfor

+ Bước 2:

forall Pi, j, where 1 ≤ i ≤ n do in parallel

A[C[i] + 1] ← A[i]

endfor

* Ví dụ:

|  |  |  |
| --- | --- | --- |
| 8 | 4 | 10 |

+ Cho mảng A =

+ Sắp xếp theo hạng được thực hiện bằng cách so sánh từng phần tử trong mảng A từ P1, 1, P1,2 . . . P3, 3.

Sau khi so sánh từng phần tử trong mảng A, các thứ hạng của từng phần tử được gán vào trong mảng C như sau:

|  |  |  |
| --- | --- | --- |
| 1 | 0 | 2 |

C =

Sau khi đã có thứ hạng cho các phần tử trong mảng C, tiến hành sắp xếp lại mảng A theo thứ tự tăng dần trong mảng C và ta được kết quả:

|  |  |  |
| --- | --- | --- |
| 4 | 8 | 10 |

A =

1. Thuật toán sắp xếp so sánh và đổi chỗ:

Đây là 1 trong những thuật toán sắp xếp đơn giản nhất, thuật toán thực hiện việc so sánh 2 phần tử liền kề nhau và nếu chúng chưa theo thứ tự yêu cầu của bài toán thì tiến hành đổi chỗ vị trí của chúng với nhau. Quá trình trên đươc lặp cho đến khi tất cả các cặp đều ở vị trí thỏa mãn điều kiện cho trước.

* Đoạn mã chương trình tuần tự sắp xếp mảng giảm dần:

for (i = n – l; i > 0; i--)

for (j = 0; j < l; j++)

{

k = j + 1;

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

{

t = a[j];

a[j] = a[k];

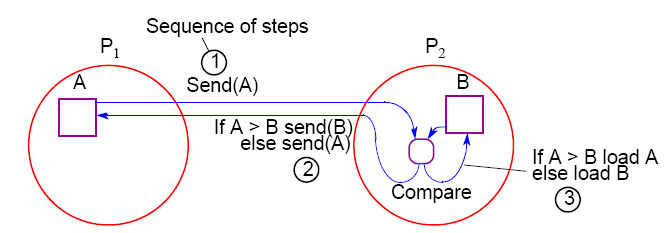
a[k] = t;

}

}

Đối với việc tính toán song song, chúng ta sử dụng mô hình IPC message passing để giao tiếp giữa các tiến trình với nhau, có 2 kiểu tính toán, mỗi kiểu bao gồm 2 chu trình P1, P2:

1. Tính toán song song bất đối xứng:



+ Chu trình P1 gửi số A cho P2. Khi đó P2 so sánh giá trị A với B, nếu A > B thì gửi B về chu trình P1, ngược lại gửi chu trình A.

+ Mã chương trình:

* Trên P1: send (&A, P2);

recv (&A, P2);

* Trên P2: recv (&A, P1);

if (A > B)

{

send (&B, P1);

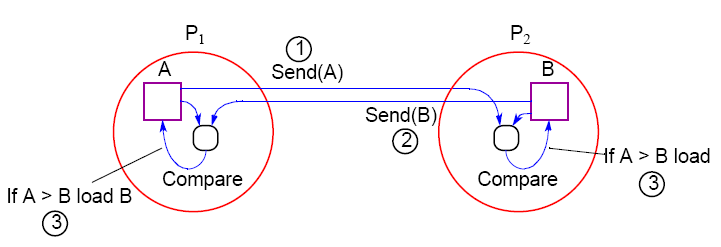
B = A;

}

else

send (&A, P1);

1. Tính toán song song đối xứng:



+ Mỗi chu trình gửi số của nó đến 1 chu trình khác. Tại đây, mỗi chu trình đều so tiến hành so sánh giá trị A với B. Trên P1, nếu A > B thì gán A = B, ngược lại trên P2 nếu A > B thì gán B = A.

+ Mã chương trình:

* Trên P1: send (&A, P2);

recv (&B, P2);

if (A > B) A = B;

* Trên P2: recv (&A, P1);

send (&B, P1);

if (A > B) B = A;

* Có 1 hạn chế là kiểu tính toán như trên dễ bị hiện tượng deadlock.