Lecture 5 : Một số thuật toán song song

1. Thuật toán xử lý mảng song song
2. Thuật toán tính số PI song song
3. Thuật toán tìm dãy con chung dài nhất song song
4. Thuật toán nhân ma trận

1. Xử lý mảng

– Ví dụ: Cần tính toán các phần tử của một mảng 2 chiều theo hàm fcn(x,y): x là cột, y là dòng; Biết rằng các phần tử được tính một cách độc lập.

–
```
for (i = 1,n)
    for ( j = 1,n)
        a(i,j) = fcn(i,j)
```

1. Xử lý mảng – Song song theo dữ liệu

- Chia mảng thành các khối nhỏ. Các phân tử trong các khối sẽ được tính toán độc lập trên các tiến trình khác nhau.
- **Tiến trình Pi sẽ thực hiện như sau:**
  - **for (i = mystart-i, myend-i)**
    for (j = 1,n)
    a(i,j) = fcn(i,j)

1. Xử lý mảng – Song song theo mô hình Master/Slaves

```
if I am MASTER
  do until no more jobs
     send to WORKER next job
     receive results from WORKER
  end do
  tell WORKER no more jobselse
endif
if I am WORKER
    do until no more jobs
      receive from MASTER next job
      calculate array element: a(i,j) = fcn(i,j)
      send results to MASTER
     end do
endif
```

2. Tính số PI

Xem xét phương pháp tính số PI sau đây:

- Vẽ một hình tròn nội tiếp hình vuông.
- Sinh ra các điểm ngẫu nhiên bên trong hình vuông.
- Xác định số điểm trong hình vuông mà cũng nằm trong hình tròn.
- Cho **r** là số điểm trong hình tròn chia cho số điểm trong hình vuông
- PI ~ 4 r
- Chú ý rằng, càng nhiều điểm ngẫu nhiên được sinh ra thì phép tính xấp xỉ số PI sẽ càng chính xác hơn.

2. Tính số PI

```
#include <stdio.h>
#include <stdlib.h>
void main()
{
float x,y;
unsigned int k,n, c=0;

n =20000;
for (k=0; k<=n;k++)
{
     x= (float) rand()/RAND_MAX;
     y= (float) rand()/RAND_MAX;
    if (x*x+y*y<=1) c++;
}
printf("%5.3f",(float) 4*c/n);
}
```

2. Tính số PI bằng thuật toán song song

- Chiến lược trong lập trình song song là bẻ gãy vòng lặp thành nhiều phần mà có thể được thực thi bởi các tác vụ.
- Đối với tác vụ tính toán xấp xỉ số Pi :
  - Mỗi tác vụ thực thi một phần vòng lặp.
  - Mỗi tác vụ có thể thực hiện phần việc của nó mà không yêu cần thêm bất kỳ thông tin nào từ các tác vụ khác (không phụ thuộc dữ liệu)
  - Sử dụng mô hình “Master-Slaver". Một tác vụ sẽ là tác vụ chủ, được sử dụng để tổng hợp các kết quả.

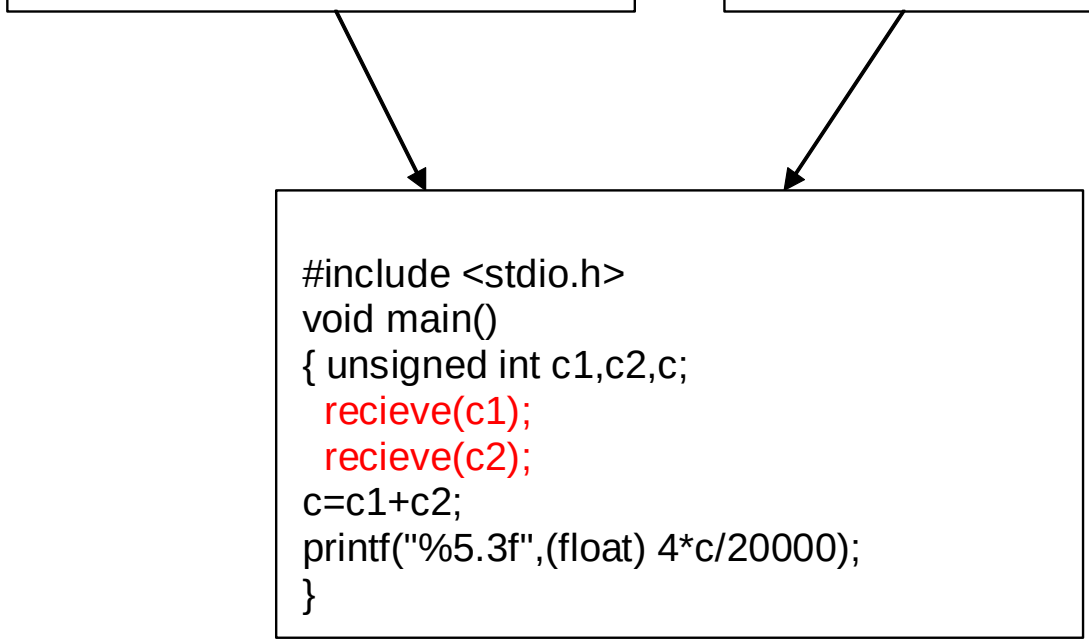
2. Tính số PI bằng thuật toán song song

```
#include <stdio.h>
#include <stdlib.h>
void main()
{
float x,y;
unsigned int k,n, c1=0;

n =10000;
for (k=0; k<=n;k++)
{
     x= (float) rand()/RAND_MAX;
     y= (float) rand()/RAND_MAX;
    if (x*x+y*y<=1) c1++;
}
send(c1);
}
```

```
#include <stdio.h>
#include <stdlib.h>
void main()
{
float x,y;
unsigned int k,n, c2=0;

n =10000;
for (k=0; k<=n;k++)
{
     x= (float) rand()/RAND_MAX;
     y= (float) rand()/RAND_MAX;
    if (x*x+y*y<=1) c2++;
}
send(c2);
}
```



```
#include <stdio.h>
void main()
{ unsigned int c1,c2,c;
  recieve(c1);
  recieve(c2);
c=c1+c2;
printf("%5.3f",(float) 4*c/20000);
}
```

3. Dãy con chung dài nhất

1. Giới thiệu bài toán dãy con chung dài nhất
2. Phân tích bài toán LCS
3. Thuật toán tuần tự
4. Song song hoá thuật toán
5. Cài đặt chương trình (pthread)

GIỚI THIỆU BÀI TOÁN

Bài toán dãy con chung dài nhất được trình bày như sau: Cho hai dãy $A=<a_1, a_2,\ldots, a_n>$ và $B=<b_1, b_2,\ldots, b_n>$ tìm một dãy dài nhất sao cho nó là dãy con của cả A và B. Cho ví dụ nếu A = <c, a, d, b, r, z > và B = <a, s, b, z>, dãy con chung dài nhất của A và B là <a, b, z>.

Định lý cấu trúc con tối ưu của một LCS

- Cho dãy $X = <x_1, x_2,..., x_n>$, ta định nghĩa tiền tố thứ i của X, i=1, 2,...,n ta có dãy $X_i = < x_1, x_2,..., x_i>$

Định lý: Cho $X = <x_1, x_2,..., x_m>$ và $Y = <y_1, y_2,..., y_n>$ là các dãy, cho $Z = <z_1, z_2,..., z_k>$ là một LCS bất kỳ của X và Y.

a. Nếu $x_m = y_n$ thì $z_k = x_m = y_n$ và Z_{k-1} là một LCS của X_{m-1} và Y_{n-1}

b. Nếu $x_m \# y_n$ thì $z_k \# x_m$ hàm ý Z là một LCS của X_{m-1} và Y

c. Nếu $x_m \# y_n$ thì $z_k \# y_n$ hàm ý Z là một LCS của X và Y_{n-1}

Định lý (tiếp)

Từ định lý trên ta thấy: để tìm LCS của X và Y ta phải xét:

- Nếu $x_m = y_n$ thì ta phải tìm một LCS của X_{m-1} và Y_{n-1} sau đó chắp $x_m = y_n$ vào LCS này cho ta LCS của X và Y.

- Nếu $x_m \# y_n$ ta phải tìm một LCS của X_{m-1} và Y và một LCS của X và Y_{n-1}. Cái nào lớn hơn thì là LCS của X và Y.

Ta có công thức đệ quy sau: (Với F(i, j) là chiều dài của một LCS của các dãy X_i và Y_j)

$$F[i, j] = \begin{cases} 0 & \text{if } i = 0 \text{ or } j = 0 \\ F[i-1, j-1] + 1 & \text{if } i, j > 0 \text{ and } x_i = y_j \\ \max\{F[i, j-1], F[i-1, j]\} & \text{if } i, j > 0 \text{ and } x_i \neq y_j \end{cases}$$

Thuật toán tuần tự

```
LCS_Length (X, Y)
        m ← length [X];
        n ← length [Y];
        for (i = 0; i <= m; ++i)
                c[i][0] = 0;
        for (j = 0; j <= m; ++j)
                c[0][j] = 0;
```

Thuật toán tuần tự (continued)

```
for (i = 1; i <= n; ++i)
        for (j = 1; j <= m; ++j)
                if (X[i] == Y[j] )
                        c[i][j] = c[i-1][j-1] + 1; b[i][j] = "↖";
                        else if (c[i-1][j] >= c[i][j-1] )
                                c[i][j] = c[i-1][j]; b[i][j] = "↑";
                        else    c[i][j] = c[i][j-1]; b[i][j] = "←";
```

return b và c;

Thời gian thực hiện của thủ tục này là O(mn), bởi mỗi khoản nhập bảng mất O(1) thời gian để tính toán.

Thuật toán tuần tự (continued)

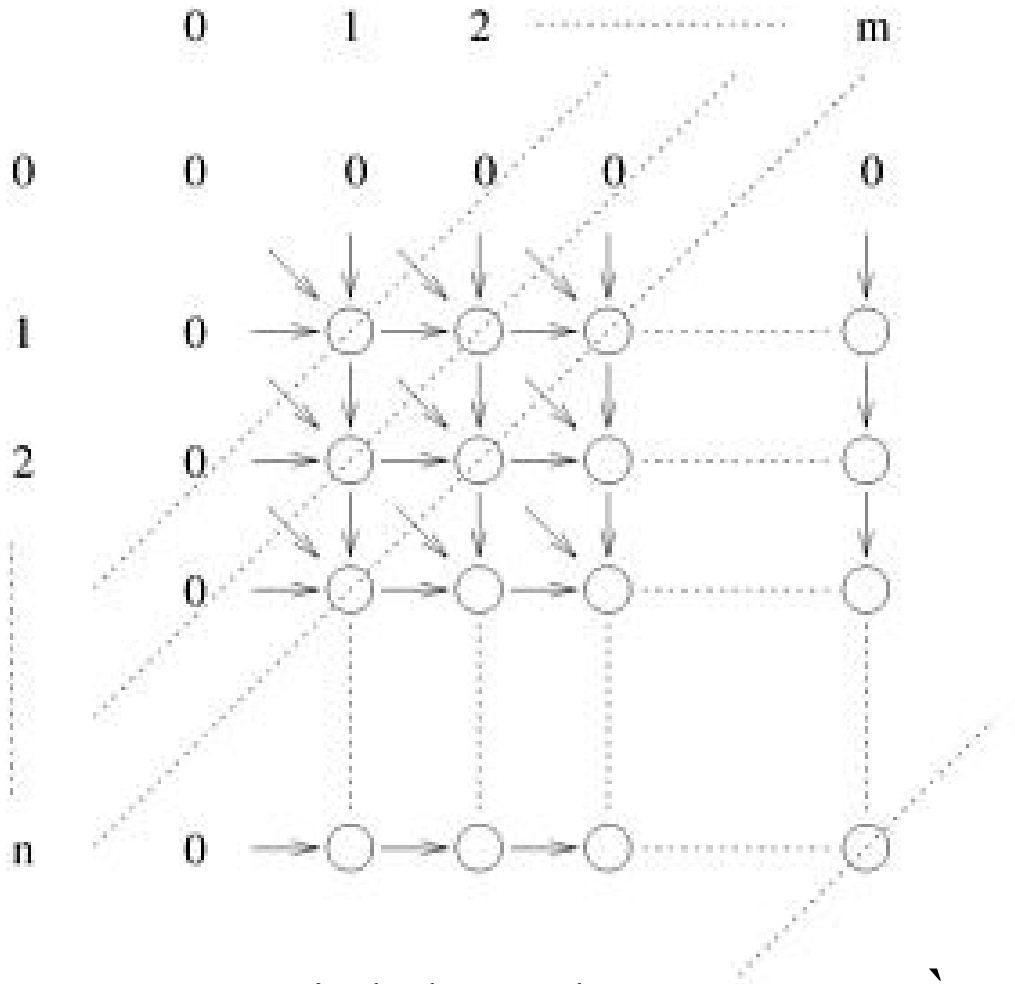
Xác định một LCS:

Bảng b mà LCS_Length trả về có thể xác định một LCS của X và Y. Ta bắt đầu tại b[m,n] và rà qua bảng theo các mũi tên. Khi gặp b[m,n]="↖" trong b[i,j], nó hàm ý rằng $x_i=y_j$ là một thành phần của LCS. Phương pháp này gặp các thành phần của LCS theo thứ tự đảo ngược. Thủ tục đệ quy dưới đây in ra một LCS của X và Y.

```
PRINT_LCS(b,X,i,j)
        if i=0 or j=0 then return;
        if b[i.j] = "↖" then PRINT_LCS(b,X,i-1,j-1);
                print x_i;
        else if b[i.j] = "↑" then PRINT_LCS(b,X,i-1,j);
        else PRINT_LCS(b,X,i,j-1);
```

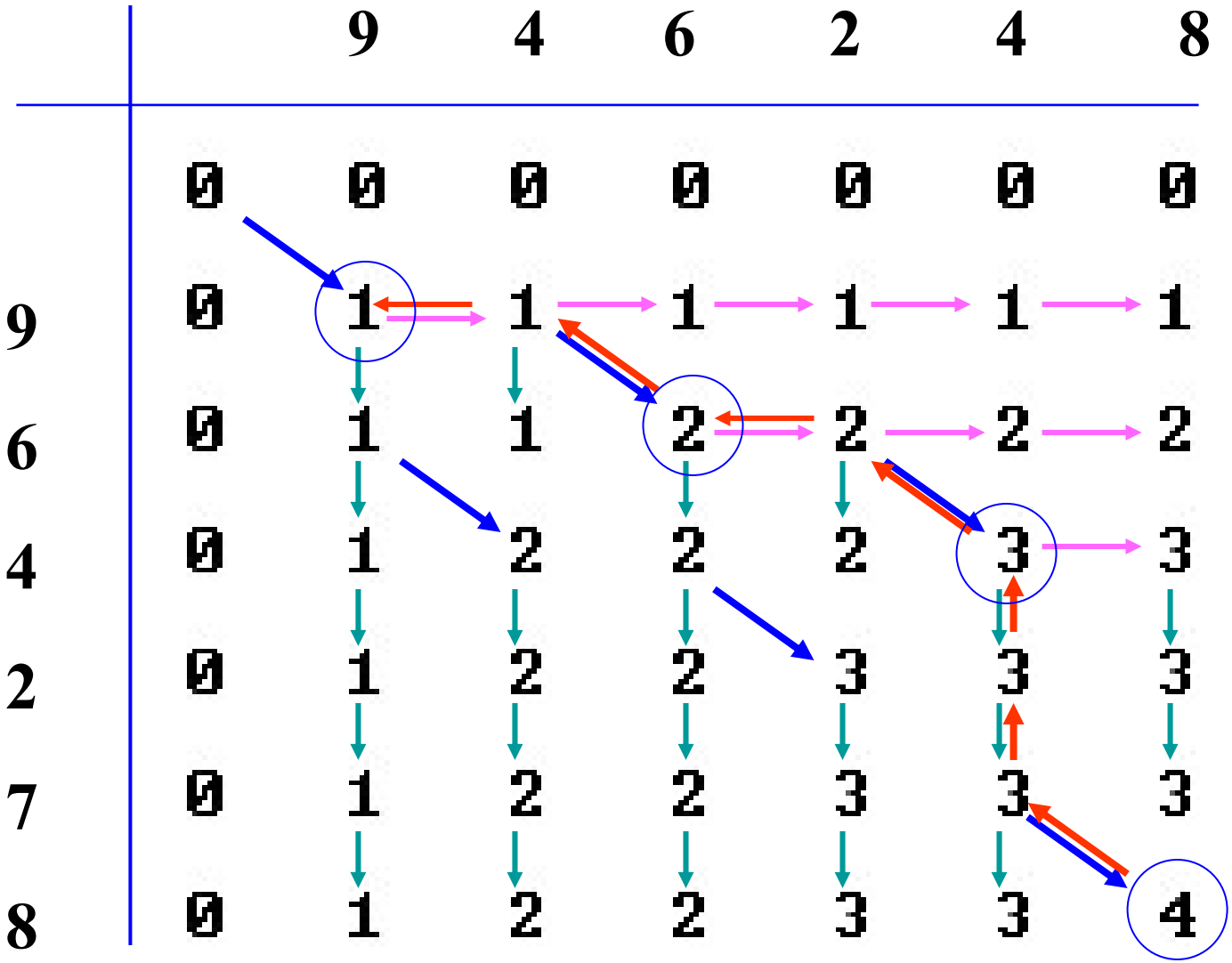
Thủ tục này mất một thời gian O(m+n), bởi ít nhất một trong số i và j được giảm lượng trong mỗi giai đoạn của đệ quy.

Thuật toán tuần tự (continued)

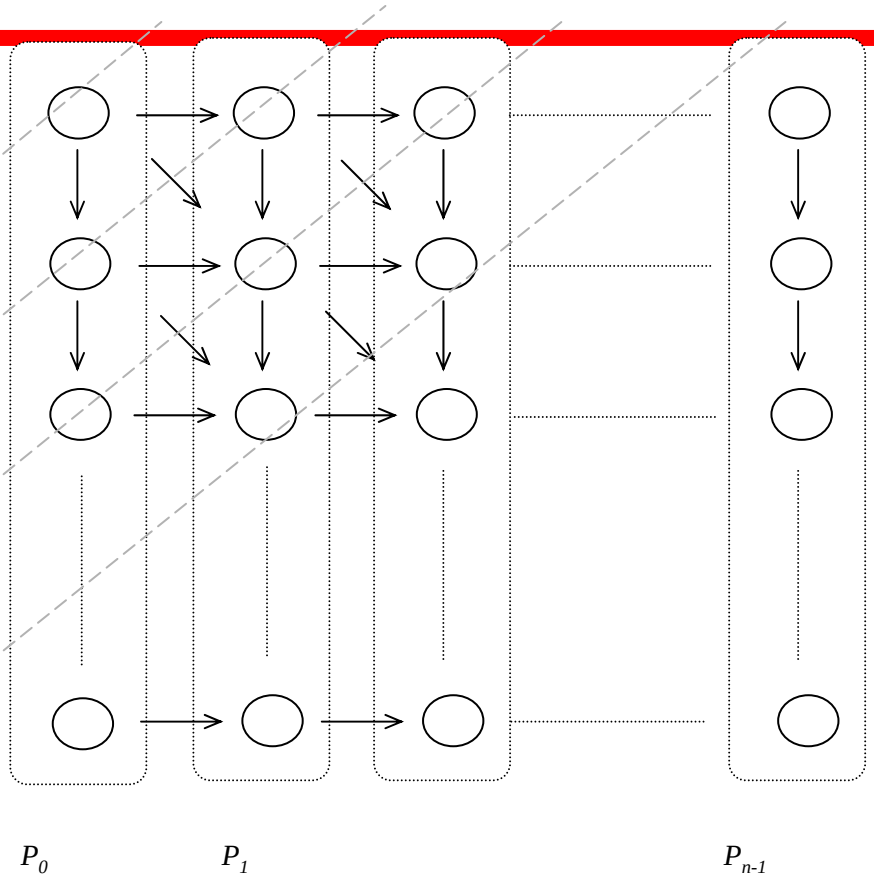


Bảng minh họa thuật toán tuần tự:

Thuật toán tuần tự (continue)



Song song hóa thuật toán



Để đơn giản hóa vấn đề chúng ta xét trường hợp n=m, tính toán song song với n phần tử xử lý. Mỗi phần tử đang xử lý P_i tính toán cột thứ i của bảng F. Như vậy, công thức song song chạy trong thời gian $\Theta(n)$.

- Bảng được tính quét theo đường chéo từ góc trên bên trái đến góc dưới bên phải. Các phần tử trên cùng một đường chéo được tính toán song song

Song song hoá thuật toán (continued)

- Khi tính F[i, j] Process P_{j-1} cần có giá trị của F[i-1, j-1] hoặc F[i, j-1] của P_{j-2}. Nó mất khoảng thời gian t_s+t_w để truyền thông, t_c- thời gian tính toán. Mỗi P_i tính toán một phần tử trên đường chéo, có 2n-1 đường chéo. Tổng thời gian chạy song song là $(2n-1)(t_c+ t_s+ t_w)$. Thời gian chạy tuần tự là n^2t_c.