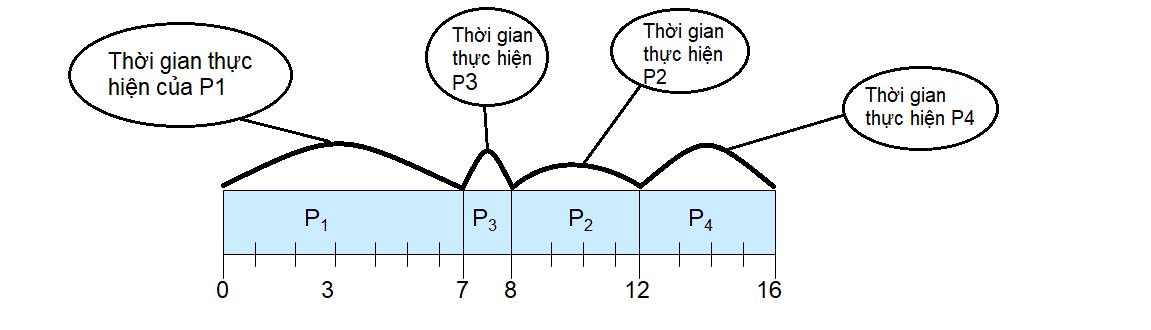
|  |  |  |
| --- | --- | --- |
| Tiến Trình (Process) | Thời gian xuất hiện (Arrival Time) | Thời gian thực hiện (Burst Time) |
| P1 | 0 | 7 |
| P2 | 2 | 4 |
| P3 | 4 | 1 |
| P4 | 5 | 4 |

**Lý Thuyết**

Tiến trình sẽ thực hiện từ thời gian xuất hiện nhỏ nhất. Nếu thời gian xuất hiện của 2 tiến trình bằng nhau thì ta sẽ so sánh thời gian thực hiện của 2 tiến trình đó, tiến trình nào có thời gian thực hiện nhỏ hơn thì tiến trình đó thực hiện trc.



Từ đây tính được thời gian chờ của mỗi tiến trình

P1 thực hiện đầu tiên nên thời gian chờ = 0  
P3 phải chờ P1 thực hiện xong mới được thực hiện nên thời gian chờ = thời gian cuối của tiến trình vừa kết thúc – thời gian xuất hiện của P3

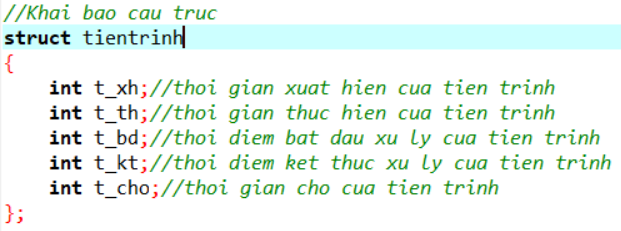
VD: thời gian chờ P3 = 7 – 4 = 3

Tương tự P2 phải đợi P1 P3 thực hiện xong mới có thể thực hiện nên thời gian chờ của P2 = 8 – 2 = 6   
Tương tự thời gian chờ P4 = 12 - 5 = 7

Thời gian chờ trung bình sẽ là trung bình cộng thời gian chờ của P1 P2 P3 P4 = (0+3+6+7)/4=4

**Ý Tưởng Giải Thuật**

* Tạo 1 kiểu cấu trúc **tiến trình**, để lưu trữ các thông tin của tiến trình đó gồm( Tgian xuất hiện, Tgian thực hiện, Tđiểm bắt đầu xử lý của tiến trình, Tđiểm kết thúc xử lý của tiến trình, Tgian Chờ của tiến trình).



* Tạo 1 mảng với kiểu cấu trúc **tiến trình** để lưu trữ các tiến trình. Tạo thêm một biến cục bộ để lưu trữ thời gian cuối cùng của tiến trình vừa kết thúc.
* Dùng lặp for để nhập và in dữ liệu của **thời gian thực hiện(Arrival Time)** và **thời gian thực hiện(Burst Time)**

**for(int i=0; i<num; i++)**

**{**

**Cout<<”Nhap du lieu tien trinh thu ”<<i+1;**

**cout<<"Nhap thoi gian xuat hien: ";**

**cin>>a[i].t\_xh;**

**cout<<"Nhap thoi gian thuc hien: ";**

**cin>>a[i].t\_th;**

**sum+=a[i].t\_th;**

**}**

* Tạo hàm tính TBC thời gian chờ

Gọi a[] là mảng chứa các tiến trình, n là số lượng tiến trình

**Function SJF(tientrinh a[], int n)**

**{**

*B1. Tạo biết để lưu thời gian cuối cùng của tiến trình vừa kết thúc*

**Int timeout;**

*B2. Xử lý trong vòng lặp với số lượng lặp = số tiến trình*

**For(i=0; i<n; i++)**

**{**

*B2.1 nếu thời gian xuất hiện của tiến trình vừa nạp >= thời gian kết thúc của tiến trình trước thi phải chọn tiến trình có thời gian XUAT HIEN nhỏ nhất tiếp theo*

**if(a[i].t\_xh >= timeout**)

**{**

B2.2. Sắp xếp lại các tiến trình theo thứ tự, tiến trình có tgian xuất hiện ngắn nhất thì đứng trước. Nếu có cùng tgian xuất hiện thì so sánh tiếp tgian thực hiện.

**}**

**else**

**{**

B2.2. Sắp xếp lại các tiến trình còn lại. Nếu (tgian thực hiện của tiến trình đứng sau < đứng trước) và tgian xuất hiện của tiến trình đứng sau <= **timeout** , hoặc có 2 tgian thực hiện bằng nhau nhưng tgian xuất hiện của đứng sau < đứng trước.

**}**

*B2.3. Xử lý và tính tgian chờ*

**if (i == 0***)//xử lý với tiến trình đầu tiên.(tgian xuất hiện bé nhất)*

**{**

**a[i].t\_bd = a[i].t\_xh;//tiến trình đầu tiên thì t\_bd =t\_xh**

**a[i].t\_kt = a[i].t\_bd + a[i].t\_th;**

**}**

**else** *//Ngược lại*

**{**

**if(a[i].t\_xh <= a[i - 1].t\_kt***)//nếu tiến trình sau xuất hiện trước khi tiến trình đang thực hiện kết thúc*

**a[i].t\_bd = a[i - 1].t\_kt;**

**else***//nếu tiến trình đang thực hiện mà vẫn chưa có tiến trình mới xuất hiện*

**a[i].t\_bd = a[i].t\_xh;**

*//Tính thời gian kết thúc*

**a[i].t\_kt = a[i].t\_bd + a[i].t\_th;**

**}**

*//Tính thời gian chờ*

**a[i].t\_cho=a[i].t\_bd-a[i].t\_xh;**

**timeout = a[i].t\_kt;** *//Gán giá trị kết thúc*

**}**

**}**

* **Nếu vẫn chưa hiểu nên đọc lại ý tưởng kết hợp với đọc code bài SJF\_khong\_uu\_tien**
* Sau khi kết thúc giải thuật trên ta sẽ được các thời gian chờ của từng tiến trính. Áp dung công thức ta sẽ được:
  + **Ttbc = ( Tổng thời gian chờ của các tiến trình)/số lượng tiến trình**