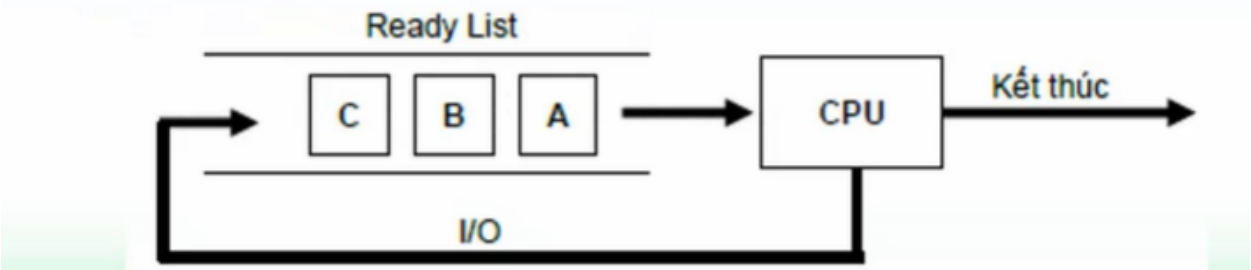
TÌM HIỂU VỀ CHIẾN LƯỢC FCFS

NỘI DUNG

1. Ý tưởng Đã hoàn tất

Thuật toán FCFS hoạt động dựa trên nguyên tắc đơn giản: **Ai đến trước được phục vụ trước**. Nghĩa là, tiến trình nào đến trước sẽ được CPU xử lý trước, bất kể thời gian thực thi của tiến trình đó là bao nhiêu.



Hình: Chiến lược FIFO

2. Các bước thực hiện Đã hoàn tất

Tạo một hàng đợi: Các tiến trình đến sẽ được đưa vào hàng đợi theo thứ tự đến.

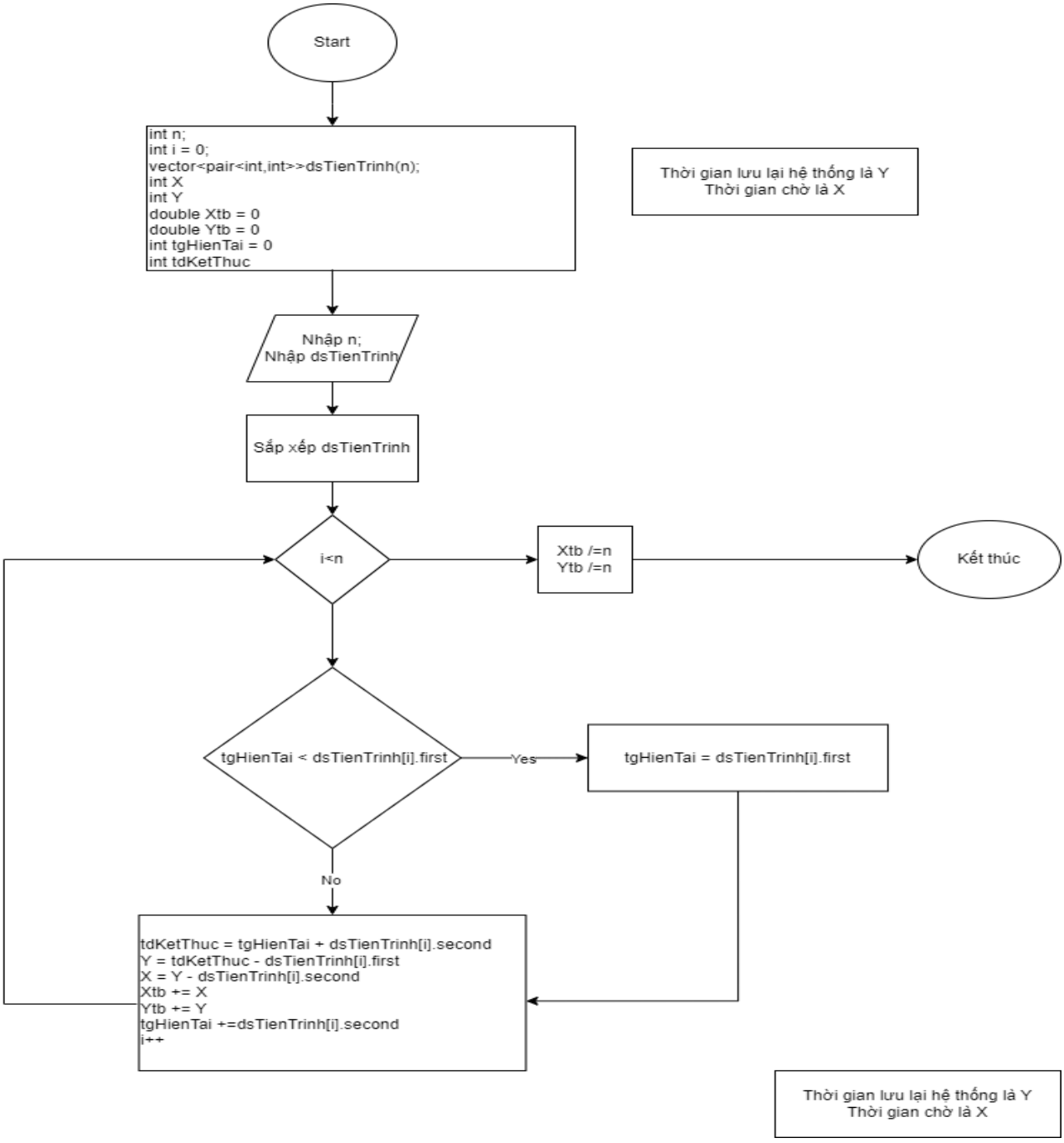
Lấy tiến trình đầu tiên: Khi CPU rảnh, tiến trình ở đầu hàng đợi sẽ được chọn để thực thi.

Thực thi: Tiến trình được chọn sẽ được thực thi cho đến khi hoàn thành hoặc bị gián đoạn bởi một sự kiện khác (như ngắt).

Loại bỏ tiến trình: Sau khi hoàn thành hoặc bị gián đoạn, tiến trình sẽ được loại bỏ khỏi hàng đợi.

Lặp lại: Quá trình này sẽ được lặp lại cho đến khi tất cả các tiến trình đều được xử lý xong.

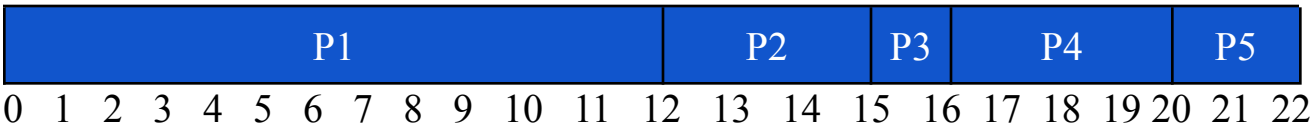
3. Lưu đồ thuật toán Đã hoàn tất



4. Ví dụ minh hoạ Đã hoàn tất

Tiến trình (Process)	Thời điểm xuất hiện (Arrival Time)	Thời gian sử dụng CPU (Burst Time)
P1	**0**	**12**
P2	**1**	**3**
P3	**3**	**1**
P4	**6**	**4**
P5	**10**	**2**

Bước 1: Vẽ biểu đồ Gantt



0: P1 vào RQ P1 dùng CPU 1: P2 vào RQ 3: P3 vào RQ 6: P4 vào RQ 10: P5 vào RQ	12: P1 kết thúc P2 dùng CPU 15: P2 kết thúc P3 dùng CPU 16: P3 kết thúc P4 dùng CPU 20: P4 kết thúc P5 dùng CPU 22: P5 kết thúc

Bước 2: Tính thời gian lưu lại hệ thống và thời gian chờ của mỗi tiến trình

Thời gian lưu lại hệ thống của 1 tiến trình = Thời điểm tiến trình kết thúc sử dụng CPU - Thời điểm xuất hiện

Thời gian chờ của 1 tiến trình = Thời gian lưu lại hệ thống của tiến trình - Thời gian tiến trình sử dụng CPU

Tiến trình	Thời gian lưu lại hệ thống	Thời gian chờ
P1	12 - 0 = 12	0
P2	15 - 1 = 14	12 - 1 = 11
P3	16 - 3 = 13	15 - 3 = 12
P4	20 - 6 = 14	16 - 6 = 10
P5	22 - 10 = 12	20 - 10 = 10

Bước 3: Tính thời gian chờ trung bình

Kết luận: Thời gian chờ trung bình : $\frac{(0 + 11 + 12 + 10 + 10)}{5} = 8.6$

5. Mô tả source code Đã hoàn tất

```
/*
    N      N         A      N   N  H    H
    NN     N        A A     NN  N  H    H
    N  N   N       AAAAA    N N N  HHHHH
    N    NN      AA    AA   N  NN  H    H
    N     N ** AA      AA N    N  H    H
*/

#include<bits/stdc++.h>
using namespace std;

#define REP(i , l, r) for(int i=l; i<=r; i++)
```

```
#define REPD(i , l, r) for(int i=l;i>=r;i--)
#define REPS(i , l, r) for(int i=l;i<r;i++)
#define present(t, x) (t.find(x) != t.end())   // Kiểm tra xem
value có trong Set,Map hay không
#define all(a) a.begin(),a.end()
#define sz(a) int((a).size())

using ll = long long;
const int MOD = (int) 1e9+7;
const int INF = (int) 1e9+1;
inline ll gcd(ll a,ll b){ll r;while(b){r=a%b;a=b;b=r;}return
a;}
inline ll lcm(ll a,ll b){return a/gcd(a,b)*b;}

void FileIO(){
    freopen("input.txt","r", stdin);
    freopen("output.txt","w",stdout);
}




// Cấu trúc lưu trữ thông tin tiến trình
struct TienTrinh {
    int id;            // ID của tiến trình
    int thoiGianXuatHien;  // Thời gian xuất hiện
    int thoiGianSuDungCPU; // Thời gian sử dụng CPU
    int thoiGianHoanThanh; // Thời gian hoàn thành
    int thoiGianCho;       // Thời gian chờ
    int thoiGianQuayVong;  // Thời gian quay vòng
};

// Hàm sắp xếp theo thời gian xuất hiện
bool soSanhThoiGianXuatHien(TienTrinh t1, TienTrinh t2) {
    return t1.thoiGianXuatHien < t2.thoiGianXuatHien;
```

```
}

// Hàm thực hiện thuật toán FCFS
void FCFS(vector<TienTrinh>& dsTienTrinh) {
    int soTienTrinh = dsTienTrinh.size();
    int thoiGianHienTai = 0;

    // Sắp xếp tiến trình theo thời gian xuất hiện
    sort(dsTienTrinh.begin(), dsTienTrinh.end(),
soSanhThoiGianXuatHien);

    // Tính toán thời gian hoàn thành, thời gian chờ và thời
gian quay vòng
    for (int i = 0; i < soTienTrinh; i++) {
        if (thoiGianHienTai < dsTienTrinh[i].thoiGianXuatHien)
{
            thoiGianHienTai = dsTienTrinh[i].thoiGianXuatHien;
// Nếu CPU rảnh, đợi đến khi tiến trình xuất hiện
        }
        dsTienTrinh[i].thoiGianHoanThanh = thoiGianHienTai +
dsTienTrinh[i].thoiGianSuDungCPU;
        dsTienTrinh[i].thoiGianQuayVong =
dsTienTrinh[i].thoiGianHoanThanh -
dsTienTrinh[i].thoiGianXuatHien;
        dsTienTrinh[i].thoiGianCho =
dsTienTrinh[i].thoiGianQuayVong -
dsTienTrinh[i].thoiGianSuDungCPU;
        thoiGianHienTai += dsTienTrinh[i].thoiGianSuDungCPU;
    }

    // In kết quả
    cout <<
"TienTrinh\tXuatHien\tSuDungCPU\tHoanThanh\tCho\tQuayVong\n";
    for (int i = 0; i < soTienTrinh; i++) {
        cout << "P" << dsTienTrinh[i].id << "\t\t" <<
dsTienTrinh[i].thoiGianXuatHien << "\t\t"
             << dsTienTrinh[i].thoiGianSuDungCPU << "\t\t" <<
```

```
dsTienTrinh[i].thoiGianHoanThanh << "\t\t"
             << dsTienTrinh[i].thoiGianCho << "\t" <<
dsTienTrinh[i].thoiGianQuayVong << endl;
    }

    // Tính toán thời gian chờ và thời gian quay vòng trung
bình
    double thoiGianChoTB = 0, thoiGianQuayVongTB = 0;
    for (int i = 0; i < soTienTrinh; i++) {
        thoiGianChoTB += dsTienTrinh[i].thoiGianCho;
        thoiGianQuayVongTB += dsTienTrinh[i].thoiGianQuayVong;
    }
    thoiGianChoTB /= soTienTrinh;
    thoiGianQuayVongTB /= soTienTrinh;

    cout << "Thoi gian cho trung binh: " << thoiGianChoTB <<
endl;
    cout << "Thoi gian quay vong trung binh: " <<
thoiGianQuayVongTB << endl;
}

int main() {
    int soTienTrinh;

    // Nhập số lượng tiến trình
    cout << "Nhap so tien trinh: ";
    cin >> soTienTrinh;

    vector<TienTrinh> dsTienTrinh(soTienTrinh);

    // Nhập thông tin tiến trình
    for (int i = 0; i < soTienTrinh; i++) {
        cout << "Nhap thoi gian xuat hien va thoi gian su dung
CPU cho tien trinh " << i + 1 << ": ";
        cin >> dsTienTrinh[i].thoiGianXuatHien >>
dsTienTrinh[i].thoiGianSuDungCPU;
        dsTienTrinh[i].id = i + 1;
```

```
    }

    // Gọi thuật toán FCFS
    FCFS(dsTienTrinh);
    system("pause");

    return 0;
}
```