

**ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH**

**TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN**







**HỆ ĐIỀU HÀNH**

**LỚP: IT007.O18.1**

**BÁO CÁO BÀI TẬP THỰC HÀNH 4**

**Họ tên: Trần Đình Khánh Đăng**

**MSSV: 22520195**



**4.3: Sinh viên chuẩn bị**

**4.3.5.1:** Vẽ sơ đồ giải thuật của các giải trình lập lịch tiến trình:

* FCFS (First Come First Served)

Ảnh có chứa văn bản, ảnh chụp màn hình, biểu đồ, Phông chữ

Mô tả được tạo tự động

Hình 1: Sơ đồ giải thuật FCFS

* RR (Round Robin)

Ảnh có chứa văn bản, biểu đồ, ảnh chụp màn hình, thiết kế

Mô tả được tạo tự động

Hình 2: Sơ đồ giải thuật RR

* SJF (Shortest Job First)

Ảnh có chứa văn bản, biểu đồ, ảnh chụp màn hình, hàng

Mô tả được tạo tự động

Hình 3: Sơ đồ giải thuật SJF

* SRT (Shortest Remain Time)

Ảnh có chứa văn bản, biểu đồ, ảnh chụp màn hình

Mô tả được tạo tự động

Hình 4: Sơ đồ giải thuật SRT

**4.3.5.2:** Giải thích các thuật ngữ sau:

|  |  |  |
| --- | --- | --- |
| **TT** | **Thuật ngữ** | **Mô tả** |
| 1 | Arrival time | Thời điểm một tiến trình nạp vào hệ thống |
| 2 | Burst time | Tổng thời gian một tiến trình thực hiện |
| 3 | Quantum time (timeslice) | Thời gian định mức trong thuật toán Round Robin |
| 4 | Response time | **Thời gian đáp ứng:** khoảng thời gian process nhận yêu cầu lần đầu tiên đến khi yêu cầu đầu tiên được đáp ứng |
| 5 | Waiting time | **Thời gian chờ:** Tổng thời gian một process đợi trong ready queue |
| 6 | Turnaround time | **Thời gian hoàn thành:** Là khoảng thời gian từ lúc nạp đến lúc kết thúc của một chương trình |
| 7 | Average waiting time | **Thời gian chờ trung bình:** Tổng thời gian chờ của các tiến trình chia cho số tiến trình |
| 8 | Average turnaround time | **Thời gian hoàn thành trung bình:** Tổng thời gian hoàn thành của các tiến trình chia cho số tiến trình |

**4.5: Bài tập ôn tập**

**Câu 1:** Viết chương trình mô phỏng giải thuật SJF với các yêu cầu sau:

* Nhập số lượng process
* Nhập process name, arrival time, burst time.
* In ra process name, response time, waiting time, turnaround time, average waiting time, average turnaround time.

**Chương trình:**

class shortestJobFirst:public processing {

public:

    void calculate() {

        // sap xep cac process theo arriveTime tang dan

        auto cmpArrive = [](const process& a, process& b) {

            return a.arriveTime < b.arriveTime;

        };

        sort(processList.begin(), processList.end(), cmpArrive);

        // hang doi uu tien phuc vu cho viec lay ra tien trinh co burstTime lon nhat

        auto cmpBurst = [](const process& a, process& b) {

            return a.burstTime > b.burstTime;

        };

        priority\_queue<process, vector<process>, decltype(cmpBurst)> pq(cmpBurst);

        // khoi tao: dua vao hang doi cac tien trinh co arriveTime thap nhat

        int time = processList[0].arriveTime; // thoi gian hien tai

        int index = 1; // so thu tu cua process duoc xu li tiep theo (theo thu tu da sort)

        pq.push(processList[0]);

        while (index < numberProcess && processList[index].arriveTime <= time)

            pq.push(processList[index++]);

        while (pq.size() || index < numberProcess) {

            if (pq.size() == 0) {

                // hang doi rong thi nap cac tien trinh co arriveTime nho nhat

                time = processList[index].arriveTime;

                pq.push(processList[index++]);

                while (index < numberProcess && processList[index].arriveTime == time)

                    pq.push(processList[index++]);

            }

            // lay ra tien trinh co burstTime nho nhat da duoc vao hang doi

            process runNow = pq.top();

            pq.pop();

            recordList.push\_back({runNow.name, time, time + runNow.burstTime});

            // runNow.burstTime + time la finishTime

            resultList[runNow.id - 1] = {runNow.name, runNow.arriveTime, runNow.burstTime, runNow.burstTime + time, time};

            time += runNow.burstTime;

            // them vao hang doi cac tien trinh co arriveTimee <= time

            while (index < numberProcess && processList[index].arriveTime <= time)

                pq.push(processList[index++]);

        }

    }

};

**Bộ Test:**

|  |  |  |
| --- | --- | --- |
| **Process** | **Arrival Time** | **Burst Time** |
| **P1** | **0** | **12** |
| **P2** | **2** | **7** |
| **P3** | **5** | **8** |
| **P4** | **9** | **3** |
| **P5** | **12** | **6** |

**Kết quả khi chạy tay:**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Name** | **Arr** | **Burst** | **Start** | **TAT** | **Finish** |
| P1 | 0 | 12 | 0 | 12 | 12 |
| P2 | 2 | 7 | 21 | 26 | 28 |
| P3 | 5 | 8 | 28 | 31 | 36 |
| P4 | 9 | 3 | 12 | 6 | 15 |
| P5 | 12 | 6 | 15 | 9 | 21 |
| Average waiting time: 9.6 (ms)  Average turnaround time: 16.8 (ms) | | | | | |

**Kết quả khi chạy chương trình:**

Ảnh có chứa ảnh chụp màn hình, văn bản, Phần mềm đồ họa, Phần mềm đa phương tiện

Mô tả được tạo tự động

**Câu 2:** Viết chương trình mô phỏng giải thuật SRT với các yêu cầu sau:

* Nhập số lượng process
* Nhập process name, arrival time, burst time.
* In ra process name, response time, waiting time, turnaround time, average waiting time, average turnaround time.

**Chương trình:**

class shortestRemainingTimeFirst: public processing {

public:

    void calculate() {

        // sap xep cac process theo arriveTime tang dan

        auto cmpArrive = [](const process& a, process& b) {

            return a.arriveTime < b.arriveTime;

        };

        sort(processList.begin(), processList.end(), cmpArrive);

        // hang doi uu tien phuc vu cho viec lay ra tien trinh co burstTime lon nhat

        auto cmpBurst = [](const process& a, process& b) {

            return a.burstTime > b.burstTime;

        };

        // burstTime cua cac process trong hang doi se co y nghia la tong thoi gian can duoc cap phat CPU con lai

        priority\_queue<process, vector<process>, decltype(cmpBurst)> pq(cmpBurst);

        // khoi tao: dua vao hang doi cac tien trinh co arriveTime thap nhat

        int time = processList[0].arriveTime; // thoi gian hien tai

        int index = 1; // so thu tu cua process duoc xu li tiep theo (theo thu tu da sort)

        pq.push(processList[0]);

        while (index < numberProcess && processList[index].arriveTime <= time)

            pq.push(processList[index++]);

        // mang dung de danh dau tien trinh da duoc thuc thi lan nao hay chua (index duoc danh theo thu tu input)

        vector<int> isExecute(numberProcess, 0);

        while (pq.size() || index < numberProcess) {

            if (pq.size() == 0) {

                // hang doi rong thi nap cac tien trinh co arriveTime nho nhat

                time = processList[index].arriveTime;

                pq.push(processList[index++]);

                while (index < numberProcess && processList[index].arriveTime == time)

                    pq.push(processList[index++]);

            }

            // lay ra tien trinh co burstTime (thoi gian can CPU con lai) nho nhat da duoc vao hang doi

            process runNow = pq.top();

            pq.pop();

            // thoi diem cuoi cung tien trinh duoc dung cpu: toi khi co tien trinh khac hoac hoan thanh tien trinh

            int maxTime = (index == numberProcess) ? time + runNow.burstTime : processList[index].arriveTime;

            // neu xu li chua xong

            if (maxTime < time + runNow.burstTime) {

                // neu chua duoc cap phat lan nao

                if (!isExecute[runNow.id - 1]) {

                    // danh dau da duoc cap phat CPU, luu truoc burstTime va firstExecute

                    isExecute[runNow.id - 1] = 1;

                    resultList[runNow.id - 1].burstTime = runNow.burstTime;

                    resultList[runNow.id - 1].respondTime = time; // luu firstExecute tam vao day

                }

                // them lai vao hang doi voi burstTime moi

                runNow.burstTime -= maxTime - time;

                pq.push(runNow);

                recordList.push\_back({runNow.name, time, maxTime});

                time = maxTime;

            } else { // hoan thanh tien trinh

                int burstTime = (isExecute[runNow.id - 1]) ? resultList[runNow.id - 1].burstTime : runNow.burstTime;

                int firstExecute = (isExecute[runNow.id - 1]) ? resultList[runNow.id - 1].respondTime : time;

                // runNow.burstTime + time la finishTime

                resultList[runNow.id - 1] = result(runNow.name, runNow.arriveTime, burstTime, time + runNow.burstTime, firstExecute);

                isExecute[runNow.id - 1] = 1;

                recordList.push\_back({runNow.name, time, time + runNow.burstTime});

                time += runNow.burstTime;

            }

            // them vao hang doi cac tien trinh co arriveTime bang voi time hien tai

            while (index < numberProcess && processList[index].arriveTime == time)

                pq.push(processList[index++]);

        }

    }

};

**Bộ Test:**

|  |  |  |
| --- | --- | --- |
| **Process** | **Arrival Time** | **Burst Time** |
| **P1** | **0** | **12** |
| **P2** | **2** | **7** |
| **P3** | **5** | **8** |
| **P4** | **9** | **3** |
| **P5** | **12** | **6** |

**Kết quả khi chạy tay:**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Name** | **Arr** | **Burst** | **Start** | **TAT** | **Finish** |
| P1 | 0 | 12 | 0 | 36 | 36 |
| P2 | 2 | 7 | 2 | 7 | 9 |
| P3 | 5 | 8 | 18 | 21 | 26 |
| P4 | 9 | 3 | 9 | 3 | 12 |
| P5 | 12 | 6 | 12 | 6 | 18 |
| Average waiting time: 7.4 (ms)  Average turnaround time: 14.6 (ms) | | | | | |

**Kết quả khi chạy chương trình:**

**Ảnh có chứa ảnh chụp màn hình, văn bản, phần mềm, Phần mềm đa phương tiện

Mô tả được tạo tự động**

**Câu 3:** Viết chương trình mô phỏng giải thuật RR với các yêu cầu sau (giả sử tất cả các tiến trình đều có arrival time là 0):

* Nhập số process
* Nhập quantum time.
* Nhập process name, burst time.
* In ra Gantt chart với các thông số: process name, start processor time, stop processor time.
* In ra average waiting time và average turnaround time.

**Chương trình:**

class roundRobin: public processing {

private:

    // thuat toan RR co them mot thuoc tinh la quantum time

    int quantumTime;

public:

    // override ham readInput

    void readInput() {

        processing::readInput();

        cout << "Enter quantum time: ";

        cin >> quantumTime;

    }

    void calculate() {

        // sap xep theo arriveTime

        auto cmpArrive = [](const process& a, process& b) {

            return a.arriveTime < b.arriveTime;

        };

        sort(processList.begin(), processList.end(), cmpArrive);

        // deque luu lai cac tien trinh theo thu tu la arriveTime

        // burstTime cua cac process trong hang doi se co y nghia la tong thoi gian can duoc cap phat CPU con lai

        deque<process> q;

        // khoi tao: dua vao deque tien trinh co arriveTime nho nhat

        int index = 1; // so thu tu cua process duoc xu li tiep theo (theo thu tu da sort)

        int time = processList[0].arriveTime; // thoi diem hien tai

        q.push\_front(processList[0]);

        // danh dau tien trinh da duoc execute hay chua

        vector<int> isExecute(numberProcess, 0);

        while (q.size() || index != numberProcess) {

            if (q.size() == 0) {

                // neu deque rong thi them vao tien trinh co arriveTime nho nhat

                time = processList[index].arriveTime;

                q.push\_front(processList[index++]);

            }

            // lay ra tien trinh o dau hang doi (tien trinh vao hang doi som nhat)

            process runNow = q.back();

            q.pop\_back();

            // xu li toi khi het tien trinh hoac het quantumTime

            int maxTime = time + min(quantumTime, runNow.burstTime);

            // neu hêt quantumTime van chua xong

            if (maxTime < time + runNow.burstTime) {

                // neu chua duoc execute lan nao

                if (!isExecute[runNow.id - 1]) {

                    // danh dau da duoc cap phat CPU, luu truoc burstTime va firstExecute

                    isExecute[runNow.id - 1] = 1;

                    resultList[runNow.id - 1].burstTime = runNow.burstTime;

                    resultList[runNow.id - 1].respondTime = time; // firstExecute duoc luu tam vao respondTime

                }

                // burstTime con lai

                runNow.burstTime -= quantumTime;

                recordList.push\_back({runNow.name, time, maxTime});

                time = maxTime;

            } else {

                int burstTime = (isExecute[runNow.id - 1]) ? resultList[runNow.id - 1].burstTime : runNow.burstTime;

                int firstExecute = (isExecute[runNow.id - 1]) ? resultList[runNow.id - 1].respondTime : time;

                // runNow.burstTime + time la finishTime

                resultList[runNow.id - 1] = result(runNow.name, runNow.arriveTime, burstTime, time + runNow.burstTime, firstExecute);

                isExecute[runNow.id - 1] = 1;

                recordList.push\_back({runNow.name, time, time + runNow.burstTime});

                time = maxTime;

                runNow.burstTime = 0; // khong can xu li nua

            }

            // them vao cac tien trinh chua duoc thuc thi co arriveTime nho hon bang time

            while (index < numberProcess && processList[index].arriveTime <= time)

                q.push\_front(processList[index++]);

            // neu tien trinh vua xu li chua hoan thanh thi day tien trinh vao deque

            if (runNow.burstTime != 0)

                q.push\_front(runNow);

        }

    }

};

**Bộ Test:**

|  |  |  |
| --- | --- | --- |
| **Process** | **Arrival Time** | **Burst Time** |
| **P1** | **0** | **12** |
| **P2** | **2** | **7** |
| **P3** | **5** | **8** |
| **P4** | **9** | **3** |
| **P5** | **12** | **6** |

**Kết quả khi chạy tay:**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Name** | **Arr** | **Burst** | **Start** | **TAT** | **Finish** |
| P1 | 0 | 12 | 0 | 30 | 30 |
| P2 | 2 | 7 | 4 | 17 | 19 |
| P3 | 5 | 8 | 12 | 29 | 34 |
| P4 | 9 | 3 | 19 | 13 | 22 |
| P5 | 12 | 6 | 22 | 24 | 36 |
| Average waiting time: 15.4 (ms)  Average turnaround time: 22.6 (ms) | | | | | |

**Kết quả khi chạy chương trình:**

Ảnh có chứa ảnh chụp màn hình, văn bản, hàng

Mô tả được tạo tự động

Full Code: <https://ideone.com/Li6Y2P>