HỆ ĐIỀU HÀNH  
BÁO CÁO LAB 4

Nhóm: 8

Thông tin thành viên:

o Dương Thuận Trí- 22521517

o Trương Quốc Vinh- 22521682

o Nguyễn Thị Kim Ngân- 21521174

o Nông Tiến Dũng- 20521213

Lớp: 1

**CHECKLIST**

**3.5. BÀI TẬP THỰC HÀNH**

|  |  |  |
| --- | --- | --- |
|  | **BT 1** | **BT 2** |
| **Vẽ lưu đồ giải thuật** |  |  |
| **Chạy tay lưu đồ giải thuật** |  |  |
| **Hiện thực code** |  |  |
| **Chạy code và kiểm chứng** |  |  |

**3.6. BÀI TẬP ÔN TẬP**

|  |  |
| --- | --- |
|  | **BT 1** |
| **Vẽ lưu đồ giải thuật** |  |
| **Chạy tay lưu đồ giải thuật** |  |
| **Hiện thực code** |  |
| **Chạy code và kiểm chứng** |  |

**Tự chấm điểm:** 10

**2.5. BÀI TẬP THỰC HÀNH**

# Giải thuật Shortest-Job-First

* A black background with white squares

  Description automatically generated**Lưu đồ:**
* **Chạy tay lưu đồ:**

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

* Tại giây 0:

|  |  |  |
| --- | --- | --- |
| Input[5] | ReadyQueue[5] | TerminatedArray[5] |
| P2(2, 7) | P1(0, 12) |  |
| P3(5, 8) |  |  |
| P4(9, 3) |  |  |
| P5(12, 6) |  |  |
|  |  |  |
| iRemain = 4 | iReady = 1 | iTerminated = 0 |

* Tại giây 12:

|  |  |  |
| --- | --- | --- |
| Input[5] | ReadyQueue[5] | TerminatedArray[5] |
|  | P4(9, 3) | P1(0, 12) |
|  | P5(12, 6) |  |
|  | P2(2, 7) |  |
|  | P3(5, 8) |  |
|  |  |  |
| iRemain = 0 | iReady = 4 | iTerminated = 1 |

* Tại giây 15:

|  |  |  |
| --- | --- | --- |
| Input[5] | ReadyQueue[5] | TerminatedArray[5] |
|  | P5(12, 6) | P1(0, 12) |
|  | P2(2, 7) | P4(9, 3) |
|  | P3(5, 8) |  |
|  |  |  |
|  |  |  |
| iRemain = 0 | iReady = 3 | iTerminated = 2 |

* Tại giây 21:

|  |  |  |
| --- | --- | --- |
| Input[5] | ReadyQueue[5] | TerminatedArray[5] |
|  | P2(2, 7) | P1(0, 12) |
|  | P3(5, 8) | P4(9, 3) |
|  |  | P5(12, 6) |
|  |  |  |
|  |  |  |
| iRemain = 0 | iReady = 2 | iTerminated = 3 |

* Tại giây 28:

|  |  |  |
| --- | --- | --- |
| Input[5] | ReadyQueue[5] | TerminatedArray[5] |
|  | P3(5, 8) | P1(0, 12) |
|  |  | P4(9, 3) |
|  |  | P5(12, 6) |
|  |  | P2(2, 7) |
|  |  |  |
| iRemain = 0 | iReady = 1 | iTerminated = 4 |

* Tại giây 36:

|  |  |  |
| --- | --- | --- |
| Input[5] | ReadyQueue[5] | TerminatedArray[5] |
|  |  | P1(0, 12) |
|  |  | P4(9, 3) |
|  |  | P5(12, 6) |
|  |  | P2(2, 7) |
|  |  | P3(5, 8) |
| iRemain = 0 | iReady = 0 | iTerminated = 5 |

* **Code:**

#include <stdio.h>

#include <stdlib.h>

#include <stdbool.h>

#include <time.h>

#include <limits.h>

#define SORT\_BY\_ARRIVAL 0

#define SORT\_BY\_PID 1

#define SORT\_BY\_BURST 2

#define SORT\_BY\_START 3

typedef *struct*

{

*int* iPID;

*int* iArrival, iBurst;

*int* iStart, iFinish, iWaiting, iResponse, iTaT;

} PCB;

*void* inputProcess(*int* *n*, PCB *P*[])

{

        srand((*unsigned*) time(NULL));

    for(*int* i = 0 ; i < n;i++){

        P[i].iPID= i+1;

        P[i].iArrival = rand() % 21;

        P[i].iBurst = (rand()% 11) +2;

    }

}

*void* printProcess(*int* *n*, PCB *P*[])

{

    for(*int* i = 0 ; i < n;i++){

        printf("PID: %d\n",P[i].iPID);

        printf("Arrival Time: %d\n",P[i].iArrival);

        printf("Burst Time: %d\n",P[i].iBurst);

        printf("Start Time: %d\n",P[i].iStart);

        printf("Finish Time: %d\n",P[i].iFinish);

        printf("Waiting Time: %d\n",P[i].iWaiting);

        printf("Response Time: %d\n",P[i].iResponse);

        printf("Turn Around Time: %d\n\n",P[i].iTaT);

    }

}

*void* exportGanttChart(*int* *n*, PCB *P*[])

{

    printf("\nGantt Chart\n");

    for(*int* i = 0 ; i < n;i++){

        printf("P%d\t",P[i].iPID);

    }

    printf("\n");

    for(*int* i = 0 ; i < n;i++){

        printf("%d\t",P[i].iStart);

    }

    printf("%d\t",P[n-1].iFinish);

    printf("\n");

}

*void* pushProcess(*int* \**n*, PCB *P*[], PCB *Q*)

{

    P[\*n] = Q;

    (\*n)++;

}

*void* removeProcess(*int* \**n*, *int* *index*, PCB *P*[]){

    for(*int* i = index ; i < \*n - 1;i++){

        P[i] = P[i+1];

    }

    (\*n)--;

}

*int* swapProcess(PCB \**P*, PCB \**Q*) {

    PCB temp = \*P;

    \*P = \*Q;

    \*Q = temp;

}

*int* partition (PCB *P*[], *int* *low*, *int* *high*, *int* *iCriteria*) {

    if (iCriteria == SORT\_BY\_ARRIVAL) {

*int* pivot = P[high].iArrival;

*int* i = (low-1);

        for(*int* j = low; j <= high; j++)

        {

            if (P[j].iArrival < pivot)

            {

                i++;

                swapProcess(&P[i], &P[j]);

            }

        }

        swapProcess(&P[i+1], &P[high]);

        return (i+1);

    } else if (iCriteria == SORT\_BY\_PID) { // Sort by PID

*int* pivot = P[high].iPID;

*int* i = (low-1);

        for(*int* j = low; j <= high; j++)

        {

            if (P[j].iPID < pivot)

            {

                i++;

                swapProcess(&P[i], &P[j]);

            }

        }

        swapProcess(&P[i+1], &P[high]);

        return (i+1);

    } else if (iCriteria == SORT\_BY\_BURST) {

*int* pivot = P[high].iBurst;

*int* i = (low-1);

        for(*int* j = low; j <= high; j++)

        {

            if (P[j].iBurst < pivot)

            {

                i++;

                swapProcess(&P[i], &P[j]);

            }

        }

        swapProcess(&P[i+1], &P[high]);

        return (i+1);

    } else if (iCriteria == SORT\_BY\_START) {

*int* pivot = P[high].iStart;

*int* i = (low-1);

        for(*int* j = low; j <= high; j++)

        {

            if (P[j].iStart < pivot)

            {

                i++;

                swapProcess(&P[i],&P[j]);

            }

        }

        swapProcess(&P[i+1], &P[high]);

        return (i+1);

    }

}

*void* quickSort(PCB *P*[], *int* *low*, *int* *high*, *int* *iCriteria*) {

    if(low < high) {

*int* pi = partition(P, low, high, iCriteria);

    quickSort(P, low, pi-1, iCriteria);

    quickSort(P, pi+1, high, iCriteria);

    }

}

*void* calculateAWT(*int* *n*, PCB *P*[])

{

*int* sum = 0;

    for(*int* i = 0 ; i < n;i++){

        sum += P[i].iWaiting;

    }

    printf("Average Waiting Time: %f\n",(*float*)sum/n);

}

*void* calculateATaT(*int* *n*, PCB *P*[]){

*int* sum = 0;

    for(*int* i = 0 ; i < n;i++){

        sum += P[i].iTaT;

    }

    printf("Average itat Time: %f\n",(*float*)sum/n);

}

*int* main()

{

    PCB Input[10];

    PCB ReadyQueue[10];

    PCB TerminatedArray[10];

*int* iNumberOfProcess;

    printf("Please input number of Process: ");

    scanf("%d", &iNumberOfProcess);

*int* iRemain = iNumberOfProcess, iReady = 0, iTerminated = 0;

    inputProcess(iNumberOfProcess, Input);

    quickSort(Input, 0, iNumberOfProcess - 1,SORT\_BY\_ARRIVAL);

    pushProcess(&iReady, ReadyQueue, Input[0]);

    removeProcess(&iRemain, 0, Input);

    ReadyQueue[0].iStart = ReadyQueue[0].iArrival;

    ReadyQueue[0].iFinish = ReadyQueue[0].iStart + ReadyQueue[0].iBurst;

    ReadyQueue[0].iResponse = ReadyQueue[0].iStart -ReadyQueue[0].iArrival;

    ReadyQueue[0].iWaiting = ReadyQueue[0].iResponse;

    ReadyQueue[0].iTaT = ReadyQueue[0].iFinish -ReadyQueue[0].iArrival;

*bool* flag=false;

    while (iTerminated < iNumberOfProcess)

    {

        while (iRemain > 0)

        {

            if (Input[0].iArrival <= ReadyQueue[0].iFinish)

            {

                pushProcess(&iReady, ReadyQueue, Input[0]);

                removeProcess(&iRemain, 0, Input);

                flag=false;

                continue;

            }

            else{

                if(iReady==0){

                     flag=true;

                     pushProcess(&iReady, ReadyQueue, Input[0]);

                      removeProcess(&iRemain, 0, Input);

                      break;

                }

                else {flag=false;break;}

            }

        }

        if (iReady > 0)

        {

            pushProcess(&iTerminated, TerminatedArray,ReadyQueue[0]);

            removeProcess(&iReady, 0, ReadyQueue);

            quickSort(ReadyQueue, 0, iReady-1,SORT\_BY\_BURST);

            if(flag)  ReadyQueue[0].iStart = ReadyQueue[0].iArrival;

            else ReadyQueue[0].iStart = TerminatedArray[iTerminated - 1].iFinish;

            ReadyQueue[0].iFinish = ReadyQueue[0].iStart + ReadyQueue[0].iBurst;

            ReadyQueue[0].iResponse = ReadyQueue[0].iStart -ReadyQueue[0].iArrival;

            ReadyQueue[0].iWaiting = ReadyQueue[0].iResponse;

            ReadyQueue[0].iTaT = ReadyQueue[0].iFinish - ReadyQueue[0].iArrival;

        }

    }

            printProcess(iTerminated, TerminatedArray);

    printf("\n===== SJF Scheduling =====\n");

    exportGanttChart(iTerminated, TerminatedArray);

    quickSort(TerminatedArray, 0, iTerminated - 1,SORT\_BY\_PID);

    calculateAWT(iTerminated, TerminatedArray);

    calculateATaT(iTerminated, TerminatedArray);

    return 0;

}

* Test case 1:

A screenshot of a computer

Description automatically generatedA screenshot of a computer program

Description automatically generated

* Test case 2:

A screenshot of a computer program

Description automatically generated

A screenshot of a computer

Description automatically generated

* Test case 3:

A screenshot of a computer program

Description automatically generated

A screenshot of a computer program

Description automatically generated

# Giải thuật Shortest-Remaining-Time-First hoặc Round Robin

Trả lời: **Giải thuật Shortest-Remaining-Time-First**

* **Lưu đồ:**

# A diagram of a flowchart Description automatically generated

A diagram of a computer program

Description automatically generated with medium confidence

Link lưu đồ: <https://app.diagrams.net/#G1v88DIaL9lPyO3XHpyHlyouRyuEkaeudo>

* Test case:

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

* Các tiến trình được đưa vào. Tại thời điểm 0, tiến trình P1 được thêm vào ReadyQueue[]

|  |  |  |
| --- | --- | --- |
| Input[5] | ReadyQueue[5] | TerminatedArray[5] |
| P2(2, 7) | P1(0, 12) |  |
| P3(5, 8) |  |  |
| P4(9, 3) |  |  |
| P5(12, 6) |  |  |
|  |  |  |
| iRemain = 4 | iReady = 1 | iTerminated = 0 |

* Trong khi P1 chạy đến thời điểm 2 thì P2 được đưa vào ReadyQueue[].

|  |  |  |
| --- | --- | --- |
| Input[5] | ReadyQueue[5] | TerminatedArray[5] |
| P3(5, 8) | P1(0, 12) |  |
| P4(9, 3) | P2(2, 7) |  |
| P5(12, 6) |  |  |
|  |  |  |
|  |  |  |
| iRemain = 3 | iReady = 2 | iTerminated = 0 |

* Khi P2 vào thì P2 trưng dụng và chiếm CPU (vì burst time của P2 nhỏ hơn burst time còn lại của P1 ) . Trong lúc đó thì P3 cũng vào ReadyQueue[] tại thời điểm 5.

|  |  |  |
| --- | --- | --- |
| Input[5] | ReadyQueue[5] | TerminatedArray[5] |
| P4(9, 3) | P2(2, 7) |  |
| P5(12, 6) | P3(5, 8) |  |
|  | P1(0, 10) |  |
|  |  |  |
|  |  |  |
| iRemain = 2 | iReady = 3 | iTerminated = 0 |

* Sau khi P2 thực thi xong và được đưa vào TerminatedArray[], tại thời điểm 9 thì P4 cũng vào ReadyQueue[] và nó chiếm CPU để thực thi. ( vì có burst time nhỏ nhất trong các tiến trình có trong ReadyQueue[] (sort by burst) .

|  |  |  |
| --- | --- | --- |
| Input[5] | ReadyQueue[5] | TerminatedArray[5] |
| P5(12, 6) | P4(9, 3) | P2(2, 7) |
|  | P3(5, 8) |  |
|  | P1(0, 10) |  |
|  |  |  |
|  |  |  |
| iRemain = 1 | iReady = 3 | iTerminated = 1 |

* Sau khi P4 thực thi xong và được đưa vào TerminatedArray[], tại thời điểm 12 thì P5 cũng được đưa vào ReadyQueue[] và nó chiếm CPU để thực thi. (vì có burst time nhỏ nhất trong các tiến trình có trong ReadyQueue[] (sort by burst).

|  |  |  |
| --- | --- | --- |
| Input[5] | ReadyQueue[5] | TerminatedArray[5] |
|  | P5(12, 6) | P2(2, 7) |
|  | P3(5, 8) | P4(9, 3) |
|  | P1(0, 10) |  |
|  |  |  |
|  |  |  |
| iRemain = 0 | iReady = 3 | iTerminated = 2 |

* Các tiến trình tiếp tục thực thi lần lượt theo thứ tự P5, P3, P1 (tiến trình có burst time nhỏ hơn sẽ được thực thi trước (sort by burst)).

|  |  |  |
| --- | --- | --- |
| Input[5] | ReadyQueue[5] | TerminatedArray[5] |
|  |  | P2(2, 7) |
|  |  | P4(9, 3) |
|  |  | P5(12, 6) |
|  |  | P3(5, 8) |
|  |  | P1(0, 10) |
| iRemain = 0 | iReady = 0 | iTerminated = 5 |

Sau khi các tiến trình thực thi xong thì sẽ lần lượt được chuyển sang TerminatedArray[], lúc này iTerminated = 5, chính là số lượng tiến trình ban đầu nhập vào và giải thuật kết thúc tại thời điểm 36.

* **Code:**

A screen shot of a computer program

Description automatically generated

A screen shot of a computer program

Description automatically generated

A screen shot of a computer program

Description automatically generated

A computer screen shot of a program

Description automatically generated

A computer screen shot of a program code

Description automatically generated

A screen shot of a computer program

Description automatically generated

A screen shot of a computer

Description automatically generated  
A screen shot of a computer program

Description automatically generated

A screen shot of a computer program

Description automatically generated

A screen shot of a computer program

Description automatically generated

A computer screen shot of text

Description automatically generated

A computer screen with many colorful text

Description automatically generated with medium confidence

A screen shot of a computer program

Description automatically generated

A screen shot of a computer program

Description automatically generated

A screen shot of a computer program

Description automatically generated

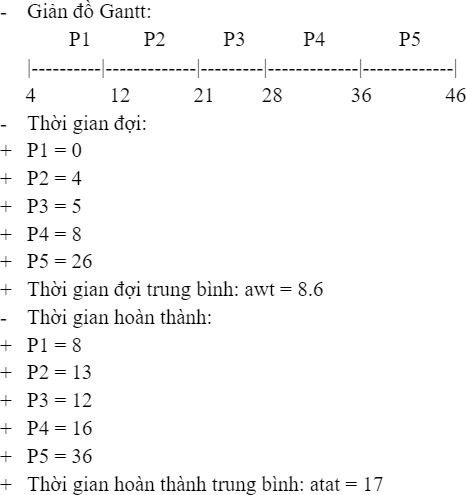
A screen shot of a computer program

Description automatically generated

* Test case 1:

A screen shot of a computer

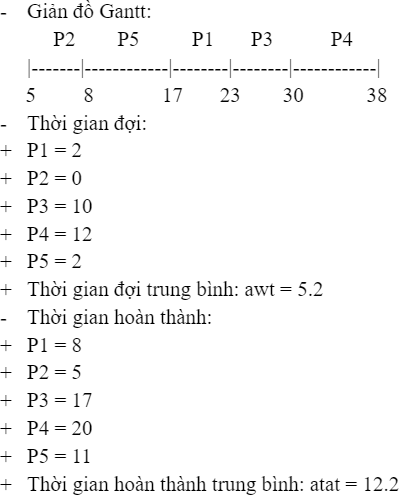
Description automatically generated



* Test case 2:

A screenshot of a computer program

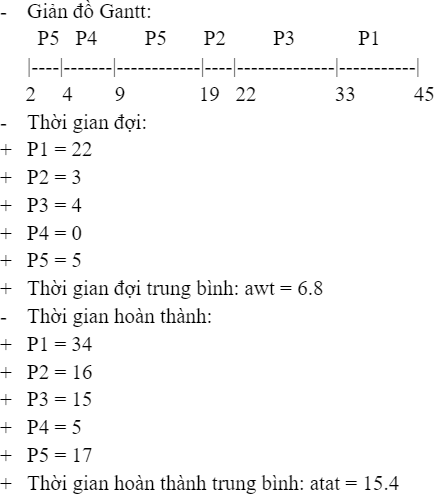
Description automatically generated



* Test case 3:

A screen shot of a computer

Description automatically generated



**2.6. BÀI TẬP ÔN TẬP**

# Giải thuật Shortest-Remaining-Time-First hoặc Round Robin

Trả lời: **Round Robin**

* **Lưu đồ:**

A diagram with text on it

Description automatically generated

A graph with text on it

Description automatically generated

A diagram of a diagram

Description automatically generated with medium confidence

A diagram of a flowchart

Description automatically generated

A diagram of a flowchart

Description automatically generated

A diagram of a company

Description automatically generated

A diagram with text and symbols

Description automatically generated with medium confidence

Link lưu đồ: https://viewer.diagrams.net/?tags=%7B%7D&highlight=0000ff&edit=\_blank&layers=1&nav=1&title=Round%20Robin.drawio#R7V1bc5s4FP41ntk%2BuMMd%2B9HOrd1tmzTxbtJ96WCj2GwwuIATu79%2BxdUgDiA74mInM53GCAGSjr5zl9QTz5abK0dbLb7aOjJ7AqdveuJ5TxAGAo%2F%2F9wu2YYEscWHB3DH0sIjfFdwZv1FUGFdbGzpyMxU92zY9Y5UtnNmWhWZepkxzHPslW%2B3RNrNfXWlzlCu4m2lmvvTe0L1F1C2Z25V%2FQsZ8EX%2BZ56I7Sy2uHBW4C023X1JF4kVPPHNs2wt%2FLTdnyPTHLh6X8LnLgrtJwxxkeTQPWPcb4cfnxXCGuP76fL4dyvfrvhS1zdvGHUY67n90aTvewp7blmZe7ErHjr22dOS%2FlcNXuzpfbHuFC3lc%2BB%2FyvG1ETG3t2bho4S3N6C7aGN6D%2F%2FhHObr6kbpzvoneHFxs4wvLc7aph%2FzLH%2Bl7u8eCq%2Fi5%2FChFA%2Bfaa2eGSoYmnm2aM0deST0hrOePW%2BoDEQ2ukL1EuD24goNMzTOes%2FNKi6bnPKm3oyD%2BERFxD4JG733WzHX0pZ6gmLj94yn%2BMfd%2F3E1Gt5Ok1ImL4xL82emujJwdpomR58%2BCl4XhobuVFozhC8Z%2BlsaF4%2F6MHA9tSkcquiuJEXIi1jGILl92OORjcC1SGIzrMR9b%2BR0sFSCoBIvYKbAI1WD5bK3Wnv%2Beb%2BvlFDnXjzeOPUOuSwsYzPVX%2Fk88qJppItOeO9oSV1whx8CdQA5572Z3owpjj8YGxSKTEebEQRZziRxOgU6AQKfUBTrlWEB3OHhESvCorMETPXpjG5aX4rvD7ByQOYK2YUOjpwjyJs04nOJiNSpvzvBwcyE25XEOm%2FJ5Cp47CXeLNH37fY3wa%2BkfmvQu%2BN541BuPl4aleUgfOY623ecFaLnC1bHeiyvk7oYjb9yipWZY%2BFfQYc4C35TUxd1IqnJlVSfIiVpNWx%2B3tbwmFdfLArB%2BVYFkWyIt2%2BJr41uD0%2BdbKiXfGrbDt5SG%2BZZaxrdSE0H5tbaDUtvRkdOf2aaNhf0oeNT5o99Pl38IERc9QSol%2BzLA%2BLnva83y1kv8a4J1Dfzn1%2BFQP3IFR25dweH53Jg2wCkOR%2F2QEvU8c8OYDvaq0Czsh9Xqyh0mEDVcOy1YlfbhIgPjfUz4iXFRDSCJNYBeN%2FAU1vLN2l2kJnqo8nIYeDb%2Bk1G%2F4zmfqLNj%2Fx%2BstS%2FtZ0S%2BNX7WV5v7%2FSPSVCWCW4l5QIkcACiVBaDU%2FybfHhYGhxZj%2BexBefi0%2BtKSV6t5d65ECzvmZvbrYCdVw67C1PUB89G4wz33EuMOqIANWwN%2FpRA5lR%2BFSqAP3SJ3ZVsuKmtM3No%2BbVs7jvvEkRY7s5U87mUFslDl2iQppLkclSSlNUD5QbcgXWopRpLUCZVc3ye0CtVE5PqNedYMU5uaKJCrWXC8AXON52kVUCZRIEhetuOQbl5eDvLgerrXf35z%2Fto8TS8vb6%2BtR1n8vm0vpHP573y0vbm9OB%2Bt5N9Xc%2B2%2Bf%2B62ZE3Hfrf4949e2ge3p9%2BtboOZFzjCWyqQMqbAZM69SiFsbz5Rb%2BM3hfMs96bQib6rtvIruMVtBj4kl7YMd5IvfQL%2FCBvB1BkQo6aMtWMuGrDvPwj3vKBoS5%2BjRkw8F3L8QKfYLOzldO02o9SQVIFi9BJkzfAiA%2FY8UB6lzV%2FSdHspS5eT640nohcwAeKYlJq4vdV5J0pbjBccd%2FE4h7luZjsgILIzBBpyT8YdpeNIuzhkzI1iO47rHvuRCJuKV6EUIcioYuJMAWPX3LGzH4WW%2FTCP6tEO%2FCdLd1ZbfTx60PnJHI2e%2F9TtxC18zOMMd4w5m4cZlSoTjIojRXTdjEqpZlTGo8%2Bl0o7gnSsop0EVuZYuDctwF91jZ6JKsLMhZOtKddm6YHowJDyOCma00UqJufFFO%2FDKLXfNK38P%2FnkaCQ%2Fot3Q9fXHacTIUDnPdvEcggiL8gJL3VAaD8avoDFJmXKw0HMw0C4RtsI1p0%2FaM2ZWy4OaD3AIHOOfrczKCemQr6WOtMYAcbGmtpH39SCpJ6vhDRQxF5sofyLqRDjD%2FauZHIpQuEaLOXWkWiPtH2%2FL6L9E092Fv2c4yDrYRSL9MYTh8YQGMMRaDj3qO%2FYTOQpZybtkWCgIHpkkUpcMKuDnRlBWk%2BDp6sX%2FfB7ox08yRacwtXLY0dD3AgxYVzPB0DuIXDHiFQtqckJLGQy4v0jPKjlfkFefJqQ6%2FpMoEq%2BYaHH4w4vC%2BKqhSqU5r33DUhrWYeZUrs51FkQl1MrTZkaoR6oAuiG4t2oqbXe0n2K2UKHZm7ps20qR7UyB1BxngdXX5A0CzNDbLTj34LdEuyYop0hVslK6dqsP6zPnXdiZotPCpaDFCtNioMO%2BT5gNJ%2BLT4M3X1PxO8LexEaSZdbVZ3fsyCSiEPtKbuqqplrMcqaFCnTH5CjRcGQOhIBPPx4kgDe%2FDmTbW3o8eLQtt6PJSU1yEXTKXMikVRpcxSZNYy6yBXDOnzEWNy15qgE49SmfZYmTwdSbZ4uayfo8ynQkvlcihV0tXk6AYyh4i9PUQO0G1lcMWuKHysiwPLxbn2786yQ5xlOecpZME06iyT35KzjEyellr3VcrFqe%2FvCDuAxCrhI%2BDF1v2hfCuZgM37CGI1qlrf6tbqE6U4IsTa7rvYrMKVz1ea5Xlni1BreXTsZZF74JTXrogQNmtbPA1js2XXdmPYBBavwHt2MLeFXodNiuUDWLTM1rgxaHQ%2FyS4hOMv4xALj6QNhBFymjI%2Fdiybafm9q23lDpCqKkF6TOMHTwBrWZTmoey47cNF8iQcgDTNk%2BQMd7BA6MzXXNWbZgZutnecEhVlM4kcj8A0CnQbz2dQ188hSdW6wQImqtLEHkCsuY55eRjLYgiSM3JvImccPBWJGMcpTyTeZb2D1kppXDS4sPT%2BRk01AY4ms2%2BupiS5oNwfNqswMDB1itEQZWggAacEsJC0YKXu3c9h6Egh%2FEa%2BqDdo5cCz09BYCg%2F2EFgLDA0LL9V%2B7pIqMi6tcziXIiA0PyTCREkVYWbFhWEs%2FolAE9ZSBe9qa%2Bl3a7urUEtBx7%2Bc1G9Y82lMPWo3yq4OrTgjZKcgDNUZT1Z5EPJnMymyThVZX8Xcmqa6UzVZuzNCtFS1xs1%2B9jxGJs34ErYqEhn02Lerg%2FkGqQgo9IJQCgnRYmwJUnGz0ruMeouOqpLLRpC8fJnE%2BIPpmomUYYZAgrC9eBsnB7ip%2BzYgcOrVc5Orb87WsnxRbYZakL4JL9Ap3vyzI6auUZrXnabJwCmenbB3iEszvq02nBScNtL79mMAN5ESD9Vqz6cDWQCbdKY56a7uWlrX6aAxp5gwoFymQAH0djFHxpCw7hCjXxo8zNOe9r9eX3NW3B9n9qn86NmWinjTVvDJBJKnxKhkUqFmZaC4fo3AzlB5FsmmgAnDpkyK6JfHJTCgJiBCAGf217RSqtoG3Wh1SZZKHXVS4GRHV3HEtDGBXKSY76rYSSDGoSHlUSpAYZIFKkO75DVrejEuDp46Ls3BowJZz8fqLd6chgxR7Xubb9llBiyhIOViR60SpHXQkv6jaHUWbX5RbE1WjYwukHY21fFq0q9T%2BD6VdnXYEHDMtZq268QxyVp8J9iO25DNWEz2GzJILeK4bcDn%2FBi%2BuNiC%2FrdCZtNnTPJgQab0Jay2eEfFv6g9hRr2eBQqRWqDqJGVBb7OlJzAAlWIy0Ob6bqjO%2Ba9aOWjvT%2FQEcZgdzbQUjY4JvflMuoqLpPYe%2FaInJr2iMLVNPRwCJoPT0FQq7B%2Fb2VRO6lfP69yIp7Q26mZGEy7Z49VPTDfGa8c9bObs1cfuTYy3xGMi%2BzcgeGwtBxf3Gu6YNY%2BuJtrkoJnwTiyWxEoWrbcoFaCabVH5CEl4o61Tmw5QHgbXLWwd4ajnYl3h6JMSrnEwdZ6Eqj9ON2d7dQ32AKVMV1pnUM6RU%2BjygTysWQM55U3ixaSZTBPJAKefAG5tRC5mYrbu9PSWnZZtY1m5UqK98wjLmt1AJKSCn73%2BxQXJv3WHc3wtNAzVFIVzkoBPwZ49ofg9kRE6iU4AREpMj1JSV%2BzOBFG6XRI1Po5ZXbMYLW9g1nYrWkvuVi8CORTweX61bVnx5iISuRiQcGg0iSc3Wcu9iV1EAqQddBbKSdOuesQPpV3%2BZNWaaQdGk%2BohHrvzu9uhcnK07N5UlnMJGvXFDGEqC28OotWnHx8crac9R5kZ9Y4oub10%2FXf18WwdO2W81Zxd2GnInba63LJqLNKsvwW9arvl9MwPKunumafNnCNCyeHza6DZ8eWyjpatfhk7SHvq6mQfkHtCCMCBeMk6Fdb7KcHH4OYVlbeTtjuAhr%2BunE6Q03RXr%2BjWock1nu5e2tGjZTUK6XICD3Sri9U8m97Tv1%2BXX%2F5Z%2FHz4%2BRN98q6cJXQQ9emyGvIcDmVY56YH%2BNKxfWVvhwE8qRZfbR35Nf4H

* **Test case:**

Cho 5 tiến trình có số liệu như sau (cho q = 4):

|  |  |  |
| --- | --- | --- |
| **Process** | **Arrival Time** | **Burst Time** |
| P1 | 0 | 6 |
| P2 | 3 | 4 |
| P3 | 7 | 3 |
| P4 | 10 | 5 |
| P5 | 12 | 1 |

Đầu tiên ta có:

|  |  |  |  |
| --- | --- | --- | --- |
| **Input** | **Ready Queue** | **Temp** | **Terminated** |
| P1 (0,6) |  |  |  |
| P2 (3, 4) |  |  |  |
| P3 (7, 3) |  |  |  |
| P4 (10, 5) |  |  |  |
| P5 (12, 1) |  |  |  |
| iRemain = 5 | iReady = 0 | iTemp = 0 | iTerminated = 0 |

P1 được vào hàng chờ Ready:

|  |  |  |  |
| --- | --- | --- | --- |
| **Input** | **Ready Queue** | **Temp** | **Terminated** |
| P2 (3, 4) | P1 (0, 6) |  |  |
| P3 (7, 3) |  |  |  |
| P4 (10, 5) |  |  |  |
| P5 (12, 1) |  |  |  |
|  |  |  |  |
| iRemain = 4 | iReady = 1 | iTemp = 0 | iTerminated = 0 |

Vì burst time của P1 (iBurst = 6) lớn hơn quantum time (q = 4) nên P1 sẽ thực hiện trong khoảng thời gian tối đa là 4 giây. Như vậy, thời điểm kết thúc của P1 là 4.

Trong khoảng thời gian từ giây 0 đến giây 4 có P2 xuất hiện (iArrival = 3). Lúc này P2 sẽ được thêm vào hàng chờ Ready

|  |  |  |  |
| --- | --- | --- | --- |
| **Input** | **Ready Queue** | **Temp** | **Terminated** |
| P3 (7, 3) | P1 (0, 6) |  |  |
| P4 (10, 5) | P2 (3, 4) |  |  |
| P5 (12, 1) |  |  |  |
|  |  |  |  |
|  |  |  |  |
| iRemain = 3 | iReady = 2 | iTemp = 0 | iTerminated = 0 |

Sau khi P1 thực hiện xong sẽ được đẩy xuống cuối của hàng chờ Ready, đồng thời sẽ cập nhật lại iRemainingTime = 2 (đã thực hiện 4 giây trước đó), iPause sẽ đánh dấu lần vào hàng chờ Ready lần thứ 2 tại giây thứ 4. P1 cũng sẽ được đẩy qua hàng Temp (các chỉ số thể hiện ở cột Temp là **(iStart, iFinish)**, ở các cột còn lại là (iArrival, iBurst) ):

|  |  |  |  |
| --- | --- | --- | --- |
| **Input** | **Ready Queue** | **Temp** | **Terminated** |
| P3 (7, 3) | P2 (3, 4) | P1 (0, 4) |  |
| P4 (10, 5) | P1 (4, 2) |  |  |
| P5 (12, 1) |  |  |  |
|  |  |  |  |
|  |  |  |  |
| iRemain = 3 | iReady = 2 | iTemp = 1 | iTerminated = 0 |

P2.iStart = P1.iFinish ( = 4)

Tương tự như trên, các tiến trình sẽ thực thi tuần tự đến khi iTerminated = iNumberOfProcess = 5:

|  |  |  |  |
| --- | --- | --- | --- |
| **Input** | **Ready Queue** | **Temp** | **Terminated** |
| P4 (10, 5) | P2 (3, 4) | P1 (0, 4) |  |
| P5 (12, 1) | P1 (4, 2) |  |  |
|  | P3 (7, 3) |  |  |
|  |  |  |  |
|  |  |  |  |
| iRemain = 2 | iReady = 3 | iTemp = 1 | iTerminated = 0 |

|  |  |  |  |
| --- | --- | --- | --- |
| **Input** | **Ready Queue** | **Temp** | **Terminated** |
| P4 (10, 5) | P1 (4, 2) | P1 (0, 4) | P2 (3, 4) |
| P5 (12, 1) | P3 (7, 3) | P2 (4, 8) |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
| iRemain = 2 | iReady = 2 | iTemp = 2 | iTerminated = 1 |

|  |  |  |  |
| --- | --- | --- | --- |
| **Input** | **Ready Queue** | **Temp** | **Terminated** |
| P4 (10, 5) | P3 (7, 3) | P1 (0, 4) | P2 (3, 4) |
| P5 (12, 1) |  | P2 (4, 8) | P1 (0, 6) |
|  |  | P1 (8, 10) |  |
|  |  |  |  |
|  |  |  |  |
| iRemain = 2 | iReady = 1 | iTemp = 3 | iTerminated = 2 |

|  |  |  |  |
| --- | --- | --- | --- |
| **Input** | **Ready Queue** | **Temp** | **Terminated** |
|  | P3 (7, 3) | P1 (0, 4) | P2 (3, 4) |
|  | P4 (10, 5) | P2 (4, 8) | P1 (0, 6) |
|  | P5 (12, 1) | P1 (8, 10) |  |
|  |  |  |  |
|  |  |  |  |
| iRemain = 0 | iReady = 3 | iTemp = 3 | iTerminated = 2 |

|  |  |  |  |
| --- | --- | --- | --- |
| **Input** | **Ready Queue** | **Temp** | **Terminated** |
|  | P3 (7, 3) | P1 (0, 4) | P2 (3, 4) |
|  | P4 (10, 5) | P2 (4, 8) | P1 (0, 6) |
|  | P5 (12, 1) | P1 (8, 10) |  |
|  |  |  |  |
|  |  |  |  |
| iRemain = 0 | iReady = 3 | iTemp = 3 | iTerminated = 2 |

|  |  |  |  |
| --- | --- | --- | --- |
| **Input** | **Ready Queue** | **Temp** | **Terminated** |
|  | P4 (10, 5) | P1 (0, 4) | P2 (3, 4) |
|  | P5 (12, 1) | P2 (4, 8) | P1 (0, 6) |
|  |  | P1 (8, 10) | P3 (7, 3) |
|  |  | P3 (10, 13) |  |
|  |  |  |  |
| iRemain = 0 | iReady = 2 | iTemp = 4 | iTerminated = 3 |

|  |  |  |  |
| --- | --- | --- | --- |
| **Input** | **Ready Queue** | **Temp** | **Terminated** |
|  | P5 (12, 1) | P1 (0, 4) | P2 (3, 4) |
|  | P4 (17, 1) | P2 (4, 8) | P1 (0, 6) |
|  |  | P1 (8, 10) | P3 (7, 3) |
|  |  | P3 (10, 13) |  |
|  |  | P4 (13, 17) |  |
| iRemain = 0 | iReady = 2 | iTemp = 5 | iTerminated = 3 |

|  |  |  |  |
| --- | --- | --- | --- |
| **Input** | **Ready Queue** | **Temp** | **Terminated** |
|  | P4 (17, 1) | P1 (0, 4) | P2 (3, 4) |
|  |  | P2 (4, 8) | P1 (0, 6) |
|  |  | P1 (8, 10) | P3 (7, 3) |
|  |  | P3 (10, 13) | P5 (12, 1) |
|  |  | P4 (13, 17) |  |
|  |  | P5 (17, 18) |  |
| iRemain = 0 | iReady = 1 | iTemp = 6 | iTerminated = 4 |

|  |  |  |  |
| --- | --- | --- | --- |
| **Input** | **Ready Queue** | **Temp** | **Terminated** |
|  |  | P1 (0, 4) | P2 (3, 4) |
|  |  | P2 (4, 8) | P1 (0, 6) |
|  |  | P1 (8, 10) | P3 (7, 3) |
|  |  | P3 (10, 13) | P5 (12, 1) |
|  |  | P4 (13, 17) | P4 (10, 5) |
|  |  | P5 (17, 18) |  |
|  |  | P4 (18, 19) |  |
| iRemain = 0 | iReady = 0 | iTemp = 6 | iTerminated = 5 |

Cuối cùng ta sẽ vẽ biểu đồ Gantt dựa vào số liệu và thứ tự tiến trình của cột Temp, tính Turnaround Time trung bình và Waiting Time trung bình dựa vào cột Terminated.

A line with numbers and a line

Description automatically generated

iATaT = (10+5+6+9+6)/5 = 7.2

iAWT = (4+1+3+4+5)/5 = 3.4

* **Code:**
* #include <stdio.h>
* #include <stdlib.h>
* #include <time.h>
* #include <stdbool.h>
* #include <limits.h>
* #define SORT\_BY\_ARRIVAL 0
* #define SORT\_BY\_PID 1
* #define SORT\_BY\_BURST 2
* #define SORT\_BY\_START 3
* typedef *struct*{
* *int* iPID;
* *int* iArrival, iBurst;
* *int* iStart, iFinish, iWaiting, iResponse, iTaT;
* *int* iPause, iRemaining;
* } PCB;
* *void* inputProcess(*int* *n*, PCB *P*[]) {
* srand((*unsigned*) time(NULL));
* for (*int* i = 0; i < n; i++){
* P[i].iPID= i+1;
* P[i].iArrival = rand() % 21;
* P[i].iBurst = (rand()% 11) +2;
* P[i].iRemaining = P[i].iBurst;
* P[i].iPause = P[i].iArrival;
* P[i].iStart = P[i].iFinish = P[i].iResponse = P[i].iTaT = P[i].iWaiting = 0;
* }
* }
* *void* printProcess(*int* *n*, PCB *P*[]) {
* for (*int* i = 0; i < n; i++){
* printf("P%d(%d, %d)\n", P[i].iPID, P[i].iArrival, P[i].iBurst);
* }
* }
* *void* exportGanttChart (*int* *n*, PCB *P*[]) {
* printf("Gantt Chart:\n");
* printf("%d", P[0].iStart);
* for (*int* i = 0; i < n; i++){
* printf("\_\_\_P[%d]\_\_\_%d", P[i].iPID, P[i].iFinish);
* }
* printf("\n");
* }
* *void* pushProcess(*int* \**n*, PCB *P*[], PCB *Q*) {
* P[\*n] = Q;
* (\*n)++;
* }
* *void* removeProcess(*int* \**n*, *int* *index*, PCB *P*[]) {
* for(*int* i = index; i < \*n; i++){
* P[i]=P[i+1];
* }
* (\*n)--;
* }
* *void* swapProcess(PCB \**P*, PCB \**Q*) {
* PCB temp = \*P;
* \*P = \*Q;
* \*Q = temp;
* }
* *int* partition (PCB *P*[], *int* *low*, *int* *high*, *int* *iCriteria*) {
* if (iCriteria == SORT\_BY\_ARRIVAL) {
* *int* pivot = P[high].iArrival;
* *int* i = (low-1);
* for(*int* j = low; j <= high; j++)
* {
* if (P[j].iArrival < pivot)
* {
* i++;
* swapProcess(&P[i], &P[j]);
* }
* }
* swapProcess(&P[i+1], &P[high]);
* return (i+1);
* } else if (iCriteria == SORT\_BY\_PID) { // Sort by PID
* *int* pivot = P[high].iPID;
* *int* i = (low-1);
* for(*int* j = low; j <= high; j++)
* {
* if (P[j].iPID < pivot)
* {
* i++;
* swapProcess(&P[i], &P[j]);
* }
* }
* swapProcess(&P[i+1], &P[high]);
* return (i+1);
* } else if (iCriteria == SORT\_BY\_BURST) {
* *int* pivot = P[high].iBurst;
* *int* i = (low-1);
* for(*int* j = low; j <= high; j++)
* {
* if (P[j].iBurst < pivot)
* {
* i++;
* swapProcess(&P[i], &P[j]);
* }
* }
* swapProcess(&P[i+1], &P[high]);
* return (i+1);
* } else if (iCriteria == SORT\_BY\_START) {
* *int* pivot = P[high].iStart;
* *int* i = (low-1);
* for(*int* j = low; j <= high; j++)
* {
* if (P[j].iStart < pivot)
* {
* i++;
* swapProcess(&P[i],&P[j]);
* }
* }
* swapProcess(&P[i+1], &P[high]);
* return (i+1);
* }
* }
* *void* quickSort(PCB *P*[], *int* *low*, *int* *high*, *int* *iCriteria*) {
* if(low < high) {
* *int* pi = partition(P, low, high, iCriteria);
* quickSort(P, low, pi-1, iCriteria);
* quickSort(P, pi+1, high, iCriteria);
* }
* }
* *void* calculateAWT(*int* *n*, PCB *P*[]) {
* *int* sum = 0;
* *float* avg = 0;
* for (*int* i = 0; i < n; i++) {
* sum += P[i].iWaiting;
* }
* if (n > 0) avg = (*float*) sum/n;
* printf("Average Waiting time: %f\n", avg);
* }
* *void* calculateATaT(*int* *n*, PCB *P*[]) {
* *int* sum = 0;
* *float* avg = 0;
* for (*int* i = 0; i < n; i++) {
* sum += P[i].iTaT;
* }
* if (n > 0) avg = (*float*) sum/n;
* printf("Average Turnaround time: %f\n", avg);
* }
* *int* main()
* {
* PCB Input[10];
* PCB ReadyQueue[10];
* PCB Temp[40];
* PCB TerminatedArray[10];
* *int* iNumberOfProcess;
* printf("Please input number of Process: ");
* scanf("%d", &iNumberOfProcess);
* *int* iRemain = iNumberOfProcess, iReady = 0, iTerminated = 0, iTemp = 0;
* inputProcess(iNumberOfProcess, Input);
* quickSort(Input, 0, iNumberOfProcess - 1, SORT\_BY\_ARRIVAL);
* printf("--Input--\n");
* printProcess(iRemain, Input);
* *int* q;
* printf("\nPlease input quantum time: ");
* scanf("%d", &q);
* pushProcess(&iReady, ReadyQueue, Input[0]);
* removeProcess(&iRemain, 0, Input);
* ReadyQueue[0].iStart = ReadyQueue[0].iArrival;
* ReadyQueue[0].iResponse = ReadyQueue[0].iStart - ReadyQueue[0].iArrival;
* *bool* flag=false;
* while (iTerminated < iNumberOfProcess) {
* if (ReadyQueue[0].iRemaining <= q)
* ReadyQueue[0].iFinish = ReadyQueue[0].iStart + ReadyQueue[0].iRemaining;
* else
* ReadyQueue[0].iFinish = ReadyQueue[0].iStart + q;
* while (iRemain > 0) {
* if (Input[0].iArrival <= ReadyQueue[0].iFinish) {
* flag=false;
* pushProcess(&iReady, ReadyQueue, Input[0]);
* removeProcess(&iRemain, 0, Input);
* continue;
* }
* else{
* if(iReady==0){
* flag=true;
* pushProcess(&iReady, ReadyQueue, Input[0]);
* removeProcess(&iRemain, 0, Input);
* break;
* }
* else {flag=false;break;}
* }
* }
* if (iReady > 0) {
* ReadyQueue[0].iTaT += ReadyQueue[0].iFinish - ReadyQueue[0].iPause;
* ReadyQueue[0].iWaiting += ReadyQueue[0].iStart - ReadyQueue[0].iPause;
* ReadyQueue[0].iPause = ReadyQueue[0].iFinish;
* if (ReadyQueue[0].iRemaining <= q) {
* ReadyQueue[0].iRemaining = 0;
* pushProcess(&iTemp, Temp, ReadyQueue[0]);
* pushProcess(&iTerminated, TerminatedArray, ReadyQueue[0]);
* removeProcess(&iReady, 0, ReadyQueue);
* }
* else {
* ReadyQueue[0].iRemaining -= q;
* pushProcess(&iTemp, Temp, ReadyQueue[0]);
* pushProcess(&iReady, ReadyQueue, ReadyQueue[0]);
* removeProcess(&iReady, 0, ReadyQueue);
* }
* if(flag)  ReadyQueue[0].iStart = ReadyQueue[0].iArrival;
* else ReadyQueue[0].iStart = Temp[iTemp-1].iFinish;
* ReadyQueue[0].iResponse = ReadyQueue[0].iStart - ReadyQueue[0].iArrival;
* }
* }
* printProcess(iTerminated, TerminatedArray);
* printf("\n===== Round Robin Scheduling =====\n");
* exportGanttChart(iTemp, Temp);
* calculateAWT(iTerminated, TerminatedArray);
* calculateATaT(iTerminated, TerminatedArray);
* return 0;
* }

**Test case 1:**

A black background with white dots

Description automatically generated

A black background with white text

Description automatically generated **Test case 2:**

**Test case 3:**

A screen shot of a computer

Description automatically generated