THE NATIONAL INSTITUTE OF ENGINEERING, MYSURU

(An Autonomous Institute under VTU, Belagavi)

In partial fulfillment of the requirements for the award of degree of

Bachelor of Engineering

in

Computer Science and Engineering

Submitted by

Adarsh A (4NI19CS006)
Anup Siddu R S (4NI19CS024)

Course Instructor

Dr. Jayasri B S

Professor & Dean (EAB)

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

THE NATIONAL INSTITUTE OF ENGINEERING

Mysore-570 008

2020-2021

INDEX

1. PAGE REPLACEMENT ALGORITHM - FIFO (First In, First Out)

Page Replacement Algorithm- It decides which page to remove, also called swap out when a new page needs to be loaded into the main memory. Page Replacement happens when a requested page is not present in the main memory and the available space is not sufficient for allocation to the requested page.

A page replacement algorithm tries to select which pages should be replaced so as to minimize the total number of page misses. There are many different page replacement algorithms.

FIFO - This is the simplest page replacement algorithm. In this algorithm, the operating system keeps track of all pages in the memory in a queue, the oldest page is in the front of the queue. When a page needs to be replaced page in the front of the queue is selected for removal.

Page- (**memory page**, or **virtual page)** is a fixed-length contiguous block of virtual memory, described by a single entry in the page table.

- It is the smallest unit of data for memory management in a virtual memory operating system.

Page frame- is the smallest fixed-length contiguous block of physical memory into which memory pages are mapped by the operating system.

Advantages –

1. It is simple and easy to understand & implement.

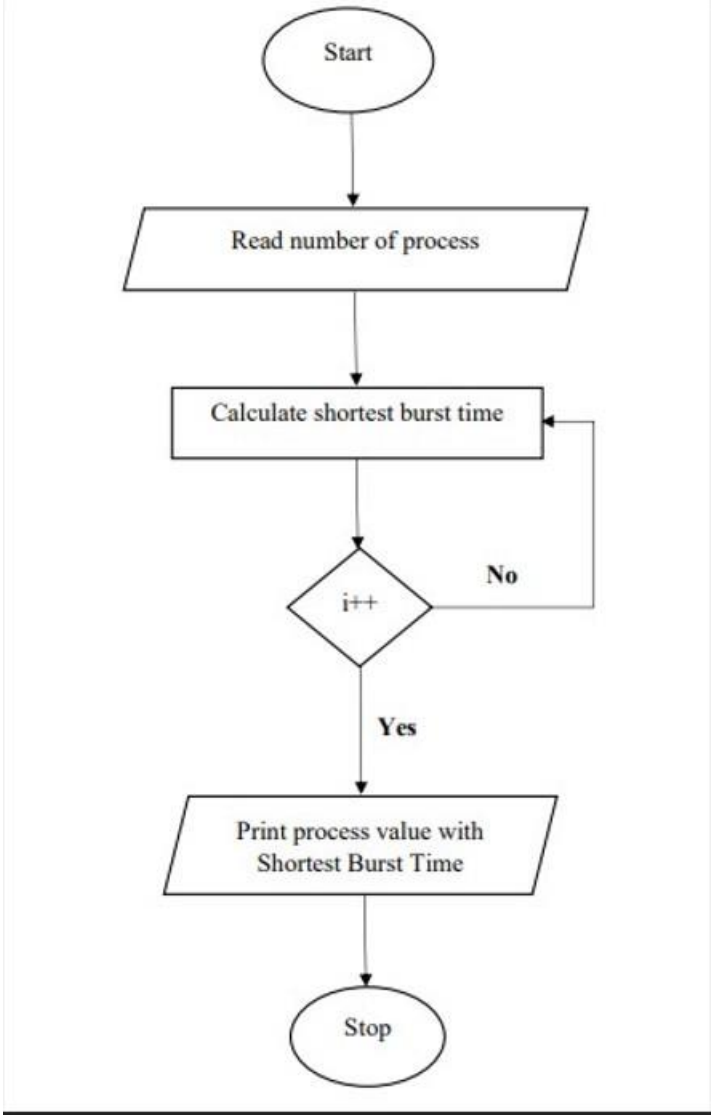
Disadvantages –

1. The process effectiveness is low.
2. When we increase the number of frames while using FIFO, we are giving more memory to processes. So, page fault should decrease, but here the page faults are increasing. This problem is called as Belady's Anomaly.

Algorithm – FIFO

1. Start the process
2. Declare the size with respect to page length
3. Check the need of replacement from the page to memory
4. Check the need of replacement from old page to new page in memory
5. Form a queue to hold all pages
6. Insert the page require memory into the queue
7. Check for bad replacement and page fault
8. Get the number of processes to be inserted
9. Display the values
10. Stop the process

FLOW CHART


Start
Read number of process
Calculate shortest burst time
i++
No
Yes
Print process value with
Shortest Burst Time
Stop

```
{ int i,j,n,a[50],frame[10],no,k,avail,count=0, count1=0;
 for(i=0;i<no;i++)
        frame[i] <- -1;    //set all frames to -1 at beginning
    j <-0;                 // represents frame number (0,1,2) for 3 frames
    cout<<"\nref string\t \tpage frames\n";
    for(i=1;i<=n;i++)
    {
        cout<<"\t"<<a[i];   // a[i] prints ref string
        avail <- 0;                 // to check faults
        for(k=0;k<no;k++)
          if(frame[k]==a[i]){
                avail <- 1;            //to check hits
                count1++;
                cout<<"  ->";
                for(k=0;k<no;k++)cout<<frame[k];
          }
           if (avail==0){
            frame[j] <- a[i];// sets ref string value to frame no. specified by J
             j=(j+1)%no;          //to indicate next frame to be replaced
             count++ ;
             for(k=0;k<no;k++)cout<<frame[k];
            }
        cout<<"\n";
    }
}
```

Page Hit – If CPU tries to retrieve the needed page from main memory, and that page is existed in the main memory (RAM), then it is known as “PAGE HIT”.

Page fault- is an interruption that occurs when a software program attempts to access a memory block not currently stored in the system's RAM. This exception tells the operating system to find the block in virtual memory so it can be sent from a device's storage (SSD or HD) to RAM.

Program -

```
#include<iostream>
using namespace std;
int main()
{
    int i,j,n,a[50],frame[10],no,k,avail,count=0, count1=0;
    cout<<"\n ENTER THE NUMBER OF PAGES:\n";
    cin>>n;
    cout<<"\n ENTER THE PAGE NUMBER :\n";
    for(i=1;i<=n;i++)
        cin>>a[i];
    cout<<"\n ENTER THE NUMBER OF FRAMES :";
    cin>>no;
    for(i=0;i<no;i++)
        frame[i]= -1;    //set all frames to -1 at beginning
    j=0;                 // represents frame number (0,1,2) for 3 frames
    cout<<"\nref string\t \tpage frames\n";
    for(i=1;i<=n;i++)
    {
        cout<<"\t"<<a[i]<<"\t";   // a[i] prints ref string
        avail=0;                 // to check faults
        for(k=0;k<no;k++)
            if(frame[k]==a[i]){
                avail=1;            //to check hits
                count1++;
                cout<<"  ->";
                 for(k=0;k<no;k++)
                    cout<<"\t"<<frame[k];
            }
            if (avail==0){
                frame[j]=a[i];       // sets ref string value to frame no.
specified by J
                j=(j+1)%no;          //to indicate next frame to be replaced
                count++;
                for(k=0;k<no;k++)
                    cout<<"\t"<<frame[k];
            }
        cout<<"\n";
    }
    cout<<"\n\nPage Fault Is: "<<count;
    cout<<"\nMiss Ratio:  "<<count<<"/"<<n<<" %";
    cout<<"\n\nNumber of Hits Is: "<<count1;
    cout<<"\nHit Ratio: "<<count1<<"/"<<n<<" %";
    return 0;
}
```

Output –

```
 ENTER THE NUMBER OF PAGES:
15

 ENTER THE PAGE NUMBER :
 7 0 1 2 0 3 0 4 2 3 0 3 2 1 2

 ENTER THE NUMBER OF FRAMES :3

ref string              page frames
        7               7       -1      -1
        0               7       0       -1
        1               7       0       1
        2               2       0       1
        0         ->    2       0       1
        3               2       3       1
        0               2       3       0
        4               4       3       0
        2               4       2       0
        3               4       2       3
        0               0       2       3
        3         ->    0       2       3
        2         ->    0       2       3
        1               0       1       3
        2               0       1       2


Page Fault Is: 12
Miss Ratio:  12/15 %

Number of Hits Is: 3
Hit Ratio: 3/15 %
PS C:\Users\adars\OneDrive\Desktop\4th sem\OOPlab>
```

2. SCHEDULING ALGORITHM – SJF (Shortest Job First)

Shortest job first (SJF) or shortest job next, is a scheduling policy that selects the waiting process with the smallest execution time to execute next. SJF is a **non-preemptive** algorithm.

To successfully implement it, the burst time/duration time of the processes should be known to the processor in advance, which is practically not feasible all the time.
This scheduling algorithm is optimal if all the jobs/processes are available at the same time. (Either Arrival time is 0 for all, or Arrival time is same)

Non-preemptive- means that once a process has been removed from the waiting queue and given CPU time, it will execute until completed or terminated. The algorithm removes processes with minimum arrival time from the waiting queue and executes them until the process is completed. The process then shifts the CPU to other processes in the waiting queue.

Advantages -

- Shortest Job first has the advantage of having a minimum average waiting time among all scheduling algorithms.
- It is a Greedy Algorithm.

Disadvantages-

- It cannot be implemented practically since burst time of the process cannot be known in advance.
- It leads to starvation for processes with larger burst time.
- Priorities cannot be set for the processes.
- Processes with larger burst time have poor response time.

Algorithm- SJF

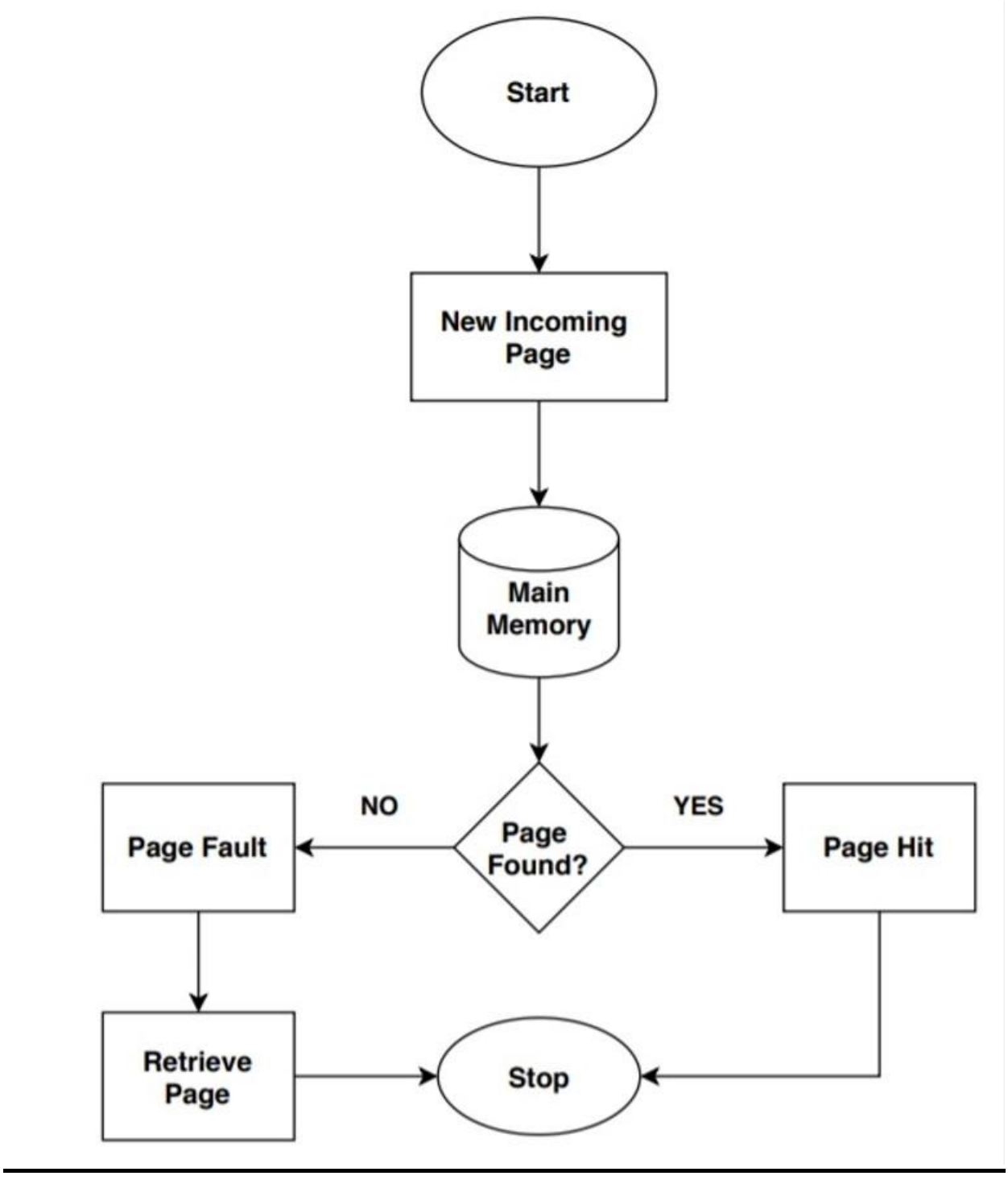
- Sort all the process according to the arrival time.
- Then select that process which has minimum arrival time and minimum Burst time.
- By the completion of previous process – if set of process has already arrived then select that process among the pool which is having minimum Burst time.

```
void arrangeArrival(int num, int mat[][7])
{                                                    // sorted acc to Arrival time-
(bubble sort)
    for (int i = 0; i < num; i++) {
        for (int j = 0; j < num - i - 1; j++) {
            if (mat[j][1] > mat[j + 1][1]) {
                for (int k = 0; k < 6; k++) {
                    swap(mat[j][k], mat[j + 1][k]);  //swaps process no., AT & BT,
                }
            }
        }
    }
}
```

```
void completionTime(int num, int mat[][7])
{
    int temp, val;           // below 3 r for only 1st process after arranging
    mat[0][3] = mat[0][1] + mat[0][2];   // CT for first process
    mat[0][5] = mat[0][3] - mat[0][1];   //TAT=CT-AT
    mat[0][4] = mat[0][5] - mat[0][2];
    mat[0][6] = mat[0][1] -mat[0][1]; //WT= TAT -BT

    for (int i = 1; i < num; i++) {   // loop start from 2nd process
        temp = mat[i - 1][3];    //store CT of prev process
        int low = mat[i][2];     //store BT of cur process
        for (int j = i; j < num; j++) {
            if (temp >= mat[j][1] && low >= mat[j][2]) { //if CT of previous >= AT
of cur process & Cur BT >= arrived BT
                low = mat[j][2];    // thn set arrived BT as low(shortest job)
                val = j;            // store that process no. with low BT in
variable
            }
            // basically checks BT of all the arrived processes n sets lowest BT
        }
        mat[val][3] = temp + mat[val][2];     // cal CT-3,WT-4,TAT-5 for j process
        mat[val][5] = mat[val][3] - mat[val][1];
        mat[val][4] = mat[val][5] - mat[val][2];
        mat[val][6] = temp - mat[val][1];
                for (int k = 0; k < 7; k++) {
            swap(mat[val][k], mat[i][k]);    // to get in order in wch cpu
allocates the process
        }
    }
}
```

Flow chart


Start
New Incoming Page
Main Memory
Page Found?
NO
YES
Page Fault
Page Hit
Retrieve Page
Stop

Program –

```
#include <iostream>
using namespace std;
int mat[10][7];     // total 6 columns req

void swap(int* a, int* b)
{
    int temp = *a;
    *a = *b;
    *b = temp;
}

void arrangeArrival(int num, int mat[][7])
{                                                  // sorted acc to Arrival time-
(bubble sort)
    for (int i = 0; i < num; i++) {
        for (int j = 0; j < num - i - 1; j++) {
            if (mat[j][1] > mat[j + 1][1]) {
                for (int k = 0; k < 6; k++) {
                    swap(mat[j][k], mat[j + 1][k]);  //swaps process no., AT & BT,
                }
            }
        }
    }
}

void completionTime(int num, int mat[][7])
{
    int temp, val;           // below 3 r for only 1st process after arranging
    mat[0][3] = mat[0][1] + mat[0][2];   // CT for first process
    mat[0][5] = mat[0][3] - mat[0][1];   //TAT=CT-AT
    mat[0][4] = mat[0][5] - mat[0][2];
    mat[0][6] = mat[0][1] -mat[0][1]; //WT= TAT -BT

    for (int i = 1; i < num; i++) {   // loop start from 2nd process
        temp = mat[i - 1][3];    //store CT of prev process
        int low = mat[i][2];     //store BT of cur process
        for (int j = i; j < num; j++) {
            if (temp >= mat[j][1] && low >= mat[j][2]) { //if CT of previous >= AT
of cur process & Cur BT >= arrived BT
                low = mat[j][2];    // thn set arrived BT as low(shortest job)
                val = j;            // store that process no. with low BT in
variable
            }
            // basically checks BT of all the arrived processes n sets lowest BT
        }
```

```
        mat[val][3] = temp + mat[val][2];     // cal CT-3,WT-4,TAT-5 for j process
        mat[val][5] = mat[val][3] - mat[val][1];
        mat[val][4] = mat[val][5] - mat[val][2];
        mat[val][6] = temp - mat[val][1];
                for (int k = 0; k < 7; k++) {
            swap(mat[val][k], mat[i][k]);    // to get in order in wch cpu
allocates the process
        }
    }
}

int main()
{
    int num, temp;
    float avgtat=0, avgwt=0,avgRT=0;
    cout << "Enter number of Process: ";
    cin >> num;
    cout << "...Enter the process ID...\n";
    for (int i = 0; i < num; i++) {
        cout << "...Process " << i + 1 << "...\n";
        cout << "Enter Process Id: ";     // 1st col- process id
        cin >> mat[i][0];
        cout << "Enter Arrival Time: ";   //2nd col- arrival time
        cin >> mat[i][1];
        cout << "Enter Burst Time: ";     //3rd col- burst time
        cin >> mat[i][2];
    }
    cout << "Before Arrange...\n";
    cout << "Process ID\tArrival Time\tBurst Time\n";
    for (int i = 0; i < num; i++) {
        cout << mat[i][0] << "\t\t" << mat[i][1] << "\t\t"
             << mat[i][2] << "\n";
    }
    arrangeArrival(num, mat);
    completionTime(num, mat);
    cout << "Final Result...\n";
    cout << "Process ID\tArrival Time\tBurst Time\tCompletion Time \tWaiting "
            "Time\tTurnaround Time\tResponse Time\n";
    for (int i = 0; i < num; i++) {
        cout << mat[i][0] << "\t\t" << mat[i][1] << "\t\t"
             << mat[i][2] << "\t\t" << mat[i][3] << "\t\t\t"
             << mat[i][4] << "\t\t"<< mat[i][5] << "\t\t"<< mat[i][6]<< "\n";
    }
```

```
    for(int i=0;i<num;i++)
    {
        avgtat+=mat[i][5];
        avgwt+=mat[i][4];
        avgRT+=mat[i][6];
    }
    avgtat= avgtat/num;
    avgwt= avgwt/num;
    avgRT= avgRT/num;
    cout<<"\n\nGrant Chart (process order): ";
    for (int i = 0; i < num; i++)
    cout<<mat[i][0]<<" ";
    cout<<"\nAverage turn around Time = "<<avgtat<<" ms";
    cout<<"\nAverage waiting Time = "<<avgwt<<" ms";
    cout<<"\nAverage Response Time = "<<avgRT<<" ms";
}
```

- **Arrival Time:** Time at which the process arrives in the ready queue.
- **Completion Time**: Time at which process completes its execution. **Burst Time:** Time required by a process for CPU execution.
- **Turn Around Time (TAT):** Time Difference between completion time and arrival time.
  Turn Around Time = Completion Time – Arrival Time.
- **Waiting Time (WT):** Time Difference between turn-around time and burst time. Waiting Time = Turn Around Time – Burst Time
- **Response time (RT):** It is the time spent when the process is in the ready state and gets the CPU for the first time

Output -

```
Enter number of Process: 5
...Enter the process ID...
...Process 1...
Enter Process Id: 1
Enter Arrival Time: 0
Enter Burst Time: 4
...Process 2...
Enter Process Id: 2
Enter Arrival Time: 2
Enter Burst Time: 6
...Process 3...
Enter Process Id: 3
Enter Arrival Time: 1
Enter Burst Time: 5
...Process 4...
Enter Process Id: 4
Enter Arrival Time: 5
Enter Burst Time: 3
...Process 5...
Enter Process Id: 5
Enter Arrival Time: 4
Enter Burst Time: 2
Before Arrange...
Process ID      Arrival Time    Burst Time
1               0               4
2               2               6
3               1               5
4               5               3
5               4               2
Final Result...
Process ID      Arrival Time    Burst Time      Completion Time         Waiting Time    Turnaround Time Response Time
1               0               4               4                       0               4               0
5               4               2               6                       0               2               0
4               5               3               9                       1               4               1
3               1               5               14                      8               13              8
2               2               6               20                      12              18              12


Grant Chart (process order): 1 5 4 3 2
Average turn around Time = 8.2 ms
Average waiting Time = 4.2 ms
Average Response Time = 4.2 ms
PS C:\Users\adars\OneDrive\Desktop\4th sem\OOPlab>
```

- In SJF Response Time will be equal to Wait time

CONCLUSION

In this way the Non preemptive Shortest Job First (SJF) algorithm which is a scheduling algorithm helps for CPU processes scheduling. In this scheduling algorithm the ready queue is organized according to the burst time of process which has shortest job.

Accordingly in a computer operating system that uses paging for virtual memory management, the Page Replacement Algorithm-(FIFO) decides which memory pages to page out, sometimes called swap out, or write to disk, when a page of memory needs to be allocated. Oldest page is replaced first.

GITHUB - https://github.com/AA-001/OS--Project

GITHUB User- Adarsh A