Bharatiya Vidya Bhavan’s

Sardar Patel Institute of Technology

Bhavan’s Campus, Munshi Nagar, Andheri (West), Mumbai-400058-India
(Autonomous College Affiliated to University of Mumbai)

Computer Engineering Department & Information Technology Engineering Department

Academic Year: 2021-2022

Class: S.Y.B.Tech **Sem.:** 4 **Course:** OS

Name	**Pratik Pujari**		
UID no.	**2020300054**	**Class:**	**Comps C Batch**
Experiment No.	7		

AIM:	To implement Disk Scheduling Algorithms
THEORY:	**Disk Scheduling** As we know, a process needs two type of time, CPU time and IO time. For I/O, it requests the Operating system to access the disk. However, the operating system must be fare enough to satisfy each request and at the same time, operating system must maintain the efficiency and speed of process execution.The technique that operating system uses to determine the request which is to be satisfied next is called disk scheduling. **Seek Time** Seek time is the time taken in locating the disk arm to a specified track where the read/write request will be satisfied. **Rotational Latency** It is the time taken by the desired sector to rotate itself to the position from where it can access the R/W heads. **Transfer Time** It is the time taken to transfer the data. **Disk Access Time** Disk access time is given as, Disk Access Time = Rotational Latency + Seek Time + Transfer Time **Disk Response Time** It is the average of time spent by each request waiting for the IO operation. **Purpose of Disk Scheduling** The main purpose of disk scheduling algorithm is to select a disk request from the queue of IO requests and decide the schedule when this request will be processed.

Goal of Disk Scheduling Algorithm

- Fairness
- High throughout
- Minimal traveling head time

Disk Scheduling Algorithms

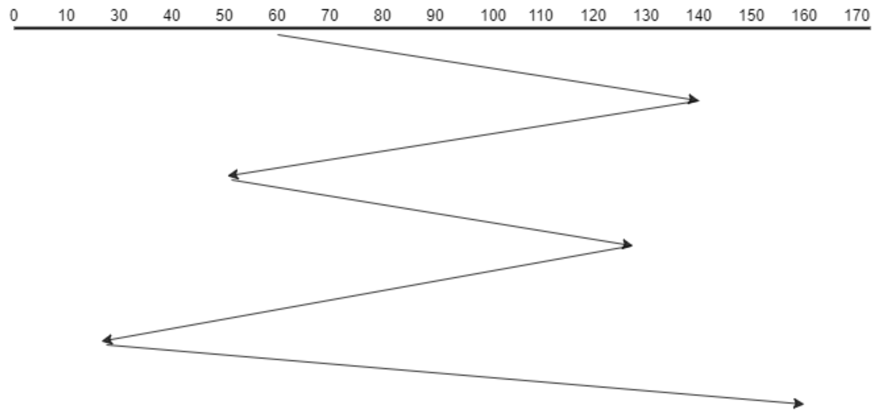
The list of various disks scheduling algorithm is given below. Each algorithm is carrying some advantages and disadvantages. The limitation of each algorithm leads to the evolution of a new algorithm.

- FCFS scheduling algorithm
- SSTF (shortest seek time first) algorithm
- SCAN scheduling
- C-SCAN scheduling
- LOOK Scheduling
- C-LOOK scheduling

First Come First Serve (FCFS)

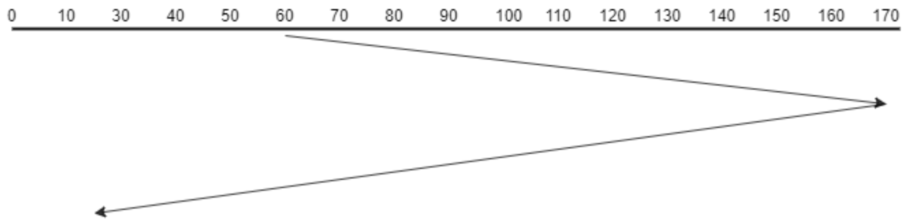
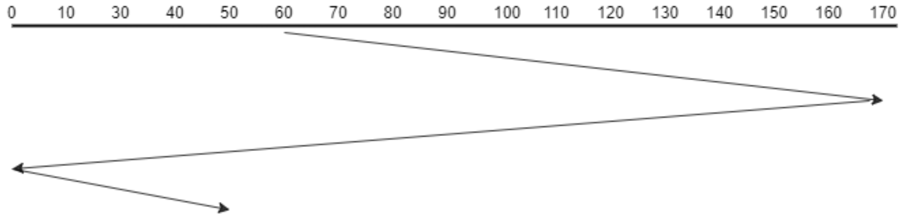
In this algorithm, the requests are served in the order they come. Those who come first are served first. This is the simplest algorithm.

Eg. Suppose the order of requests are 70, 140, 50, 125, 30, 25, 160 and the initial position of the Read-Write head is 60.



Seek Time = Distance Moved by the disk arm = (140-70)+(140-50)+(125-50)+(125-30)+(30-25)+(160-25)=480

SCAN

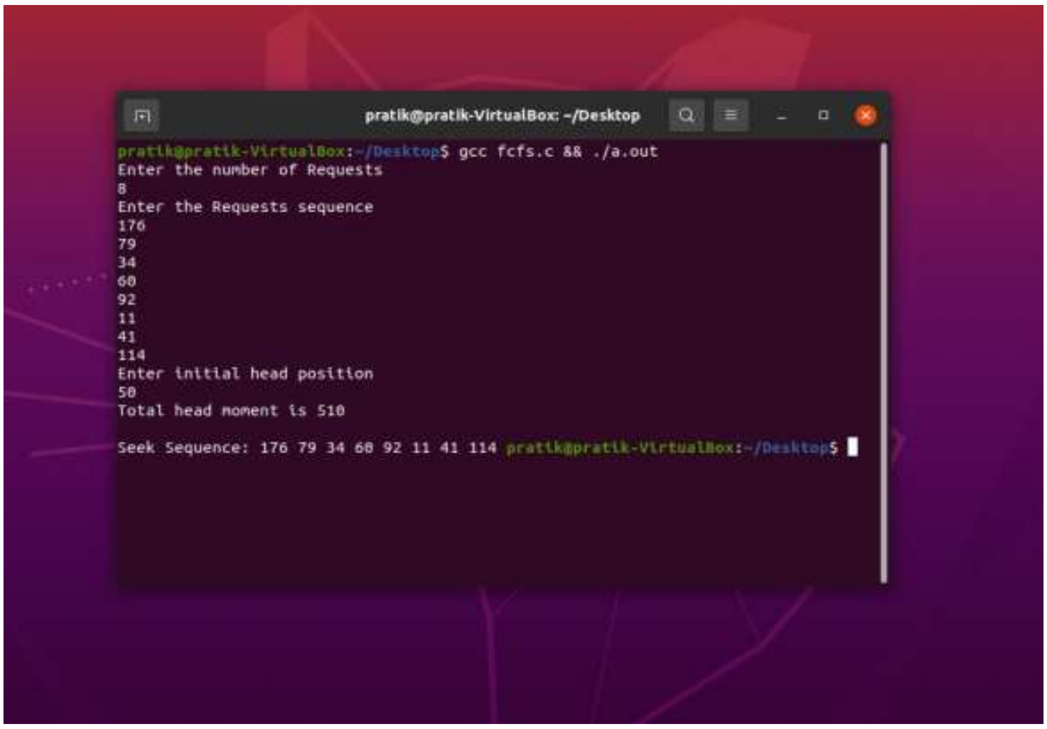
	In this algorithm, the disk arm moves in a particular direction till the end and serves all the requests in its path, then it returns to the opposite direction and moves till the last request is found in that direction and serves all of them. Eg. Suppose the order of requests are 70, 140, 50, 125, 30, 25, 160 and the initial position of the Read-Write head is 60. And it is given that the disk arm should move towards the larger value.  Seek Time = Distance Moved by the disk arm = (170-60)+(170-25)=255 **C-SCAN** This algorithm is the same as the SCAN algorithm. The only difference between SCAN and C-SCAN is, it moves in a particular direction till the last and serves the requests in its path. Then, it returns in the opposite direction till the end and doesn't serve the request while returning. Then, again reverses the direction and serves the requests found in the path. It moves circularly. Eg. Suppose the order of requests are 70, 140, 50, 125, 30, 25, 160 and the initial position of the Read-Write head is 60. And it is given that the disk arm should move towards the larger value.  Seek Time = Distance Moved by the disk arm = (170-60)+(170-0)+(50-0)=330

FCFS

CODE:

```
#include<stdio.h>
#include<stdlib.h>
int main()
{
    int RQ[100],i,n,TotalHeadMoment=0,initial;
    printf("Enter the number of Requests\n");
    scanf("%d",&n);
    printf("Enter the Requests sequence\n");
    for(i=0;i<n;i++)
     scanf("%d",&RQ[i]);
    printf("Enter initial head position\n");
    scanf("%d",&initial);

    // logic for FCFS disk scheduling

    for(i=0;i<n;i++)
    {
        TotalHeadMoment=TotalHeadMoment+abs(RQ[i]-initial);
        initial=RQ[i];
    }

    printf("Total head moment is %d",TotalHeadMoment);
    return 0;

}
```

OUTPUT:



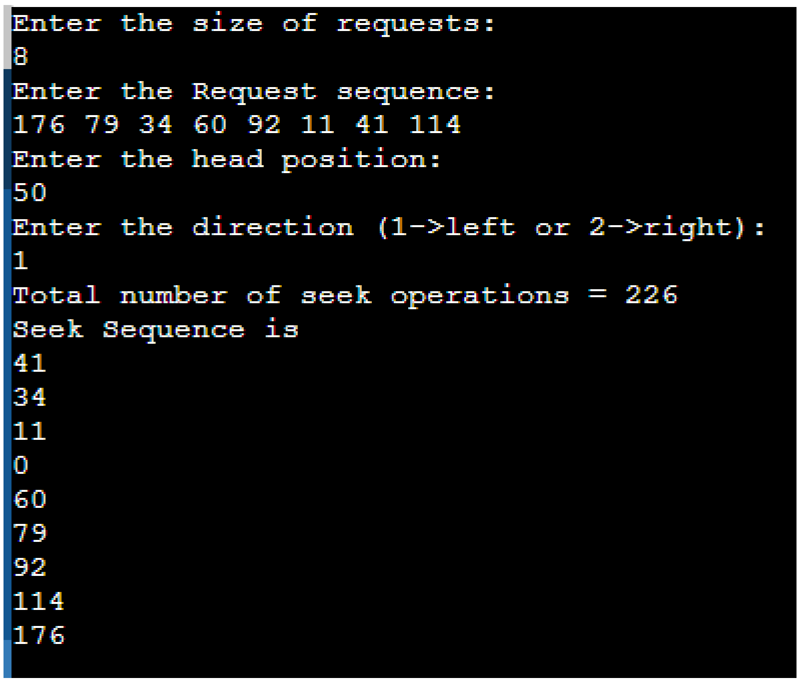
```
pratik@pratik-VirtualBox:~/Desktop$ gcc fcfs.c && ./a.out
Enter the number of Requests
8
Enter the Requests sequence
176
79
34
60
92
11
41
114
Enter initial head position
50
Total head moment is 510

Seek Sequence: 176 79 34 60 92 11 41 114 pratik@pratik-VirtualBox:~/Desktop$ 
```

SCAN

CODE:

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

int disk_size = 200;

void SCAN(int arr[], int head, string direction,int size)
{
    int seek_count = 0;
    int distance, cur_track;
    vector<int> left, right;
    vector<int> seek_sequence;

    // appending end values
    // which has to be visited
    // before reversing the direction
```

```
if (direction == "left")
    left.push_back(0);
else if (direction == "right")
    right.push_back(disk_size - 1);

for (int i = 0; i < size; i++) {
    if (arr[i] < head)
        left.push_back(arr[i]);
    if (arr[i] > head)
        right.push_back(arr[i]);
}

// sorting left and right vectors
std::sort(left.begin(), left.end());
std::sort(right.begin(), right.end());

// run the while loop two times.
// one by one scanning right
// and left of the head
int run = 2;
while (run--) {
    if (direction == "left") {
        for (int i = left.size() - 1; i >= 0; i--) {
            cur_track = left[i];

            // appending current track to seek sequence
            seek_sequence.push_back(cur_track);

            // calculate absolute distance
            distance = abs(cur_track - head);

            // increase the total count
            seek_count += distance;

            // accessed track is now the new head
            head = cur_track;
        }
        direction = "right";
        }
        else if (direction == "right") {
            for (int i = 0; i < right.size(); i++) {
                cur_track = right[i];
                // appending current track to seek sequence
                seek_sequence.push_back(cur_track);

                // calculate absolute distance
                distance = abs(cur_track - head);

                // increase the total count
                seek_count += distance;

                // accessed track is now new head
                head = cur_track;
            }
            direction = "left";
        }
    }

    cout << "Total number of seek operations = "
         << seek_count << endl;

    cout << "Seek Sequence is" << endl;

    for (int i = 0; i < seek_sequence.size(); i++) {
        cout << seek_sequence[i] << endl;
    }
}

// Driver code
int main()
{

    // request array
    int size;
    cout<<"Enter the size of requests: "<<endl;
    cin>>size;
```

	```
	        int arr[size];
	        cout<<"Enter the Request sequence: "<<endl;
	        for(int i=0;i < size;i++){
	                cin>>arr[i];}
	        cout<<"Enter the head position: "<<endl;
	        int head;
	        cin>>head;
	        cout<<"Enter the direction (1->left or 2->right): 
	"<<endl;
	        int direction;
	        cin>>direction;
	    if(direction==1)
	        SCAN(arr, head, "left",size);
	    else
	        SCAN(arr, head, "right",size);
	    return 0;
	}
	```
CODE:	

```
Enter the size of requests:
8
Enter the Request sequence:
176 79 34 60 92 11 41 114
Enter the head position:
50
Enter the direction (1->left or 2->right):
1
Total number of seek operations = 226
Seek Sequence is
41
34
11
0
60
79
92
114
176
```

Bharatiya Vidya Bhavan's

Sardar Patel Institute of Technology

Bhavan's Campus, Munshi Nagar, Andheri (West), Mumbai-400058-India
(Autonomous College Affiliated to University of Mumbai)

Computer Engineering Department &
Information Technology Engineering Department

Academic Year: 2021-2022

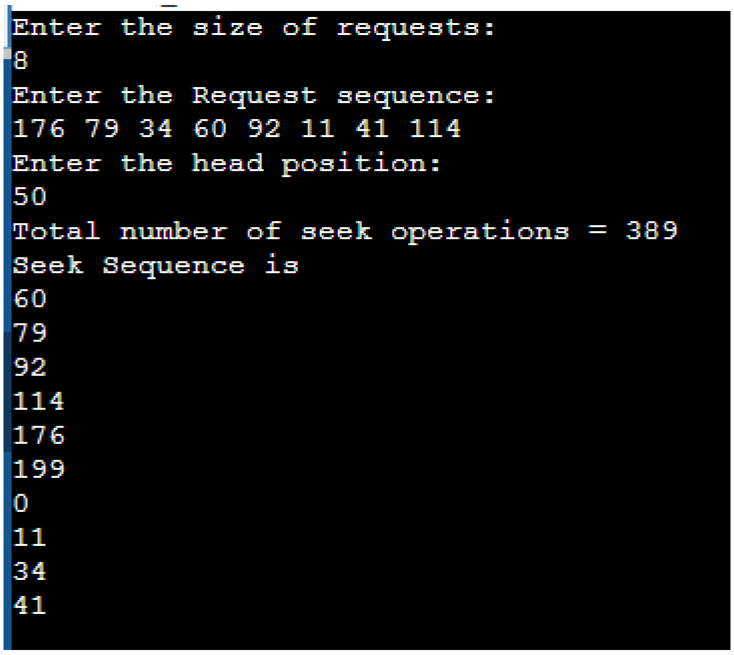
Class: S.Y.B.Tech **Sem.:** 4 **Course:** OS

C-SCAN	
CODE:	see below

```
// C++ program to demonstrate
// C-SCAN Disk Scheduling algorithm
#include <bits/stdc++.h>
using namespace std;

// Code by Vikram Chaurasia


void CSCAN(int arr[], int head,int size,int disk_size)
{
    int seek_count = 0;
    int distance, cur_track;
    vector<int> left, right;
    vector<int> seek_sequence;

    // appending end values
    // which has to be visited
    // before reversing the direction
    left.push_back(0);
    right.push_back(disk_size - 1);

    // tracks on the left of the
    // head will be serviced when
    // once the head comes back
    // to the beginning (left end).
    for (int i = 0; i < size; i++) {
        if (arr[i] < head)
            left.push_back(arr[i]);
        if (arr[i] > head)
            right.push_back(arr[i]);
    }

    // sorting left and right vectors
    std::sort(left.begin(), left.end());
    std::sort(right.begin(), right.end());
```

```
// first service the requests
// on the right side of the
// head.
for (int i = 0; i < right.size(); i++) {
    cur_track = right[i];
    // appending current track to seek sequence
    seek_sequence.push_back(cur_track);

    // calculate absolute distance
    distance = abs(cur_track - head);

    // increase the total count
    seek_count += distance;

    // accessed track is now new head
    head = cur_track;
}

// once reached the right end
// jump to the beginning.
head = 0;

// adding seek count for head returning from 199 to 0
seek_count += (disk_size - 1);

// Now service the requests again
// which are left.
for (int i = 0; i < left.size(); i++) {
    cur_track = left[i];

    // appending current track to seek sequence
    seek_sequence.push_back(cur_track);

    // calculate absolute distance
    distance = abs(cur_track - head);

    // increase the total count
```

```
        seek_count += distance;

        // accessed track is now the new head
        head = cur_track;
    }

    cout << "Total number of seek operations = "
         << seek_count << endl;

    cout << "Seek Sequence is" << endl;

    for (int i = 0; i < seek_sequence.size(); i++) {
        cout << seek_sequence[i] << endl;
    }
}

// Driver code
int main()
{
    int size;
    cout<<"Enter the size of requests: "<<endl;
    cin>>size;
        int arr[size];
        cout<<"Enter the Request sequence: "<<endl;
        for(int i=0;i < size;i++){
                cin>>arr[i];}
        cout<<"Enter the head position: "<<endl;
        int head;
        cin>>head;
    CSCAN(arr,head,size,200);
    return 0;

}
```

<table>
<tr><td><b>OUTPUT:</b></td><td>

```
Enter the size of requests:
8
Enter the Request sequence:
176 79 34 60 92 11 41 114
Enter the head position:
50
Total number of seek operations = 389
Seek Sequence is
60
79
92
114
176
199
0
11
34
41
```
</td></tr>
<tr><td><b>CONCLUSION:</b></td><td>Learnt about the different disk scheduling algorithms. The different algorithm like FCFS, SCAN, C-SCAN. Learnt about the Disc Scheduling algorithm to find the total number of seek operations done to access all the requested tracks if that algorithm is used.</td></tr>
</table>