Jaypee Institute of Information Technology,
Sector -128, Noida

Operating Systems and System Programming Lab
End Term Project

Topic- Comparative Analysis of Disk Scheduling Algorithms

Submitted by:

Rashika Agarwal	9919103028
Shubh Gupta	9919103022
Divya Shikhar Chauhan	9919103021
Vaibhav	9919103011

Submitted to-
Dr. Anubhuti Roda Mohindra
(Department of CSE & IT, JIIT-128)

December 2021

Chapter 1: Problem Statement

A process needs two types of time, CPU time and I/O time. For I/O, it requests the Operating system to access the disk. However, the operating system must be fair enough to satisfy each request and at the same time, the operating system must maintain the efficiency and speed of process execution. So, it must determine which request should be satisfied next.

In this report, we will have a comparative analysis of various disk scheduling algorithms.

Chapter 2: Abstract

Disk scheduling is a policy of the operating system to decide which I/O request is going to be satisfied foremost. The goal of disk scheduling algorithms is to maximize the throughput and minimize the response time. The present piece of investigation documents the comparative analysis of six different disk scheduling algorithms viz. First Come First Serve (FCFS), Shortest Seek Time First (SSTF), Scan, C-Scan, Look and C-look disk scheduling by comparing their head movement in different runs. The implementation is carried out by creating an interface to calculate total head movement of these six algorithms.

Chapter 3: Introduction

The file system can be viewed logically in three different divisions i.e. user, programmer interface to the file system and secondary storage structure. The lowest level of the file system is secondary storage structure and disk is the main secondary storage device that is generally divided into tracks, cylinders and sectors and stores the data permanently. The I/O operation depends on the computer system, the operating system, and the nature of the I/O channel and disk controller hardware .

The file system can be viewed logically in three different divisions i.e. user, programmer interface to the file system and secondary storage structure. The lowest level of the file system is secondary storage structure and disk is the main secondary storage device that is generally divided into tracks, cylinders and sectors and stores the data permanently. The I/O operation depends on the computer system, the operating system, and the nature of the I/O channel and disk controller hardware. The user programs make use of the data on the disk by means of I/O requests. Data is stored on both surfaces of a series of magnetic disks called platters that are connected by a single spindle. The surface of a platter is logically divided into tracks that are further subdivided into sectors and the set of tracks that are at one arm position form a cylinder.

One read-write head per disk surface is used to access the data and all read-write heads are attached to a single moving arm. The segment of the disk surface where the data is read or written must revolve under the read-write head for accessing the data.
The key responsibility of the operating system is to efficiently use the hardware of the computer system.
For most disks, the seek time leads to latency time and transfer time, so reducing the mean seek time can improve system performance to a large extent.

In multiprogramming systems, processes running concurrently may generate requests for reading and writing disk records. The operating system handles these I/O requests from the queue and processes them one by one. The algorithm used to choose which I/O request is going to be fulfilled earliest is called disk scheduling algorithm. The different disk scheduling algorithms are First Come First Serve, Shortest Seek Time First, Scan, Look, Circular Scan and Circular Look. The main objectives for any disk scheduling algorithm are minimizing the response time and maximizing the throughput.

3.1 Disk Performance Parameters:

Seek Time: Seek Time is the time taken by the arm head to move within the tracks for read or write operations. Minimum the average seek time, maximum is the efficiency of the algorithm.

Rotational Latency: Rotational latency is the time taken by the disk to position the sector under the disk arm.

Total Access Time: It is the time taken for transfer of data + Seek time + Rotational latency.

Head Movement: It is the movement of the disk head arm between the tracks of the disk. Since HeadMovement and Seek Time are dependent on each other, minimum the head movement, minimum the average seek time.

Transfer Time: The transfer time to or from the disk depends on the rotation speed of the disk.

T = b/rN

Where T = Transfer Time

b = Number of bytes to be transferred

N = Number of bytes on a track

r = Rotation speed, in revolutions per second

Thus the total average access time can be expressed as:

$$\mathbf{Ta = Ts + 1/2r + b/rN}$$

where Ts is the average seek time.

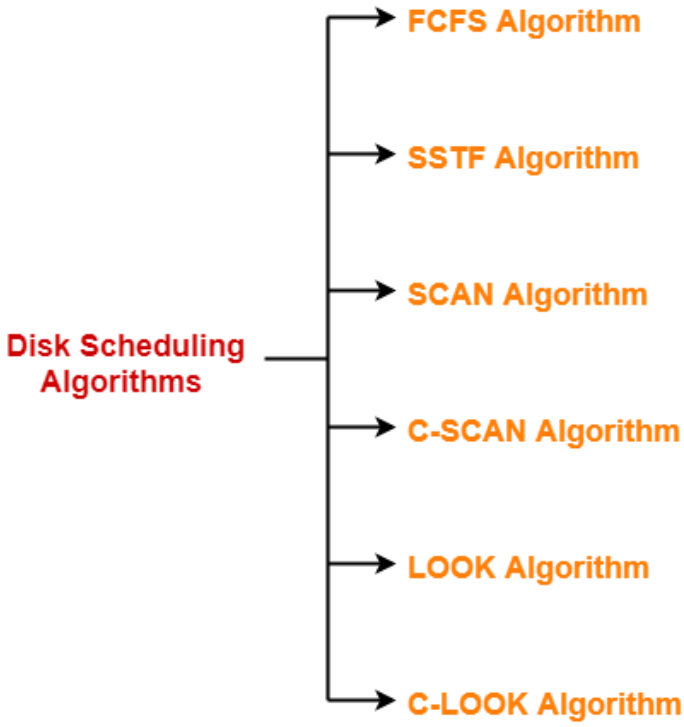
Mean Response Time: The average time spent waiting for a request to be serviced.

Disk Bandwidth: The disk bandwidth is the total number of bytes transferred, divided by the total time between the first request for service and the completion of the last transfer.

Throughput: The number of requests serviced per unit time.

3.2 Working Procedure of Disk Scheduling Algorithms

Disk scheduling algorithms are applied to decide which I/O request is going to be satisfied first. The fundamental disk scheduling algorithms are First Come First Serve, Shortest Seek Time First, Scan, Look, Circular Scan and Circular Look.



The procedures of these algorithms are described below.

3.3 First Come First Serve Disk Scheduling Algorithm (FCFS)

The functioning of this algorithm is maintained by the First in First out (FIFO) queue. With this scheme, the I/O requests are served or processed according to their arrival. Though this algorithm improves response time but fails to decrease the average seek time because this algorithm needs a lot of random head movements and disk rotations.
Let us consider the following track requests in the disk queue to compute the total head movement of the read/write head and let us assume that the current read/write head position is at location 100 for a 200 track disk (0-199).

Track Requests: 25, 90, 135, 50, 190 and 60.

In First Come First Serve algorithm, the read/write head initially move from current location 100 to 25, then 25 to 90, then 90 to 135, then 135 to 50, then 50 to 190 and at last 190 to 60.

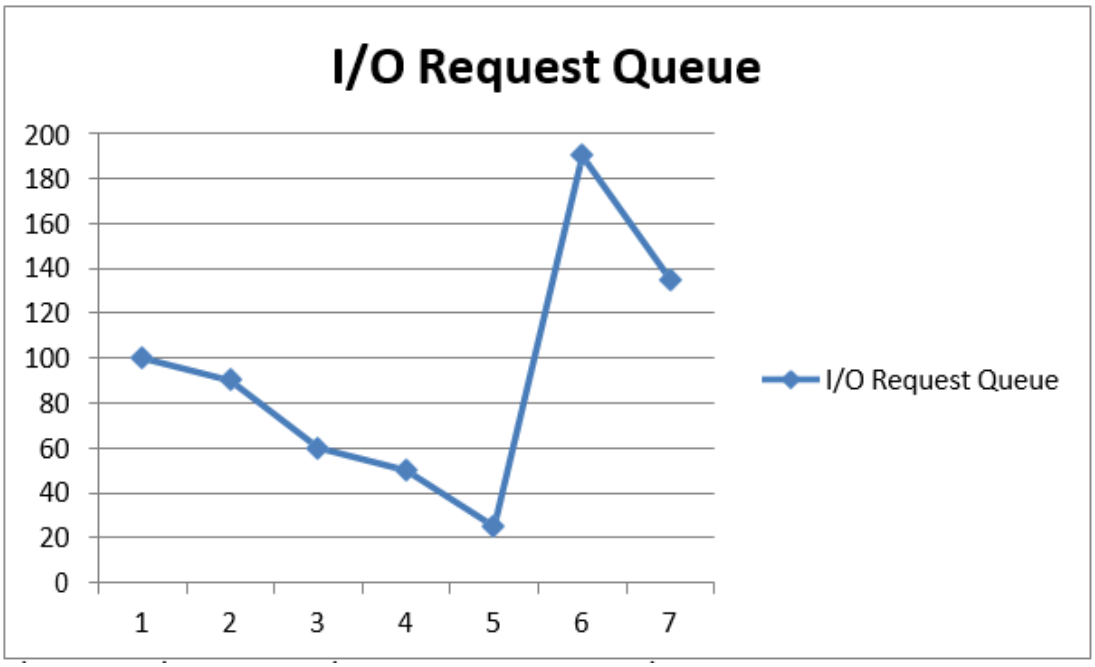


Figure 1: First Come First Serve Representation.

Total Head Movement = (100-25) + (90-25) + (135-90) + (135-50) + (190-50) + (190-60)

= 540 tracks. The major disadvantage of the FCFS disk scheduling algorithm is that it raises the mean seek time because the disk arm has to cover a long distance to accomplish a request as in this case movement of arm from 135 to 50 and then from 50 to 190.

Advantages-

- It is simple, easy to understand and implement.
- It does not cause starvation to any request.

Disadvantages-

- It results in increased total seek time.
- It is inefficient.

3.4 Shortest Seek Time First Disk Scheduling Algorithm (SSTF)

In this approach, the read/write head serves the request first that has minimum seek time from the current head position. The I/O requests that are close to the read/write head are serviced first. SSTF disk scheduling algorithm is actually a form of SJF scheduling algorithm and may cause starvation of some requests. SSTF disk scheduling algorithm has better throughput than FCFS disk scheduling algorithm but some requests may be delayed for a long time if lots of closely situated requests arrive just after it. Let us consider the same track requests in the disk queue to compute the total head movement of the read/write head and let us again assume that the current read/write head position is at location 100 for a 200 track disk (0-199).

Track Requests: 25, 90, 135, 50, 190 and 60.

In the Shortest Seek Time First algorithm, the read/write head initially moves from current location 100 to 90, then 90 to 60, then 60 to 50, then 50 to 25, then 25 to 135 and at last 135 to 190.

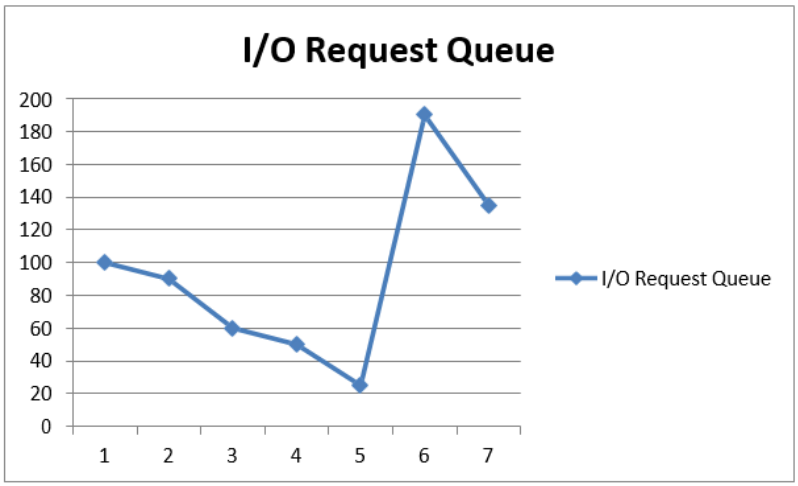


Figure 2: Shortest Seek Time First Representation.

Total Head Movement = (100-90) + (90-60) + (60-50) + (50-25) + (135-25) + (190-135)

= 240 tracks. The major disadvantage of SSTF disk scheduling algorithm is that some requests have to wait for a long time if new requests with shorter seek time keep arriving but the advantage is that the throughput is higher as compared to FIFO disk scheduling algorithm and produces less head movements.

Advantages-

- It reduces the total seek time as compared to FCFS.
- It provides increased throughput.
- It provides less average response time and waiting time.

Disadvantages-

- There is an overhead of finding out the closest request.
- The requests which are far from the head might starve for the CPU.
- It provides high variance in response time and waiting time.
- Switching the direction of the head frequently slows down the algorithm.

3.5 Scan Disk Scheduling Algorithm (SCAN)

In the Scan disk scheduling algorithm, the read/write head starts from one end and moves towards the other end and servicing requests as it reaches each track until it reaches the other end of the disk. The direction of the read/write head reverses after reaching at the other end and servicing continues. In this way, the read/write head continuously swings from end to end**.** Let us consider the same track requests in the disk queue to compute the total head movement of the read/write head and let us again assume that the current read/write head position is at location 100 for a 200 track disk (0-199).

Track Requests: 25, 90, 135, 50, 190 and 60.

In the Scan disk scheduling algorithm, the read/write head initially moves from current location 100 to 90, then 90 to 60, then 60 to 50, then 50 to 25, then 25 to 0 and then to the other direction from 0 to 135 and last 135 to 190. It services all the requests at either side firstly then moves the other way that's why it is sometimes called the elevator algorithm.

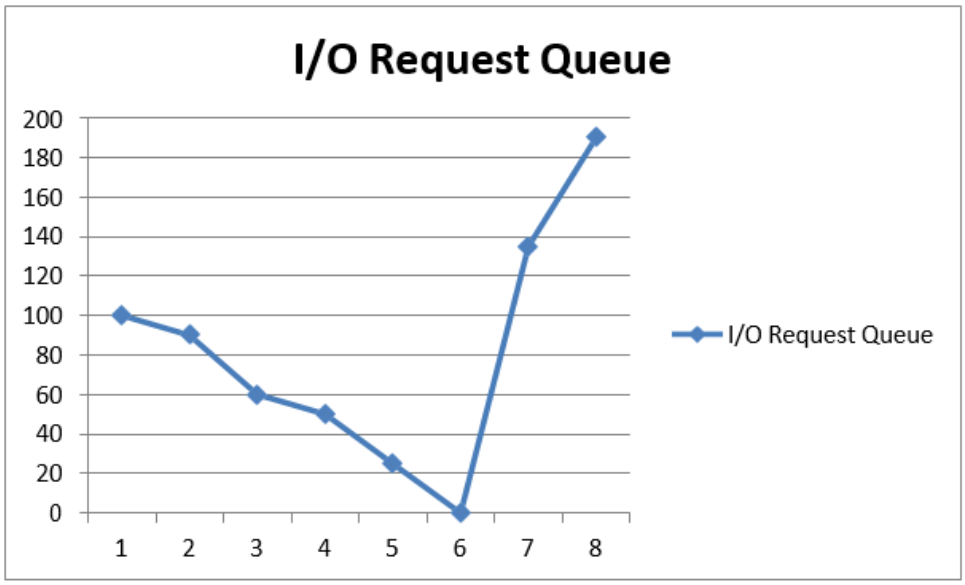


Figure 3: SCAN Representation.

Total Head Movement = (100-90) + (90-60) + (60-50) + (50-25) + (25-0) + (135-0) +

(190-135) = 290 tracks. The major disadvantage of the Scan disk scheduling algorithm is that the disk arm always starts from the beginning towards one end even if other numbers of requests are present on the other end. The throughput of the Scan algorithm is better than FCFS and also prevents starvation.

Advantages-

- It is simple, easy to understand and implement.
- It does not lead to starvation.
- It provides low variance in response time and waiting time.

Disadvantages-

- It causes long waiting times for the cylinders just visited by the head.
- It causes the head to move till the end of the disk even if there are no requests to be serviced.

3.6 Look Disk Scheduling Algorithm (LOOK)

Look disk scheduling algorithm is the improved version of Scan disk scheduling algorithm and avoids the starvation problem. In Look disk scheduling algorithm, the arm goes only as far as final requests in each direction and then reverses direction without going all the way to the end . In this way it improves both response time and throughput. Let us consider the same track requests in the disk queue to compute the total head movement of the read/write head and let us again assume that the current read/write head position is at location 100 for a 200 track disk (0-199).

Track Requests: 25, 90, 135, 50, 190 and 60.

In Look disk scheduling algorithm, the read/write head initially moves from current location 100 to 90, then 90 to 60, then 60 to 50, then 50 to 25, then to the other direction from 25 to 135 and then at last from 135 to 190.

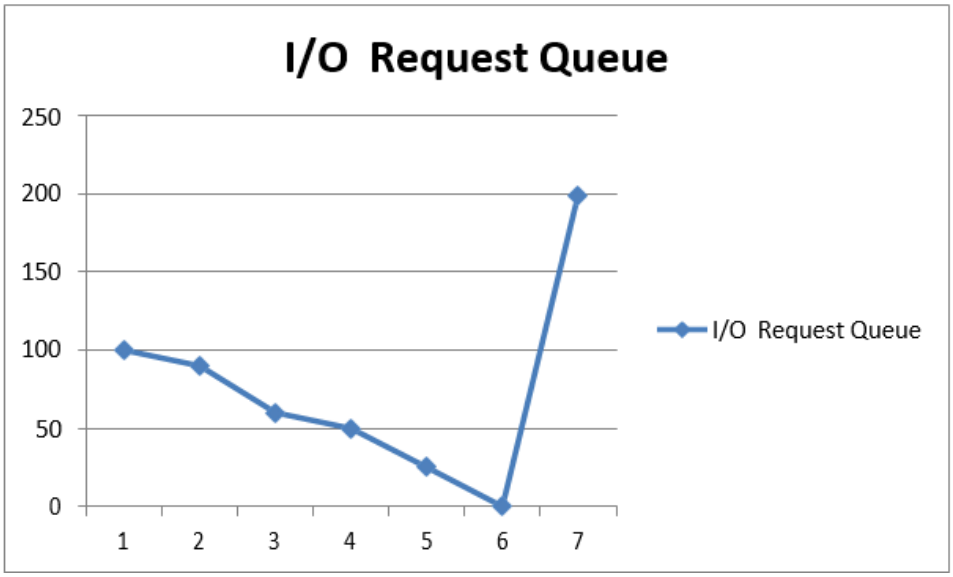


Figure 4: LOOK Representation.

Total Head Movement = (100-90) + (90-60) + (60-50) + (50-25) + (135-25) + (190-135)

= 240 tracks. The major advantage of Look disk scheduling algorithm is that it improves response time and throughput than Scan disk scheduling algorithm.

Advantages-

- It does not cause the head to move till the ends of the disk when there are no requests to be serviced.
- It provides better performance as compared to SCAN Algorithm.
- It does not lead to starvation.
- It provides low variance in response time and waiting time.

Disadvantages-

- There is an overhead of finding the end requests.
- It causes long waiting times for the cylinders just visited by the head.

3.7 Circular Scan Disk Scheduling Algorithm (C-SCAN)

Cyclic Scan or Circular Scan disk scheduling algorithm is an improved version of Scan disk scheduling algorithm and known as one directional scan. It starts its scan towards the nearest end and services the requests all the way to the end. Let us consider the same track requests in the disk queue to compute the total head movement of the read/write head and let us again assume that the current read/write head position is at location 100 for a 200 track disk (0- 199).

Track Requests: 25, 90, 135, 50, 190 and 60.

In the C-Scan disk scheduling algorithm, the read/write head initially move from current location 100 to 90, then 90 to 60, then 60 to 50, then 50 to 25, then 25 to 0 and then to the other direction from 0 to 199, then 199 to 190 and at last 190 to 135. It services all the requests at either side firstly up to the end and then moves the other end's without fulfilling any requests in the way and then starts servicing as shown in the figure below.

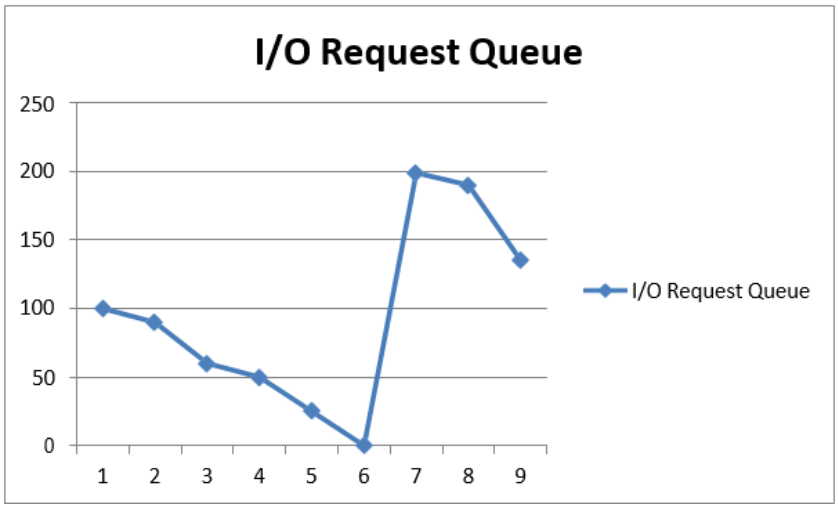


Figure 5: C-Scan Representation.

Total Head Movement = (100-90) + (90-60) + (60-50) + (50-25) + (25-0) + (199-0) + (199-190) + (190-135) = 363 tracks. The C-Scan disk scheduling algorithm satisfies requests only when the head moves in one direction and not satisfies any requests when it moves back.

Advantages-

- The waiting time for the cylinders just visited by the head is reduced as compared to the SCAN Algorithm.
- It provides uniform waiting time.
- It provides better response time.

Disadvantages-

- It causes more seek movements as compared to SCAN Algorithm.
- It causes the head to move till the end of the disk even if there are no requests to be serviced.

3.8 Circular Look Disk Scheduling Algorithm (C-LOOK)

Cyclic Look or Circular Look disk scheduling algorithm is an improved version of C-Scan disk scheduling algorithm. Arm goes only as far as the last request in each direction and then reverses direction immediately without first going all the way to the end of the disk. Let us consider the same track requests in the disk queue to compute the total head

movement of the read/write head and let us again assume that the current read/write head position is at location 100 for a 200 track disk (0-199).

Track Requests: 25, 90, 135, 50, 190 and 60.

In C-Look disk scheduling algorithm, the read/write head initially moves from current location 100 to 90, then 90 to 60, then 60 to 50, then 50 to 25 and then to the other direction from 25 to 190 and at last 190 to 135. It services all the requests at either side firstly up to the last request and then moves the other end's last request without fulfilling any requests in the way and then starts servicing as shown in the figure below.

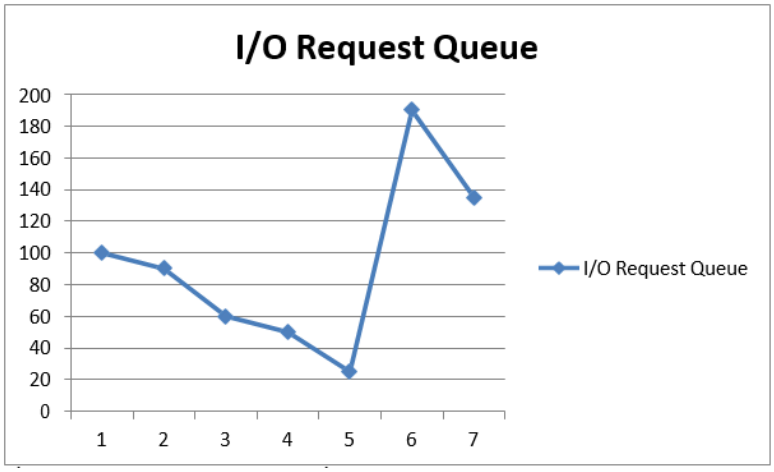


Figure 6: C-LOOK Representation.

.

Total Head Movement = (100-90) + (90-60) + (60-50) + (50-25) + (190-25) + (190-135)

= 295 tracks. The major advantage of the C-Look disk scheduling algorithm is that it results in higher throughput and lower response time.

Advantages-

- It does not cause the head to move till the ends of the disk when there are no requests to be serviced.
- It reduces the waiting time for the cylinders just visited by the head.
- It provides better performance as compared to LOOK Algorithm.
- It does not lead to starvation.
- It provides low variance in response time and waiting time.

Disadvantages-

- There is an overhead of finding the end requests.

3.9 Comparison of different scheduling algorithms

Table -1 compares the average head movement of six disk scheduling algorithms for the first five runs and their average. Similar requests are assigned for every individual run for all six algorithms and their total head movement is calculated.

S.No.	FCFS	SSTF	Scan	Look	C-Scan	C-Look
1	540	240	290	240	363	295
2	631	276	280	276	348	306
3	264	217	270	224	393	289
4	322	189	235	189	363	189
5	640	235	275	245	378	300
Average	**479**	**231**	**270**	**235**	**369**	**276**

Table 1: Average head movement of first five runs and their average.

The Shortest Seek Time First algorithm produces the minimum head movement of 240 in the first run, 276 in the second run, 217 in the third run, 189 in the fourth run, 235 in the last run and average head movement of all the runs is 231. From the above table, it is very much clear that for the different five runs, the Shortest Seek Time First disk scheduling algorithm has minimum total head movement in all cases as compared to other five algorithms. A graphical representation is shown in the figure 8 with the help of data evaluated in table 1.

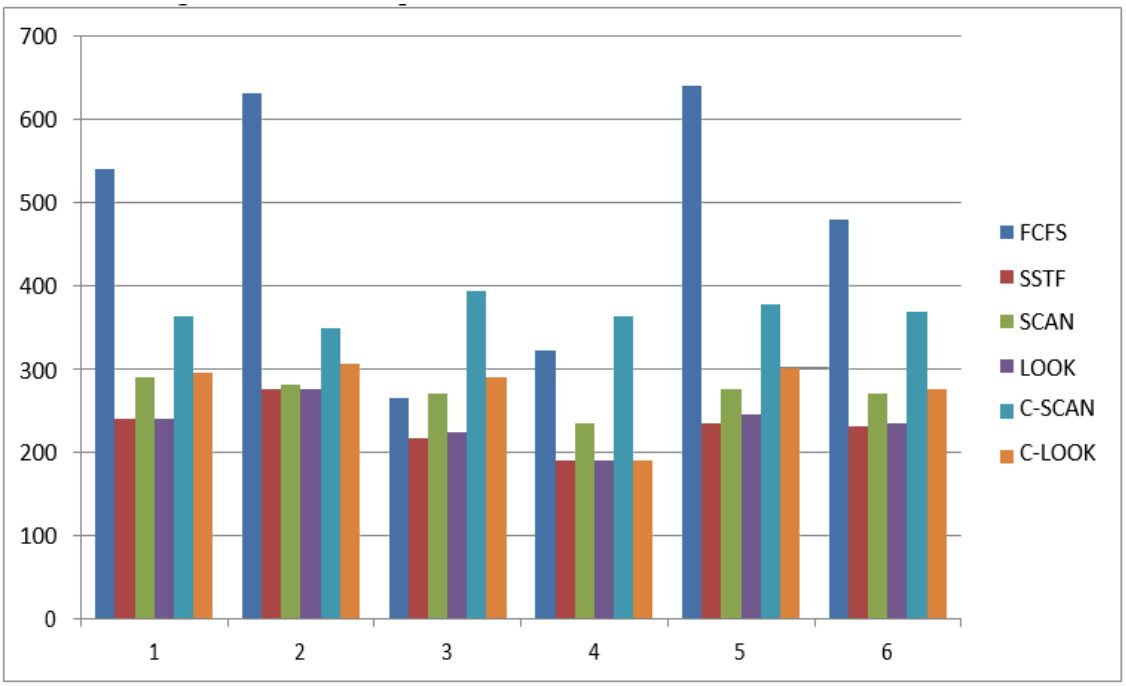


Figure 7: Comparison between six disk scheduling algorithms in graphical representation.

The figure 7 gives comparative details of the six disk scheduling algorithms. The X axis represents five runs and the average case and the Y axis is used for calculating average total head movement of each algorithm. From the various runs, it is concluded that Shortest Seek Time First has the minimum average head movement than all other five algorithms for the similar requests. Thus the Shortest Seek Time First algorithm is a good criterion for selecting the requests for I/O from the disk queue.

Chapter 4: Workflow Diagram

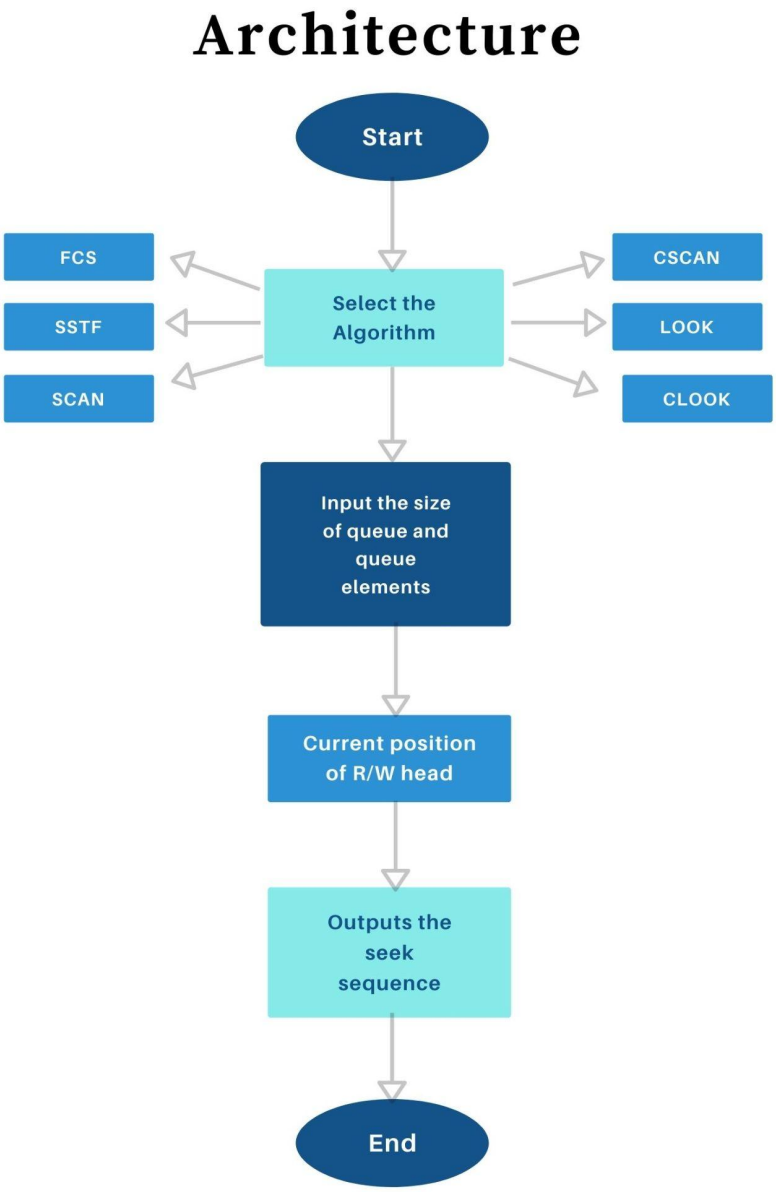


Figure 8: Flow diagram

Chapter 5: Results

We have calculated the average total head movement after entering the various runs for the requests of different algorithms because total head movement is the criteria for analyzing the disk scheduling algorithms. After comparing the total head movement of various algorithms, we have found that the Shortest Seek Time First disk scheduling algorithm has the least average head movement than the others discussed above in context to total head movement.

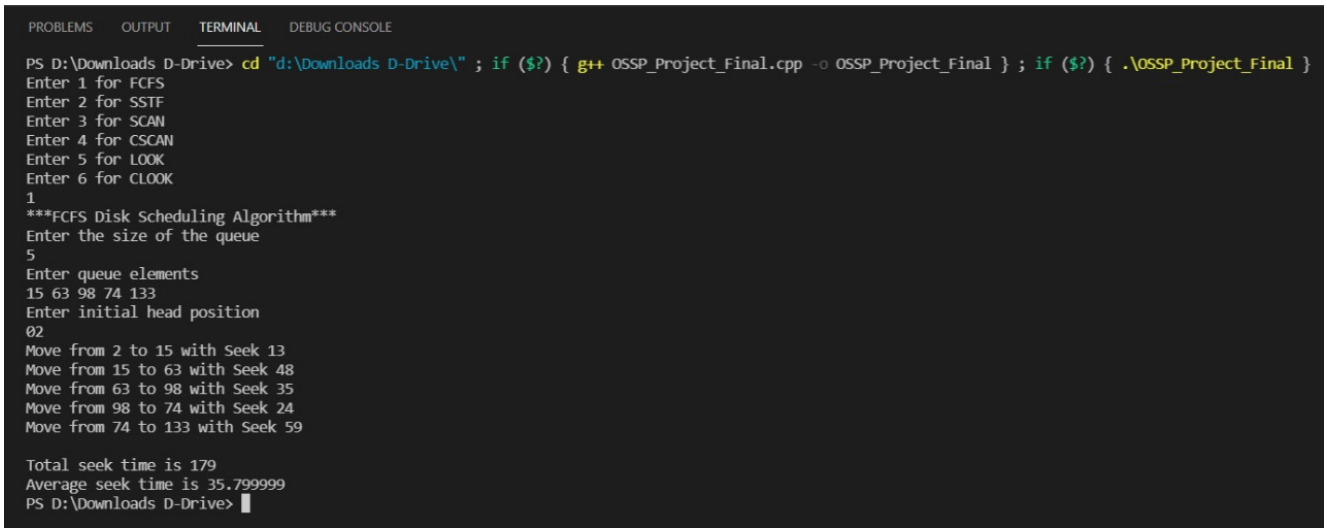


Figure 9: FCFS

Figure 10: SSTF

```
PROBLEMS  OUTPUT  TERMINAL  DEBUG CONSOLE

PS D:\Downloads D-Drive> cd "d:\Downloads D-Drive\" ; if ($?) { g++ OSSP_Project_Final.cpp -o OSSP_Project_Final } ; if ($?) { .\OSSP_Project_Final }
Enter 1 for FCFS
Enter 2 for SSTF
Enter 3 for SCAN
Enter 4 for CSCAN
Enter 5 for LOOK
Enter 6 for CLOOK
3
Enter the size of the queue
5
Enter queue elements
15 63 98 74 133
Enter initial head position
100
Total number of seek operations = 233
Seek Sequence is
98 -> 74 -> 63 -> 15 -> 0 -> 133 ->
PS D:\Downloads D-Drive>
```

Figure 10: SCAN

```
PROBLEMS  OUTPUT  TERMINAL  DEBUG CONSOLE

PS D:\Downloads D-Drive> cd "d:\Downloads D-Drive\" ; if ($?) { g++ OSSP_Project_Final.cpp -o OSSP_Project_Final } ; if ($?) { .\OSSP_Project_Final }
Enter 1 for FCFS
Enter 2 for SSTF
Enter 3 for SCAN
Enter 4 for CSCAN
Enter 5 for LOOK
Enter 6 for CLOOK
4
Input no of disk locations      5
Enter initial head position     02
Enter disk positions to be read
15 63 98 74 133
Disk head moves from 2 to 0 with seek 2
Disk head moves from 0 to 199 with seek 199
Disk head moves from 199 to 133 with seek 66
Disk head moves from 133 to 98 with seek 35
Disk head moves from 98 to 74 with seek 24
Disk head moves from 74 to 63 with seek 11
Disk head moves from 63 to 15 with seek 48
Total seek time is 385
Average seek time is 77.000000
PS D:\Downloads D-Drive>
```

Figure 11: CSCAN

```
PROBLEMS  OUTPUT  TERMINAL  DEBUG CONSOLE

PS D:\Downloads D-Drive> cd "d:\Downloads D-Drive\" ; if ($?) { g++ OSSP_Project_Final.cpp -o OSSP_Project_Final } ; if ($?) { .\OSSP_Project_Final }
Enter 1 for FCFS
Enter 2 for SSTF
Enter 3 for SCAN
Enter 4 for CSCAN
Enter 5 for LOOK
Enter 6 for CLOOK
5
Input the number of disk locations      5
Enter initial head position     88
Enter disk positions to read
15 63 98 74 133
Disk head moves from 88 to 74 with seek 14
Disk head moves from 74 to 63 with seek 11
Disk head moves from 63 to 15 with seek 48
Disk head moves from 15 to 98 with seek 83
Disk head moves from 98 to 133 with seek 35
Total seek time is 191
Average seek time is 38.200001
PS D:\Downloads D-Drive>
```

Figure 12: LOOK

Figure 13: CLOOK

Chapter 6: Conclusion

The report analyzes the six disk scheduling algorithms viz. First Come First Serve, Shortest Seek Time First, Scan, Look, C-Scan and C-Look. Different disk scheduling algorithms can be used depending upon the load of the system. The performance depends upon the number and types of requests.

References

[1] https://www.geeksforgeeks.org/disk-scheduling-algorithms/

[2] https://www.gatevidyalay.com/disk-scheduling-disk-scheduling-algorithms/

[3] http://www.cs.iit.edu/~cs561/cs450/disksched/disksched.html

[4] https://www.javatpoint.com/os-disk-scheduling

[5] https://solver.assistedcoding.eu/disk_scheduling