An Improved FCFS (IFCFS) Disk Scheduling Algorithm

Manish Kumar Mishra
Department of Information Technology
College of Computing and Informatics
Haramaya University
Ethiopia

ABSTRACT
Since the time movable head disk came into existence, the I/O performance has been improved by proper scheduling of disk accesses. Disk scheduling involves a careful examination of pending requests to determine the most efficient way to service the requests. The two most common types of scheduling are seek optimization and rotational (or latency) optimization. Most of the scheduling algorithms concentrate on reducing seek times for a set of requests, because seek times tend to be an order of magnitude greater than latency times. Some of the most important scheduling algorithms are First-Come-First-Served (FCFS), Shortest Seek Time First (SSTF), SCAN, Circular Scan (C-SCAN) and LOOK. FCFS is the simplest form of disk scheduling algorithm. This algorithm is simple to implement, but it generally does not provide the fastest service. This paper describes an improvement in FCFS. A simulator program has been designed and tested the improved FCFS. After improvement in FCFS it has been found that the service is fast and seek time has been reduced drastically.

General Terms
Operating System, Disk Scheduling.

Keywords
Disk Scheduling, Seek Time, Average Seek Time, FCFS, IFCFS.

1. INTRODUCTION
In multiprogrammed computing systems, inefficiency is often caused by improper use of rotational storage devices such as disk. In this type of system, many processes may be generating requests for reading and writing disk records. Sometimes these processes make requests faster than they can be serviced by the moving head-disks, as a result waiting lines or queues build up for each device [1]. Which process should be selected next for service, is an important question, because it affects the effectiveness of the service. The main aim of the disk scheduling algorithms is to reduce or minimize the seek time for a set of requests [2]. The disk performance can be optimized by installing a magnetic disk that can result in high transfer rates. Magnetic disk is a collection of platters. Information is stored by recording it magnetically on the platters. A read-write disk head is located on top of each surface of every platter. The heads are attached to a disk arm that moves all the heads as a unit. The surface of a platter is logically divided into circular tracks, which are subdivided into sectors [3]. A cylinder is made up of set of tracks that are at one arm position. Disks are currently four orders of magnitude slower than main memory, so many researches are going on to enhance the efficiency of disks [4]. Scheduling algorithms for moving head-disks have been studied for many years, but which algorithm is "best" is still an open question [5]. Most scheduling algorithms in use today are variations of a few central themes. FCFS is still a preferable choice [5]. It is easy to implement and it is fair in the sense that once a request has arrived, its place in the schedule is fixed. This study focuses on improving the effectiveness of FCFS.

1.1 Disk Performance Parameters
The disk I/O operations mainly depend on the computer system, the operating system, and the nature of the I/O channel and disk controller hardware [6]. The time taken to position the disk arm at the desired cylinder is called the Seek Time, and the time for the desired sector to rotate to the disk head is called the Rotational Latency. The sum of seek time and rotational latency is known as Access Time. The transfer time mainly depends on the rotational speed of the disk. The total number of bytes transferred, divided by the total time between the first request for service and the completion of the last transfer is called the disk Bandwidth [3]. These are the disk performance parameters and they can be improved by scheduling the servicing of disk I/O requests in a good order.

1.2 Disk Scheduling Algorithms
Disk scheduling algorithms are used to allocate the services to the I/O requests on the disk. Some of the most important scheduling algorithms are First-Come-First-Served (FCFS), Shortest Seek Time First (SSTF), SCAN, Circular Scan (C-SCAN) and LOOK. FCFS is the simplest form of disk scheduling algorithm. In this scheduling, I/O requests are served as per their arrival. The request that arrive first, is served first so the name First-Come-First-Served. In SSTF algorithm, the request with the minimum seek time from the current head position is served first. In this algorithm, I/O requests at the edges of the disk surface may get starved [7]. SSTF gives substantial improvement over FCFS. In SCAN algorithm, the disk arm starts from one end of the disk and moves to the other end of the disk. While moving from one end to the other end of the disk, it serves the requests as it reaches each cylinder. When it reaches to other end, the direction of head movement is reversed. SCAN gives better performance than FCFS and SSTF. In C-SCAN, the disk head moves from one end to the other end of the disk, serving the request along the way. When the disk head reaches to the other end, it immediately returns back to the beginning of the disk. In return trip, it does not serve any request. The waiting time increases in C-SCAN [2]. In LOOK algorithm, the arm goes only as far as the final request in each direction [3]. The direction reverses immediately, without going all the way to the end of the disk.

2. IFCFS ALGORITHM
The improved FCFS (IFCFS) disk scheduling algorithm works similar to FCFS but with a small improvement. IFCFS move the disk head with the intention to serve the first I/O request. On the way going to serve the first request, if there is any request waiting from the current disk head position to the

first request, will be served. After serving first request and the requests that were served on the way, disk head will move to the next request waiting in the queue. On the way going to serve this request, if there is any request waiting from the current disk head position to the next waiting request, will be served etc. IFCFS guarantees the performance improvement over FCFS.

Following is the proposed IFCFS disk scheduling algorithm

Step 1. START
Step 2. Make a queue of the I/O requests say REQUEST.
Step 3. Do steps 4, 5 and 6 WHILE queue REQUEST becomes empty.
Step 4. Move the disk head from the current position say K to the first request in the queue REQUEST. Serve the requests that are on the way of the disk head from the current position K to the first request in the queue REQUEST.
Step 5. Serve the first request in the queue REQUEST.
Step 6. Remove the requests from the queue REQUEST that are already served.
Step 7. END

3. PERFORMANCE EVALUATION

To evaluate the performance, I assumed that all the I/O requests are independent of each other and have equal priority. The requests are stored in the request queue. The minimum track number is 0 and the maximum track number is 100 on each platter. The seek time has been taken as the performance parameter.

3.1 Experiments Performed

For performance evaluation of my proposed IFCFS algorithm, I have taken three different cases. Six, eight and ten I/O requests have been taken into consideration in case 1, case 2 and case 3 respectively. In each case, the experimental results of proposed IFCFS algorithm have been compared with FCFS algorithm.

Case 1: I consider the disk queue with request for I/O to blocks on cylinders 80, 50, 30, 40, 5 and 10. If disk head is presently at cylinder 20, it will first move to cylinder 80. Since three I/O requests on cylinders 50, 30 and 40 will be on the way going from cylinder 20 to 80, these three requests will be served before the disk head reached to cylinder 80. After serving request at cylinder 80, the request queue will be checked for next request. Since the requests on cylinders 50, 30 and 40 have been already served, the next available request is on cylinder 5. The disk head will now move to cylinder 5 from cylinder 80. On the way going to cylinder 5, the request on cylinder 10 will be served. The total head movement is 135 cylinders. Using the same example, the total head movement in FCFS is 160 cylinders. Table 1 shows the comparison of result of proposed IFCFS with FCFS algorithm. Figure 1 and Figure 2 shows the representation of FCFS and IFCFS respectively. Figure 3 shows the comparison of average seek time of FCFS and IFCFS.

Table 1. Comparison of FCFS and IFCFS (Case 1)

Algorithms	Total Head Movement	Average Seek Time
FCFS	160	26.67
IFCFS	135	22.50

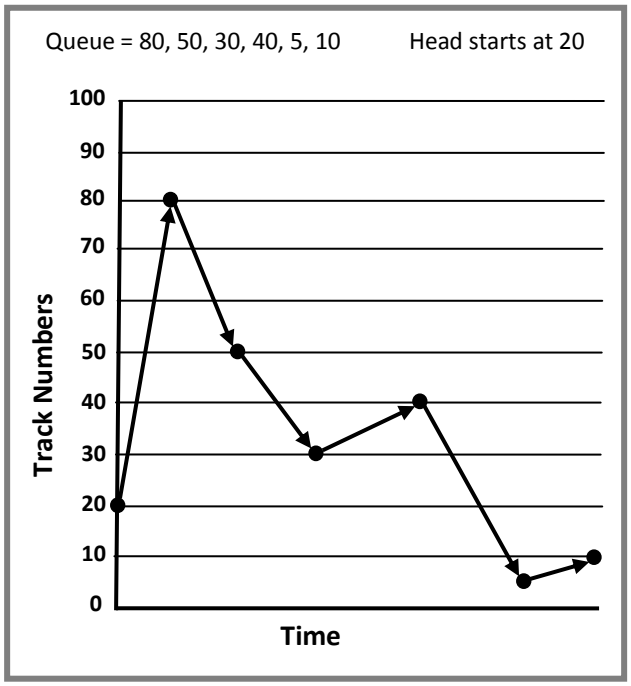

Fig 1: Representation of FCFS (Case 1)

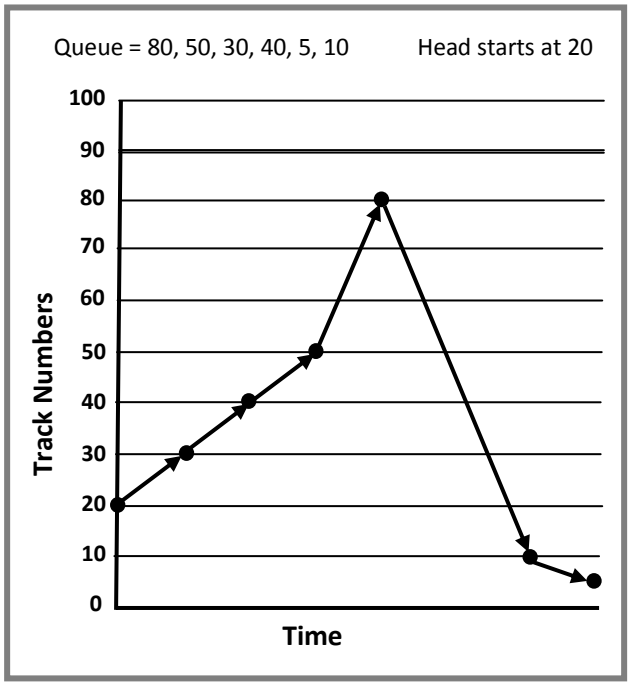

Fig 2: Representation of IFCFS (Case 1)

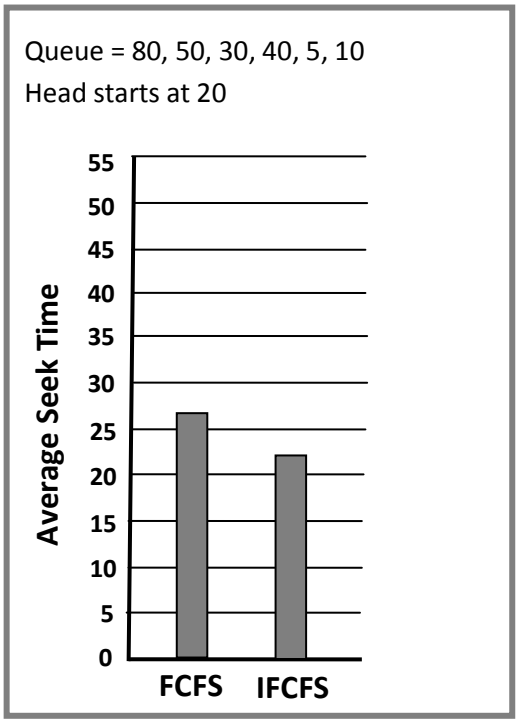

Fig 3: Comparison of Average Seek Time (Case 1)

Case 2: The disk queue with request for I/O to blocks on cylinders 100, 5, 50, 90, 75, 10, 80 and 60 has been taken into consideration. If disk head is presently at cylinder 40, it will first move to cylinder 100. Since five I/O requests on cylinders 50, 90, 75, 80 and 60 will be on the way going from cylinder 40 to 100, these five requests will be served before the disk head reached to cylinder 100. After serving request at cylinder 100, the request queue will be checked for next request. Since the requests on cylinders 50, 90, 75, 80 and 60 have been already served, the next available request is on cylinder 5. The disk head will now move to cylinder 5 from cylinder 100. On the way going to cylinder 5, the request on cylinder 10 will be served. The total head movement is 155 cylinders. Using the same example, the total head movement in FCFS is 410 cylinders. Table 2 shows the comparison of result of proposed IFCFS with FCFS algorithm. Figure 4 and Figure 5 shows the representation of FCFS and IFCFS respectively. Figure 6 shows the comparison of average seek time of FCFS and IFCFS.

Table 2. Comparison of FCFS and IFCFS (Case 2)

Algorithms	Total Head Movement	Average Seek Time
FCFS	410	51.25
IFCFS	155	19.38

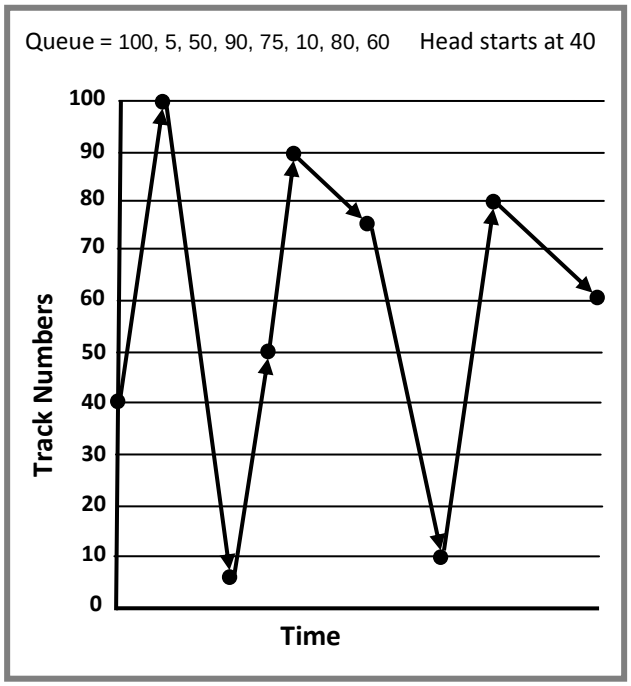

Fig 4: Representation of FCFS (Case 2)

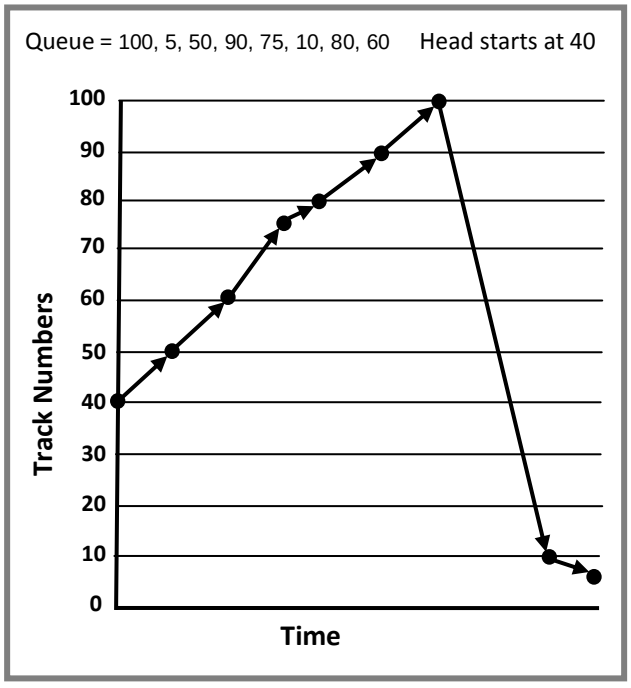

Fig 5: Representation of IFCFS (Case 2)

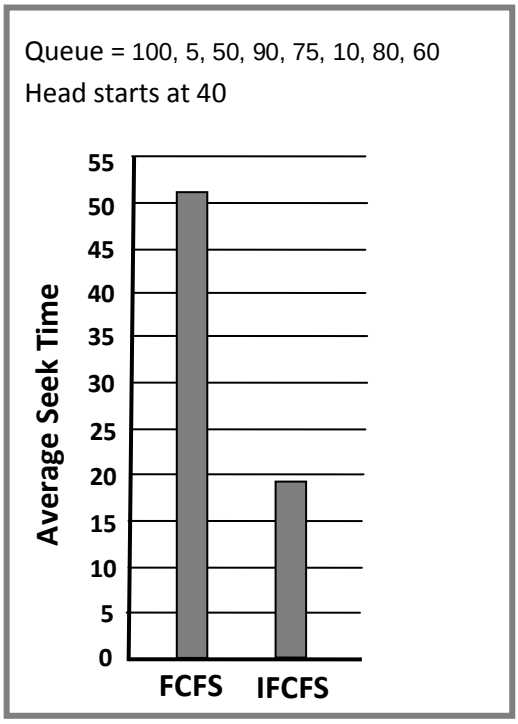

Fig 6: Comparison of Average Seek Time (Case 2)

Case 3: The disk queue with request for I/O to blocks on cylinders 20, 35, 5, 95, 75, 55, 85, 45, 40 and 15 has been taken into consideration. If disk head is presently at cylinder 50, it will first move to cylinder 20. Since three I/O requests on cylinders 35, 45 and 40 will be on the way going from cylinder 50 to 20, these three requests will be served before the disk head reached to cylinder 20. After serving request at cylinder 20, the request queue will be checked for next request. Since the requests on cylinders 35, 45 and 40 have been already served, the next available request is on cylinder 5. The disk head will now move to cylinder 5 from cylinder 20. On the way going to cylinder 5, the request on cylinder 15 will be served. After serving request at cylinder 5, the request queue will be checked for next request. Next available request is on cylinder 95. Since three I/O requests on cylinders 75, 55 and 80 will be on the way going from cylinder 5 to 95, these three requests will be served before the disk head reached to cylinder 95. The total head movement is 135 cylinders. Using the same example, the total head movement in FCFS is 305 cylinders. Table 3 shows the comparison of result of proposed IFCFS with FCFS algorithm. Figure 7 and Figure 8 shows the representation of FCFS and IFCFS respectively. Figure 9 shows the comparison of average seek time of FCFS and IFCFS.

Table 3. Comparison of FCFS and IFCFS (Case 3)

Algorithms	Total Head Movement	Average Seek Time
FCFS	305	30.50
IFCFS	135	13.50

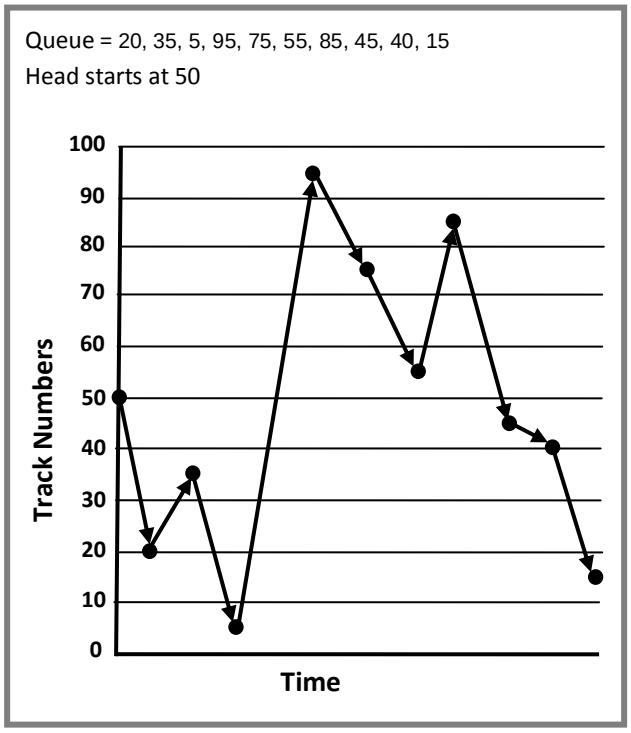

Fig 7: Representation of FCFS (Case 3)

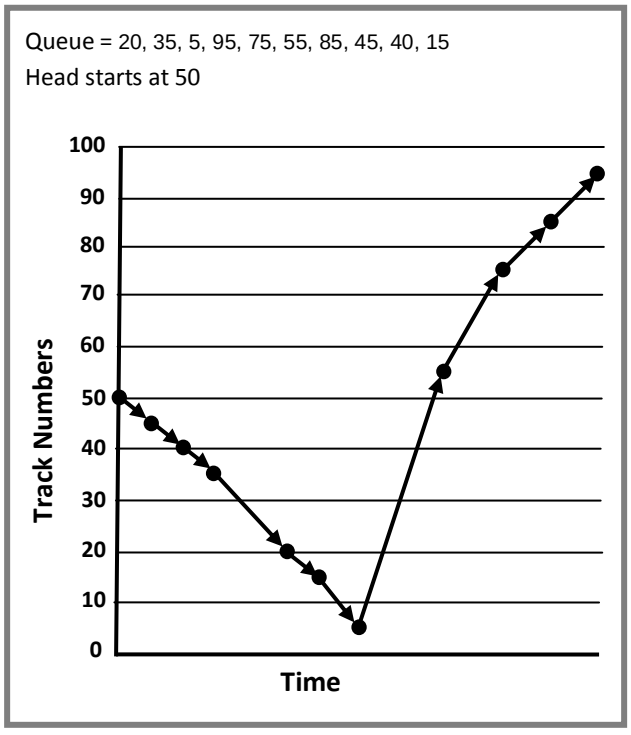

Fig 8: Representation of IFCFS (Case 3)

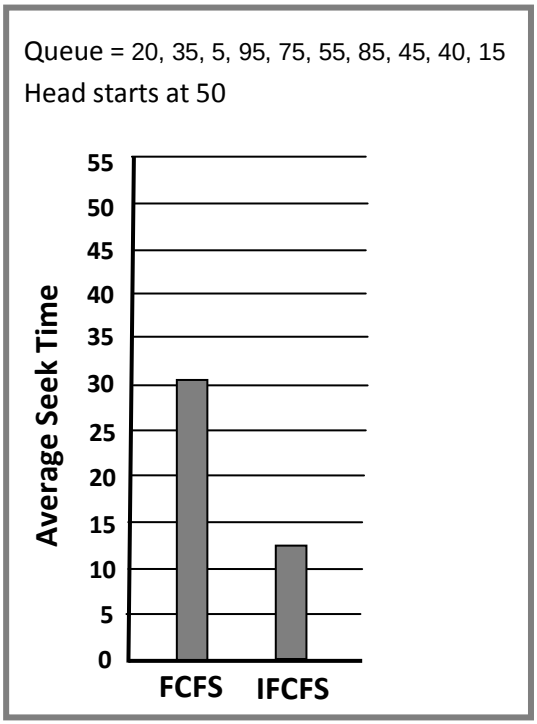

Fig 9: Comparison of Average Seek Time (Case 3)

4. CONCLUSION

Simulation results shows that the proposed IFCFS disk scheduling algorithm is always giving better performance than FCFS. After improvement in FCFS it has been found that the service is fast and seek time has been reduced drastically. This algorithm can be implemented to improve the performance in the systems in which FCFS is a preferable choice.

5. REFERENCES

[1] H. M. Deitel, "Operating Systems", 2nd Edn., Pearson Education Pte. Ltd., 2002, ISBN 81-7808-035-4.

[2] Sourav Kumar Bhoi, Sanjaya Kumar Panda, and Imran Hossain Faruk, "Design and Performance Evaluation of an Optimized Disk Schedduling Algorithm (ODSA)", International Journal of Computer Applications, Vol. 40, No. 11, Feb 2012, pp. 28-35.

[3] A. Silberschatz, P. B. Galvin, and G. Gagne, "Operating System Concepts", 7th Edn., John Wiley and Sons Inc, 2005, ISBN 0-471-69466-5.

[4] W. Stallings, "Operating Systems", 4th Edn., Pearson Education Pte. Ltd., 2007, ISBN 81-7808-503-8.

[5] Robert Geist, Stephen Daniel, "A Continuum of Disk Scheduling Algorithms" ACM Transactions on Computer Systems" Vol. 5, No. 1, Feb 1987, pp. 77-92

[6] C. Staelin, G. Amir, D. B. Ovadia, R. Dagan, M. Melamed and D. Staas, " Real-time disk scheduling algorithm allowing concurrent I/O requests", HP Laboratories, HPL-2009-344.

[7] A. L. N. Reddy, Jim Wyllie and K. B. R. Wijayaratne, "Disk Scheduling in a Multimedia I/O System" ACM Transactions on Multimedia Computing, Communications and Applications" Vol. 1, No. 1, Feb 2005, pp. 37-59