Logo, company name

Description automatically generated

**Comparison Between Disk Scheduling Algorithms**

|  |  |
| --- | --- |
| Aya Elneanaei Fouda | 202002609 |
| Lina Eyad | 202000202 |
| Kerolos salama | 202001113 |
| Bola Nader | 202000645 |

**Presented to:**

Dr. Eman Kamel Mohamed Gawish

TA. [Salma Khaled Hegazy](https://moodle.nu.edu.eg/user/view.php?id=8860&course=6631)

Rameez Barakat Hassan

# Table of Contents

Table of Contents

[Table of Contents 2](#_Toc135774113)

[Section 1: Abstract 4](#_Toc135774114)

[Section 2: Introduction 4](#_Toc135774115)

[Section 3: Literature review 5](#_Toc135774116)

[3.1 FCFS Scheduling Algorithm 5](#_Toc135774117)

[3.2 SSTF Scheduling Algorithm 6](#_Toc135774118)

[3.3 SCAN Scheduling Algorithm 7](#_Toc135774119)

[3.4 C-SCAN Scheduling Algorithm 7](#_Toc135774120)

[3.5. LOOK Scheduling Algorithm 7](#_Toc135774121)

[3.6.C-LOOK Scheduling Algorithm 8](#_Toc135774122)

[3.7 SMCC Scheduling Algorithm 8](#_Toc135774123)

[3.8 IFCFS Scheduling Algorithm 9](#_Toc135774124)

[3.9 HDSA Scheduling Algorithm 10](#_Toc135774125)

[Section 4: Advantages and Disadvantages algorithms : 11](#_Toc135774126)

[Section 5: Methodology 14](#_Toc135774127)

[Case1 : 14](#_Toc135774128)

[Case2 16](#_Toc135774129)

[Case3 18](#_Toc135774130)

[Case 4: 20](#_Toc135774131)

[Case 5 : 23](#_Toc135774132)

[Case6: 25](#_Toc135774133)

[Section 6: Results and Analysis 27](#_Toc135774134)

[Section 7: Conclusion 28](#_Toc135774136)

[References 29](#_Toc135774137)

# Section 1: Abstract

Operating systems use disc scheduling as a method of determining which I/O request will be handled first. Disc scheduling algorithms aim to increase throughput while reducing reaction times. In the current piece of research, nine distinct disc scheduling methods are compared and analyzed. Shortest seek time, first come, first served Initially, scan, look, C-Scan and C-look, IFCFS and SMCC throughout many runs. The implementation is done in C++ language by building an interface to figure out how much each of these nine algorithms will move the head overall.

# Section 2: Introduction

The scheduling mechanism used to conduct I/O operations has a significant impact on

computer systems’ performance. A good scheduling algorithm should be quick and fair. In other

words, algorithms should be able to serve as many I/O requests as possible in a given amount of time without causing starvation for some of them.

Disk scheduling algorithms are primarily concerned with reducing the overall time required to serve requests (access time) and the average time between the arrival of requests and their completion (waiting time). The access time for any I/O consists of the time needed for the disk arm to move the head to the requested track (seek time) in addition to the time required for the disk to rotate to the specified sector on track (rotational latency). Since seek time is the dominant factor in access time, scheduling algorithms normally neglect rotational latency (Muqaddas et al., 2009).

Researchers are focusing their efforts on improving traditional disc scheduling algorithms including FCFS, SSTF, SCAN, C-SCAN, LOOK, and C-LOOK. According to the literature, the SSTF and LOOK algorithms are the top competitors among traditional scheduling algorithms (Javed & Khan, 2000). In this paper, we will implement these common algorithms as well as three new ones (IFCFS, HDSA, SMCC). Then, using three different examples, we will evaluate their performance by calculating the total and average seek time for each one.

# Section 3: Literature review

# 3.1 FCFS Scheduling Algorithm

First-Come, First-Serve (FCFS) is a simple disk scheduling algorithm that serves requests in the order in which they arrive. It is the easiest to implement and requires no additional overhead, making it a popular choice for simple systems with low I/O loads. It does not provide the fastest service. Also, no rearranging of requests is performed. In the first come first serve disk scheduling algorithm the request that arrives first is served first. However, FCFS can lead to poor performance if the requests are not distributed evenly across the disk, or if there are long seek times between requests. This is because FCFS does not consider the position of the disk head or the relative distance between requests. As a result, FCFS can lead to high average seek times and long waiting times for requests that are located far from the current position of the disk head.

Several studies have evaluated the performance of FCFS and compared it with other disk scheduling algorithms. For example, a study by M. Harchol-Balter and A. Silberschatz (1996) found that FCFS performs poorly under heavy I/O loads, especially when the requests are not uniformly distributed across the disk. The study also found that other algorithms, such as SCAN and LOOK, can provide better performance in such scenarios. Overall, while FCFS is simple and easy to implement, it may not provide optimal performance in many real-world scenarios. Other disk scheduling algorithms, such as SSTF and SCAN, can provide better performance by considering the position of the disk head and the relative distance between requests.

# 3.2 SSTF Scheduling Algorithm

The Shortest Seek Time First (SSTF) disk scheduling algorithm is a popular algorithm that serves requests in the order of their relative distance from the current position of the disk head. The algorithm selects the request that is closest to the current position of the disk head, reducing the average seek time and improving disk throughput. Several studies have evaluated the performance of SSTF and compared it with other disk scheduling algorithms. For example, a study by A. Al-Dubai et al. (2009) found that SSTF performs better than other algorithms, such as FCFS and SCAN, in terms of average response time, disk utilization, and throughput. The study also found that SSTF can improve the performance of the disk subsystem under heavy I/O loads. However, SSTF can also have some drawbacks. For example, it may cause starvation of requests that are far away from the current position of the disk head. This can lead to some requests waiting indefinitely, while other requests are served quickly, leading to poor fairness. Overall, SSTF is a popular disk scheduling algorithm that can improve the performance of the disk subsystem by reducing the average seek time and improving disk throughput. However, the choice of algorithm depends on the specific workload and system configuration, as well as the goals and constraints of the system.

# 3.3 SCAN Scheduling Algorithm

In the SCAN disc scheduling method, the head begins at one end of the disc and proceeds to the other end, completing requests one at a time along the way. To access the disc, the head first moves in one direction before turning around and repeating the process. As a result, this algorithm sometimes goes by the name elevator algorithm because it functions like an elevator. Because of this, the requests in the middle receive greater attention, and those arriving after the disc arm must wait.

# 3.4 C-SCAN Scheduling Algorithm

The circular SCAN (C-SCAN) scheduling algorithm is a modified version of the SCAN disc scheduling method that addresses the SCAN algorithm's inefficiency by providing more consistent service to the requests. Similar to SCAN (Elevator Algorithm), C-SCAN moves the head from one end to the other end while servicing all requests. However, as soon as the head reaches the opposite end, it quickly travels back to the disk's beginning without attending to any requests on the way there (see the chart below) and begins servicing once it arrives. Since it essentially interprets the cylinders as a circular list that loops around from the last cylinder to the first one, this algorithm is also known as the "Circular Elevator Algorithm".

# 3.5. LOOK Scheduling Algorithm

LOOK is a more sophisticated version of the SCAN (elevator) disc scheduling technique that provides significantly faster seek times than the other algorithms. The LOOK algorithm handles requests similarly to the SCAN method, but it also "looks" ahead as if there are additional tracks in the same direction that need to be served. If no requests are outstanding in the moving direction, the head changes course and begins servicing requests in the opposite direction. The key reason for the LOOK algorithm's superior performance over SCAN is that the head is not permitted to go past the end of the disc in this method

# 3.6.C-LOOK Scheduling Algorithm

This paper aims to make a performance analysis of various disk scheduling algorithms based on various factors. SCAN And C-SCAN algorithms move the disk arm across the full width of the disk. But in practice, neither of these algorithms is implemented this way. In the C-LOOK disk scheduling algorithm, the arm goes only as far as the final request in each direction. Then it reverses the direction immediately without going all the way to the end of the disk (C.Mallikarjuna & P.Chitti Babu, 2016).

# 3.7 SMCC Scheduling Algorithm

The paper proposes a new disk scheduling algorithm (SSMC) and compares its

performance with the other common scheduling algorithms such as FCFS, SSTF, LOOK,

SCAN…etc. SMCC's major purpose is to reduce the number of head movements and seek time,

therefore boosting the algorithm's performance.

The algorithm begins by sorting the requests in ascending order and then extracts the

midway request from the sorted queue. The algorithm then compares the current head pointer

against the request for the midpoint. If the current head pointer is less than the midpoint request,

the algorithm will service each request starting with the first and ending with the last in the sorted list. Otherwise, it will serve the requests from the last request until we reach the first request in the sorted list. Finally, we will calculate the total number of head movements and average seek time. Results indicate that the proposed algorithm reduces the number of head movements and seek time, therefore enhancing disc drive performance. For a few requests, our algorithm is equal to SSTF / LOOK disk scheduling algorithms (Kumar & Rajendra, 2015)

# 3.8 IFCFS Scheduling Algorithm

Improved Fair Credit-based Flow Control with Selective Acknowledgment (IFCFS) scheduling algorithm is a packet scheduling technique designed to ensure efficient and fair resource allocation among different data flows in computer networks. The algorithm builds upon the concept of credit-based flow control, which assigns credits to each flow based on its packet size and inter-arrival time to provide a fair share of the available bandwidth. Additionally, IFCFS incorporates selective acknowledgment (SACK), a mechanism that allows the sender to be informed about which packets have been received and which are still missing, leading to more efficient retransmission and reduced packet loss.

The IFCFS disk scheduling algorithm works like FCFS but with a small improvement. IFCFS move the disk head to serve the first I/O request. On the way 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 the first request and the requests that were served on the way, the disk head will move to the next request waiting in the queue. On the way 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 performance improvement over FCFS

# 3.9 HDSA Scheduling Algorithm

This paper claims that the proposed algorithm (HDSA) shows performance than existing conventional disk scheduling algorithms such as FCFS, SSTF, SCAN, LOOK... etc.

First, the service request queue is divided into two new queues P and Q. The first queue P

contains requests that are lower than the initial disk head position h; while the second queue Q

contains requests that are greater than the initial disk head position h. Next, the absolute difference

between h and the lowest track request n of disk queue P and the absolute difference between h and the highest track request m of disk queue Q, are calculated as x and y respectively. If x > y, then scanning begins in disk request queue Q, servicing the requests using SSTF, then serves the

requests in queue P using SSTF as well. Otherwise, if x < y then disk request queue P takes the

67 lead and servicing operations begin immediately on the other request queue Q after P has been completed. The testing showed that the HDSA had fewer head movements than traditional disc scheduling methods. This will surely improve disc performance in computer resource management (T.A. et al., 2019).

# Section 4: Advantages and Disadvantages algorithms :

|  |  |
| --- | --- |
| FCFS | SSTF |
| Advantages:  1-The reasoning behind the First Come First Serve method is really straightforward; it just processes requests in the order they are received.  2-As a result, First Come First Serve is incredibly straightforward and simple to comprehend and apply.  3-Every process in FCFS finally has a chance to run, preventing starvation. | Advantages:  1-Compared to First Come First Serve, the total seek time is shorter.  2-Throughput improves and rises with SSTF.  3-Average wait times and response times in SSTF are shorter. |
| Disadvantages:  1-Since this scheduling strategy is nonpreemptive, the process cannot be halted mid-execution and must complete its whole course.  2-As a nonpreemptive scheduling technique, FCFS requires that the short processes at the back of the queue wait for the long processes at the front to complete.  3-FCFS's throughput is not particularly effective.  4-Only on tiny systems, when input-output efficiency is not crucial, is FCFS implemented. | Disadvantages:  1-There is a cost associated with identifying the closest request in SSTF.  2-For unreasonable requests, starvation may result.  3-Response and waiting times in the SSTF have substantial variance.  4-The method becomes slower when the Head's direction changes frequently. |

2.

|  |  |  |
| --- | --- | --- |
| **SCAN** | **C-SCAN** | **HDSA-scheduling** |
| Advantages:  1-The scan scheduling approach is straightforward, simple to comprehend, and simple to use.  2-SCAN algorithm prevents starvation.  3-Low variation occurs in response and waiting times. | Advantages:  1-The SCAN scheduling algorithm's replacement and upgraded version is called C-SCAN.  2-While responding to all inquiries in between, the Head moves from one end of the disc to the other.  3-In comparison to the SCAN Algorithm, C-SCAN reduces the waiting time for the cylinders that the head recently visited.  4-The waiting period is constant.  5-Improved response times are offered. | Advantages:  1-Improved performance: HDSA scheduling aims to optimize disk access by dynamically allocating storage space based on the access patterns of the data. This can lead to improved performance and reduced access latency.  2-Efficient space utilization: By dynamically allocating storage space, HDSA scheduling can make better use of available disk space, reducing wasted or unused storage.  3-Adaptive to workload changes: HDSA scheduling can adapt to changing workload patterns, allowing for efficient allocation and reallocation of storage space as needed. |
| Disadvantages:  1-There is a prolonged waiting period for the cylinders that the head has just visited.  2-Even when there are no requests to be served, the head moves all the way to the end of the disc in SCAN. | Disadvantages:  1-C-SCAN causes more seek motions than SCAN Algorithm does.  2-In contrast to SCAN algorithm, in C-SCAN the Head will continue to move to the end of the disc even if there are no more requests to be handled. | Disadvantages:  Increased complexity: Implementing and managing HDSA scheduling requires additional complexity in disk management systems. It may involve sophisticated algorithms and techniques to track and manage data allocation dynamically. |

3.

|  |  |  |
| --- | --- | --- |
| Look | C-Look | SMCC |
| Advantages:  1-In contrast to the SCAN algorithm, the Head will not move to the end of the disk if there are no more requests that need to be served.  2-The performance is superior to that of the SCAN Algorithm.  3-The LOOK scheduling algorithm prevents starvation.  4-The waiting time and response time have a low variance. | Advantages:  1-If there are no requests that need to be serviced, C-LOOK does not require the head to move until the disk's end.  2-There is less sitting tight time for the chambers which are simply visited by the head in C-LOOK.  3-C-LOOK outperforms the LOOK Algorithm in terms of performance.  4-In C-LOOK, starvation is avoided.  5-The waiting time and response time have a low variance. | Advantages:  Reduced head movements: The SMCC algorithm aims to minimize head movements by sorting requests and selecting a midpoint reference. This can potentially lead to improved disk performance by reducing the distance the disk head needs to travel to access data.  2-Decreased seek time: Minimizing head movements also implies a reduction in seek time. By optimizing the order of request servicing, the algorithm seeks to minimize the time required for the disk head to move to the desired location, resulting in faster data access. |
| Disadvantages:  1-Above of finding the end demands is available.  2-The cylinders that Head has just visited have to wait a long time. | Disadvantages:  There is a cost associated with finding the end requests in C-LOOK. | Disadvantages:  Limited evaluation: The paper may not provide an extensive evaluation or comparison of the SMCC algorithm with other scheduling algorithms. It's important to have a comprehensive analysis to understand the algorithm's performance in various scenarios and workloads. |

# Section 5: Methodology

We have written the code of all nine algorithms mentioned above in C++ programming.

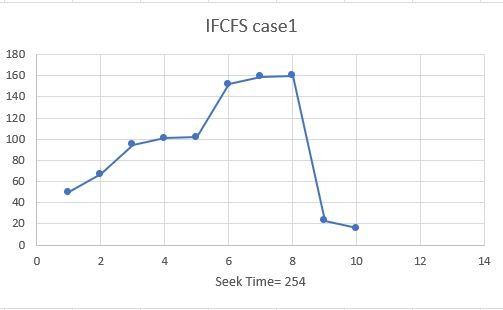
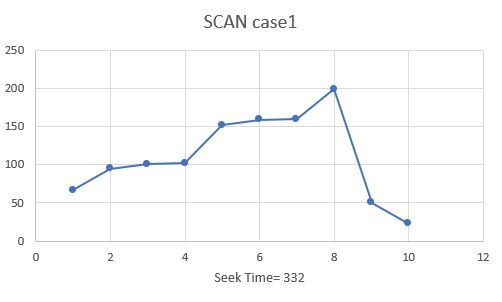
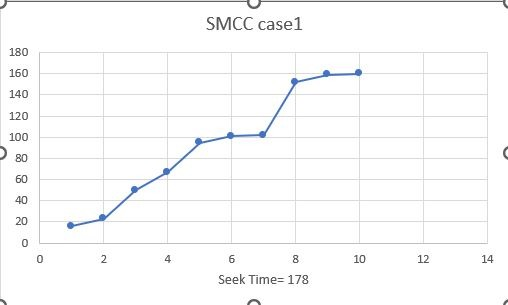
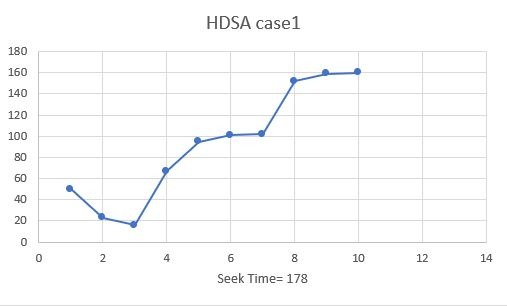
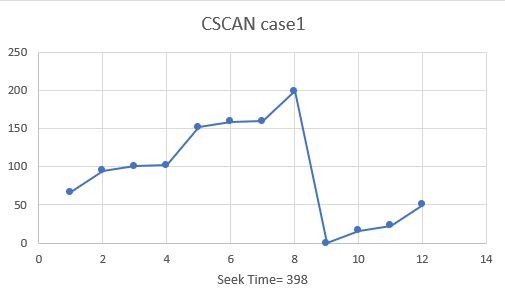
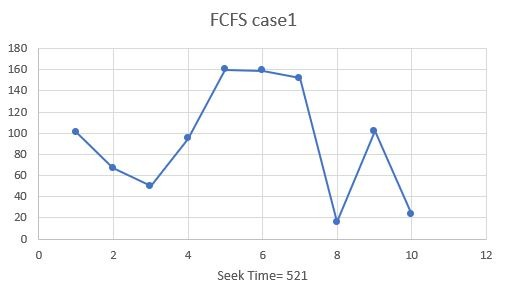
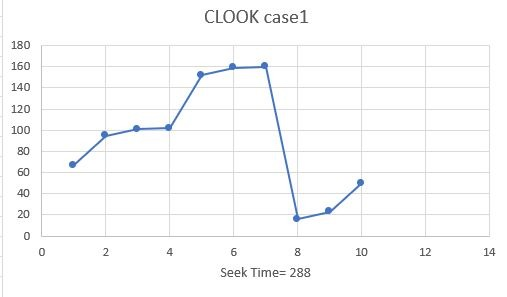
language (see appendix 7.1 to 7.9). Next, we ran the program on six cases randomly. The program serviced them and calculated the total seek time according to each of the algorithms. Finally, we compared the results of all algorithms in all six cases in terms of total seek time.

# Case1 :

Suppose a disk drive has 200 cylinders, numbered 0 to 199. Consider a disk queue with

requests for i/o to blocks on cylinder: 101, 67, 50, 95, 160, 159, 152, 16, 102, 23. Assume that disk head is currently at cylinder 50. figure 1 to figure 9 show the representation of FCFS, SSTF, SSTF, SCAN, C-SCAN, LOOK, C-LOOK, IFCFS, SMCC and HDSA disk scheduling algorithm

respectively.

******

# Case2

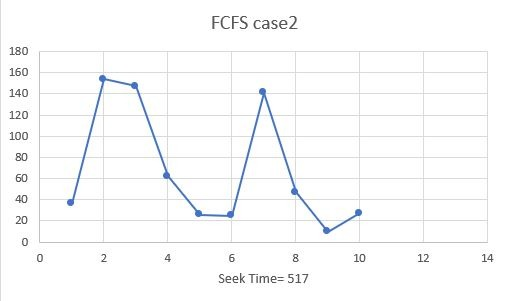
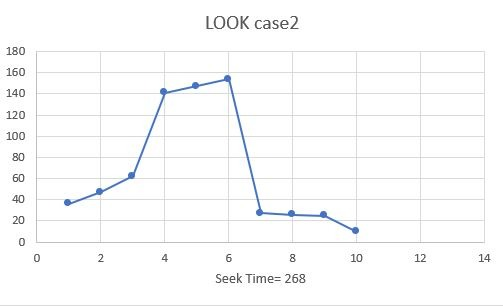
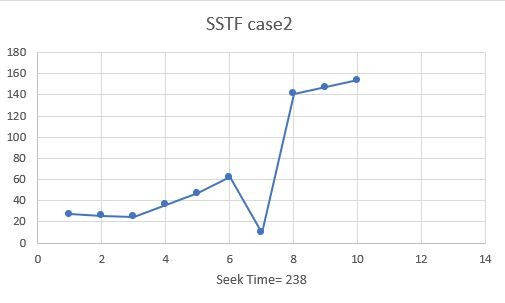
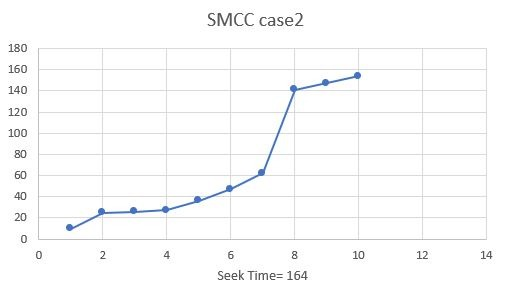
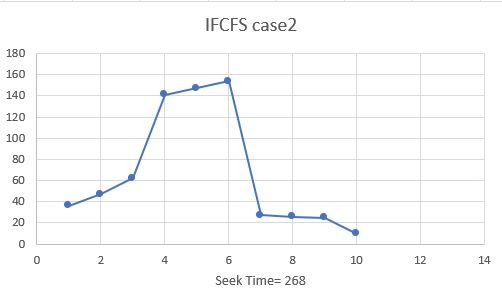
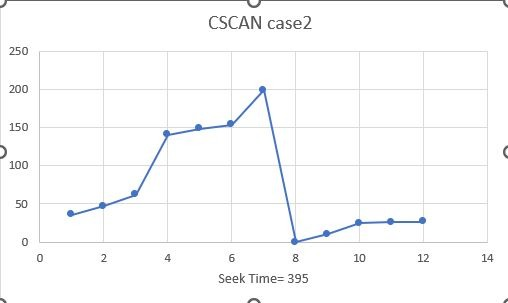
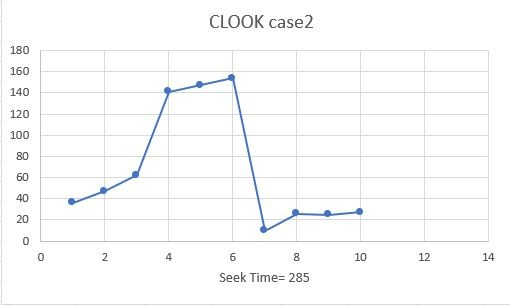
Suppose a disk drive has 200 cylinders, numbered 0 to 199. Consider a disk queue with

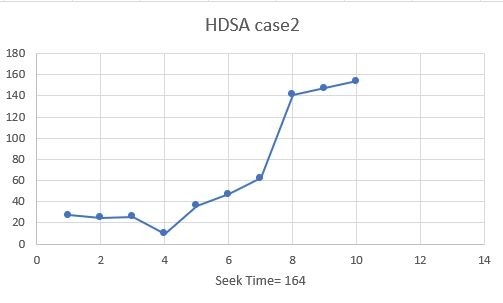
requests for i/o to blocks on cylinder:36, 154, 147, 62, 26, 25, 141, 47, 10, 27. Assume that disk

head is currently at cylinder 30 figure 10 to figure 18 show the representation of FCFS, SSTF,

SSTF, SCAN, C-SCAN, LOOK, C-LOOK, IFCFS, SMCC and HDSA disk scheduling algorithm

respectively

******

******

# Case3

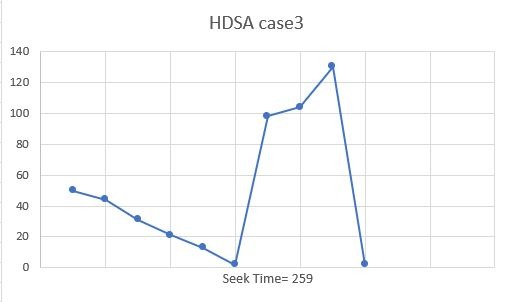
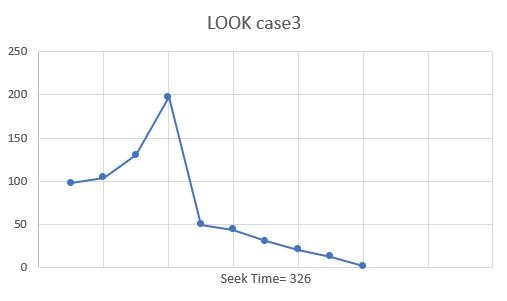
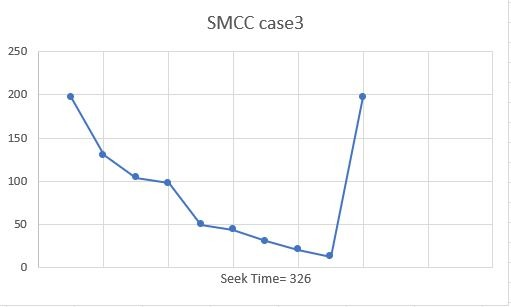
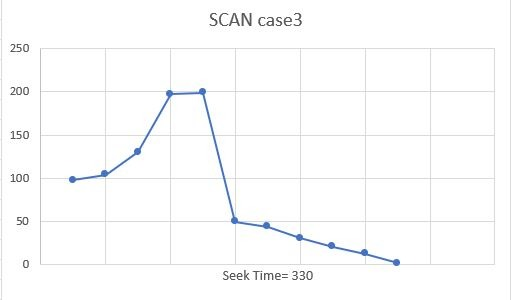
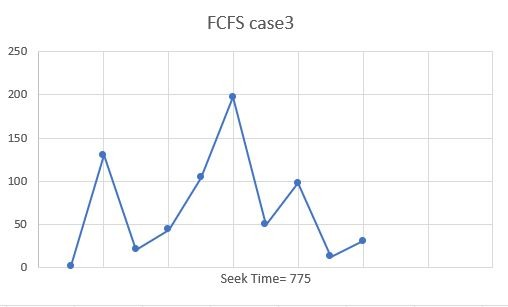
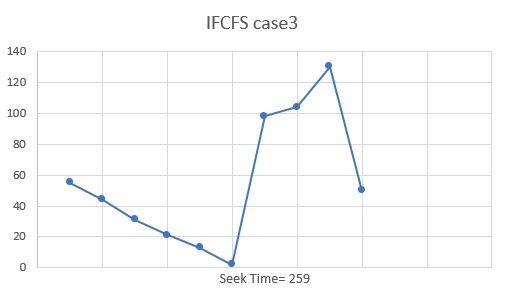
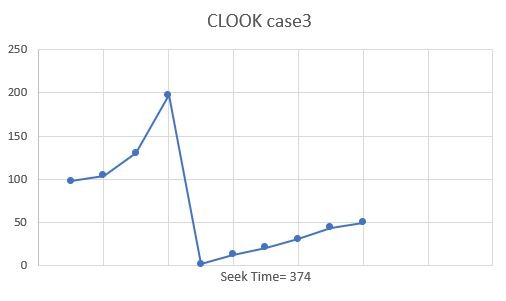
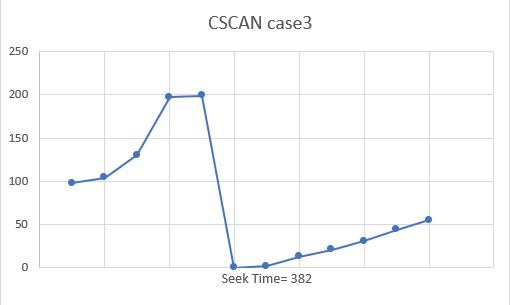
Suppose a disk drive has 200 cylinders, numbered 0 to 199. Consider a disk queue with

requests for I/o to blocks on cylinder: 2, 130, 21, 44, 104, 197, 50, 98, 13, 31. Assume that disk

head is currently at cylinder 66. figure 19 to figure 27 show the representation of FCFS, SSTF,

SSTF, SCAN, C-SCAN, LOOK, C-LOOK, IFCFS, SMCC and HDSA disk scheduling algorithm

Respectively.



# Case 4:

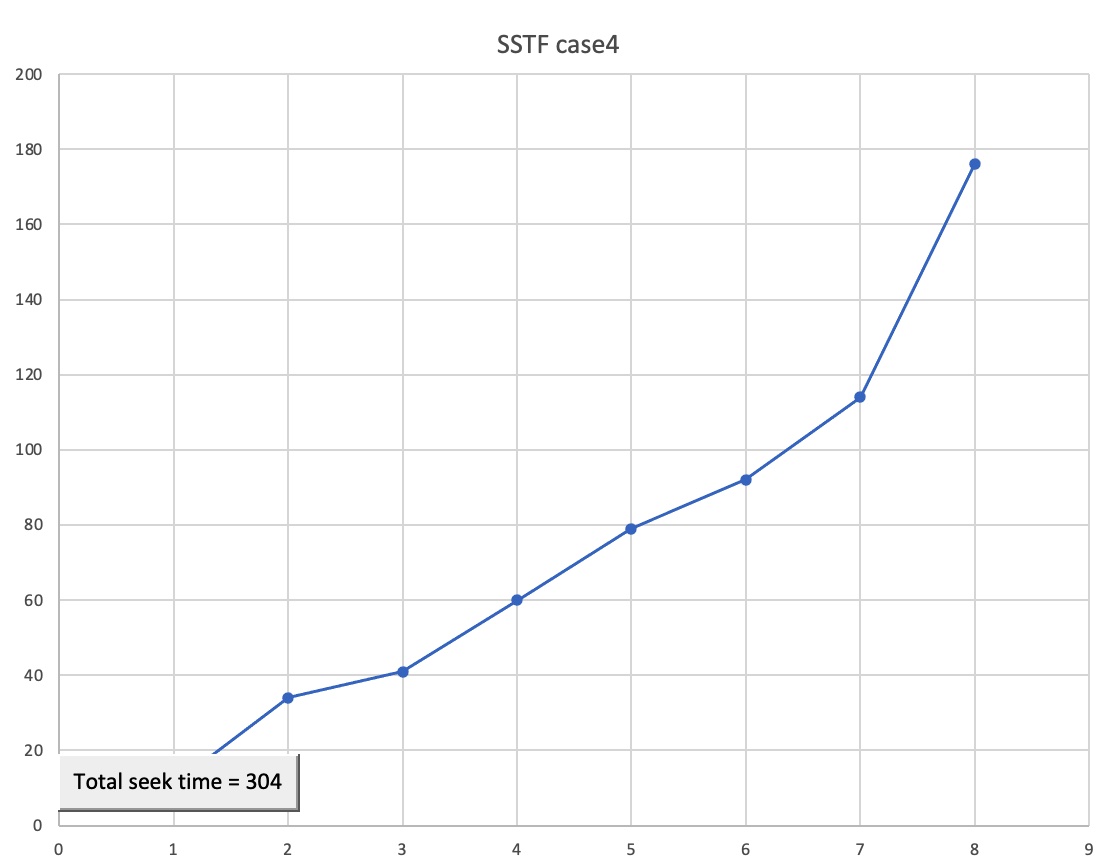
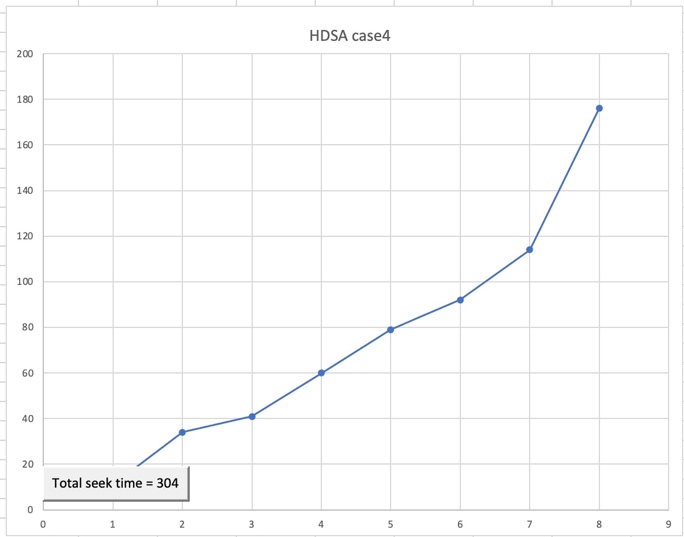
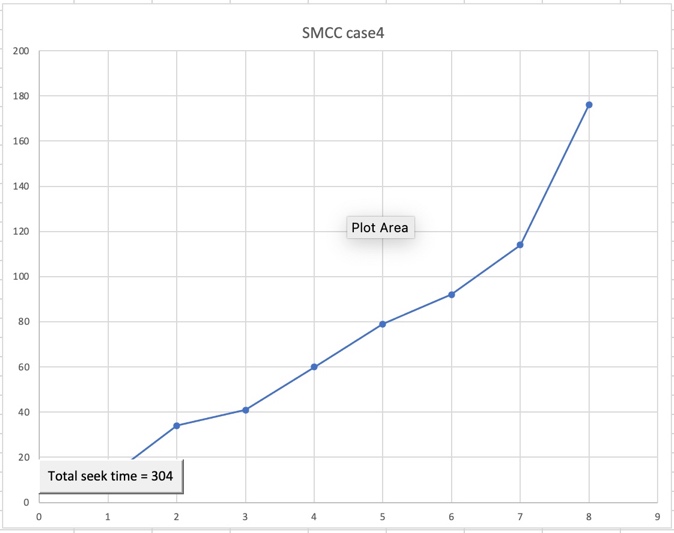
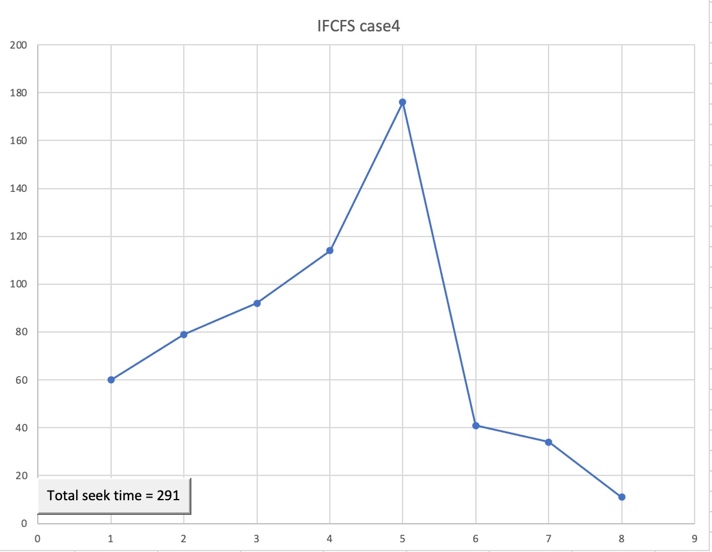
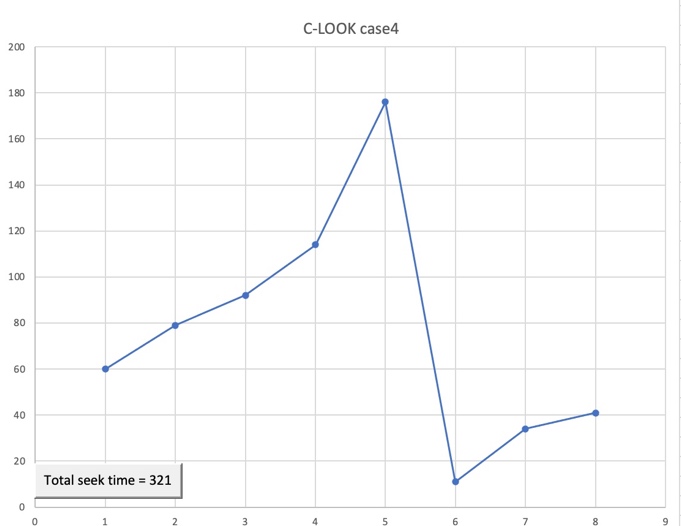
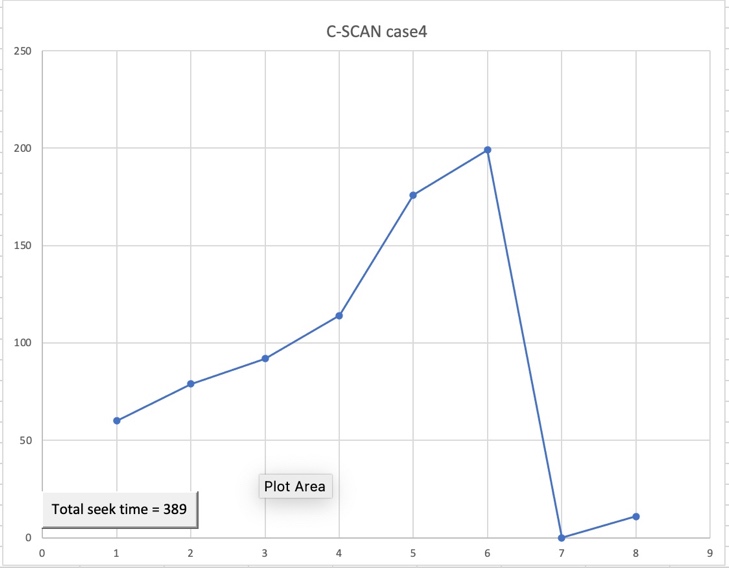
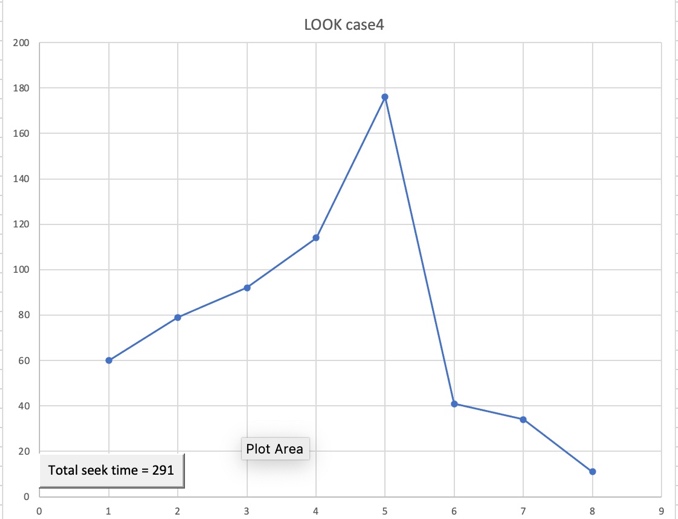
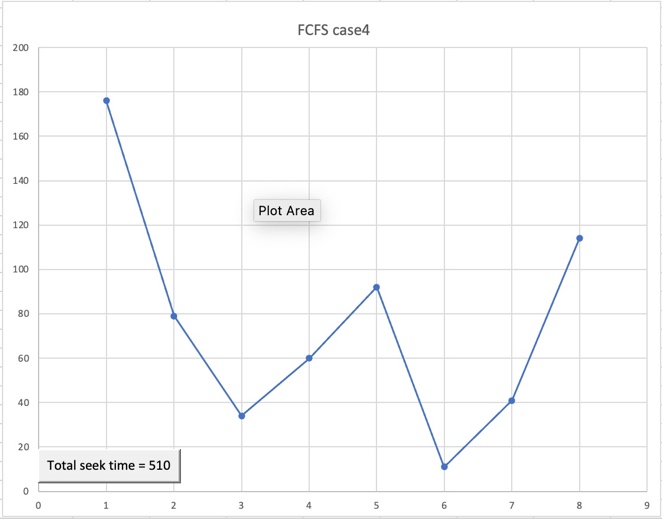
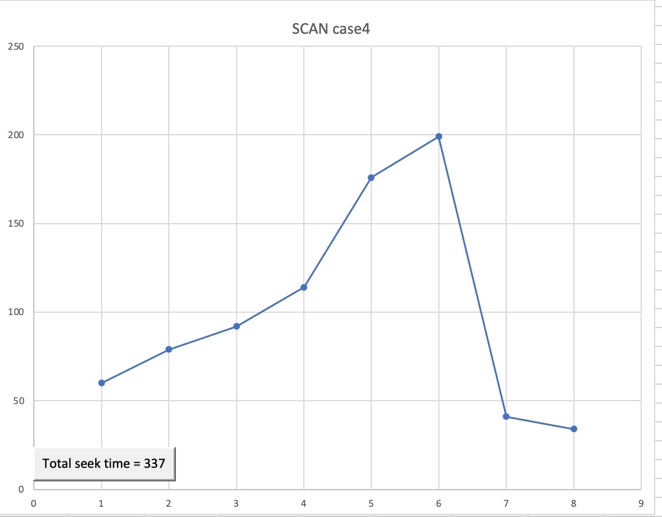
Suppose a disk drive has 200 cylinders, numbered 0 to 199. Consider a disk queue with

requests for I/o to blocks on cylinder: 176, 79, 34, 60, 92, 11, 41, 114. Assume that disk

head is currently at cylinder 50. figure 19 to figure 27 show the representation of FCFS, SSTF,

SSTF, SCAN, C-SCAN, LOOK, C-LOOK, IFCFS, SMCC and HDSA disk scheduling algorith

Respectively.

******

# Case 5 :

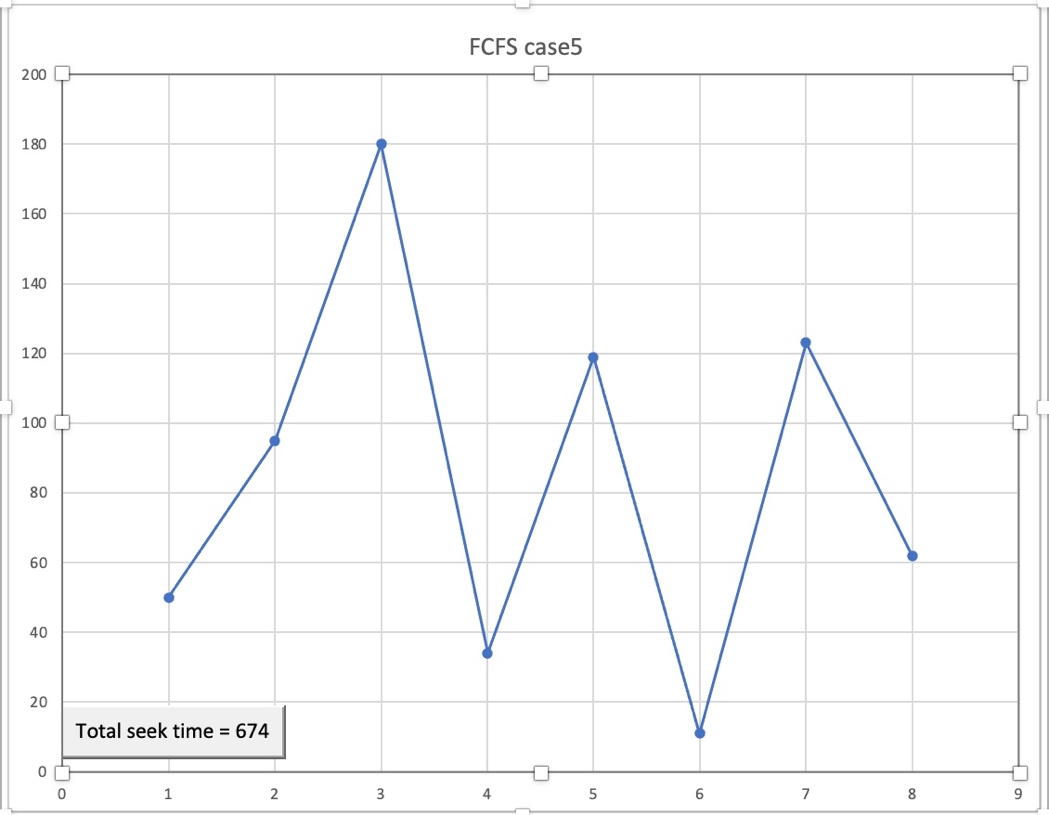
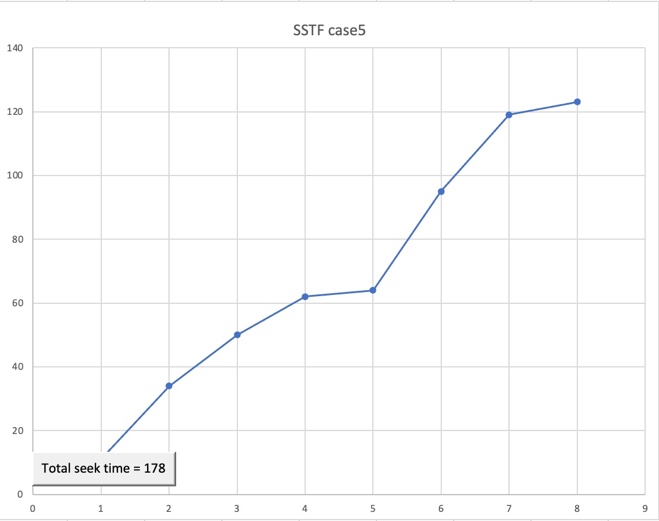
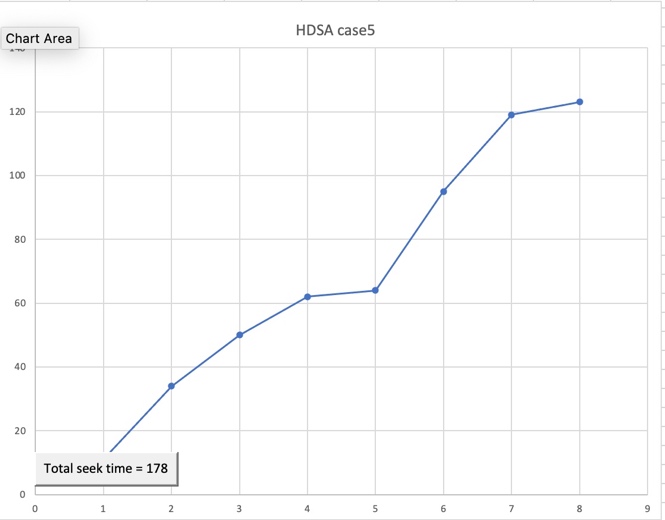
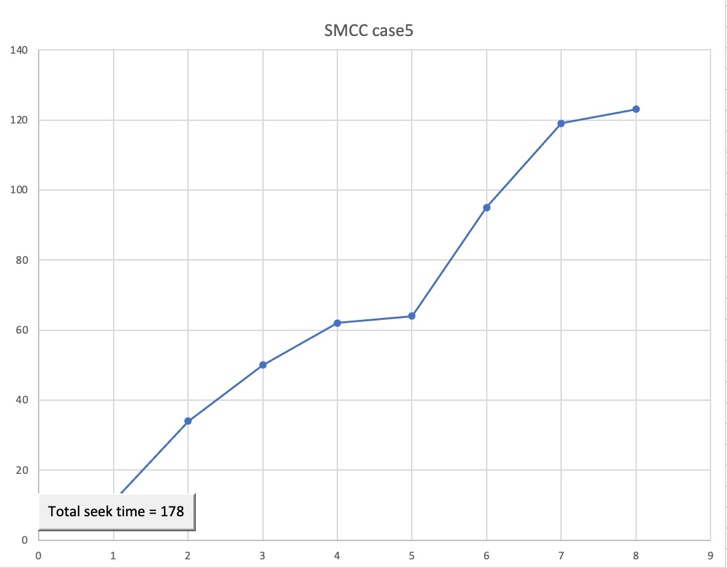
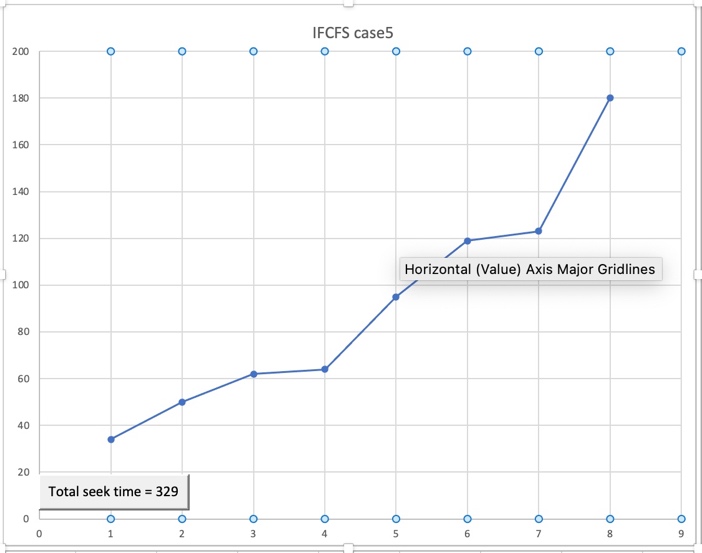
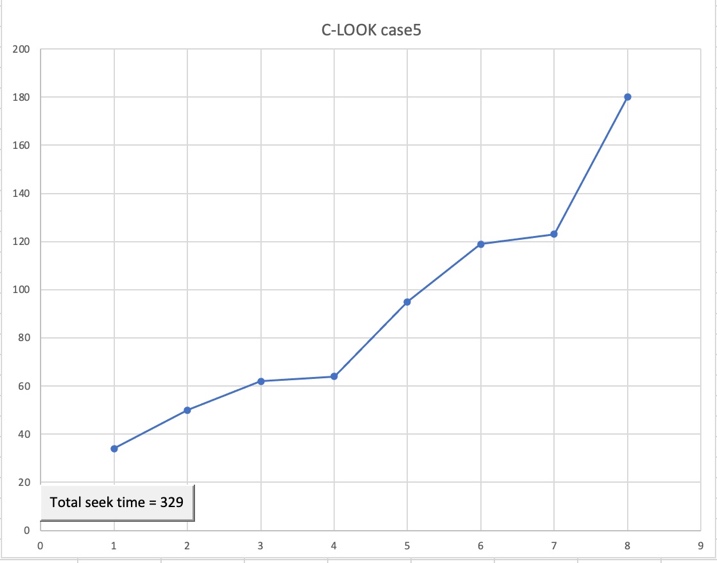
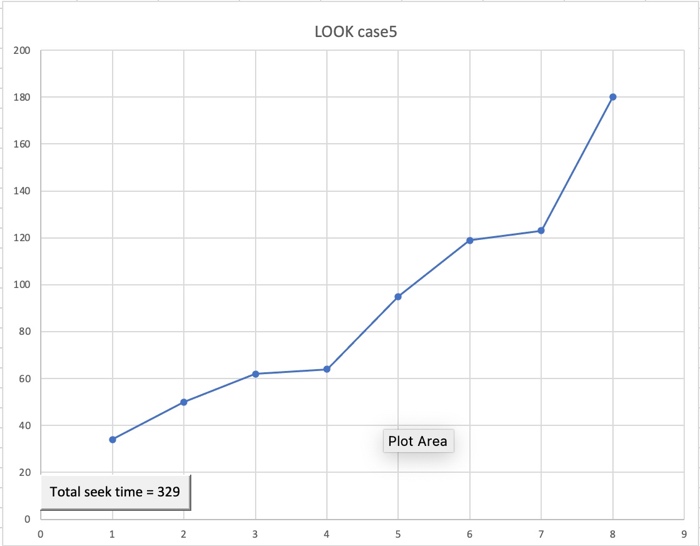
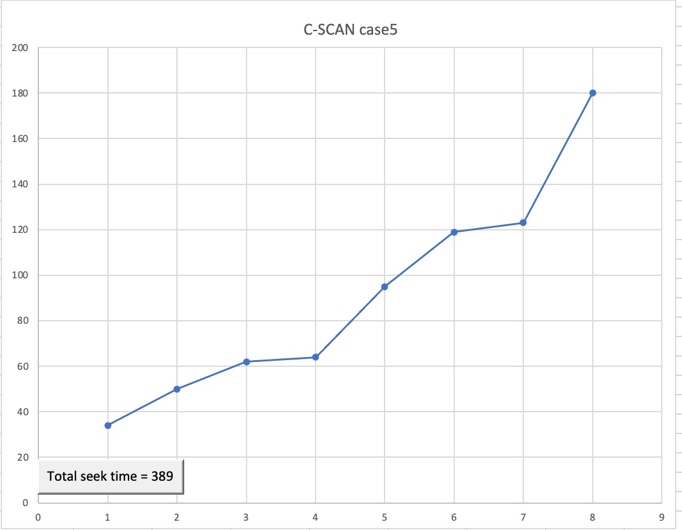
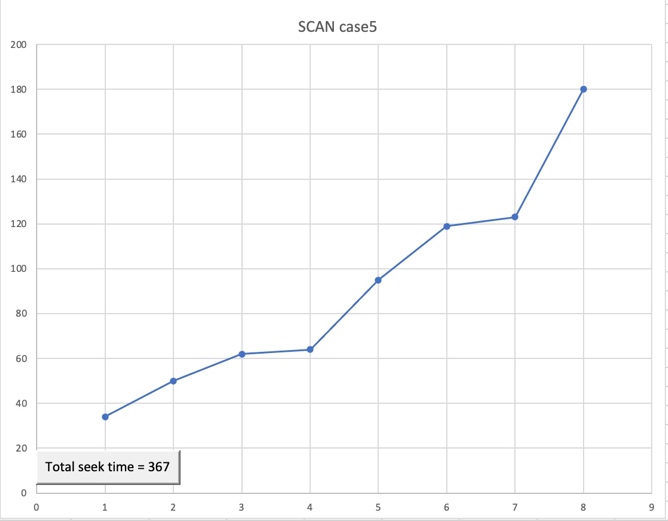
Suppose a disk drive has 200 cylinders, numbered 0 to 199. Consider a disk queue with

requests for I/o to blocks on cylinder: 50,95, 180, 34, 119, 11, 123, 62, 64. Assume that disk

head is currently at cylinder 20. figure 19 to figure 27 show the representation of FCFS, SSTF,

SSTF, SCAN, C-SCAN, LOOK, C-LOOK, IFCFS, SMCC and HDSA disk scheduling algorithm

Respectively.

******

# Case6:

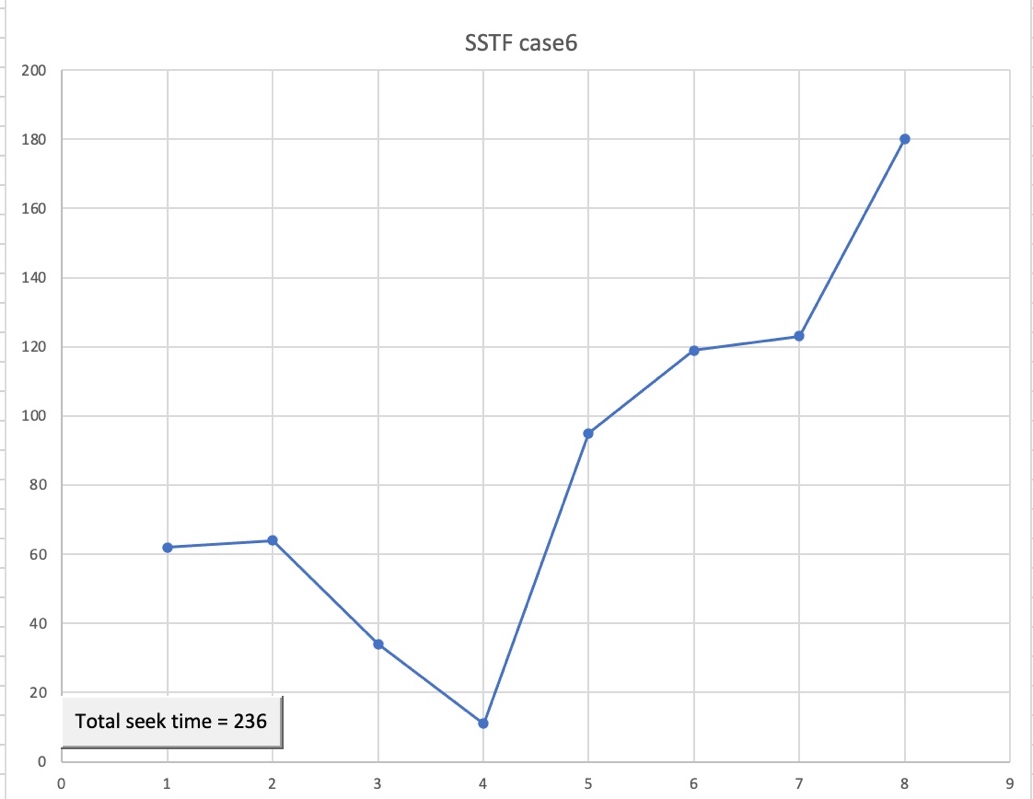
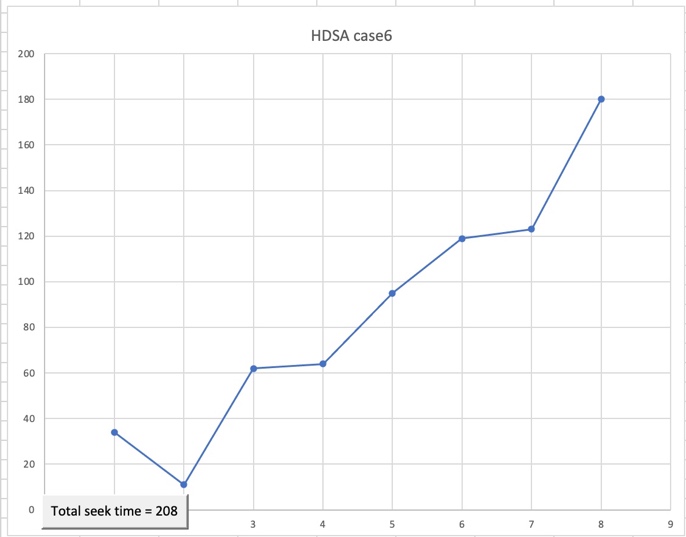
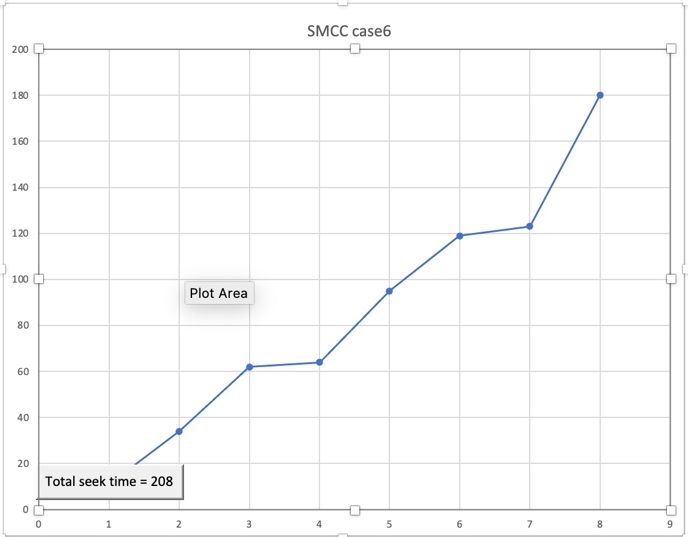
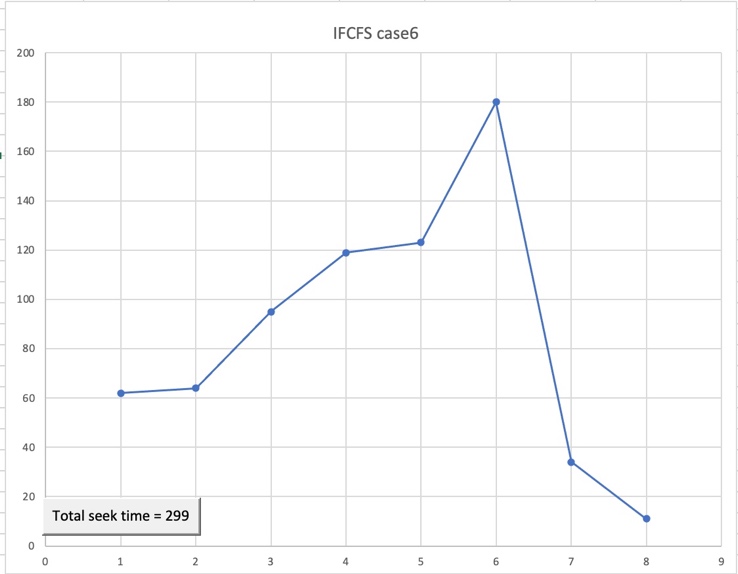
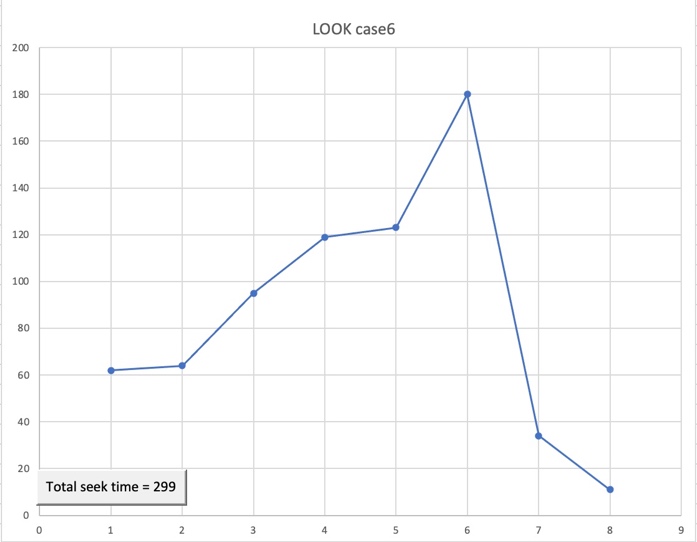
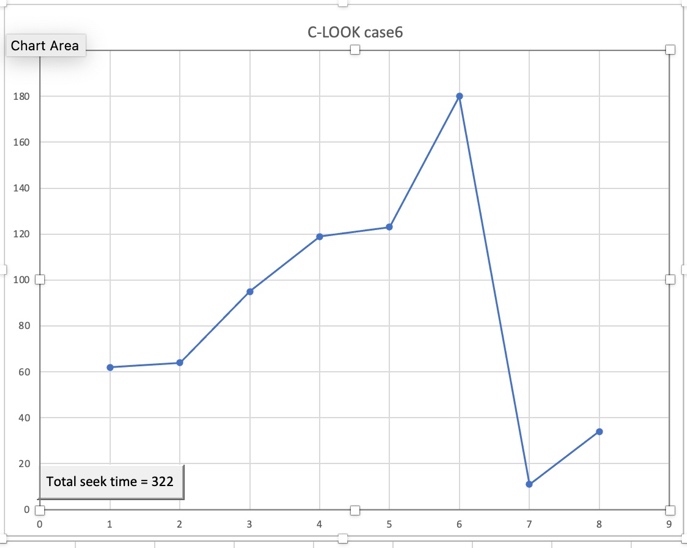
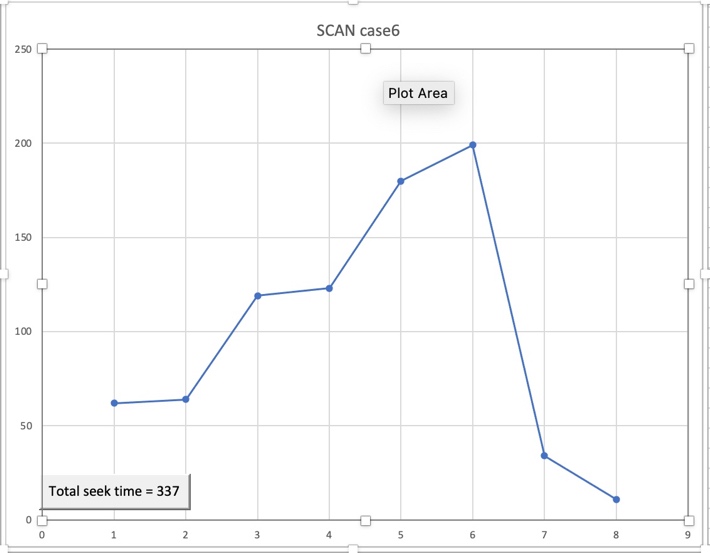
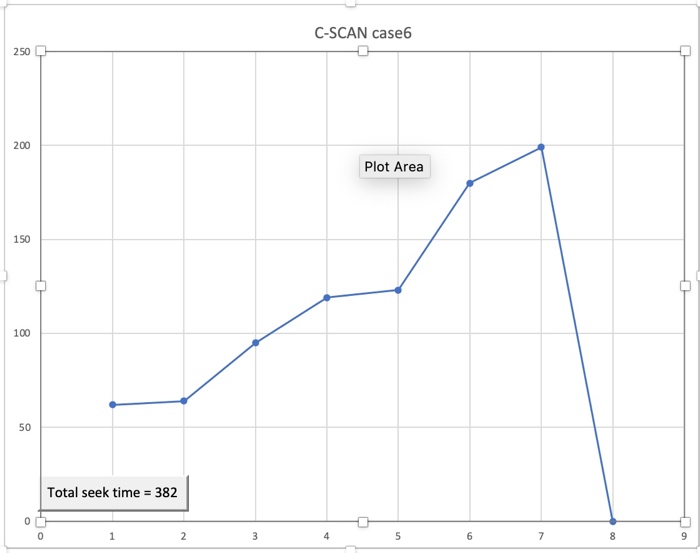
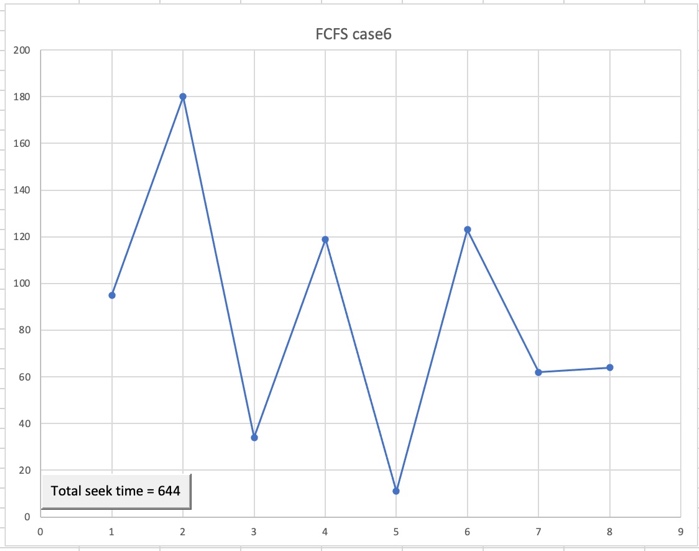
Suppose a disk drive has 200 cylinders, numbered 0 to 199. Consider a disk queue with

requests for I/o to blocks on cylinder: 95, 180, 34, 119, 11, 123, 62, 64.Assume that disk

head is currently at cylinder 50. figure 19 to figure 27 show the representation of FCFS, SSTF,

SSTF, SCAN, C-SCAN, LOOK, C-LOOK, IFCFS, SMCC and HDSA disk scheduling algorithm

Respectively.



# Section 6: Results and Analysis

Comparison of all the algorithms

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Algorithm | CASE 1 | CASE 2 | CASE 3 | CASE 4 | CASE 5 | CASE 6 |
| FCFS | 521 | 517 | 775 | 510 | 674 | 644 |
| SSTF | 254 | 238 | 259 | 204 | 178 | 236 |
| SCAN | 332 | 358 | 330 | 337 | 367 | 337 |
| C-SCAN | 398 | 395 | 382 | 389 | 389 | 382 |
| LOOK | 254 | 268 | 326 | 291 | 329 | 299 |
| C-LOOK | 288 | 285 | 374 | 321 | 329 | 322 |
| IFCFS | 254 | 268 | 259 | 291 | 329 | 299 |
| SMCC | 178 | 164 | 326 | 204 | 178 | 208 |
| HDSA | 178 | 164 | 259 | 204 | 178 | 208 |

**We conclude from table that**

We see that the HDSA algorithm had the least total seek time in all cases compared to the other scheduling algorithms. The IFCFS and SSTF are not one of the top algorithms in most of cases, however, its performance is much better than the FCFS, SCAN, C-SCAN, LOOK, and C-LOOK had a bad performance compared to the other algorithms. Moreover, we notice that the HDSA algorithm had the minimum total seek time in all six cases. Furthermore, SMCC algorithm had the minimum total seek time in five cases. As a result, we conclude that HDSA and SMCC are the most efficient algorithms in terms of total seek time.

# Section 7: Conclusion

In conclusion, we have implemented 9 algorithms including FCFS, SSTF, SSTF, SCAN, C-SCAN, LOOK, C-LOOK, IFCFS, SMCC, and HDSA in C++ programing language. After that, we compared their total seek time. The results indicated that the HDSA algorithm had the best performance in terms of total seek time. Furthermore, SMCC algorithm had the minimum total seek time in five cases. Therefore, I recommend using the HDSA algorithm since it had the least total seek time.

# References

Al-Dubai, A. Y., Ould-Khaoua, M., & Mackenzie, L. M. (2009). Performance evaluation of shortest seek time first algorithm for hard real-time systems. Journal of Systems and Software, 82(11), 1807-1814. <https://link.springer.com/article/10.1007/BF00364960>

Harchol-Balter, M., & Silberschatz, A. (1996). Scheduling for reduced CPU energy. In Proceedings of the 3rd International Workshop on Mobile Multimedia Communications (pp. 157-172). <https://dl.acm.org/doi/abs/10.1145/1151690.1151693>

Tutorials Point - Operating System Scheduling Algorithms:<https://www.tutorialspoint.com/operating_system/os_process_scheduling_algorithms.htm>

Kumar Mishra, M. (2012). An Improved FCFS (IFCFS) Disk Scheduling Algorithm. International Journal of Computer Applications, 47(13), 20–24. <https://doi.org/10.5120/7248-0298>

Geeks for Geeks - Program for Round Robin Scheduling for the same Arrival time

<https://www.geeksforgeeks.org/program-round-robin-scheduling-set-1/>

Yan, Y., & Hwang, K. (2004). A survey of I/O scheduling techniques. ACM Computing Surveys (CSUR), 36(4), 422-438. [https://dl.acm.org/doi/10.1145/1041680.1041683](https://dl.acm.org/doi/10.1145/1041680.1041683" \t "_blank)

Javed, M., & Khan, I. (2000). Simulation and performance comparison of four disk scheduling

algorithms. 2000 TENCON Proceedings. Intelligent Systems and Technologies for the

New Millennium (Cat. No.00CH37119). https://doi.org/10.1109/tencon.2000.888379

Kumar, M. M., & Rajendra, B. R. (2015). An Improved Approach to Maximize the Performance

of Disk Scheduling Algorithm by Minimizing the Head Movement and Seek Time Using

Sort Mid-Current Comparison (SMCC) Algorithm. *Procedia Computer Science*, *57*, 222–

231. https://doi.org/10.1016/j.procs.2015.07.468

Kumar Mishra, M. (2012). An Improved FCFS (IFCFS) Disk Scheduling Algorithm.

*International Journal of Computer Applications*, *47*(13), 20–24.

[https://doi.org/10.5120/7248-0298](file:///Users/kerosalama/Desktop/os%20_firstone.docx)

Muqaddas, A., Hanady, M., & Salman, A. (2009, October). *S-LOOK: A Preemptive Disk*

*Scheduling Algorithm for Offline and Online Environments*. Computer Science and

Information Technologies, LVIV, UKRAINE.

[https://www.researchgate.net/publication/237498360\_S LOOK\_A\_Preemptive\_Disk\_Scheduling\_Algorithm\_for\_Offline\_and\_Online\_Environments](file:///Users/kerosalama/Desktop/os%20lt.docx)

T.A., A., B.O., A., & S.F., O. (2019). A Hybridized Disk Scheduling Algorithm (HDSA).

*International Journal of Scientific and Research Publications (IJSRP)*, *9*(3), p8789.

[https://doi.org/10.29322/ijsrp.9.03.2019.p8789](file:///Users/kerosalama/Desktop/os%20lt.docx)

Rasool, Saman & Gakher, Ritika. (2015). Reformed FCFS Disk Scheduling Algorithm.

International Journal of Computer Applications. 127. 38-41. 10.5120/ijca2015906592.

<https://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.735.3339&rep=rep1&type=pdf>

M.A, Kashem & Saha, Sandipon & Naderuzzaman, Mohammad. (2013). A New Approach of

Disk Scheduling Algorithm.

[https://www.researchgate.net/publication/301890138\_A\_New\_Approach\_of\_Disk\_Scheduling\_](file:///Users/kerosalama/Desktop/os lt.docx)

[Algorithm](file:///Users/kerosalama/Desktop/os lt.docx)

C.Mallikarjuna & P.Chitti Babu.(2016). Performance Analysis of Disk Scheduling Algorithms.

[https://www.academia.edu/25944127/Performance\_Analysis\_of\_Disk\_Scheduling\_Algorithms](file:///Users/kerosalama/Desktop/os%20_firstone.docx)