**Batch: C1**

**Roll No.: 16010122221**

**Experiment No. 09**

**Grade: AA / AB / BB / BC / CC / CD /DD**

**Signature of the Staff In-charge with date**

**TITLE:** Implementation of Disk scheduling policies

**AIM:** Implementation of Disk scheduling algorithms - FCFS,SSTF,SCAN, CSCAN, LOOK

## Expected Outcome of Experiment:

**CO 4.** To understand various Memory, I/O and File management techniques.

## Books/ Journals/ Websites referred:

1. **Silberschatz A., Galvin P., Gagne G. “Operating Systems Principles”, Willey Eight edition.**

## Achyut S. Godbole , Atul Kahate “Operating Systems”, McGraw Hill Third Edition.

1. **Sumitabha Das “ UNIX Concepts & Applications”, McGraw Hill Second Edition.**

## Pre Lab/ Prior Concepts:

Memory management, process transitions, Relation between CPU scheduling and disc scheduling

# Assigned Algorithm 1 Details: First-Come, First-Served (FCFS):

**Overview:** FCFS (First-Come, First-Served) is the simplest disk scheduling algorithm. It processes disk I/O requests in the order they arrive. This method does not consider the position of the disk head or the distance to the next request, which can lead to inefficient disk usage and increased seek times.

**Performance Characteristics:**

* **Total Seek Time:** The total time taken for the disk arm to move to all requested positions.
* **Complexity:** O(n), where n is the number of requests, because it processes each request sequentially.
* **Efficiency:**
* **Advantages:** Simple and easy to implement.
* **Disadvantages:** Can result in significant seek time, particularly for a large number of scattered requests.

# Assigned Algorithm 2 Details: Shortest Seek Time First (SSTF):

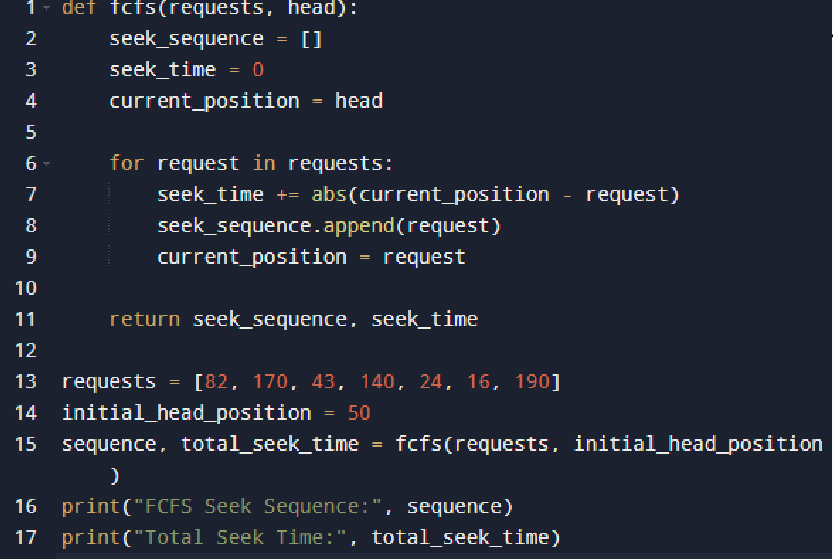
**Overview:** SSTF (Shortest Seek Time First) selects the disk I/O request that is closest to the current head position, thereby minimizing the seek time for each individual request. This strategy is more efficient than FCFS but can lead to starvation.

**Performance Characteristics:**

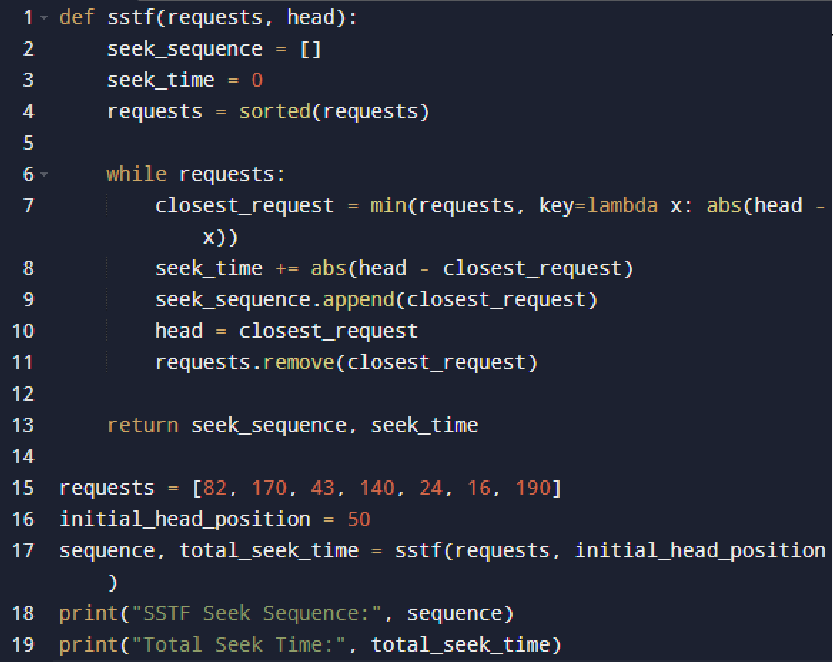
* **Total Seek Time:** Usually lower than FCFS due to the prioritization of closer requests.
* **Complexity:** O(n²) in the worst case, since finding the closest request requires searching through the remaining requests.
* **Efficiency:**
* **Advantages:** Reduces average seek time significantly.
* **Disadvantages:** Starvation can occur for requests that are far from the current head if newer requests are consistently closer.

**Source code:**

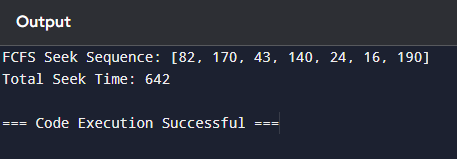
1. FCFS:

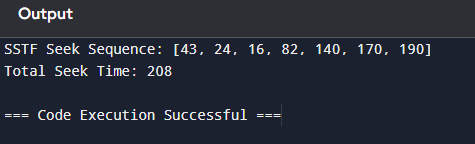


1. SSTF:



**Output screenshots:**

* 1. FCFS:
  2. SSTF:



**Conclusion:**

The FCFS algorithm is straightforward but can lead to inefficient disk usage, resulting in higher seek times. In contrast, SSTF is more efficient as it reduces total seek time by prioritizing the nearest request. However, SSTF may lead to starvation, where some requests may wait indefinitely if newer requests keep arriving closer to the head.

**Post Lab Descriptive Questions**

1. **Compare the performance of FCFS, SSTF, SCAN, C-SCAN, and LOOK in terms of total seek time for the same set of disk requests. Which algorithm gives the best performance and why?** FCFS generally has the highest total seek time due to its lack of optimization. SSTF improves performance significantly by minimizing seek time for each request. SCAN and C-SCAN further optimize performance by moving the head in a specific direction and servicing requests in that path, reducing overall wait times. LOOK enhances SCAN by only servicing requests that are present, preventing unnecessary movements to the end of the disk. Overall, SCAN or LOOK often provides the best performance due to their systematic approach to handling requests.

### Explain how SSTF can lead to starvation. Can this problem occur in SCAN or LOOK? Justify your answer.

SSTF can lead to starvation when requests far from the current head position are repeatedly overshadowed by closer requests, causing them to wait indefinitely. SCAN and LOOK do not suffer from this problem to the same extent because they service requests in one direction until they reach the end or the closest request in that direction, ensuring that all requests are eventually addressed. However, under heavy load or with a skewed distribution of requests, SCAN and LOOK can also lead to longer wait times, but they systematically cover the entire range of requests.

### How does the size of the disk (number of tracks) influence the performance of the SCAN, C-SCAN, and LOOK algorithms? Provide examples to support your answer.

The size of the disk affects the performance of SCAN, C-SCAN, and LOOK significantly. For larger disks, the distance the head travels increases, which can lead to longer seek times if requests are spread out. For instance, if a disk has 200 tracks and requests are spread from track 0 to track 199, SCAN will have to travel the entire length, resulting in higher total seek times compared to a smaller disk with requests clustered closer together. C-SCAN improves on this by always returning to the starting point after reaching the end, providing a more uniform wait time. Conversely, LOOK can be more efficient on larger disks if requests are concentrated within certain ranges, as it limits unnecessary movements beyond the furthest request in its current direction.

## Date: