Department of Computer Science & Engineering

Course Title: Operating System Lab

Course Code: CSE 406

Lab Report No: Mid Q2

Lab Report: SSTF Disk Scheduling

Submitted To:	**Submitted By:**
Atia Rahman Orthi	**Name: Amirul Islam Papon**
Lecturer,	**Reg No: 21201076**
Department of CSE, UAP	**Sec: B**

Objective

To understand and implement the SSTF (Shortest Seek Time First) disk scheduling algorithm and analyze its performance in terms of head movement.

Theory

Disk scheduling algorithms determine the order in which pending disk I/O requests are serviced. The SSTF algorithm selects the request that is closest to the current head position, thereby minimizing the seek time.

This algorithm improves average response time compared to FCFS, but it may cause starvation for distant requests.

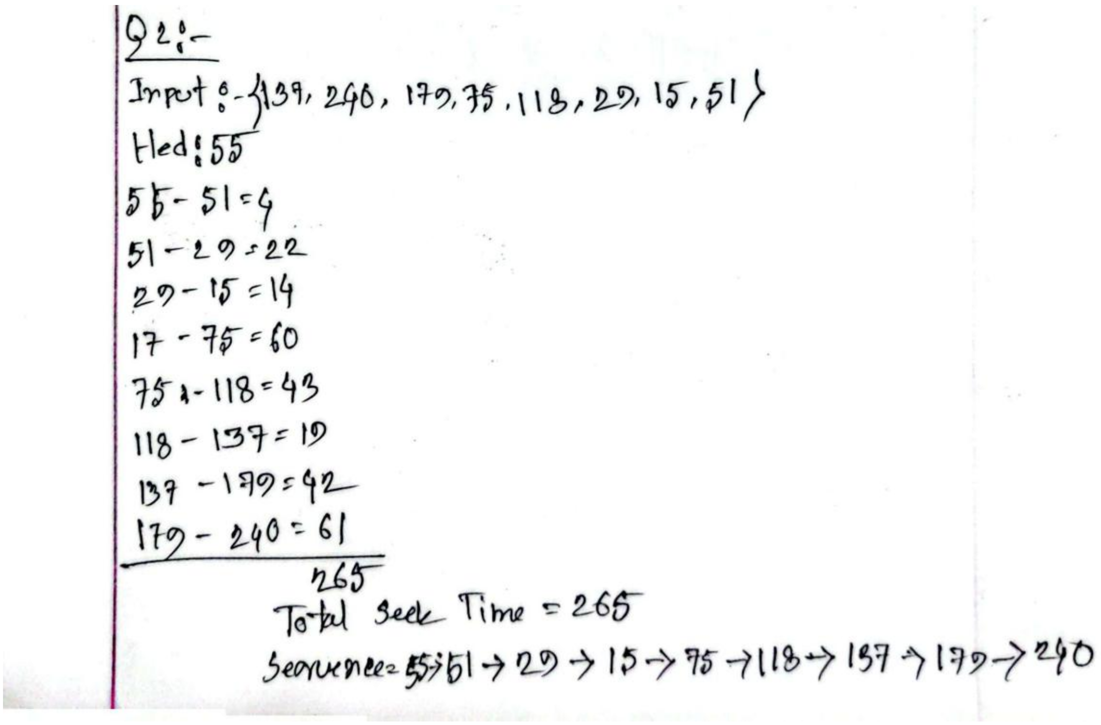
Algorithm Steps

1. Start with the initial head position.
2. From the list of pending requests, choose the one with the shortest seek time (smallest distance from the current head position).
3. Move the head to that request, service it, and remove it from the list.
4. Repeat until all requests are serviced.

Input

- Disk I/O Requests: [137, 240, 179, 75, 118, 29, 15, 51]
- Initial Head Position: 55

Solve



Q2:-

Input :- {137, 240, 179, 75, 118, 29, 15, 51}

Hed: 55

55 - 51 = 4

51 - 29 = 22

29 - 15 = 14

17 - 75 = 60

75 - 118 = 43

118 - 137 = 19

137 - 179 = 42

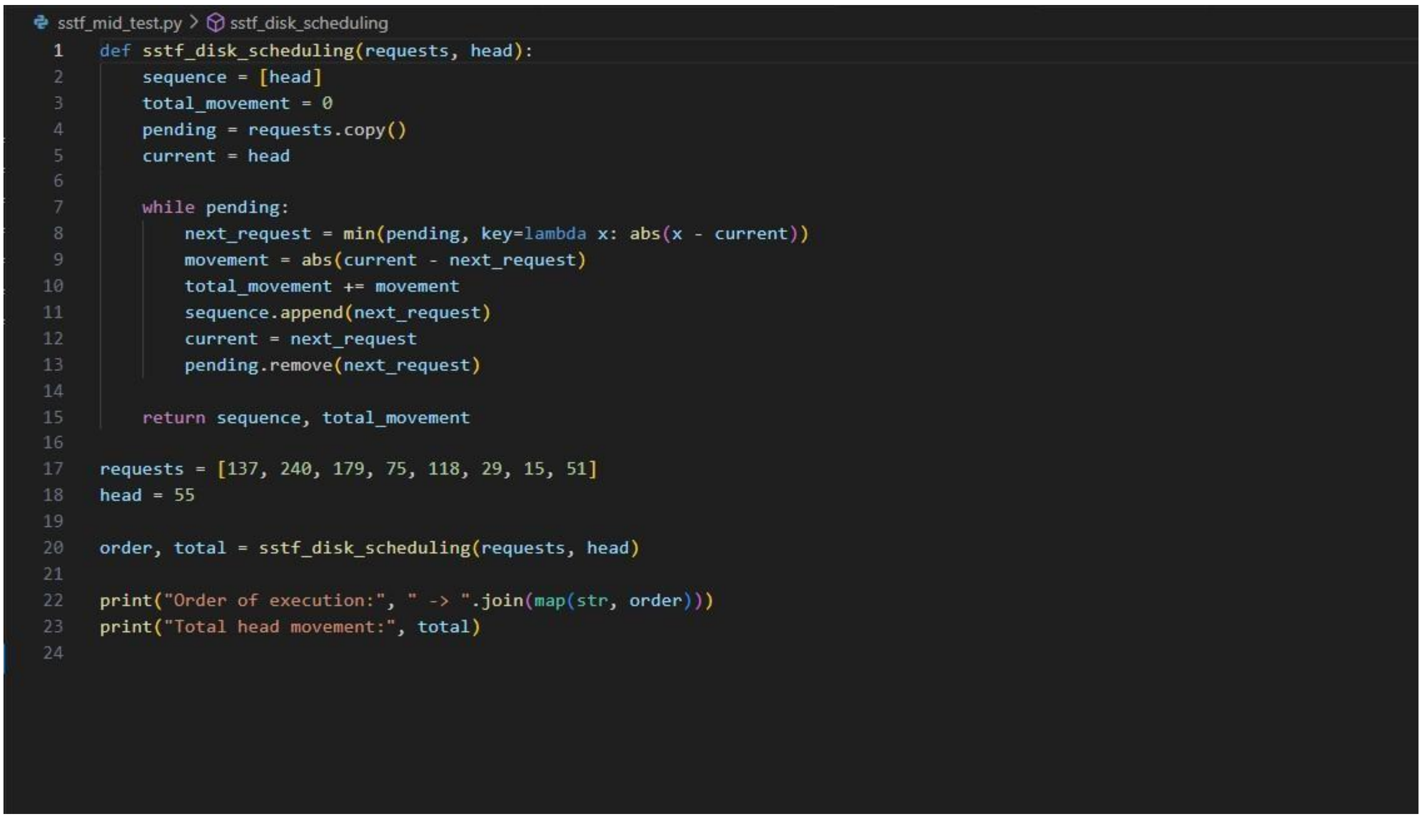
179 - 240 = 61

265

Total Seek Time = 265

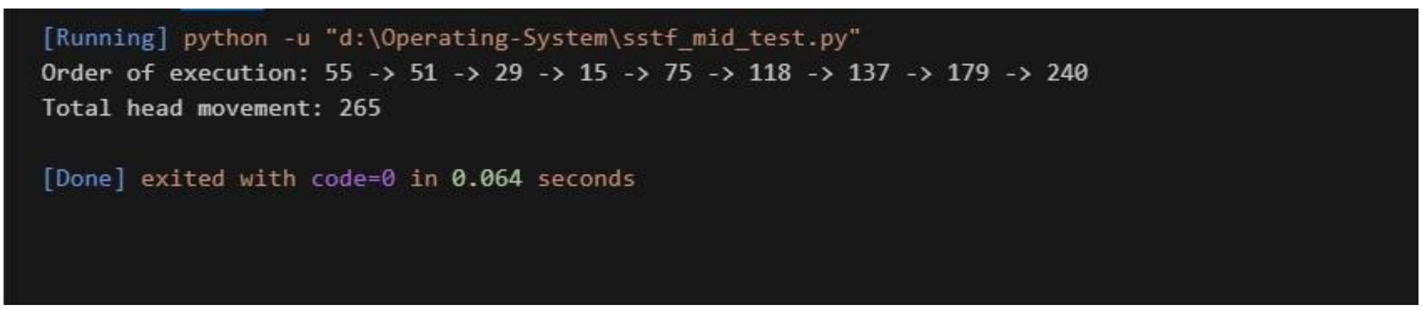
Sequence= 55→51 → 29 → 15 → 75 → 118 → 137 → 179 → 240

Source Code



```python
def sstf_disk_scheduling(requests, head):
    sequence = [head]
    total_movement = 0
    pending = requests.copy()
    current = head

    while pending:
        next_request = min(pending, key=lambda x: abs(x - current))
        movement = abs(current - next_request)
        total_movement += movement
        sequence.append(next_request)
        current = next_request
        pending.remove(next_request)

    return sequence, total_movement

requests = [137, 240, 179, 75, 118, 29, 15, 51]
head = 55

order, total = sstf_disk_scheduling(requests, head)

print("Order of execution:", " -> ".join(map(str, order)))
print("Total head movement:", total)
```

Output



```
[Running] python -u "d:\Operating-System\sstf_mid_test.py"
Order of execution: 55 -> 51 -> 29 -> 15 -> 75 -> 118 -> 137 -> 179 -> 240
Total head movement: 265

[Done] exited with code=0 in 0.064 seconds
```

Advantages

- Reduces the total seek time compared to FCFS.
- Prioritizes nearer requests, improving overall efficiency.

Disadvantages

- Can cause starvation for requests that are far from the current head position.
- Performance may degrade with continuous addition of close requests.

Conclusion

The SSTF algorithm is an effective disk scheduling technique for minimizing seek time. However, its potential for starvation means it may not be suitable for real-time systems. The SSTF algorithm was successfully implemented and tested, showing a total head movement of 265 cylinders for the given input.

Github Link: https://github.com/Amirul-Islam-Papon/Operating-System/blob/main/sstf_mid_test.py