Department Of Computer Science and Engineering

Course Title: Operating System Lab

Course Code: CSE 406

Title: SCAN Disk Scheduling Algorithm

Submitted To
Atia Rahman Orthi
Lecturer
Department Of Computer Science & Engineering

Submitted By
Md. Atikul Islam Atik
Reg No:21201063
Roll:63
Sec:B1

Lab MID Exam

Submission Date: 15.4.2025

Request Sequence: `{137, 240, 179, 75, 118, 29, 15, 51}`

Initial Head Position: `55`

Output: `410`

Step-by-Step Execution:

1. Initial Position: 55
2. Requests >= 55 (Right side): 75, 118, 137, 179, 240

   Requests < 55 (Left side): 51, 29, 15 → (Reversed as per SCAN logic)

Head Movement Order:

`55 → 75 → 118 → 137 → 179 → 240 → 51 → 29 → 15`

Step-by-step head movement and seek calculation:

1. The head starts at position 55.
2. Requests are divided into two parts:
   - Right side (greater than or equal to 55): 75, 118, 137, 179, 240
   - Left side (less than 55): 51, 29, 15
3. The right side is processed in ascending order: 75 → 118 → 137 → 179 → 240
4. After reaching the farthest request on the right (240), it reverses and serves the left side in descending order: 51 → 29 → 15

Seek operation calculation:

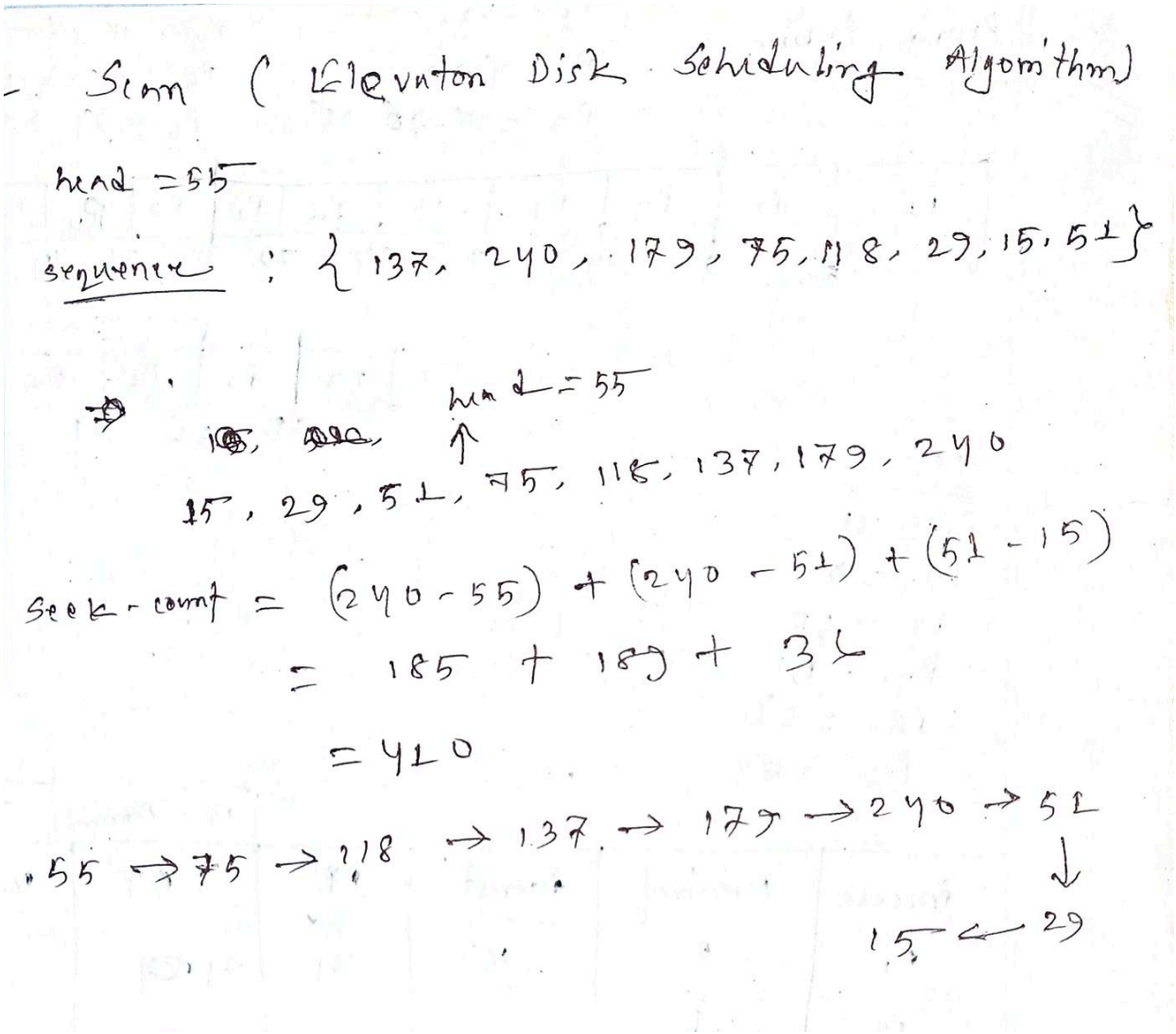
- From 55 to 75: absolute difference is |75 - 55| = 20
- From 75 to 118: |118 - 75| = 43
- From 118 to 137: |137 - 118| = 19
- From 137 to 179: |179 - 137| = 42
- From 179 to 240: |240 - 179| = 61
- From 240 to 51: |51 - 240| = 189
- From 51 to 29: |29 - 51| = 22
- From 29 to 15: |15 - 29| = 14

Total Seek Count:

20 + 43 + 19 + 42 + 61 + 189 + 22 + 14 = 410

Code Link (Github)

Hand Calculation



Algorithm (SCAN Disk Scheduling)

1. Sort all disk I/O requests in ascending order.
2. Divide the sorted list into two groups:
    - Requests smaller than the initial head position.
    - Requests greater than or equal to the initial head position.

3. Move the disk head in the current direction (usually towards the higher cylinder numbers) and service all requests along the way.
4. When the head reaches the end in that direction, it reverses direction.
5. Service the remaining requests in the opposite direction, continuing from the end back toward the starting point.

Conclusion:

The SCAN disk scheduling algorithm operates similarly to an elevator: it moves in one direction, servicing requests until it reaches the end, and then reverses to handle the remaining ones. This approach helps reduce the variance in response times and avoids starvation of requests. Compared to FCFS and SSTF, SCAN provides a more balanced and predictable performance, especially when the request queue is heavily populated on both sides of the current head position.