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

Submission Date: 10.4.2025

Input: 14,41,53,65,67,98,122,124,183,199

Head=53

Output:331

Code snippet

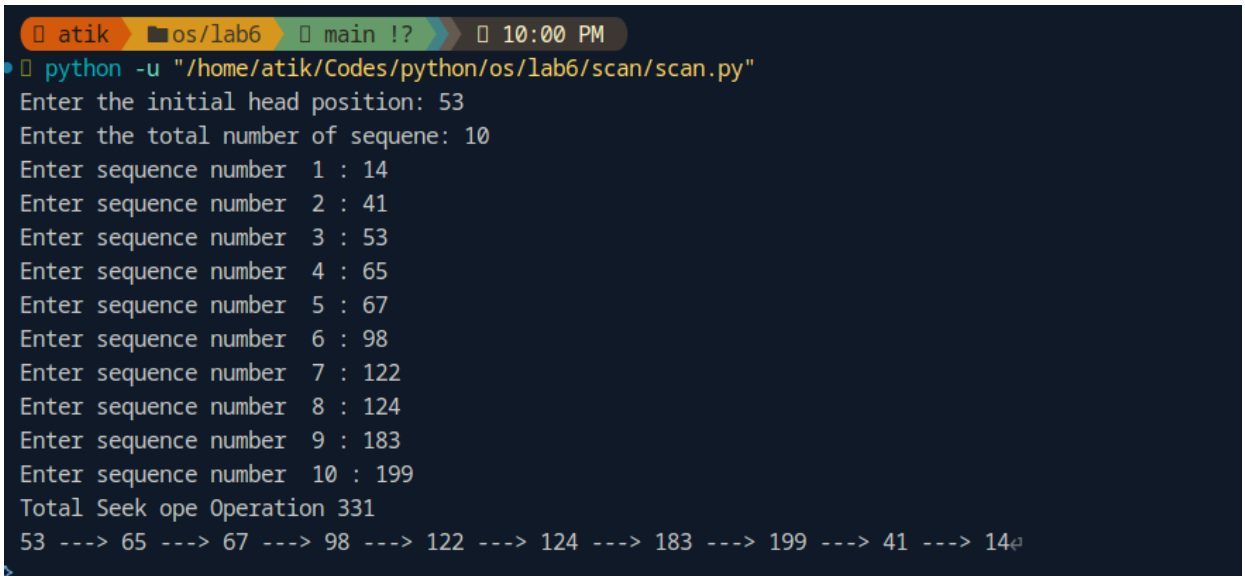
```
def scan_disk_sheduling(request_sequence, initial_head):
   current_head = initial_head
   sequence = []
   total_seektime = 0
   request_sequence = sorted(request_sequence)

   left = [r for r in request_sequence if r < current_head]
   right = [r for r in request_sequence if r >= current_head]
   left.reverse()

   for request in  right + left:
       if request > 0:
           total_seektime += abs(request - current_head)
       sequence.append(request)
       current_head = request
   return total_seektime, sequence
```

```python
def take_input():
   head = int(input("Enter the initial head position: "))
   n = int (input("Enter the total number of sequence: "))

   req_sequences = []

   for i in range(n):
       req = int(input(f'Enter sequence number  {i+1} : '))
       req_sequences.append(req)

   return req_sequences, head

def print_sequence(sequences):
   for i in range(len(sequences)):
       if i < len(sequences)-1:
           print(sequences[i], end=" ---> ")
       else:
           print(sequences[i], end="")
```

```python
if __name__ == "__main__":
    inputs = take_input()
    req_sequences = inputs[0]
    head = inputs[1]

    res = scan_disk_sheduling(req_sequences, head)
    print("Total Seek ope Operation", res[0])
    print_sequence(res[1])
```

Code Link (Github)

Output Snippet



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.