**Department Of Computer Science and Engineering**

**Course Title:** Operating System Lab

**Course Code:** CSE 406

**Title:** C-SCAN Disk Scheduling Algorithm

| Submitted To | Submitted By |
|---|---|
| **Atia Rahman Orthi** | **Md. Atikul Islam Atik** |
| Lecturer | Reg No:21201063 |
| Department Of Computer Science & | Roll:63 |
| Engineering | Sec:B1 |

Submission Date: 10.4.2025

**Input: 0,14,41,53,65,67,98,122,124,183,199**

**Head=53**

**Output:386**

## Code snippet

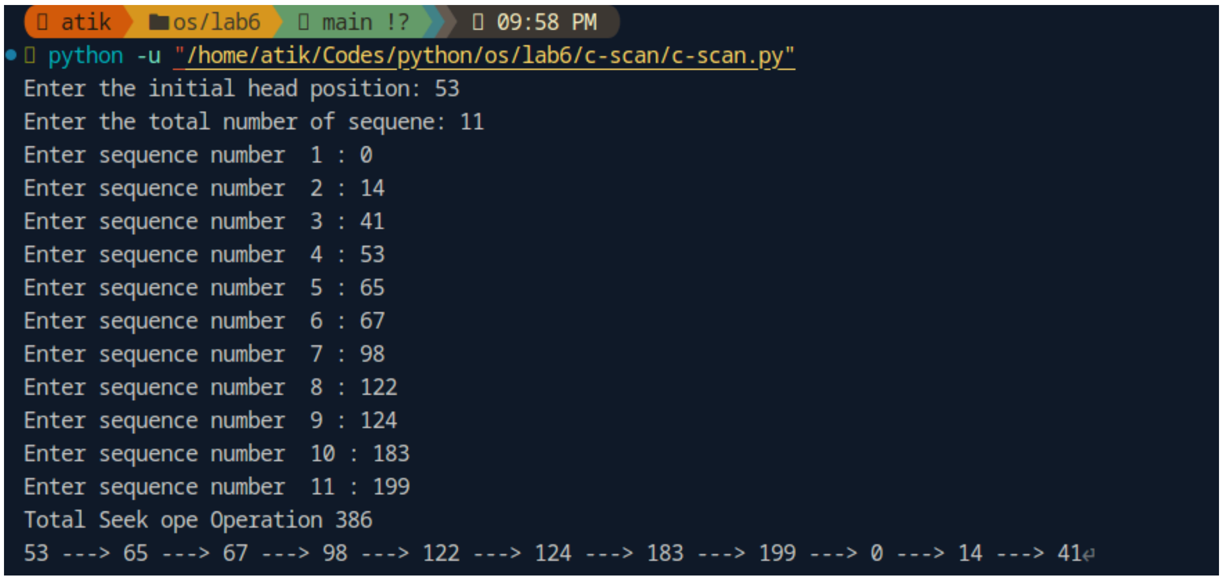
```python
def scan_disk_sheduling(request_sequence, initial_head):
   current_head = initial_head
   sequence = []
   total_seektime = 0
   request_sequence = sorted(request_sequence)

   left = [r for r in request_sequence if r < current_head]
   right = [r for r in request_sequence if r >= current_head]



   for request in  right + left:
       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



```
 atik  os/lab6  main !?  09:58 PM
• python -u "/home/atik/Codes/python/os/lab6/c-scan/c-scan.py"
 Enter the initial head position: 53
 Enter the total number of sequene: 11
 Enter sequence number  1 : 0
 Enter sequence number  2 : 14
 Enter sequence number  3 : 41
 Enter sequence number  4 : 53
 Enter sequence number  5 : 65
 Enter sequence number  6 : 67
 Enter sequence number  7 : 98
 Enter sequence number  8 : 122
 Enter sequence number  9 : 124
 Enter sequence number  10 : 183
 Enter sequence number  11 : 199
 Total Seek ope Operation 386
 53 ---> 65 ---> 67 ---> 98 ---> 122 ---> 124 ---> 183 ---> 199 ---> 0 ---> 14 ---> 41
```

## Algorithm (C-SCAN Disk Scheduling)

1. Begin by sorting the list of disk I/O requests in ascending order.
2. Divide the sorted requests into two parts:
    - Requests greater than or equal to the initial head position.
    - Requests less than the initial head position.
3. Move the disk head in one direction (towards the higher-numbered cylinders), servicing each request until the end of the disk is reached.
4. Once the end is reached, the head moves to the start of the disk **without servicing any requests during this jump**.
5. Continue servicing the remaining requests (those that were initially less than the head) from the beginning of the disk towards the initial position.
6. The total seek time is calculated by summing the absolute movements of the head for all serviced requests and the jump from end to start.

## Conclusion:

The Circular SCAN (C-SCAN) algorithm ensures a more consistent and fair servicing time for disk I/O requests by always moving the head in a single direction. Unlike traditional SCAN, C-SCAN treats the disk as a circular queue, reducing the variability in wait time between the requests at the start and end of the disk. Although it includes a jump from the last cylinder back to the first, this approach benefits systems with heavy and uniformly distributed I/O loads by providing predictable performance.