Department of Computer Science & Engineering

Course Title: Operating System Lab

Course Code: CSE 406

Lab Report No: 08

Lab Report: Scan Disk Scheduling

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

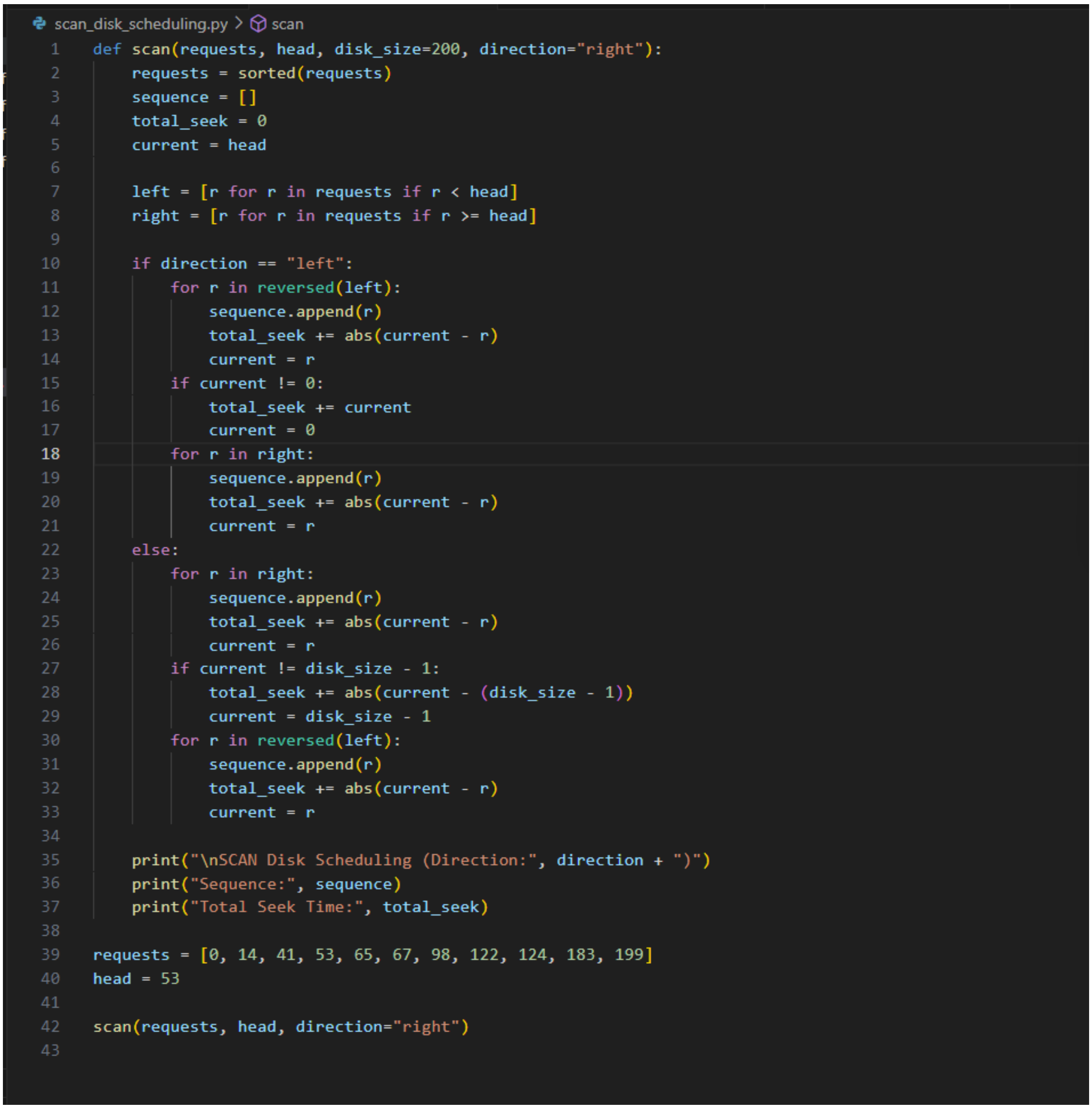
Problem Statement:

To implement the SCAN disk scheduling algorithm that moves the disk arm towards one end, servicing requests in order, and then reverses direction to continue servicing remaining requests.

Steps:

- Accept the request sequence and the initial head position.
- Divide the requests into two lists: those greater than and less than the current head.
- Move the head in the specified direction (right) servicing all requests.
- Once the end is reached, reverse direction and continue servicing.
- Calculate the total seek time and servicing sequence.

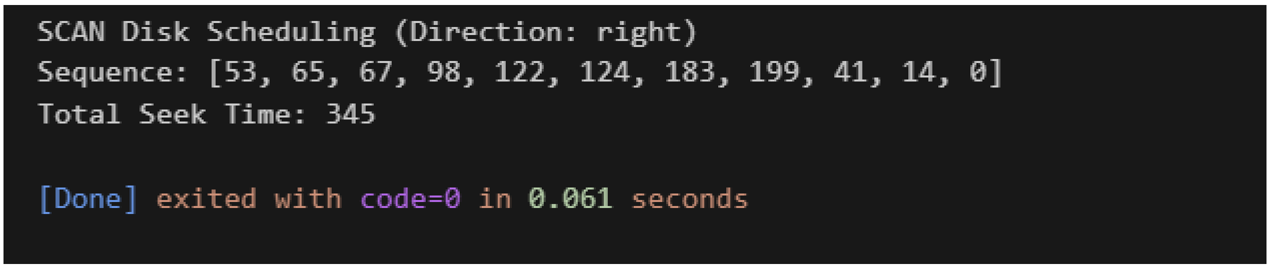
Code:



scan_disk_scheduling.py > scan

```python
def scan(requests, head, disk_size=200, direction="right"):
    requests = sorted(requests)
    sequence = []
    total_seek = 0
    current = head

    left = [r for r in requests if r < head]
    right = [r for r in requests if r >= head]

    if direction == "left":
        for r in reversed(left):
            sequence.append(r)
            total_seek += abs(current - r)
            current = r
        if current != 0:
            total_seek += current
            current = 0
        for r in right:
            sequence.append(r)
            total_seek += abs(current - r)
            current = r
    else:
        for r in right:
            sequence.append(r)
            total_seek += abs(current - r)
            current = r
        if current != disk_size - 1:
            total_seek += abs(current - (disk_size - 1))
            current = disk_size - 1
        for r in reversed(left):
            sequence.append(r)
            total_seek += abs(current - r)
            current = r

    print("\nSCAN Disk Scheduling (Direction:", direction + ")")
    print("Sequence:", sequence)
    print("Total Seek Time:", total_seek)

requests = [0, 14, 41, 53, 65, 67, 98, 122, 124, 183, 199]
head = 53

scan(requests, head, direction="right")
```

Output:



```
SCAN Disk Scheduling (Direction: right)
Sequence: [53, 65, 67, 98, 122, 124, 183, 199, 41, 14, 0]
Total Seek Time: 345

[Done] exited with code=0 in 0.061 seconds
```

Discussion:

SCAN works like an elevator – it moves in one direction and services all requests until the end, then reverses. This ensures better fairness and avoids starvation.

Given:

Requests = [0, 14, 41, 53, 65, 67, 98, 122, 124, 183, 199]
Head = 53
Direction = "right"

The head moves to the end (199) servicing requests along the way, then reverses and services the lower requests.

Conclusion:

SCAN improves fairness over SSTF and avoids starvation, making it suitable for multi-user systems or high-volume disk operations.

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