Operating Systems Lab
Fall 2024-25(L59+60)

Student Name:- Parth Suri
Registration No:-22BDS0116

Class No. :- VL2024250102445

Faculty Name:- Rahul Srivastava

To implement and analyze FCFS, SSTF, and SCAN disk scheduling algorithms to optimize seek time.

Theory:
1. FCFS (First-Come, First-Served): Processes requests in arrival order. Simple but may cause high seek times.
2. SSTF (Shortest Seek Time First): Services the request nearest to the current head position, reducing seek time.
3. SCAN: Moves the head in one direction, servicing requests, then reverses direction (elevator algorithm).

Pseudocode:

1. FCFS:
Input: Requests R[], Initial head position head
1. Initialize seek_time = 0, current = head
2. For each request in R[]: a. seek_time += abs(request - current) b. current = request
3. Return seek_time and order

2. SSTF:
Input: Requests R[], Initial head position head
1. Initialize seek_time = 0, current = head
2. While R[] is not empty:
a. Find closest_request to current
b. seek_time += abs(closest_request - current)
c. current = closest_request, remove it from R[]
3. Return seek_time and order

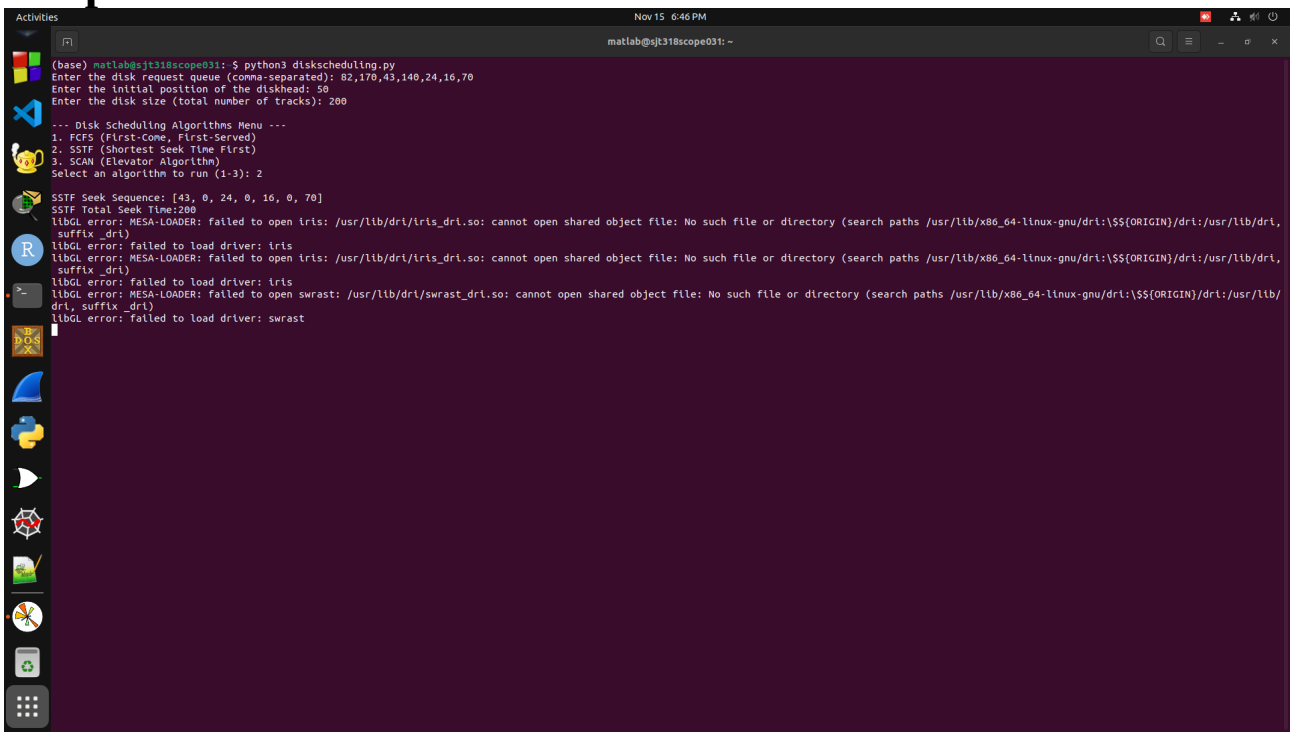
3. SCAN:
Input: Requests R[], Initial head position head, Direction (up/down)
1. Sort R[], split into lower and higher than head
2. If direction is up:
a. Service higher requests in ascending order, then lower in descending
3. If direction is down:
a. Service lower requests in descending order, then higher in ascending
4. Calculate seek_time and return order

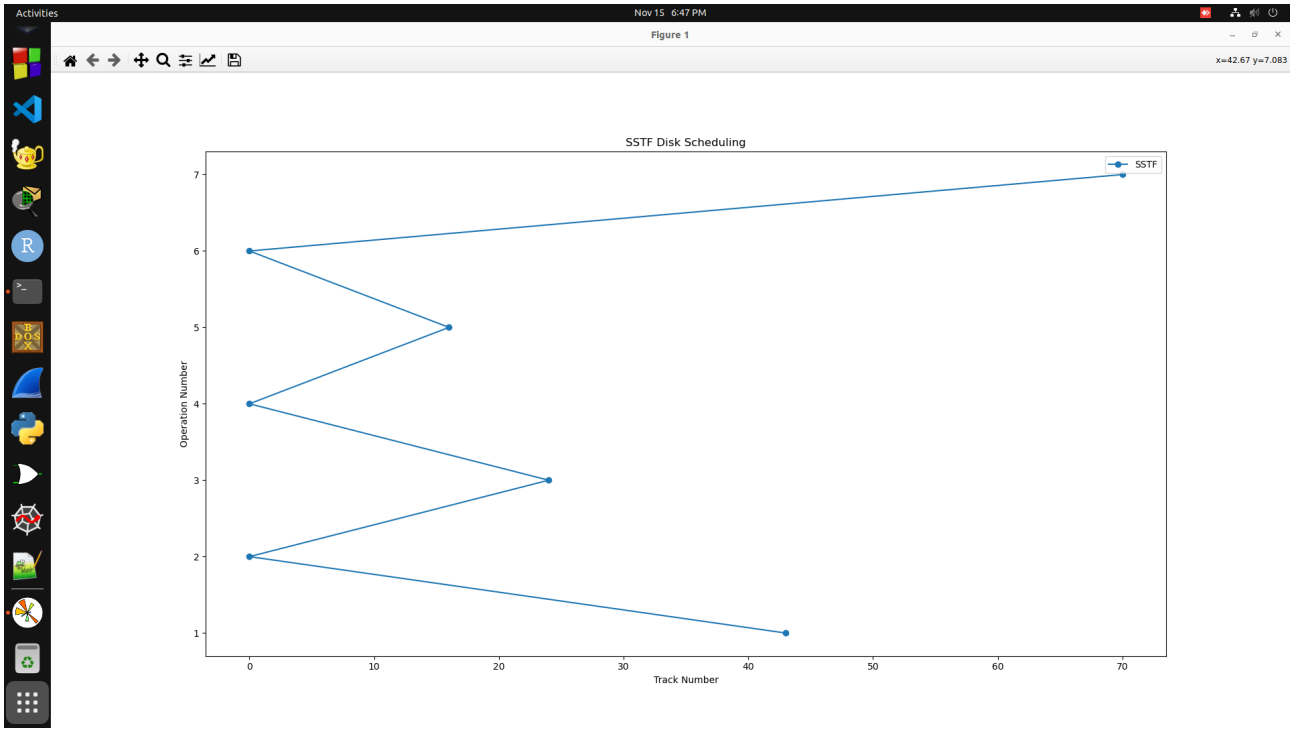
Code:-

```
import matplotlib.pyplot as plt
# Disk Scheduling Algorithms
def fcfs(requests, head):
    seek_sequence = []
    seek_time = 0
    current_position = head
    for track in requests:
        seek_sequence.append(track)
        seek_time += abs(track - current_position)
        current_position = track
        return seek_sequence, seek_time
def scan(requests, head, disk_size):
    requests.sort()
    seek_sequence = []
    seek_time = 0
    current_position = head
    left = [track for track in requests if track < head]
    right = [track for track in requests if track >= head]
    for track in reversed(left):
        seek_sequence.append(track)
        seek_time += abs(track - current_position)
        current_position = track
        seek_sequence.append(0)
        seek_time += abs(current_position - 0)
        current_position = 0
    for track in right:
        seek_sequence.append(track)
        seek_time += abs(track - current_position)
        current_position = track
        return seek_sequence, seek_time
def sstf(requests, head):
    # Initialize variables
    seek_time = 0
    current = head
    order = [] # To store the order of requests serviced
    # Process all requests
    while requests:
        # Find the closest request to the current head position
        closest_request = min(requests, key=lambda x: abs(x - current))
        # Add the distance to the seek time
        seek_time += abs(closest_request - current)
        # Update current head position
```

```
            current = closest_request
            # Append the serviced request to the order
            order.append(closest_request)
            # Remove the serviced request from the list
            requests.remove(closest_request)
    return seek_time, order
def main():
    requests = list(map(int, input("Enter the disk request queue (comma-separated):
").split(',')))
    head = int(input("Enter the initial position of the diskhead: "))
    disk_size = int(input("Enter the disk size (total number of tracks): "))
    while True:
        print("\n--- Disk Scheduling Algorithms Menu ---")
        print("1. FCFS (First-Come, First-Served)")
        print("2. SSTF (Shortest Seek Time First)")
        print("3. SCAN (Elevator Algorithm)")
        choice = input("Select an algorithm to run (1-3): ")
        if choice == '1':
            seek_sequence, seek_time = fcfs(requests, head)
            algorithm_name = "FCFS"
        elif choice == '2':
            seek_sequence, seek_time = scan(requests, head,disk_size)
            algorithm_name = "SSTF"
        elif choice == '3':
            seek_sequence, seek_time = c_scan(requests, head,)
            algorithm_name = "SCAN"
        else:
            print("Invalid choice. Please select a valid option.")
            continue
        print(f"\n{algorithm_name} Seek Sequence: {seek_sequence}")
        print(f"{algorithm_name} Total Seek Time:{seek_time}")
        # Plot the seek sequence with x and y axes swapped
        plt.plot(seek_sequence, list(range(1,len(seek_sequence) + 1)),
marker='o',
        label=algorithm_name)
        plt.title(f"{algorithm_name} Disk Scheduling")
        plt.xlabel("Track Number")
        plt.ylabel("Operation Number")
        plt.legend()
        plt.show()
if __name__ == "__main__":
    main()
```

Output:-


Activities
Nov 15 6:46 PM
matlab@sjt318scope031: ~
(base) matlab@sjt318scope031:~$ python3 diskscheduling.py
Enter the disk request queue (comma-separated): 82,170,43,140,24,16,70
Enter the initial position of the diskhead: 50
Enter the disk size (total number of tracks): 200
--- Disk Scheduling Algorithms Menu ---
1. FCFS (First-Come, First-Served)
2. SSTF (Shortest Seek Time First)
3. SCAN (Elevator Algorithm)
Select an algorithm to run (1-3): 2
SSTF Seek Sequence: [43, 0, 24, 0, 16, 0, 70]
SSTF Total Seek Time:200
libGL error: MESA-LOADER: failed to open iris: /usr/lib/dri/iris_dri.so: cannot open shared object file: No such file or directory (search paths /usr/lib/x86_64-linux-gnu/dri:\$${ORIGIN}/dri:/usr/lib/dri, suffix _dri)
libGL error: failed to load driver: iris
libGL error: MESA-LOADER: failed to open iris: /usr/lib/dri/iris_dri.so: cannot open shared object file: No such file or directory (search paths /usr/lib/x86_64-linux-gnu/dri:\$${ORIGIN}/dri:/usr/lib/dri, suffix _dri)
libGL error: failed to load driver: iris
libGL error: MESA-LOADER: failed to open swrast: /usr/lib/dri/swrast_dri.so: cannot open shared object file: No such file or directory (search paths /usr/lib/x86_64-linux-gnu/dri:\$${ORIGIN}/dri:/usr/lib/dri, suffix _dri)
libGL error: failed to load driver: swrast


Activities
Nov 15 6:47 PM
Figure 1
x=42.67 y=7.083
SSTF Disk Scheduling
SSTF
Operation Number
Track Number
1
2
3
4
5
6
7
0
10
20
30
40
50
60
70