CSE 406 – Lab Report 5: Disk Scheduling using SSTF (Shortest Seek Time First)

Submitted By:

Sharif Md. Yousuf
ID: 22101128
Section: C-2
4th Year, 1st Semester
Spring 2025

Submitted To:

Atia Rahman Orthi
Lecturer
Department of Computer Science & Engineering
University of Asia Pacific

Date of Submission:
16 August, 2025 (Saturday)

1 Problem Statement

Implement a disk head scheduling simulator for the Shortest Seek Time First (SSTF) algorithm. Given an initial head position and a list of pending cylinder requests, at each step the algorithm must select the unfinished request with the minimum absolute seek distance from the current head location. The program outputs the order in which requests are served, the individual head movements, the total head movement, and the average seek per serviced request.

Input

- First line: two integers n and $head$ where $n > 0$ is the number of requests and $head$ is the initial cylinder position.
- Next either:
  - A single line containing n integers (space separated), or
  - n lines each containing one integer.

  Each integer is a non-negative cylinder number.

Example inputs (both equivalent forms):

```
8 50
176 79 34 60 92 11 41 114
```

or

```
8 50
176
79
34
60
92
11
41
114
```

Output

- Line: Initial head position.
- Line: The order in which requests are serviced (SSTF sequence) separated by arrows.
- A table showing per-move index, from, to, and movement distance.
- Two summary lines: total head movement and average seek per move (formatted to two decimals).

Illustrative output fragment:

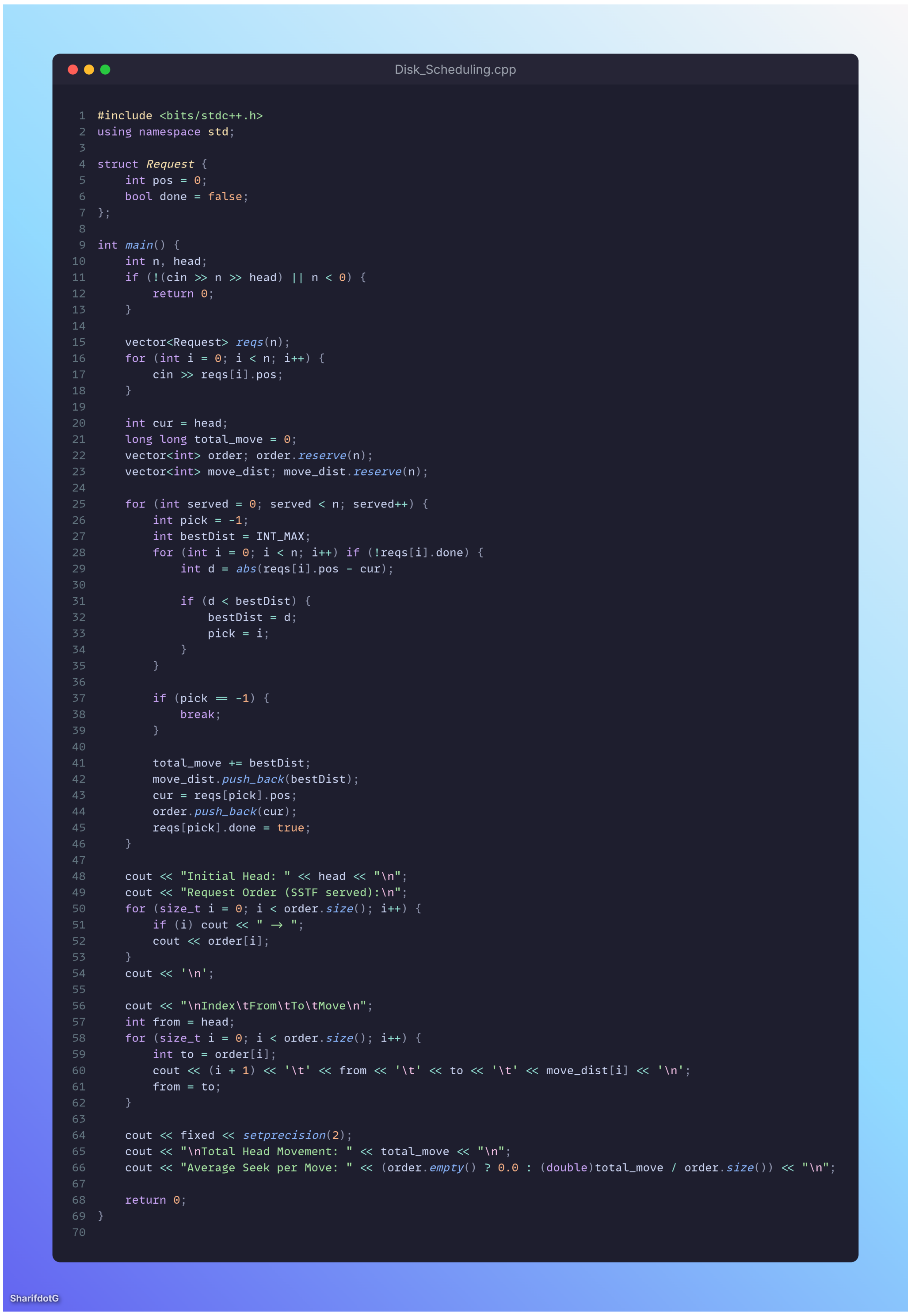
```
Initial Head: 50
Request Order (SSTF served):
41 -> 34 -> 60 -> 79 -> 92 -> 114 -> 176 -> 11

Index From To Move
1 50 41 9
...

Total Head Movement: 183
Average Seek per Move: 22.88
```

2 Objective

The goals of this lab were to:

- Understand mechanical disk access latency components and why seek minimization matters
- Implement the SSTF heuristic to reduce cumulative head movement
- Track per-move distances and compute aggregate performance metrics
- Compare SSTF qualitatively to FCFS and LOOK/SCAN families (discussion)
- Identify fairness and starvation trade-offs inherent to SSTF

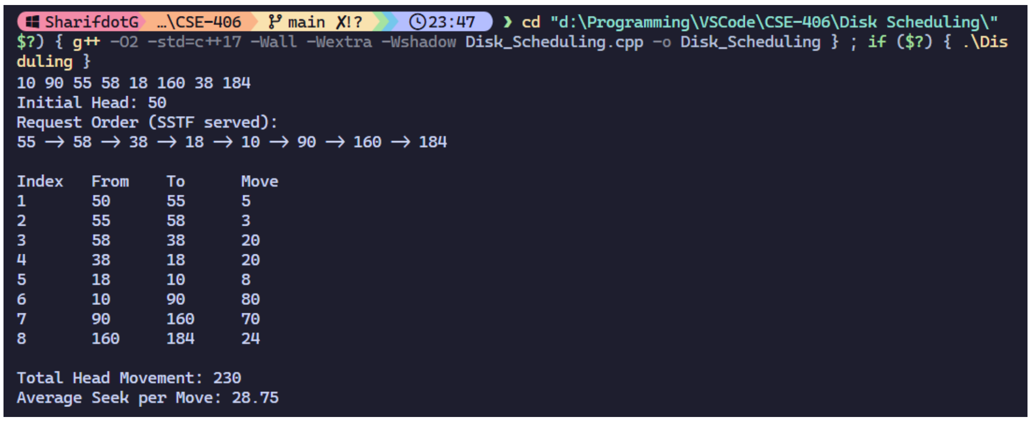
3 Source Code Screenshot



```cpp
#include <bits/stdc++.h>
using namespace std;

struct Request {
    int pos = 0;
    bool done = false;
};

int main() {
    int n, head;
    if (!(cin >> n >> head) || n < 0) {
        return 0;
    }

    vector<Request> reqs(n);
    for (int i = 0; i < n; i++) {
        cin >> reqs[i].pos;
    }

    int cur = head;
    long long total_move = 0;
    vector<int> order; order.reserve(n);
    vector<int> move_dist; move_dist.reserve(n);

    for (int served = 0; served < n; served++) {
        int pick = -1;
        int bestDist = INT_MAX;
        for (int i = 0; i < n; i++) if (!reqs[i].done) {
            int d = abs(reqs[i].pos - cur);

            if (d < bestDist) {
                bestDist = d;
                pick = i;
            }
        }

        if (pick == -1) {
            break;
        }

        total_move += bestDist;
        move_dist.push_back(bestDist);
        cur = reqs[pick].pos;
        order.push_back(cur);
        reqs[pick].done = true;
    }

    cout << "Initial Head: " << head << "\n";
    cout << "Request Order (SSTF served):\n";
    for (size_t i = 0; i < order.size(); i++) {
        if (i) cout << " -> ";
        cout << order[i];
    }
    cout << '\n';

    cout << "\nIndex\tFrom\tTo\tMove\n";
    int from = head;
    for (size_t i = 0; i < order.size(); i++) {
        int to = order[i];
        cout << (i + 1) << '\t' << from << '\t' << to << '\t' << move_dist[i] << '\n';
        from = to;
    }

    cout << fixed << setprecision(2);
    cout << "\nTotal Head Movement: " << total_move << "\n";
    cout << "Average Seek per Move: " << (order.empty() ? 0.0 : (double)total_move / order.size()) << "\n";

    return 0;
}

```

Figure 1: SSTF Disk Scheduling Source Code

4 Output Screenshot



```
 SharifdotG  …\CSE-406  main X!?  23:47  ❯ cd "d:\Programming\VSCode\CSE-406\Disk Scheduling\"
$?) { g++ -O2 -std=c++17 -Wall -Wextra -Wshadow Disk_Scheduling.cpp -o Disk_Scheduling } ; if ($?) { .\Dis
duling }
10 90 55 58 18 160 38 184
Initial Head: 50
Request Order (SSTF served):
55 → 58 → 38 → 18 → 10 → 90 → 160 → 184

Index   From    To      Move
1       50      55      5
2       55      58      3
3       58      38      20
4       38      18      20
5       18      10      8
6       10      90      80
7       90      160     70
8       160     184     24

Total Head Movement: 230
Average Seek per Move: 28.75
```

Figure 2: SSTF Program Output

5 Discussion

SSTF significantly reduces average seek distance versus naive FCFS when requests are spatially clustered. However:

- **Starvation / Fairness:** Far-out requests may wait if closer requests keep arriving (in dynamic systems).
- **Directional Thrashing:** Head may oscillate locally between nearby cylinders rather than progressing outward, increasing latency for distant requests.
- **Non-Optimality:** Greedy choices can create suboptimal long tail movements.
- **Comparison to SCAN/LOOK:** Elevator algorithms provide more uniform wait times by enforcing directional sweeps, trading a small increase in total movement for fairness.

Enhancements could include LOOK/SCAN, C-SCAN variants, or incorporating request arrival times if extended to a dynamic queue. For solid-state drives (SSDs), seek-based heuristics largely lose relevance, shifting optimization toward wear-leveling and parallelism.

6 Conclusion

The SSTF disk scheduling implementation demonstrates how a simple greedy heuristic can reduce head travel distance relative to FCFS while introducing fairness concerns. Understanding SSTF provides foundational insight before adopting more balanced algorithms like SCAN or C-SCAN in real disk schedulers.