Name: Devraj Pravin Lokhande Roll No.- 64

PRN:12310051 Operating Systems SY-AIML Div: A

**LAB - 7**

1. Implement following Disk scheduling algorithms
2. 1. FCFS
3. 2. SSTF
4. 3. SCAN
5. 4. C-SCAN

#include <iostream>

#include <vector>

#include <algorithm>

#include <cmath>

#include <iomanip>

#include <climits>

using namespace std;

// Utility to calculate total seek time and print path

int calculateSeekTime(const vector<int>& path, int head) {

    int seek = 0;

    cout << "Seek Sequence: " << head;

    for (int i = 0; i < path.size(); i++) {

        cout << " -> " << path[i];

        seek += abs(path[i] - head);

        head = path[i];

    }

    cout << "\nTotal Seek Time: " << seek << "\n";

    return seek;

}

// 1. FCFS

void FCFS(vector<int> requests, int head) {

    cout << "\n=== FCFS Disk Scheduling ===\n";

    calculateSeekTime(requests, head);

}

// 2. SSTF

void SSTF(vector<int> requests, int head) {

    cout << "\n=== SSTF Disk Scheduling ===\n";

    vector<int> path;

    vector<bool> visited(requests.size(), false);

    for (int i = 0; i < requests.size(); ++i) {

        int minDist = INT\_MAX, index = -1;

        for (int j = 0; j < requests.size(); ++j) {

            if (!visited[j]) {

                int dist = abs(requests[j] - head);

                if (dist < minDist) {

                    minDist = dist;

                    index = j;

                }

            }

        }

        visited[index] = true;

        path.push\_back(requests[index]);

        head = requests[index];

    }

    calculateSeekTime(path, head);

}

// 3. SCAN (Elevator Algorithm)

void SCAN(vector<int> requests, int head, int diskSize, string direction = "left") {

    cout << "\n=== SCAN Disk Scheduling ===\n";

    vector<int> left, right;

    vector<int> path;

    for (int r : requests) {

        if (r < head) left.push\_back(r);

        else right.push\_back(r);

    }

    sort(left.begin(), left.end());

    sort(right.begin(), right.end());

    if (direction == "left") {

        for (int i = left.size() - 1; i >= 0; i--) path.push\_back(left[i]);

        path.push\_back(0);

        for (int i = 0; i < right.size(); i++) path.push\_back(right[i]);

    } else {

        for (int i = 0; i < right.size(); i++) path.push\_back(right[i]);

        path.push\_back(diskSize - 1);

        for (int i = left.size() - 1; i >= 0; i--) path.push\_back(left[i]);

    }

    calculateSeekTime(path, head);

}

// 4. C-SCAN

void CSCAN(vector<int> requests, int head, int diskSize) {

    cout << "\n=== C-SCAN Disk Scheduling ===\n";

    vector<int> left, right;

    vector<int> path;

    for (int r : requests) {

        if (r < head) left.push\_back(r);

        else right.push\_back(r);

    }

    sort(left.begin(), left.end());

    sort(right.begin(), right.end());

    for (int i = 0; i < right.size(); i++) path.push\_back(right[i]);

    path.push\_back(diskSize - 1);

    path.push\_back(0);

    for (int i = 0; i < left.size(); i++) path.push\_back(left[i]);

    calculateSeekTime(path, head);

}

// Main driver

int main() {

    int n, head, diskSize;

    cout << "Enter number of requests: ";

    cin >> n;

    vector<int> requests(n);

    cout << "Enter disk requests: ";

    for (int i = 0; i < n; i++) cin >> requests[i];

    cout << "Enter initial head position: ";

    cin >> head;

    cout << "Enter total disk size (number of cylinders): ";

    cin >> diskSize;

    FCFS(requests, head);

    SSTF(requests, head);

    SCAN(requests, head, diskSize, "left");

    CSCAN(requests, head, diskSize);

    return 0;

}

**OUTPUT:**

