K. J. Somaiya College of Engineering, Mumbai-77
(A Constituent College of Somaiya Vidyavihar University)
Department of Computer Engineering

Batch: D-2	**Roll No.:** 16010122151
Experiment No. 09	
Grade: AA / AB / BB / BC / CC / CD /DD	
Signature of the Staff In-charge with date	

TITLE: Implementation of Disk Scheduling Algorithm like FCFS, SSTF, SCAN, CSCAN, LOOK

AIM: Implementation of Disk Scheduling Algorithm like FCFS, SSTF, SCAN, CSCAN, LOOK

Expected Outcome of Experiment:

CO 4. To understand various Memory, I/O and File management techniques.

Books/ Journals/ Websites referred:

1. **Silberschatz A., Galvin P., Gagne G. "Operating Systems Principles", Willey Eight edition.**
2. **Achyut S. Godbole , Atul Kahate "Operating Systems", McGraw Hill Third Edition.**
3. **Sumitabha Das " UNIX Concepts & Applications", McGraw Hill Second Edition.**

Pre Lab/ Prior Concepts:

Knowledge of disk scheduling algorithm.

Calculation of seek time and transfer time etc

Description of the application to be implemented:

1. First Come-First Serve (FCFS)

In FCFS, the disk services requests in the order they arrive. It's a simple algorithm but can lead to long seek times. Example: If the disk head starts at track 50 and needs to service requests at tracks [95, 180, 34, 119, 11, 123, 62, 64], it will process them in the order they came in: Order: 50 -> 95 -> 180 -> 34 -> 119 -> 11 -> 123 -> 62 -> 64 This can result in a lot of unnecessary movement, increasing seek time.

2. Shortest Seek Time First (SSTF)

SSTF selects the request closest to the current head position. This minimizes seek time but can cause "starvation" for some requests if they're constantly distant from the current head position. Example: Starting at track 50 with requests [95, 180, 34, 119, 11, 123, 62, 64], SSTF would pick the closest request each time: Sequence: 50 -> 62 -> 64 -> 34 -> 11 -> 95 -> 119 -> 123 -> 180

3. Elevator (SCAN)

In SCAN, the disk head moves toward the end of the disk, servicing requests along the way, then reverses direction once it reaches the end. Example: Starting at 50 with requests [95, 180, 34, 119, 11, 123, 62, 64] and assuming the head moves towards the higher end: Sequence: 50 -> 62 -> 64 -> 95 -> 119 -> 123 -> 180, then it reverses to service requests in the opposite direction (if any).

4. Circular SCAN (C-SCAN)

C-SCAN operates similarly to SCAN but only moves in one direction (e.g., up). Once it reaches the end, it jumps back to the beginning without servicing any requests on the return trip. Example: Starting at 50 with requests [95, 180, 34, 119, 11, 123, 62, 64], moving upwards: Sequence: 50 -> 62 -> 64 -> 95 -> 119 -> 123 -> 180, then it jumps back to the beginning of the disk.

5. LOOK

LOOK is similar to SCAN, but instead of going all the way to the end of the disk, it only goes as far as the last request in each direction before reversing. Example: Starting at 50 with requests [95, 180, 34, 119, 11, 123, 62, 64]: Sequence: 50 -> 62 -> 64 -> 95 -> 119 -> 123 -> 180, and then reverses back to 11 since no requests are beyond these positions.

Implementation details:

FCFS

```
#include <iostream>
#include <vector>
#include <cmath>

using namespace std;

// Function to implement FCFS Disk Scheduling
void FCFS_DiskScheduling(const vector<int>& requestQueue, int
startTrack) {
    int n = requestQueue.size();
    int totalSeekCount = 0;
    int currentTrack = startTrack;

    cout << "Seek Sequence: " << endl;
    cout << currentTrack << " ";

    // Loop through all the requests and process them in the order
they arrive
    for (int i = 0; i < n; i++) {
        // Calculate the seek count (distance to move the disk arm)
        totalSeekCount += abs(requestQueue[i] - currentTrack);
        currentTrack = requestQueue[i];

        // Output the seek sequence
        cout << currentTrack << " ";
    }

    cout << endl;
    cout << "Total Seek Count: " << totalSeekCount << endl;
    cout << "Average Seek Count: " << (float)totalSeekCount / n <<
endl;
}

int main() {
```

```
    int n, startTrack;

    // Input number of disk requests and starting track position
    cout << "Enter the number of disk requests: ";
    cin >> n;

    vector<int> requestQueue(n);

    cout << "Enter the disk requests (space-separated): ";
    for (int i = 0; i < n; i++) {
        cin >> requestQueue[i];
    }

    cout << "Enter the starting track position: ";
    cin >> startTrack;

    // Call the FCFS Disk Scheduling function
    FCFS_DiskScheduling(requestQueue, startTrack);

    return 0;
}
```

```
hyder@HyderPresswala MINGW64 ~/Downloads/Codewithharry
$ g++ -o hyder hyder.cpp

hyder@HyderPresswala MINGW64 ~/Downloads/Codewithharry
$ ./hyder.exe
Enter the number of disk requests: 5
Enter the disk requests (space-separated): 98 183 41 122 14
Enter the starting track position: 50
Seek Sequence:
50 98 183 41 122 14
Total Seek Count: 464
Average Seek Count: 92.8

hyder@HyderPresswala MINGW64 ~/Downloads/Codewithharry
$
```

CSCAN

```cpp
#include <iostream>
#include <vector>
#include <algorithm>
#include <cmath>

using namespace std;

// Function to implement C-SCAN Disk Scheduling
void CSCAN_DiskScheduling(const vector<int>& requestQueue, int
startTrack, int diskSize) {
    int n = requestQueue.size();
    int totalSeekCount = 0;
    int currentTrack = startTrack;

    // Sorting the request queue to easily process in order
    vector<int> sortedRequests = requestQueue;
    sort(sortedRequests.begin(), sortedRequests.end());

    // Create two partitions: requests less than the start and
requests greater than the start
    vector<int> left, right;
    for (int i = 0; i < n; i++) {
        if (sortedRequests[i] < startTrack) {
            left.push_back(sortedRequests[i]);
        } else {
            right.push_back(sortedRequests[i]);
        }
    }

    // Output the seek sequence
    cout << "Seek Sequence: ";

    // First process the right side (from start to the rightmost
track)
    for (int i = 0; i < right.size(); i++) {
        cout << right[i] << " ";
    }

    // Then jump to the farthest end (disk size) and come back to
the lowest track
```

```
    totalSeekCount += (diskSize - 1 - currentTrack);
    currentTrack = diskSize - 1;
    cout << diskSize - 1 << " ";  // Jump to the farthest end

    // After jumping, process the left side (from the leftmost
track)
    for (int i = left.size() - 1; i >= 0; i--) {
        cout << left[i] << " ";
    }

    // Calculate total seek count
    totalSeekCount += (diskSize - 1 - startTrack); // Jump from
start to the end
    for (int i = 1; i < right.size(); i++) {
        totalSeekCount += abs(right[i] - right[i - 1]); // Moving to
the rightmost tracks
    }
    for (int i = 1; i < left.size(); i++) {
        totalSeekCount += abs(left[i] - left[i - 1]); // Moving back
from the leftmost tracks
    }

    cout << endl;
    cout << "Total Seek Count: " << totalSeekCount << endl;
    cout << "Average Seek Count: " << (float)totalSeekCount / n <<
endl;
}

int main() {
    int n, startTrack, diskSize;

    // Input number of disk requests, starting track position, and
disk size
    cout << "Enter the number of disk requests: ";
    cin >> n;

    vector<int> requestQueue(n);

    cout << "Enter the disk requests (space-separated): ";
    for (int i = 0; i < n; i++) {
        cin >> requestQueue[i];
```

```
    }

    cout << "Enter the starting track position: ";
    cin >> startTrack;

    cout << "Enter the total disk size: ";
    cin >> diskSize;

    // Call the C-SCAN Disk Scheduling function
    CSCAN_DiskScheduling(requestQueue, startTrack, diskSize);

    return 0;
}
```

```
hyder@HyderPresswala MINGW64 ~/Downloads/Codewithharry
$ g++ -o hyder hyder.cpp

hyder@HyderPresswala MINGW64 ~/Downloads/Codewithharry
$ ./hyder.exe
Enter the number of disk requests: 8
Enter the disk requests (space-separated): 98 183 41 122 14 124 65 67
Enter the starting track position: 53
Enter the total disk size: 200
Seek Sequence: 65 67 98 122 124 183 199 41 14
Total Seek Count: 437
Average Seek Count: 54.625

hyder@HyderPresswala MINGW64 ~/Downloads/Codewithharry
$
```

Conclusion:

We learned disk scheduling algorithms and implemented them in code

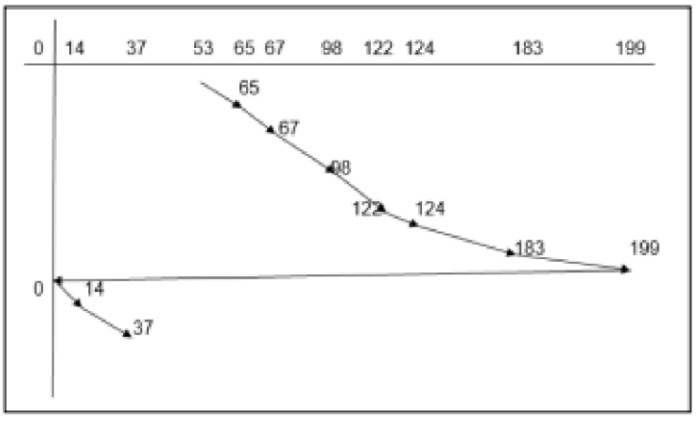
Post Lab Descriptive Questions

1. A disk drive has 200 cylinders numbered from 0 to 199. The disk head is initially at cylinder 53. The queue of pending requests in FIFO order is :
98, 183, 37, 122, 14, 124, 65, 67.

Starting from the current head position, what is the total distance travelled (in cylinders) by disk arm to satisfy the requests using CSCAN and Look. Illustrate with figures in each case.

Ans) CSCAN:
Circular scanning works just like the elevator to some extent. It begin sits scan toward the nearest end and works its way all the way to the end of the system. Once it hits the bottom or top it jumps to the other end and moves in the same direction. Keep in mind that the huge jump doesn't count as a head movement. The heaviest density of request is at the other end of the disk. The request has waited longer. This scheduling algorithm provides more uniform wait time C-SCAN moves the head from one end of the disk to the other, servicing request along the way. When the head reaches the other end, it immediately returns to the beginning of the disk without servicing any request on the return trip.

Consider an e.g. ,Queue=98, 183, 37, 122, 14, 124, 65, 67.
Head starts at 53.



The Total Distance
(53-65)+(67-65)+(98-67)+(122-98)+(124-122)+(183-124)+(199-183)+(140)+(37-14)= 159 Cylinders

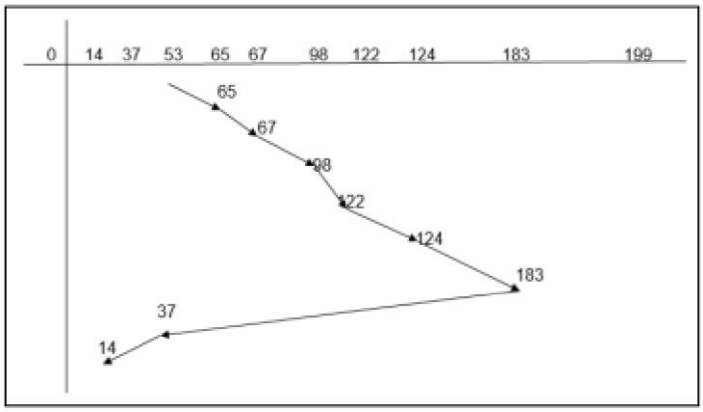
In another way:

C-Scan: 53 , 65 , 67 , 98 , 122 , 124 , 183 , 199 , 0 , 14 , 37
Look:

Let us see the same example for look scheduling queue:

98, 183, 37, 122, 14, 124, 65, 67.
Head starts: 53

Look and C-Look are the versions of SCAN and C-SCAN because they look for a request before continuing to move in a given direction.



The total distance
(65-53)+(67-65)+(98-67)+(122-98)+(124-122)+(183-124)+(183-37)+(37-14)= 299
Cylinders
In another way

LOOK: 53 , 65 , 67 , 98 , 122 , 124 , 183 , 37 , 14

1. In a hard disk, what rotates about a central spindle _____
a. Disk
b. Platter
c. Sector
d. None of the above

Ans: Platter

2. The time required to move the disk arm to the required track is known as _____

a. Latency time
b. Access time
c. Seek time
d. None of the above

Ans: Seek Time

Date: 06-11-2024 **Signature of faculty in-charge**