Experiment No.10
File Management & I/O Management A. Implement disk scheduling algorithms FCFS, SSTF.
Name Of Student:-Bhagyashri Kaleni Sutar
Roll No:-75
Date of Submission:
Marks:
Sign:

Experiment No. 10

Aim: To study and implement disk scheduling algorithms FCFS.

Objective:

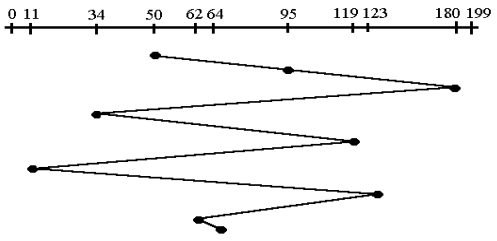
The main purpose of disk scheduling algorithm is to select a disk request from the queue of IO requests and decide the schedule when this request will be processed.

Theory:

TYPES OF DISK SCHEDULING ALGORITHMS

Although there are other algorithms that reduce the seek time of all requests, I will only concentrate on the following disk scheduling algorithms:
1. First Come-First Serve (FCFS)
2. Shortest Seek Time First (SSTF)
3. Elevator (SCAN)
4. Circular SCAN (C-SCAN)
5. C-LOOK

Given the following queue -- 95, 180, 34, 119, 11, 123, 62, 64 with the Read-write head initially at the track 50 and the tail track being at 199 let us now discuss the different algorithms.



FCFS

First Come -First Serve (FCFS)

All incoming requests are placed at the end of the queue. Whatever number that is next in the queue will be the next number served. Using this algorithm doesn't provide the best results. To determine the number of head movements you would simply find the number of tracks it took to move from one request to the next. For this case it

went from 50 to 95 to 180 and so on. From 50 to 95 it moved 45 tracks. If you tally up the total number of tracks you will find how many tracks it had to go through before finishing the entire request. In this example, it had a total head movement of 640 tracks. The disadvantage of this algorithm is noted by the oscillation from track 50 to track 180 and then back to track 11 to 123 then to 64. As you will soon see, this is the worse algorithm that one can use.

Program:

```
import java.util.Arrays;


public class DiskScheduling {


    public static void fcfs(int[] requests, int head) {
        int totalSeekTime = 0;
        int currentHead = head;


        System.out.println("FCFS Disk Scheduling:");
        System.out.print("Sequence: " + head);


        for (int request : requests) {
            totalSeekTime += Math.abs(currentHead - request);
            currentHead = request;
            System.out.print(" -> " + request);
        }


        System.out.println("\nTotal Seek Time: " + totalSeekTime);
    }


    public static void sstf(int[] requests, int head) {
```

```
int totalSeekTime = 0;
int currentHead = head;
int[] remainingRequests = requests.clone();
boolean[] visited = new boolean[remainingRequests.length];

System.out.println("SSTF Disk Scheduling:");
System.out.print("Sequence: " + head);

for (int i = 0; i < remainingRequests.length; i++) {
    int shortestSeek = Integer.MAX_VALUE;
    int shortestIndex = -1;

    for (int j = 0; j < remainingRequests.length; j++) {
        if (!visited[j]) {
            int seek = Math.abs(currentHead - remainingRequests[j]);
            if (seek < shortestSeek) {
                shortestSeek = seek;
                shortestIndex = j;
            }
        }
    }

    totalSeekTime += shortestSeek;
    currentHead = remainingRequests[shortestIndex];
    visited[shortestIndex] = true;
    System.out.print(" -> " + currentHead);
}
```

```
        System.out.println("\nTotal Seek Time: " + totalSeekTime);
    }

    public static void main(String[] args) {
        int[] requests = {98, 183, 37, 122, 14, 124, 65, 67};
        int head = 53;

        fcfs(requests, head);
        sstf(requests, head);

    }
}
```

//output

FCFS Disk Scheduling:

Sequence: 53 -> 98 -> 183 -> 37 -> 122 -> 14 -> 124 -> 65 -> 67

Total Seek Time: 640

SSTF Disk Scheduling:

Sequence: 53 -> 65 -> 67 -> 37 -> 14 -> 98 -> 122 -> 124 -> 183

Total Seek Time: 236

Conclusion:

Why is Disk Scheduling important?

Ans:- Disk scheduling is crucial in operating systems because it significantly impacts a computer's overall performance. Here's a breakdown of its importance:

- **Optimizing Disk I/O:**

- Disk drives are mechanical devices, and accessing data on them is relatively slow compared to accessing data in RAM.
-
- Disk scheduling algorithms aim to minimize the time it takes to access data on the disk, thereby improving the efficiency of disk input/output (I/O) operations.

- **Minimizing Seek Time:**
  - Seek time is the time it takes for the disk arm to move to the correct track.
  -
  - Disk scheduling algorithms prioritize requests to reduce the distance the disk arm has to travel, thus minimizing seek time.
  -
- **Reducing Rotational Latency:**
  - Rotational latency is the time it takes for the desired sector to rotate under the read/write head.
  -
  - Efficient scheduling can also contribute to reducing this latency.
  -
- **Maximizing Throughput:**
  - By optimizing disk access, disk scheduling algorithms increase the number of I/O requests that can be processed in a given time, thereby maximizing throughput.
  -
- **Ensuring Fairness:**
  - In multi-user or multi-tasking environments, disk scheduling algorithms ensure that all processes have fair access to the disk, preventing any one process from monopolizing the disk resources.
- **Improving System Responsiveness:**
  - Efficient disk scheduling leads to faster data retrieval, which improves the overall responsiveness of the computer system.

In essence, disk scheduling is about managing and optimizing the order in which disk access requests are handled, leading to a more efficient and responsive computer system.