Batch: A2	**Roll No.: 16010121045**
Experiment No. 08	
Grade: AA / AB / BB / BC / CC / CD /DD	
Signature of the Staff In-charge with date	

TITLE: Disk Scheduling Algorithms

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. **William Stallings, "Operating System Internal & Design Principles", Pearson.**
4. **Andrew S. Tanenbaum, "Modern Operating System", Prentice Hall.**

Pre Lab/ Prior Concepts:

- Knowledge of disk scheduling algorithm.
- Calculation of seek time and transfer time etc.

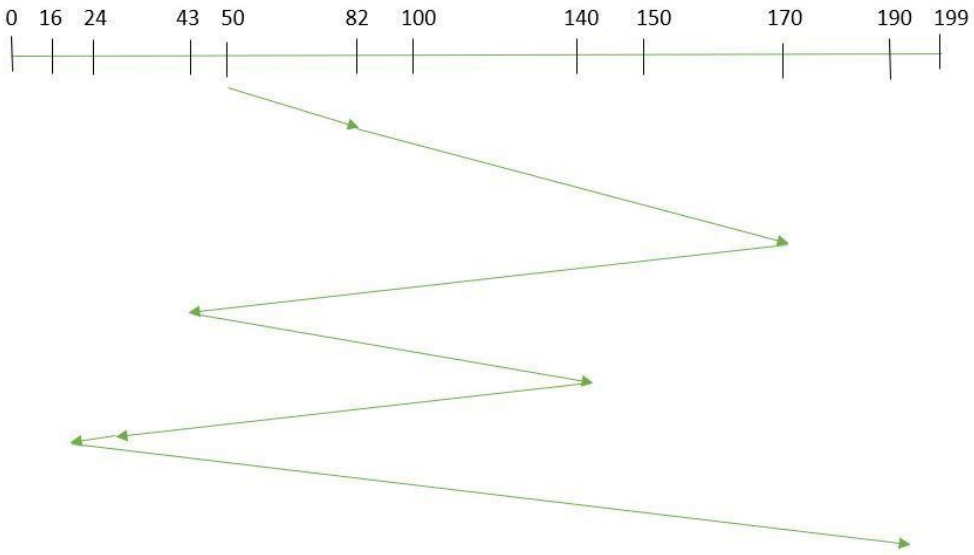
Description of the application to be implemented:

First Come-First Serve (FCFS):

FCFS is the simplest of all the Disk Scheduling Algorithms. In FCFS, the requests are addressed in the order they arrive in the disk queue. Let us understand this with the help of an example.

Example:

Suppose the order of request is- (82,170,43,140,24,16,190)
And current position of Read/Write head is: 50



So, total seek time:
=(82-50)+(170-82)+(170-43)+(140-43)+(140-24)+(24-16)+(190-16)
=642

Advantages:

- Every request gets a fair chance
- No indefinite postponement Disadvantages:
- Does not try to optimize seek time
- May not provide the best possible service

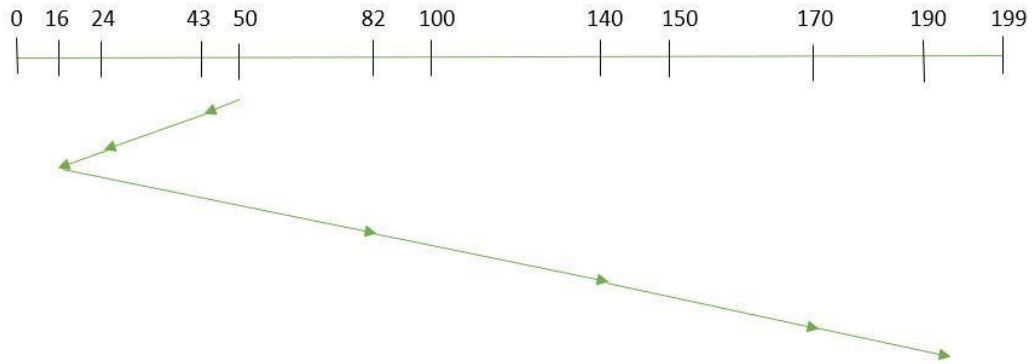
Shortest Seek Time First (SSTF):

In SSTF (Shortest Seek Time First), requests having shortest seek time are executed first. So, the seek time of every request is calculated in advance in the queue and then they are scheduled according to their calculated seek time. As a result, the request

near the disk arm will get executed first. SSTF is certainly an improvement over FCFS as it decreases the average response time and increases the throughput of system. Let us understand this with the help of an example.

Example:

Suppose the order of request is- (82,170,43,140,24,16,190)
And current position of Read/Write head is : 50



So, total seek time:

=(50-43)+(43-24)+(24-16)+(82-16)+(140-82)+(170-40)+(190-170)
=208

Advantages:

- Average Response Time decreases
- Throughput increases

Disadvantages:

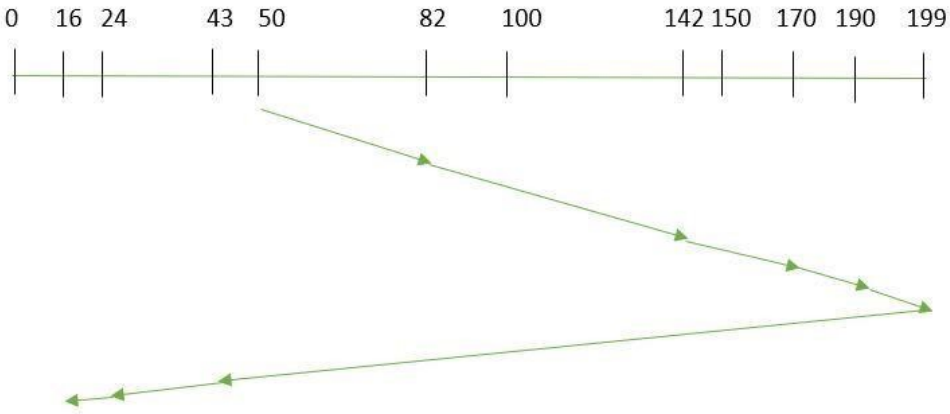
- Overhead to calculate seek time in advance
- Can cause Starvation for a request if it has higher seek time as compared to incoming requests
- High variance of response time as SSTF favours only some requests

Elevator (SCAN):

In SCAN algorithm the disk arm moves into a particular direction and services the requests coming in its path and after reaching the end of disk, it reverses its direction and again services the request arriving in its path. So, this algorithm works as an elevator and hence also known as elevator algorithm. As a result, the requests at the midrange are serviced more and those arriving behind the disk arm will have to wait.

Example:

Suppose the requests to be addressed are-82,170,43,140,24,16,190. And the Read/Write arm is at 50, and it is also given that the disk arm should move "towards the larger value".



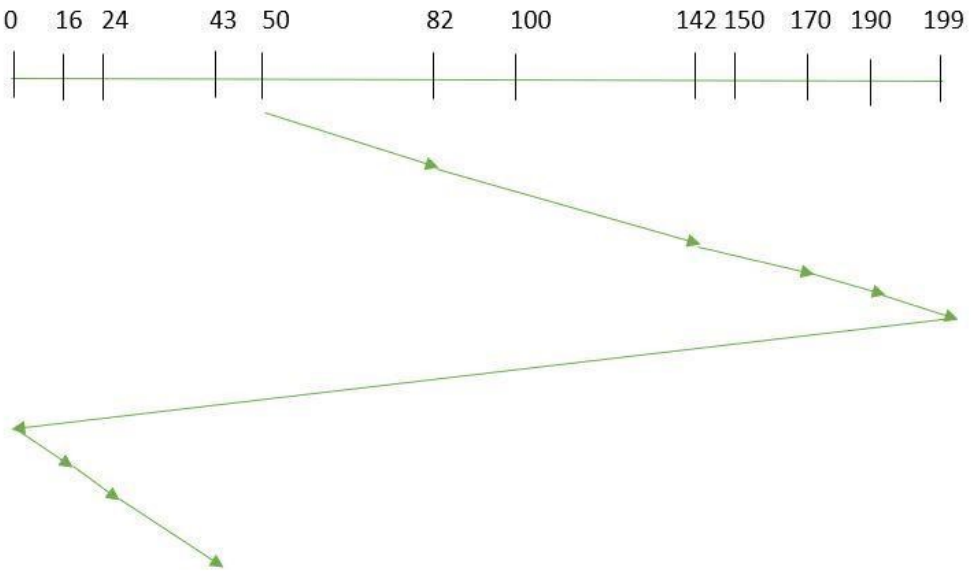
Therefore, the seek time is calculated as:

=(199-50)+(199-16)
=332 Advantages:

- High throughput
- Low variance of response time
- Average response time Disadvantages:
- Long waiting time for requests for locations just visited by disk arm

CSCAN:

In SCAN algorithm, the disk arm again scans the path that has been scanned, after reversing its direction. So, it may be possible that too many requests are waiting at the other end or there may be zero or few requests pending at the scanned area. These situations are avoided in *CSCAN* algorithm in which the disk arm instead of reversing its direction goes to the other end of the disk and starts servicing the requests from there. So, the disk arm moves in a circular fashion and this algorithm is also similar to SCAN algorithm and hence it is known as C-SCAN (Circular SCAN). **Example:**
Suppose the requests to be addressed are-82,170,43,140,24,16,190. And the Read/Write arm is at 50, and it is also given that the disk arm should move "towards the larger value".



Seek time is calculated as:

=(199-50)+(199-0)+(43-0)
=391

Advantages:

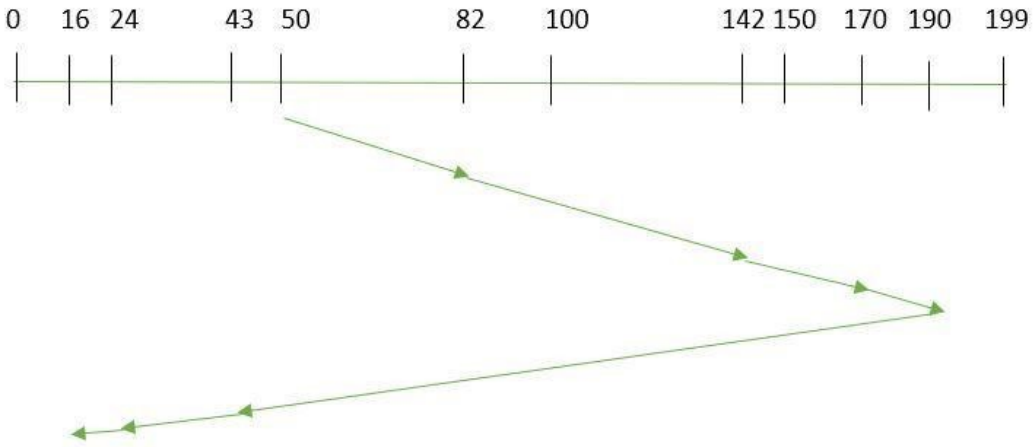
• Provides more uniform wait time compared to SCAN

LOOK:

It is similar to the SCAN disk scheduling algorithm except for the difference that the disk arm in spite of going to the end of the disk goes only to the last request to be serviced in front of the head and then reverses its direction from there only. Thus it prevents the extra delay which occurred due to unnecessary traversal to the end of the disk.

Example:

Suppose the requests to be addressed are-82,170,43,140,24,16,190. And the Read/Write arm is at 50, and it is also given that the disk arm should move "towards the larger value".



So, the seek time is calculated as:

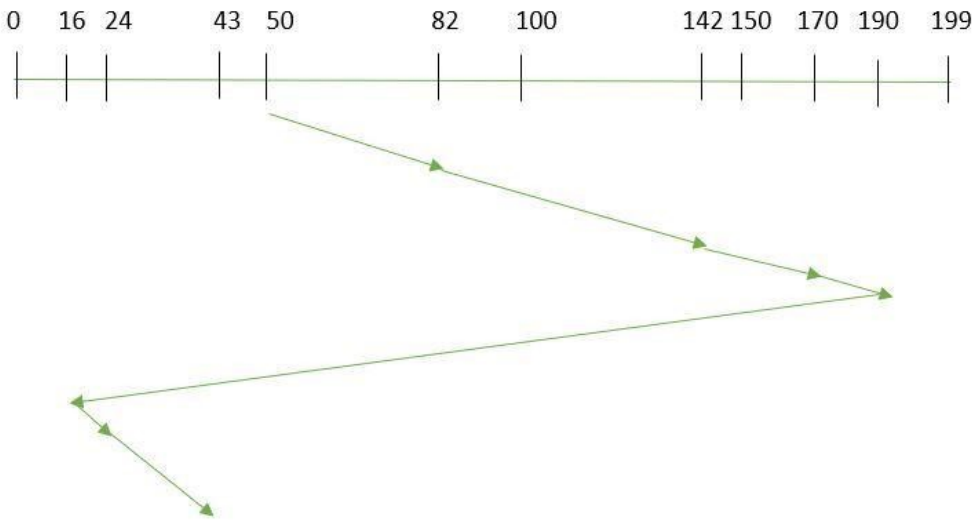
=(190-50)+(190-16)
=314

CLOOK:

As LOOK is similar to SCAN algorithm, in similar way, CLOOK is similar to CSCAN disk scheduling algorithm. In CLOOK, the disk arm in spite of going to the end goes only to the last request to be serviced in front of the head and then from there goes to the other end's last request. Thus, it also prevents the extra delay which occurred due to unnecessary traversal to the end of the disk.

Example:

Suppose the requests to be addressed are-82,170,43,140,24,16,190. And the Read/Write arm is at 50, and it is also given that the disk arm should move **"towards the larger value"**



So, the seek time is calculated as:

= (190-50) + (190-16) + (43-16)
=341

Implementation details: (printout of code)
FCFS and SSTF

```
import java.util.ArrayList;
import java.util.Collections;

class DiskScheduling {
    public static void main(String[] args) {
        ArrayList<Integer> requestQueue = new ArrayList<>();
        Collections.addAll(requestQueue, 98, 183, 37, 122,
14, 124, 65, 67);

        int headPosition = 53;

        System.out.println("FCFS Schedule:");
        fcfs(requestQueue, headPosition);

        System.out.println("\nSSTF Schedule:");
        sstf(requestQueue, headPosition);
    }
```

```
    public static void fcfs(ArrayList<Integer> requestQueue,
int headPosition) {
        int totalSeekOperations = 0;
        int currentHeadPosition = headPosition;

        for (Integer request : requestQueue) {
            int seekDistance = Math.abs(currentHeadPosition -
request);
            totalSeekOperations += seekDistance;
            currentHeadPosition = request;
            System.out.println("Move head to track " +
request);
        }

        System.out.println("Total seek operations with FCFS:
" + totalSeekOperations);
    }

    public static void sstf(ArrayList<Integer> requestQueue,
int headPosition) {
        int totalSeekOperations = 0;
        int currentHeadPosition = headPosition;

        while (!requestQueue.isEmpty()) {
            int shortestSeekTime = Integer.MAX_VALUE;
            int closestRequest = -1;

            for (Integer request : requestQueue) {
                int seekDistance =
Math.abs(currentHeadPosition - request);
                if (seekDistance < shortestSeekTime) {
                    shortestSeekTime = seekDistance;
                    closestRequest = request;
                }
            }

            totalSeekOperations += shortestSeekTime;
            currentHeadPosition = closestRequest;
```

```
            System.out.println("Move head to track " +
closestRequest);

requestQueue.remove(Integer.valueOf(closestRequest));
        }

        System.out.println("Total seek operations with SSTF:
" + totalSeekOperations);
    }
}
```

```
❯ cd "/Users/pargatsinghdhanjal/Desktop/
FCFS Schedule:
Move head to track 98
Move head to track 183
Move head to track 37
Move head to track 122
Move head to track 14
Move head to track 124
Move head to track 65
Move head to track 67
Total seek operations with FCFS: 640

SSTF Schedule:
Move head to track 65
Move head to track 67
Move head to track 37
Move head to track 14
Move head to track 98
Move head to track 122
Move head to track 124
Move head to track 183
Total seek operations with SSTF: 236
```

Conclusion: Successfully implemented the given experiment on disk scheduling.

Post Lab Descriptive Questions

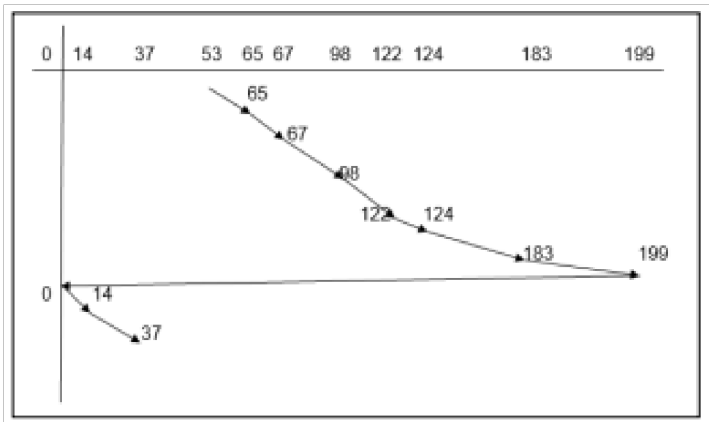
1. A disk drive has 200 cylinders numbered from 0 to 199. The disk head is initially at cylinder 53. The queue of pending requests in FIFO order is :
98, 183, 37, 122, 14, 124, 65, 67.

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 beginsits scan toward the nearest end and works itsway all the way to the end of thesystem. Once it hits the bottom or top it jumps to the other end and moves in thesame direction. Keep in mind that the huge jump doesn't count as a headmovement.The heaviest density of request is at the other end of the disk. The request haswaited 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 ofthe 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.



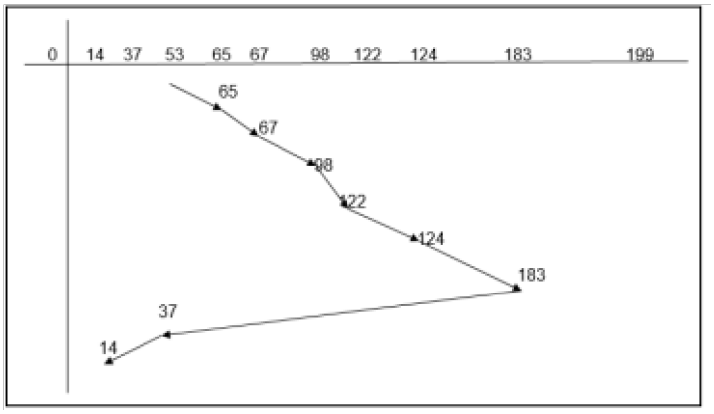
The totaldistance

(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

Post Lab Objective Questions

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

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:

Date: _____ **Signature of faculty in-charge**