ASSIGNMENT No: 8

Title: Disk scheduling algorithms

AIM: To implement C programs for Disk scheduling algorithms

1. SSTF
2. SCAN
3. C-LOOK

Theory:

- **Disk scheduling** is done by operating systems to schedule I/O requests arriving for the disk. Disk scheduling is also known as I/O Scheduling.
- Importance of Disk Scheduling in Operating System
- Multiple I/O requests may arrive by different processes and only one I/O request can be served at a time by the disk controller. Thus other I/O requests need to wait in the waiting queue and need to be scheduled.
- Two or more requests may be far from each other so this can result in greater disk arm movement.
- Hard drives are one of the slowest parts of the computer system and thus need to be accessed in an efficient manner.
- Both the access time and the bandwidth can be improved by managing the order in which disk I/O requests are serviced which is called as disk scheduling.

Important terms related to disk scheduling.

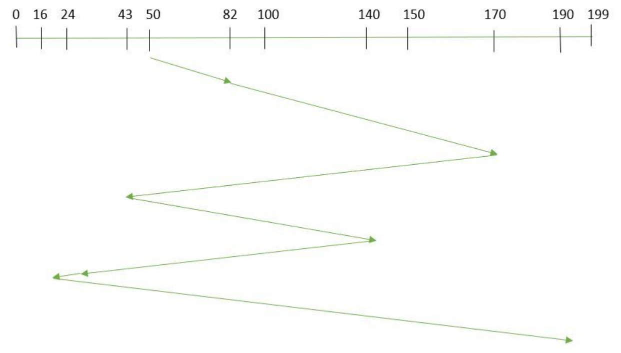
- **Seek Time:** Seek time is the time taken to locate the disk arm to a specified track where the data is to be read or written. So the disk scheduling algorithm that gives a minimum average seek time is better.
- **Rotational Latency:** Rotational Latency is the time taken by the desired sector of the disk to rotate into a position so that it can access the read/write heads. So the disk scheduling algorithm that gives minimum rotational latency is better.
- **Transfer Time:** Transfer time is the time to transfer the data. It depends on the rotating speed of the disk and the number of bytes to be transferred.
- **Disk Access Time:**

  Disk Access Time = Seek Time + Rotational Latency + Transfer Time

  Total Seek Time = Total head Movement * Seek Time
- **Disk Response Time:** Response Time is the average time spent by a request waiting to perform its I/O operation. The average Response time is the response time of all requests. Variance Response Time is the measure of how individual requests are serviced with respect to average response time. So the disk scheduling algorithm that gives minimum variance response time is better.

1. **First Come -First Serve (FCFS)**
   - FCFS is the simplest of all 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 overhead movement (total distance covered by the disk arm) =

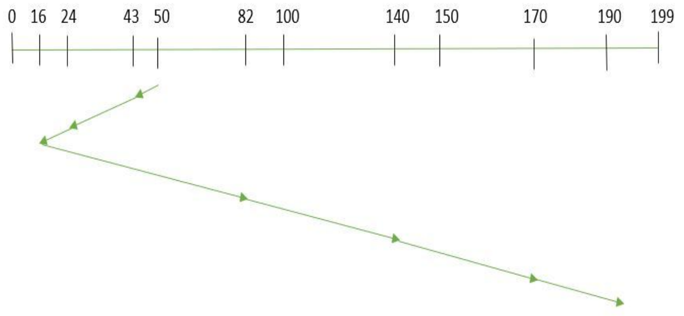
(82-50)+(170-82)+(170-43)+(140-43)+(140-24)+(24-16)+(190-16) = 642

Advantages of FCFS
Here are some of the advantages of First Come First Serve.
- Every request gets a fair chance
- No indefinite postponement

Disadvantages of FCFS
Here are some of the disadvantages of First Come First Serve.
- Does not try to optimize seek time
- May not provide the best possible service

2. **Shortest Seek Time First (SSTF)**
   - In SSTF (Shortest Seek Time First), requests having the 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 the 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 overhead movement (total distance covered by the disk arm) =
(50-43)+(43-24)+(24-16)+(82-16)+(140-82)+(170-140)+(190-170) =208

Advantages of Shortest Seek Time First
Here are some of the advantages of Shortest Seek Time First.
- The average Response Time decreases
- Throughput increases

Disadvantages of Shortest Seek Time First
Here are some of the disadvantages of Shortest Seek Time First.
- Overhead to calculate seek time in advance

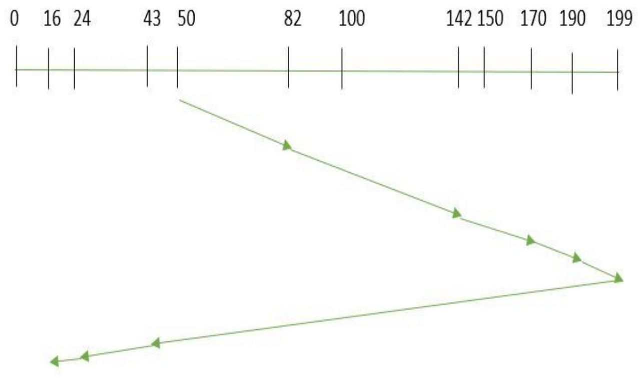
- Can cause Starvation for a request if it has a higher seek time as compared to incoming requests
- The high variance of response time as SSTF favors only some requests

3. Elevator (SCAN)

- In the SCAN algorithm the disk arm moves in a particular direction and services the requests coming in its path and after reaching the end of the disk, it reverses its direction and again services the request arriving in its path. So, this algorithm works as an elevator and is hence also known as an 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 total overhead movement (total distance covered by the disk arm) is calculated as
= (199-50) + (199-16) = 332



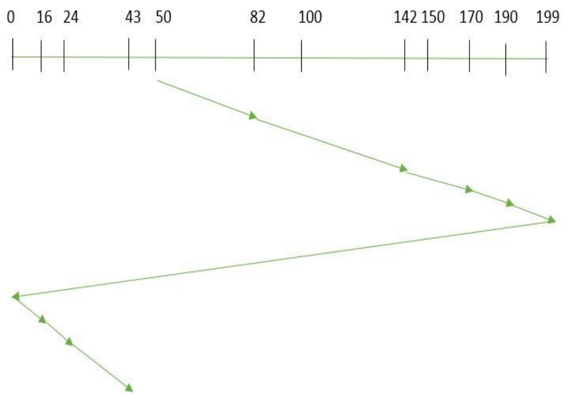
Advantages of SCAN Algorithm
Here are some of the advantages of the SCAN Algorithm.
- High throughput
- Low variance of response time
- Average response time

Disadvantages of SCAN Algorithm
Here are some of the disadvantages of the SCAN Algorithm.
- Long waiting time for requests for locations just visited by disk arm

4. **C-SCAN**

- In the 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 the *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 the SCAN algorithm 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".**
So, the total overhead movement (total distance covered by the disk arm) is calculated as:
=(199-50) + (199-0) + (43-0) = 391

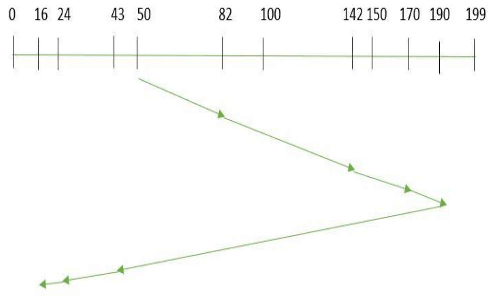
Advantages of C-SCAN Algorithm
Here are some of the advantages of C-SCAN.

- Provides more uniform wait time compared to SCAN.

5. **LOOK**

- LOOK Algorithm 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".**

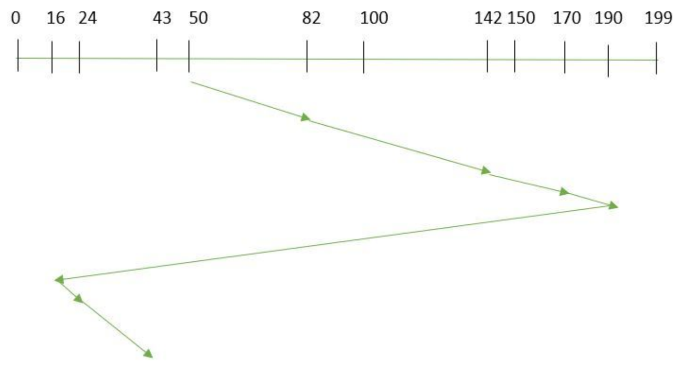
So, the total overhead movement (total distance covered by the disk arm) is calculated as:
= (190-50) + (190-16) = 314

6. **C-LOOK**

- As LOOK is similar to the SCAN algorithm, in a similar way, C-LOOK is similar to the 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:

1. Suppose the requests to be addressed are-82,170,43,140,24,16,190.
2. 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 total overhead movement (total distance covered by the disk arm) is calculated as
= (190-50) + (190-16) + (43-16) = 341

Conclusion: