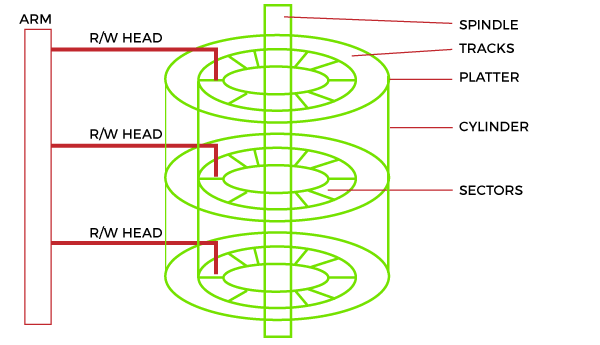
**Disk Scheduling Algorithms**

**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.

What is Disk Access Time in Disk Scheduling?

Disk Access Time is defined as the total time required by the computer to process a read/write request and then retrieve the required data from the disk storage.

There are two components in disk access time. The first component is the ***seek time*** which occurs when the read and write arm seeks the desired track. The second component is latency or wait *time* which occurs when the head write arm waits for the desired sector on the track to spin around.



Access to the data on disks is measured in terms of milliseconds. However, this is actually much slower than the processing speeds of CPUs. Although I/O is still slow, it cannot match the speed improvements of modern processors. Disk Access Time is divided into two parts:

1. Access Time
2. Data Transfer Time

**Formula**

**1. Access Time**

Access Time is defined as the setup time before the actual data transfer takes place. For example, the read/write head is on track 1, but we need to read data from another track or segment. Thus, the read/write head will move to the data block location before the actual transfer occurs. This delay is called ***Access Time***. Access Time is calculated by summation of the following:

Disk Access Time = Access Time + Data Transfer Time

* **Seek Time:** The time the read/write head takes to reach the desired output. It is known to be the most important time because it cannot create a gap. Seek time is inversely proportional to the performance. So the lesser seek time, the better will be the performance.
* Seek Time = (Number of tracks/cylinders crossed) \* (Time to cross one track/cylinder)
* **Rotational Latency:** The time read/write head is required to move from the current to the requested sector.
* Rotational latency = (Angle by which disk is rotated) / (Angular Frequency)
* **Settle Time:** Settle time is the time required by the read/write head to stop vibrating.

**2. Data Transfer Time**

Data Transfer Time is defined as the time required to transfer data between the system and the disk. Data Transfer Time is two types:

* **Internal Transfer Rate:** It is defined as the time required to move data between the disk surface and the hard disk cache.
* **External Transfer Rate:** It is defined as the time required to move data between the hard disk cache and the system.

Difference between Seek Time and Disk Access Time

Below are some differences between rotational latency and disk access time in disk scheduling, such as:

|  |  |
| --- | --- |
| **Seek Time** | **Disk Access Time** |
| Seek time is the time taken by the head to move from the current track to the one where data is present. | Disk access time is the time required by the computer to process a read/write request and retrieve the required data. |
| Seek time is always less than Disk Access Time, and since it is a sub-part of Disk Access time. | Disk access time is very large compared to Seek time. |
| Seek time doesn't consider a transfer of data. | Disk access time considers the time required to transfer data. |
| Seek time can vary a lot depending upon the distance between the current and final position and how it has been instructed to go. | Disk Access Time depends on two parts, i.e. access time and data transfer time. |
| Seek time is usually between 10-20 ms, depending upon the RPS and drive grade. | We can reduce Disk Access Time if we can reduce access time and data transfer time. |
| Seek time is measured in Average Seek Time. | Disk Access Time, = Seek time + Rotational Latency + Data Transfer Time |

**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.

*Here, we will cover the following Topics in Disk Scheduling.*

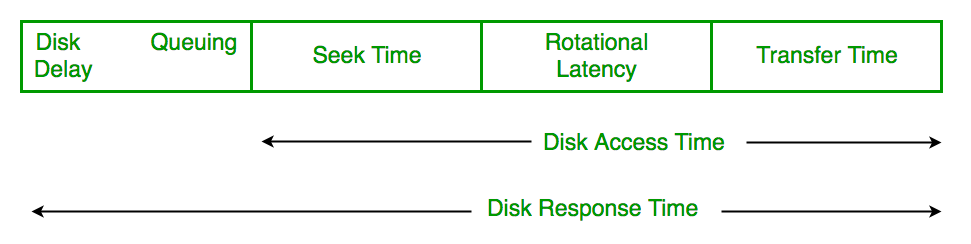
* *Important Terms Associated with Disk Scheduling*
* *Disk Scheduling Algorithms*
  + *FCFS (First Come First Serve)*
  + *SSTF (Shortest Seek Time First)*
  + *SCAN (Elevator Algorithm)*
  + *C-SCAN (CIrcular SCAN)*
  + *LOOK*
  + *C-LOOK*
  + *RSS*
  + *LIFO (Last-In First-Out)*
  + *N-Step SCAN*
  + *F-SCAN*

**Key Terms Associated with 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*



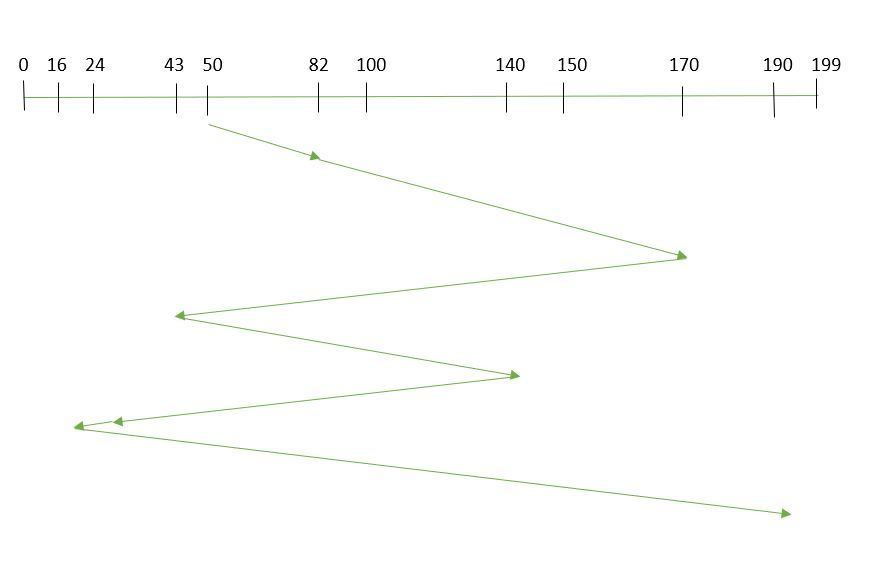
* **Disk Response Time:**Response Time is the average time spent by a request waiting to perform its I/O operation. The average Response time is the response time of all requests. Variance Response Time is the measure of how individual requests are serviced with respect to average response time. So the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

## ****Disk Scheduling Algorithms****

There are several Disk Several Algorithms. We will discuss each one of them.

### **FCFS (First Come First Serve)**

[FCFS](https://www.geeksforgeeks.org/fcfs-disk-scheduling-algorithms/) 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.



*First Come First Serve*

**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**

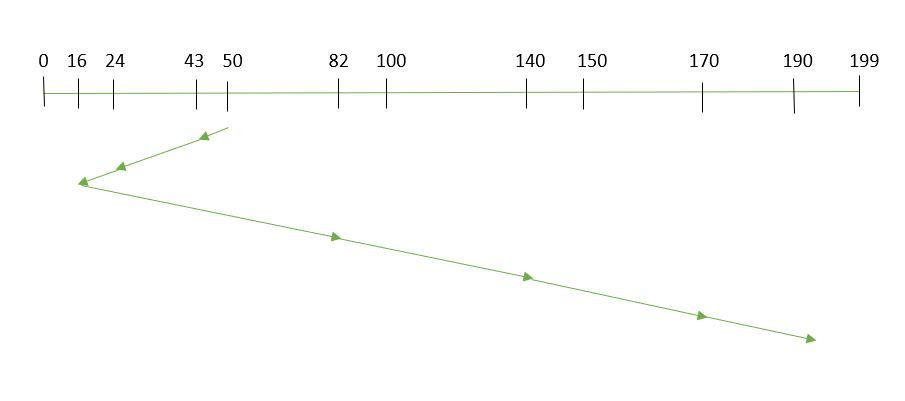
Here are some of the disadvantages of First Come First Serve.

* Does not try to optimize seek time
* May not provide the best possible service

### **SSTF (Shortest Seek Time First)**

In [SSTF (Shortest Seek Time First)](https://www.geeksforgeeks.org/program-for-sstf-disk-scheduling-algorithm/), 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:**



*Shortest Seek Time First*

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**

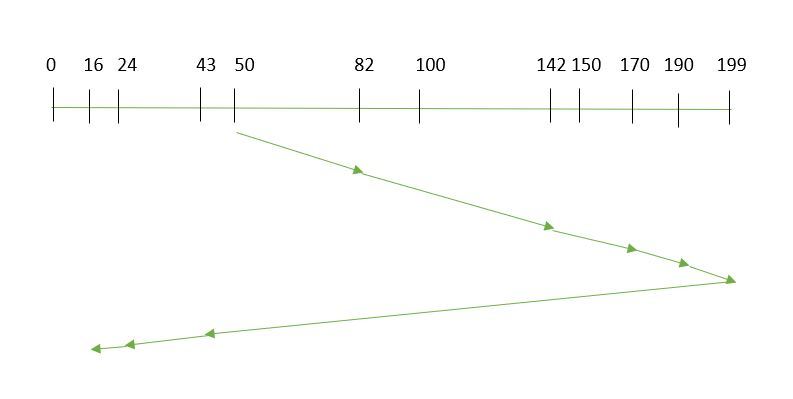
Here are some of the disadvantages of Shortest Seek Time First.

* 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

### **SCAN**

In the [SCAN algorithm](https://www.geeksforgeeks.org/scan-elevator-disk-scheduling-algorithms/) 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:**



*SCAN Algorithm*

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.

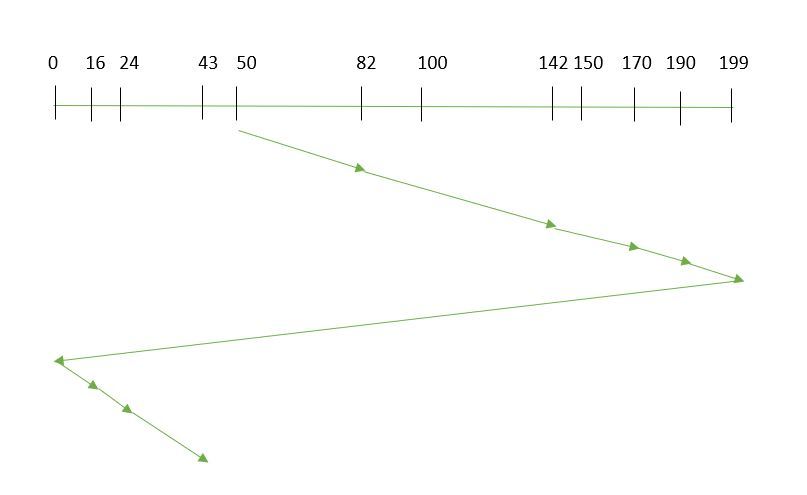
* Long waiting time for requests for locations just visited by disk arm

### **C-SCAN**

In the [SCAN algorithm](https://www.geeksforgeeks.org/c-scan-disk-scheduling-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:**



*Circular SCAN*

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

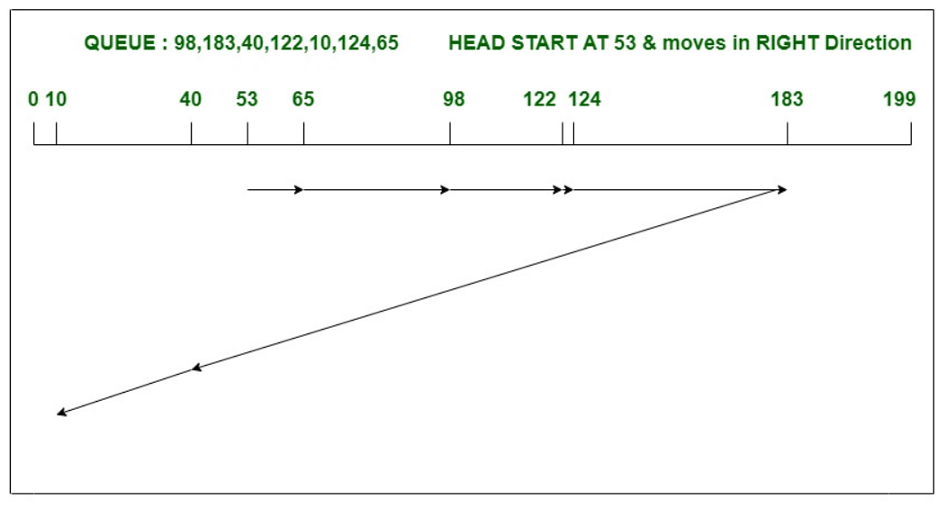
**Advantages of C-SCAN Algorithm**

Here are some of the advantages of C-SCAN.

* Provides more uniform wait time compared to SCAN.

[LOOK disk scheduling algorithm](https://www.geeksforgeeks.org/look-disk-scheduling-algorithm/)**:**  
Look Algorithm is actually an improves version of SCAN Algorithm. In this algorithm, the head starts from first request at one side of disk and moves towards the other end by serving all requests in between. After reaching the last request of one end, the head reverse its direction and returns to first request, servicing all requests in between. Unlike SCAN, in this the head instead of going till last track, it goes till last request and then direction is changed.

**Example –**  
Consider a disk with 200 tracks (0-199) and the disk queue having I/O requests in the following order as follows: 98, 183, 40, 122, 10, 124, 65. The current head position of the Read/Write head is 53 and will move in Right direction . Calculate the total number of track movements of Read/Write head using LOOK algorithm.



Total head movements,

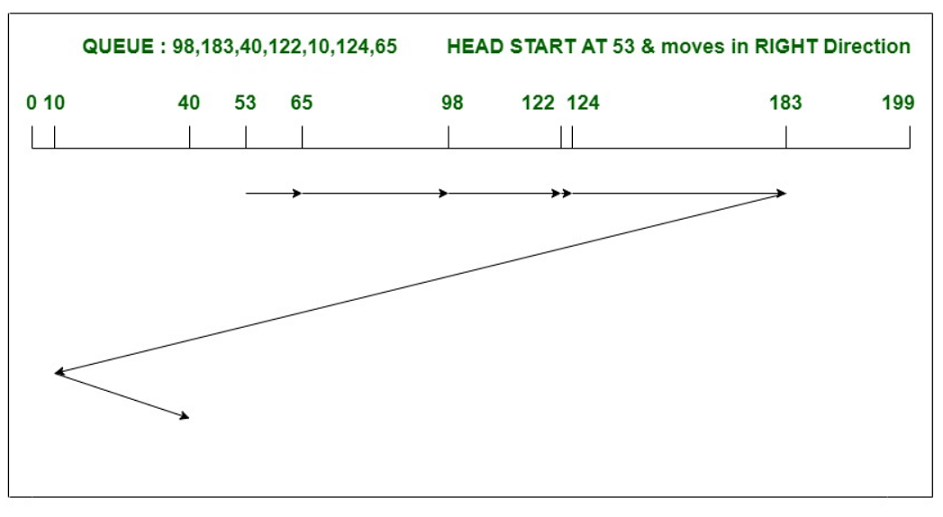
= (65-53)+(98-65)+(122-98)

+(124-122)+(183-124)+(183-40)+(40-10)

= 303

[C-LOOK disk scheduling algorithm](https://www.geeksforgeeks.org/c-look-disk-scheduling-algorithm/)**:**  
C-LOOK is the modified version of both LOOK and SCAN algorithms. In this algorithm, the head starts from first request in one direction and moves towards the last request at other end, serving all request in between. After reaching last request in one end, the head jumps in other direction and move towards the remaining requests and then satisfies them in same direction as before. Unlike LOOK, it satisfies requests only in one direction.

**Example –**  
Consider a disk with 200 tracks (0-199) and the disk queue having I/O requests in the following order as follows: 98, 183, 40, 122, 10, 124, 65. The current head position of the Read/Write head is 53 and will move in Right direction . Calculate the total number of track movements of Read/Write head using C-LOOK algorithm.

Total head movements,

= (65-53)+(98-65)+(122-98)

+(124-122)+(183-124)+(183-10)+(40-10)= 333

**Difference between LOOK and C-LOOK disk scheduling algorithm :**

|  |  |  |
| --- | --- | --- |
|  | **LOOK DISK SCHEDULING ALGORITHM** | **C-LOOK SCHEDULING ALGORITHM** |
| 1. | In LOOK, the head can serve the requests in both the directions. | In C-LOOK algorithm, head can serves the requests only in one direction. |
| 2. | It lags in performance as compared to C-LOOK. | C-LOOK algorithm has the best performance in all disk scheduling algorithms. |
| 3. | In above example of LOOK algorithm, the head moves from 53, serves all requests in right direction till it reaches the last request in one end. Then it reverses the direction and serves the remaining requests in left direction. | In above example of C-LOOK algorithm, the head moves from 53, serves all requests in right direction till it reaches the last request in one end. Then it jumps to the remaining requests and serve them in right direction only. |
| 4. | Here handling of request is not so good as compared to C-LOOK algorithm. | C-LOOK algorithm can handle requests more effectively than LOOK. |
| 5. | LOOK has higher throughput and provides low variance response time. | C-LOOK provides uniform waiting time and response time. |