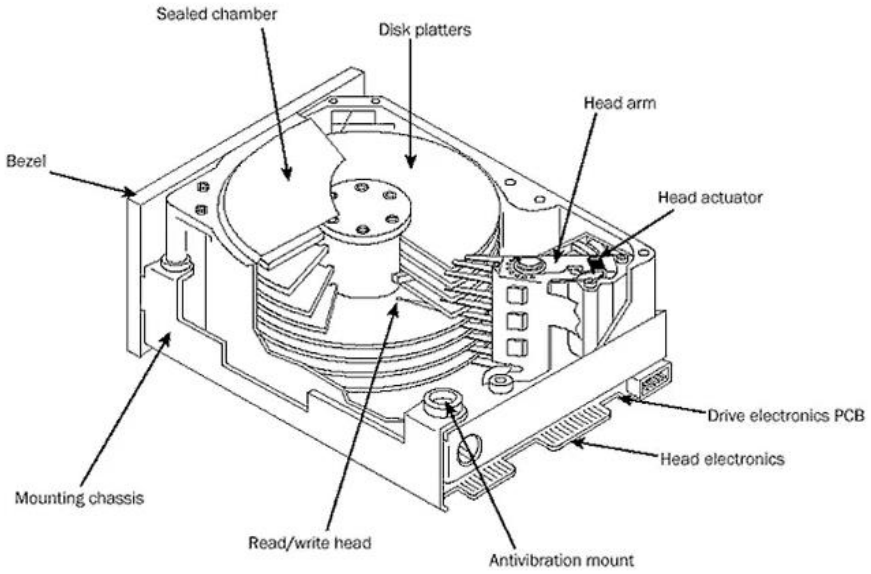
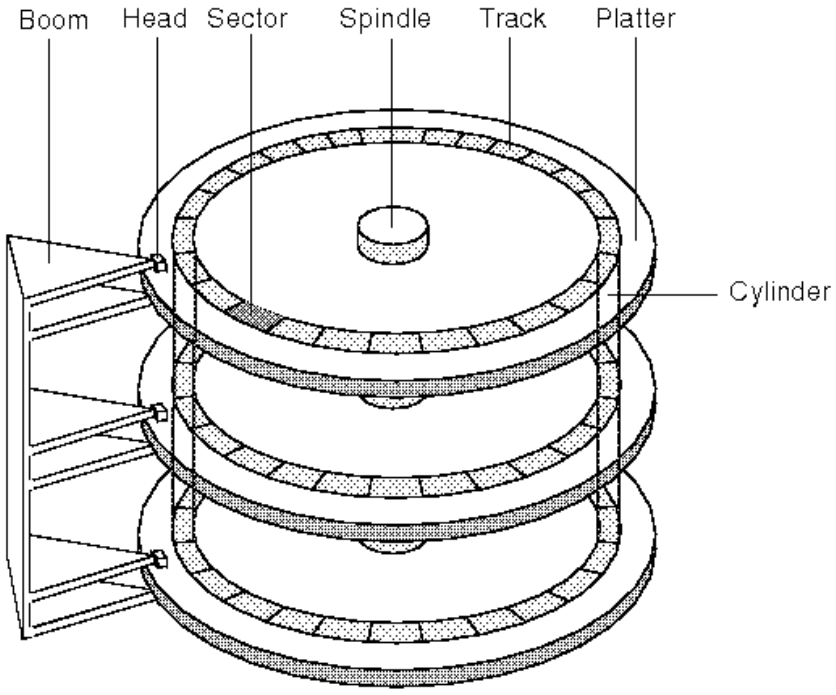
UNIT 5: Storage Management

Structure of the Disk

The hard disk is so called because it is a hardware device that encases a series of disk-like structures coated with some magnetic material. Data can be read or written to this magnetic material. These individual discs called platters, spin rapidly on a spindle and are accessed by a mechanical magnetic head. The concept is similar to our music record players or turntables used years ago. The platters may store data on both sides and the magnetic head is accessible to both the inner and outer portions of the platters.





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.

Disk scheduling is important because:

- 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 request may be far from each other so 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.

The technique that operating system uses to determine the request which is to be satisfied next is called disk scheduling.

Some Terminologies

- Seek Time: Seek time is the time taken in locating the disk arm to a specified track where the read/write request will be satisfied.
- Rotational Latency: It is the time taken by the desired sector to rotate itself to the position from where it can access the R/W heads.
- Transfer Time: It is the time taken to transfer the data.
- Disk Access Time: Disk access time is given as,

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

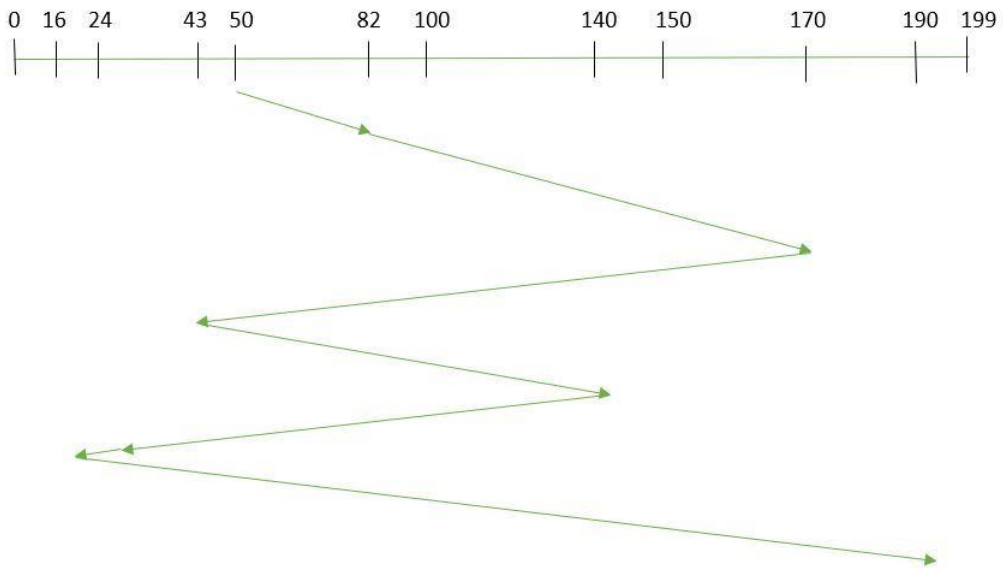
- Disk Response Time: It is the average of time spent by each request waiting for the IO operation.

Disk Scheduling Algorithms

- FCFS scheduling algorithm
- SSTF (shortest seek time first) algorithm
- SCAN scheduling
- C-SCAN scheduling
- LOOK Scheduling
- C-LOOK scheduling

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.

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



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

OR

Total Seek Time= (170-50)+(170-43)+(140-43)+(140-16)+(190-16) =
120+127+97+124+174=642

Question : Consider the following disk request sequence for a disk with 100 tracks 45, 21, 67, 90, 4, 50, 89, 52, 61, 87, 25

Head pointer starting at 50 and moving in left direction. Find the number of head movements in cylinders using FCFS scheduling.

Solution: Number of cylinders moved by the head

= (50-45)+(45-21)+(67-21)+(90-67)+(90-4)+(50-4)+(89-50)+(61-52)+(87-61)+(87-25)

= 5 + 24 + 46 + 23 + 86 + 46 + 49 + 9 + 26 + 62

= 376

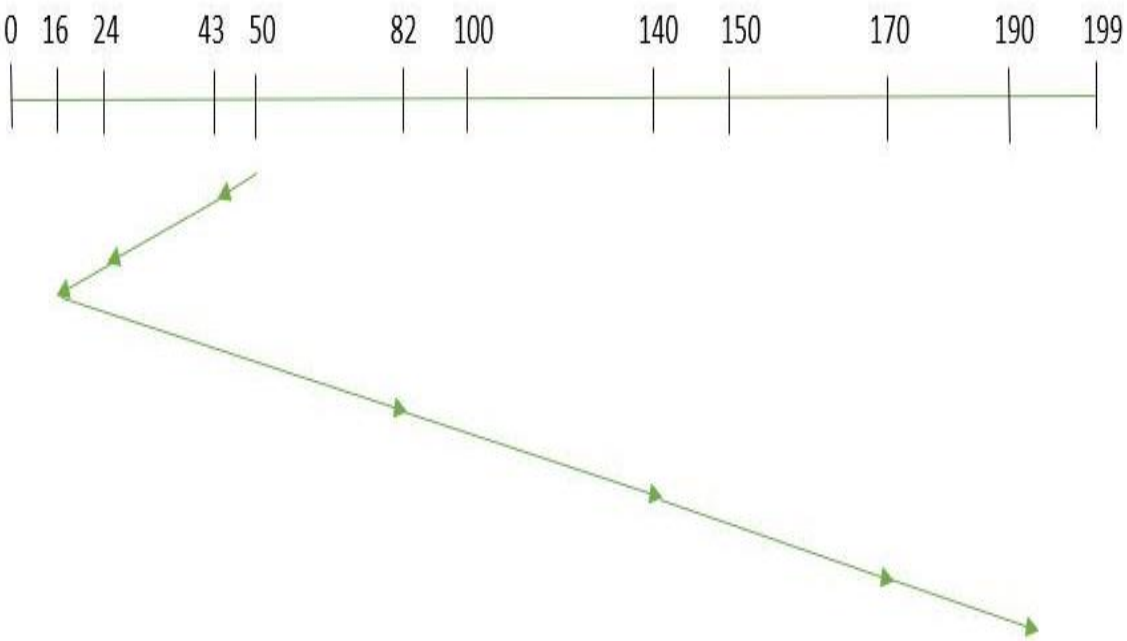
Question: Request sequence = {176, 79, 34, 60, 92, 11, 41, 114}

Initial head position = 50

Seek Time : 510

SSTF: In SSTF (Shortest Seek Time First), this algorithm services that request next which requires least number of head movements from its current position regardless of the direction. It breaks the tie in the direction of head movement.

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



Total Seek Time= (50-16)+(190-16)=208

Example: Consider a disk queue with requests for I/O to blocks on cylinders 98, 183, 41, 122, 14, 124, 65, 67. The SSTF scheduling algorithm is used. The head is initially at cylinder number 53 moving towards larger cylinder numbers on its servicing pass.

(208)

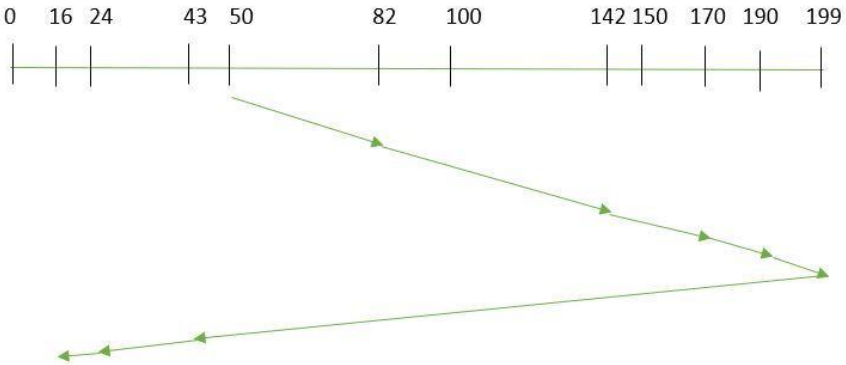
Example

Consider a disk system with 100 cylinders. The requests to access the cylinders occur in following sequence- 4, 34, 10, 7, 19, 73, 2, 15, 6, 20. Assuming that the head is currently at cylinder 50, calculate total seek time.

(50-2) + (73-2) = 48+71 = 119

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.**

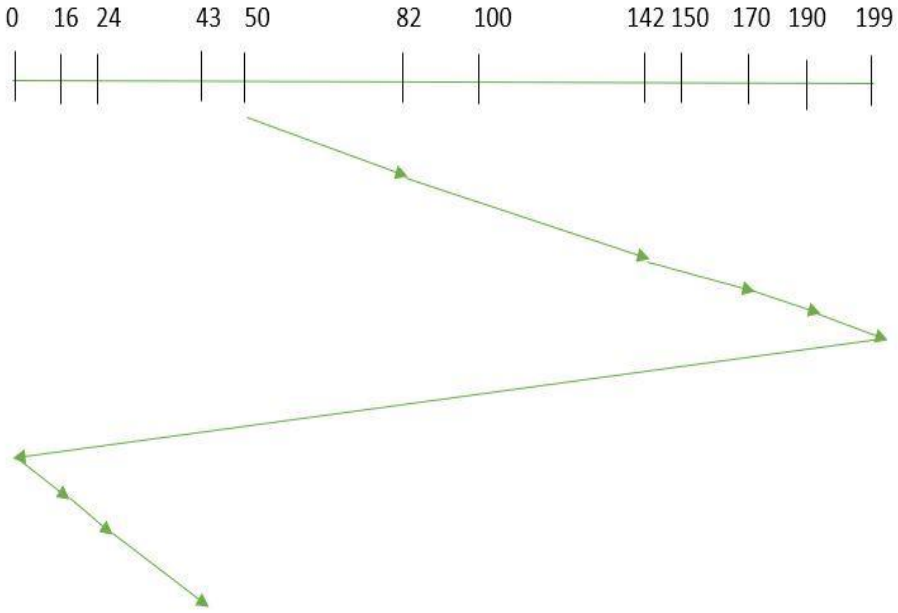
Example: Suppose the requests to be addressed are-82,170,43,140,24,16,190. And the Read/Write arm is at 50, and it is also given that the disk arm should move "towards the larger value".



seek time is calculated as=(199-50)+(199-16) =332

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.

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".

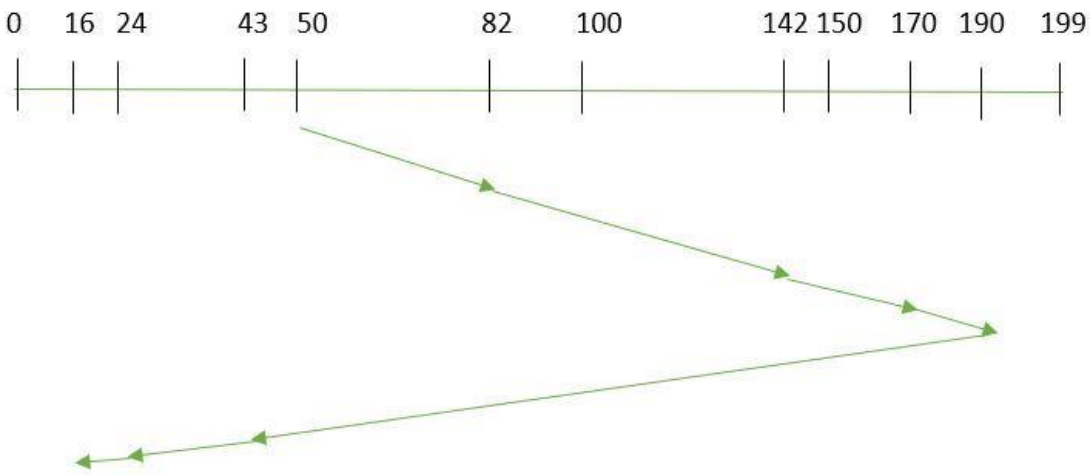


Seek time is calculated as =(199-50)+(199-0)+(43-0) =391

LOOK Disk Scheduling Algorithm-

- LOOK Algorithm is an improved version of the SCAN Algorithm.
- Head starts from the first request at one end of the disk and moves towards the last request at the other end servicing all the requests in between.
- After reaching the last request at the other end, head reverses its direction.
- It then returns to the first request at the starting end servicing all the requests in between

Example:
Suppose the requests to be addressed are-82,170,43,140,24,16,190. And the Read/Write arm is at 50. The cylinders are numbered from 0 to 199.



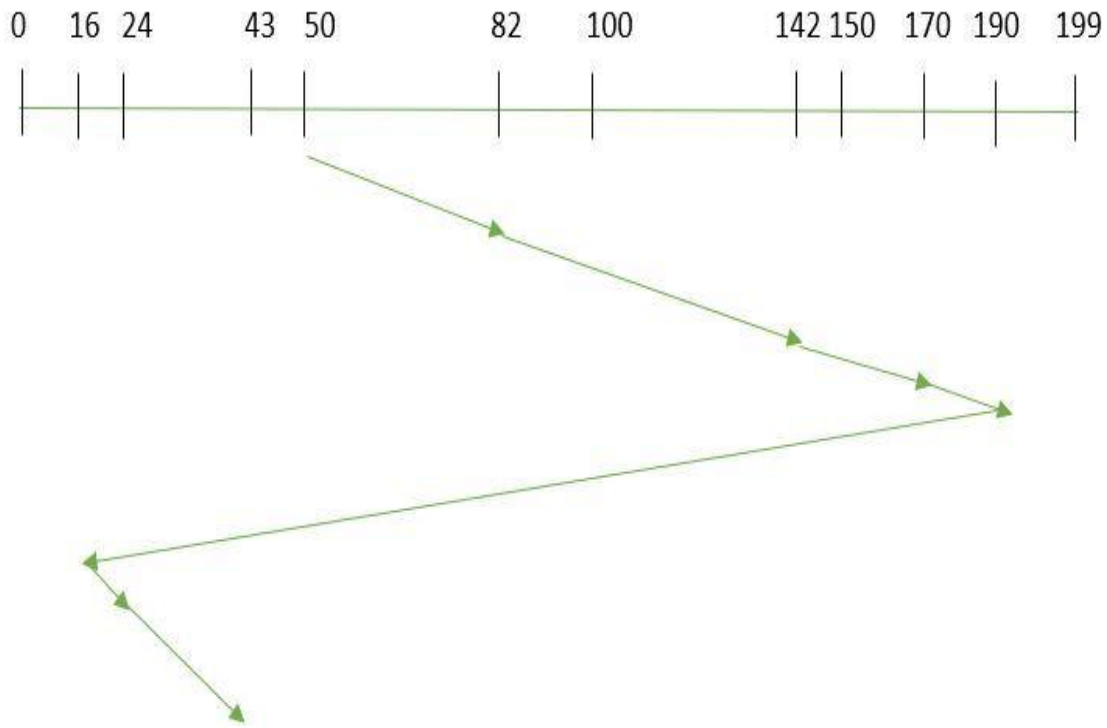
the seek time is calculated as:

=(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"



Seek time is calculated =(190-50)+(190-16)+(43-16) =341

Disk Management

The operating system is responsible for several aspects of disk management.

Disk Formatting

A new magnetic disk is a blank slate. It is just platters of a magnetic recording material. Before a disk can store data, it must be divided into sectors that the disk controller can read and write. This process is called low-level formatting (or **physical formatting).**

Low-level formatting fills the disk with a special data structure for each sector. The data structure for a sector consists of a header, a data area, and a trailer. The header and trailer contain information used by the disk controller, such as a sector number and an **error-correcting code (ECC).**

To use a disk to hold files, the operating system still needs to record its own data structures on the disk. It does so in two steps. The first step is to **partition** the disk into one or more groups

of cylinders. The operating system can treat each partition as though it were a separate disk. For instance, one partition can hold a copy of the operating system's executable code, while another holds user files. After partitioning, the second step is **logical formatting** (or creation of a file system). In this step, the operating system stores the initial file-system data structures onto the disk.

Boot Block

When a computer is powered up or rebooted, it needs to have an initial program to run. This initial program is called the bootstrap program. It initializes all aspects of the system (i.e. from CPU registers to device controllers and the contents of main memory) and then starts the operating system.

To do its job, the bootstrap program finds the operating system kernel on disk, loads that kernel into memory, and jumps to an initial address to begin the operating-system execution.

For most computers, the bootstrap is stored in read-only memory **(ROM).** This location is convenient because ROM needs no initialization and is at a fixed location that the processor can start executing when powered up or reset. And since ROM is read-only, it cannot be infected by a computer virus. The problem is that changing this bootstrap code requires changing the ROM hardware chips.

For this reason, most systems store a tiny bootstrap loader program in the boot ROM, whose only job is to bring in a full bootstrap program from disk. The full bootstrap program can be changed easily: A new version is simply written onto the disk. The full bootstrap program is stored in a partition (at a fixed location on the disk) is called **the boot blocks**. A disk that has a boot partition is called **a boot disk or system disk.**

Bad Blocks

Since disks have moving parts and small tolerances, they are prone to failure. Sometimes the failure is complete, and the disk needs to be replaced, and its contents restored from backup media to the new disk.

More frequently, one or more sectors become defective. Most disks even come from the factory with bad blocks. Depending on the disk and controller in use, these blocks are handled in a variety of ways.

The controller maintains a list of bad blocks on the disk. The list is initialized during the low-level format at the factory and is updated over the life of the disk. The controller can be told to replace each bad sector logically with one of the spare sectors. This scheme is known as sector sparing or forwarding.