Disk Scheduling

Disk Scheduling

- The operating system is responsible for using hardware efficiently —
- for the disk drives, this means having **a fast access time and disk bandwidth**
- Minimize seek time
- Seek time ≈ seek distance

Disk scheduling

- Done by operating systems to schedule I/O requests arriving for the disk.

Hard Disk

- A hard disk or fixed disk
- Is an electro-mechanical data storage device that stores and retrieves digital data using magnetic storage and one or more rigid rapidly rotating platters coated with magnetic material.
- The platters are paired with magnetic heads, usually arranged on a moving actuator arm, which read and write data to the platter surfaces.
- Data is accessed in a random-access manner, individual blocks of data can be stored and retrieved in any order.

Important?

Important because:

1) 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.
2) Two or more request may be far from each other so can result in greater disk arm movement.
3) Hard drives are one of the slowest parts of the computer system and thus need to be accessed in an efficient manner.

Some important terms:

Seek Time:

- Seek time is the time taken to move the disk arm to a specified track containing the desired sector where the data is to be read or written.
- So the disk scheduling algorithm that gives minimum average seek time is better.

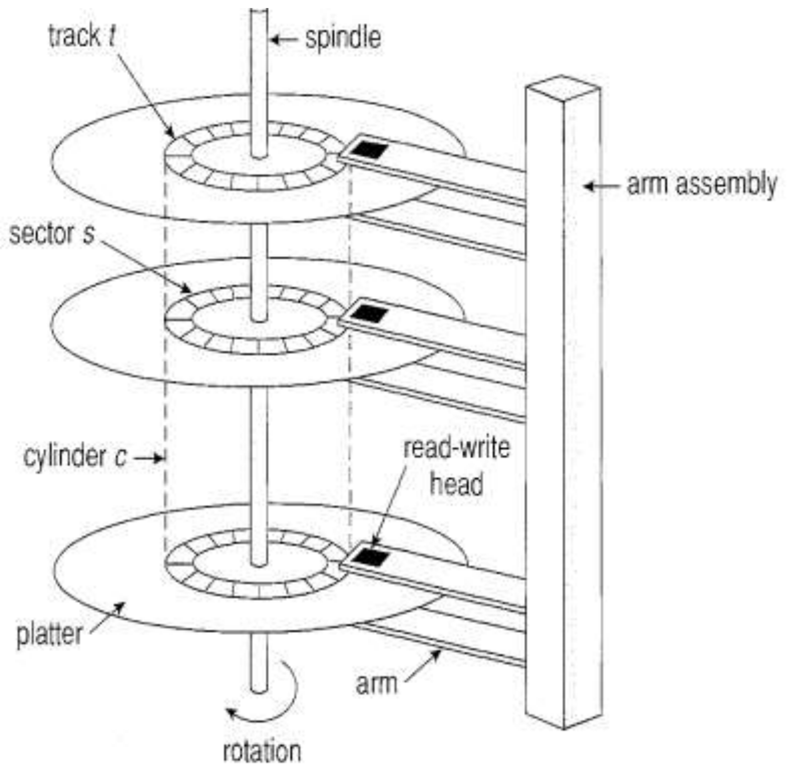


Figure 12.1 Moving-head disk mechanism.

Some important terms:

Rotational Latency:

- Rotational Latency is the time taken by the desired sector of disk to rotate to the disk head, i.e. into a position so that it can access the read/write heads.
- So the disk scheduling algorithm that gives minimum rotational latency is better

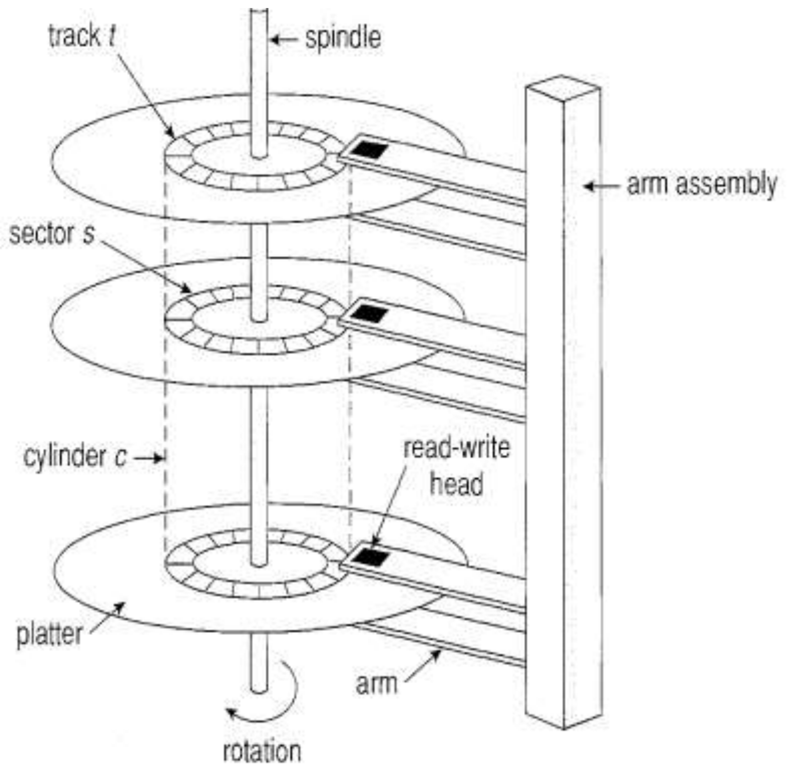


Figure 12.1 Moving-head disk mechanism.

Some important terms:

Disk bandwidth:

- is the total number of bytes transferred, divided by the total time between the first request for service and the completion of the last transfer

Anatomy of a Hard Drive

On the outside, a hard drive looks like this

Anatomy of a Hard Drive

If we take the cover off, we see that there actually is a "hard disk" inside

Anatomy of a Hard Drive

A hard drive usually contains multiple disks, called *platters*

These spin at thousands of RPM (5400, 7200, 10000, …)

Slower disks use less power

Anatomy of a Hard Drive

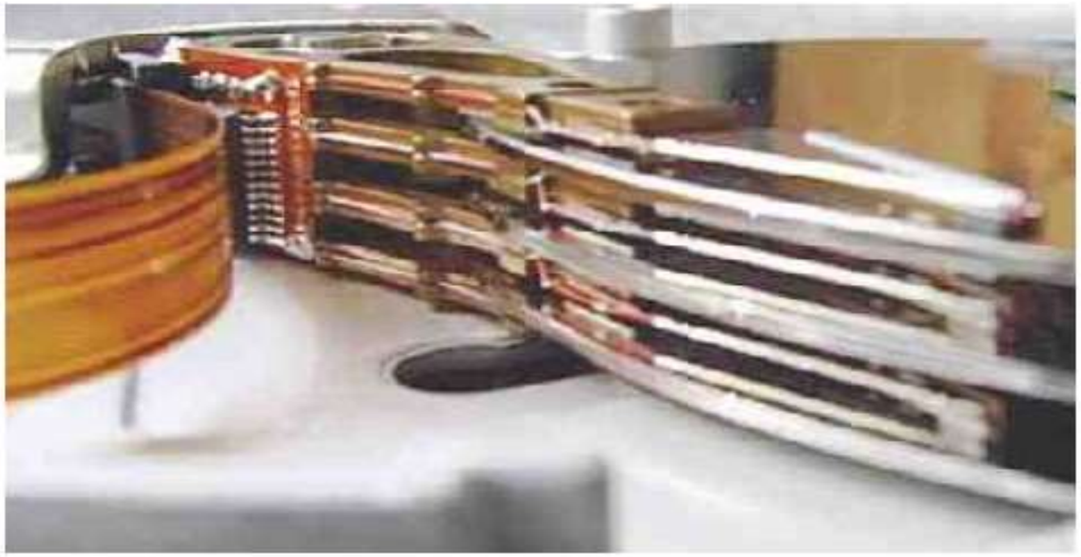
Information is written to and read from the platters by the *read/write heads* on the end of the *disk arm*

Anatomy of a Hard Drive

Both sides of each platter store information

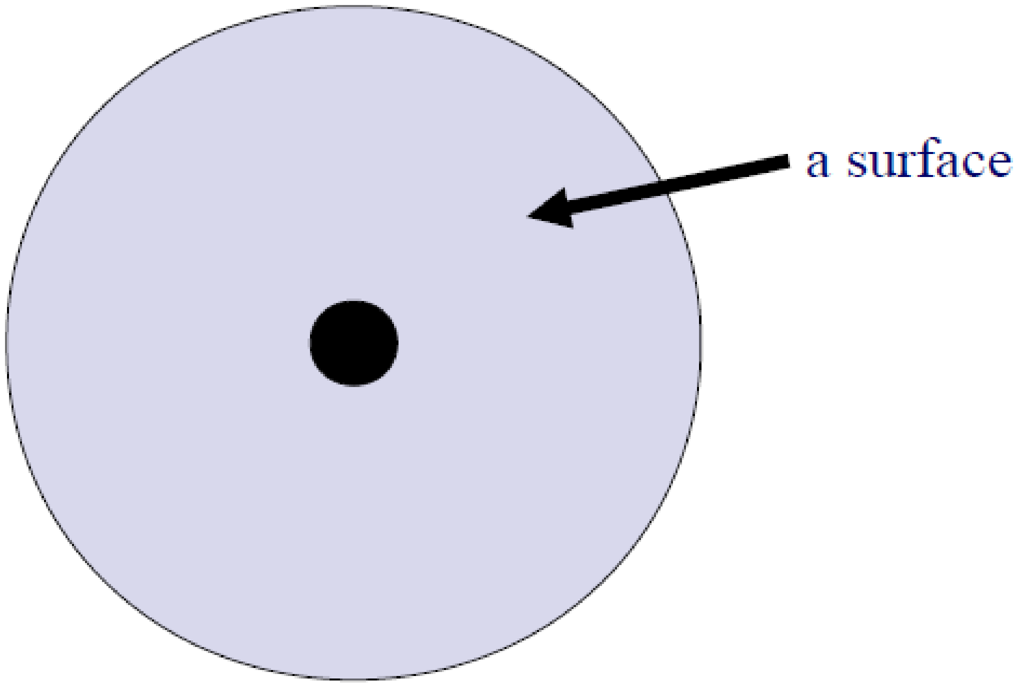
Each side of a platter is called a *surface*

Each surface has its own read/write head



Anatomy of a Hard Drive

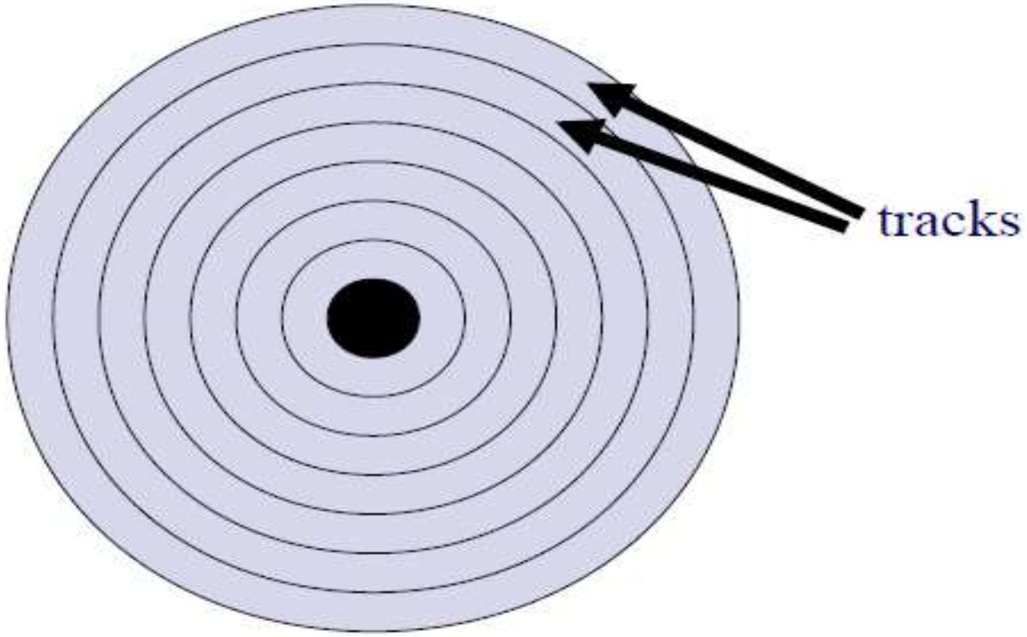
How are the surfaces organized?



Courtesy-https://www.cs.cmu.edu/~410-s14/lectures/L27_Disks1.pdf

Anatomy of a Hard Drive

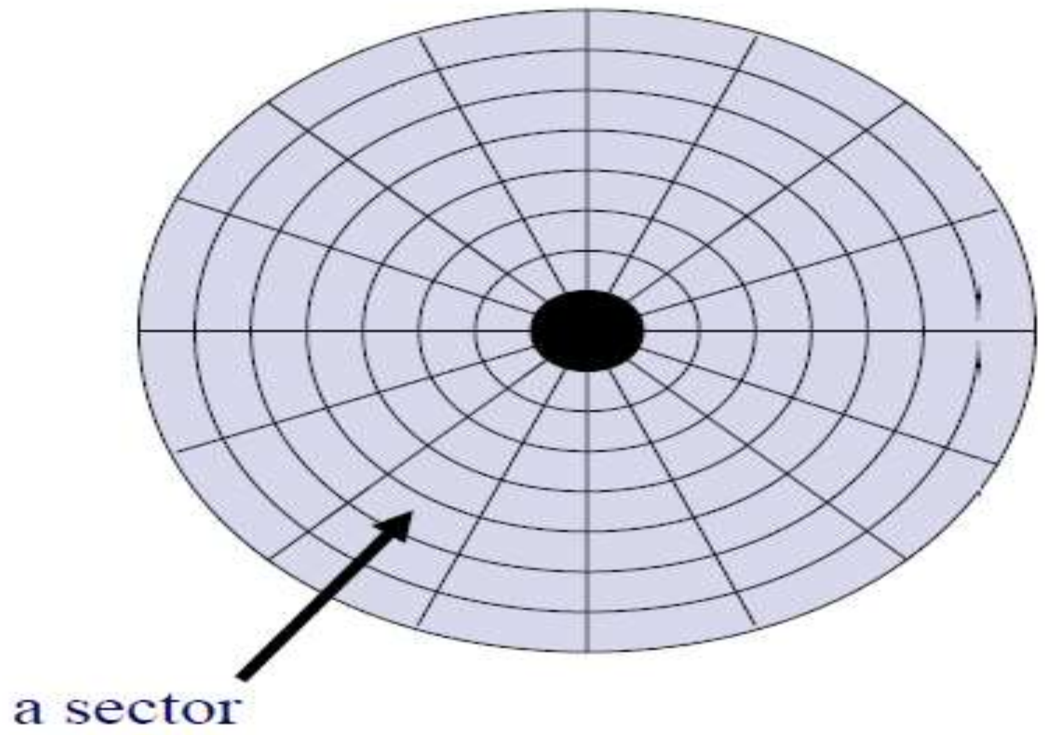
Each surface is divided by concentric circles, creating *tracks*



Anatomy of a Hard Drive

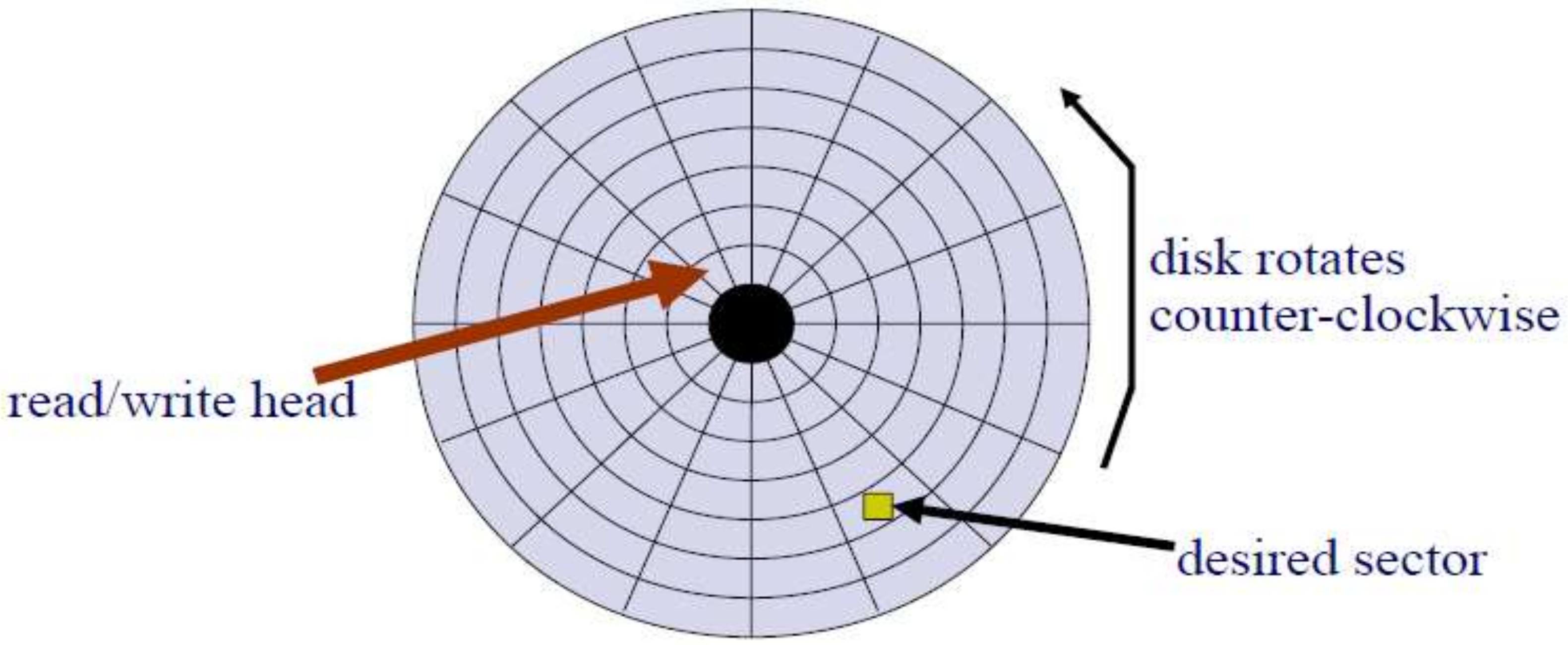
These tracks are further divided into *sectors*

A sector is the smallest unit of data transfer to or from the disk



Anatomy of a Hard Drive

Let's read in a sector from the disk



Some important terms:

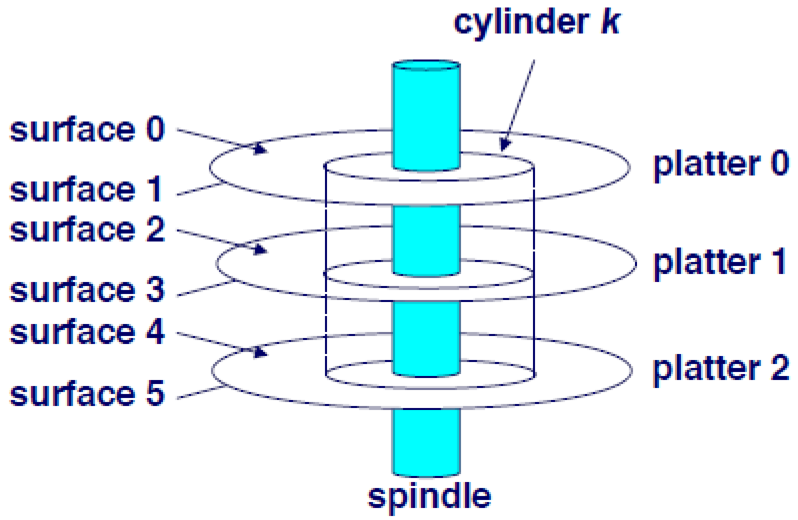
- **Transfer Time:** Transfer time is the time to transfer the data. It depends on the rotating speed of the disk and number of bytes to be transferred.

- **Disk Access Time:** Disk Access Time is:

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

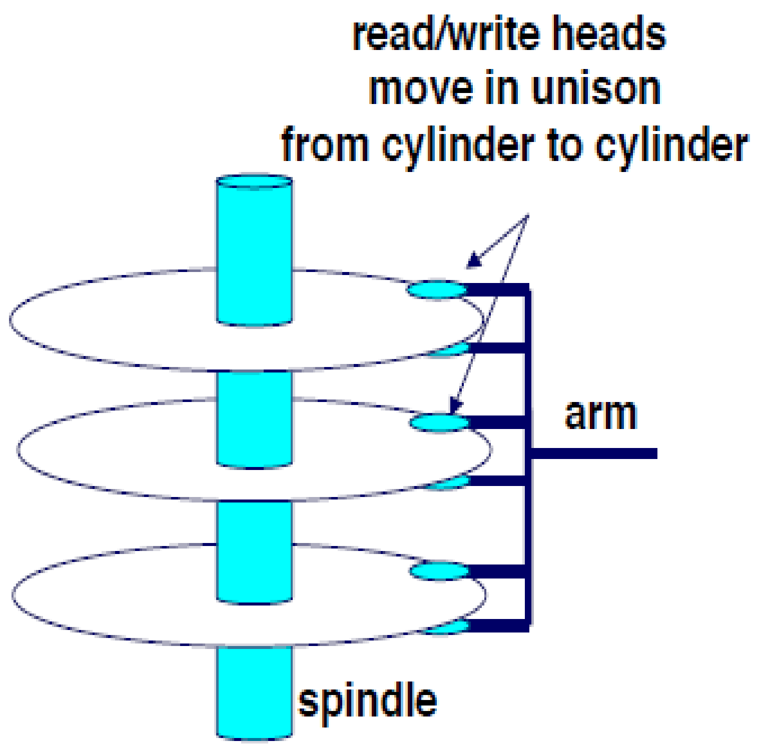
Disk Cylinder

- **Matching track across surfaces are collectively called cylinder**



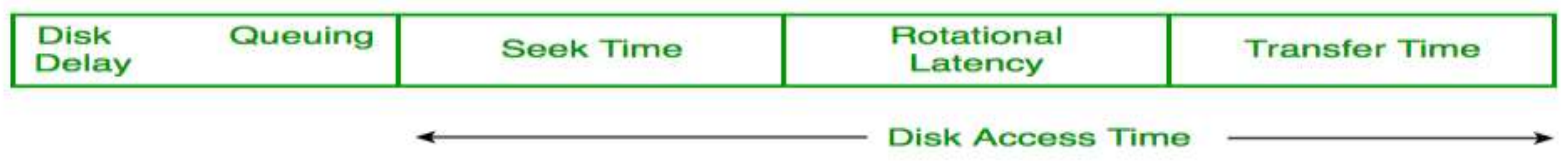
Disk Cylinder

- **Access within a cylinder is faster**
- **Heads share one arm**
  - **All Heads always on same cylinder**
  - **Active Head is aligned, others are closed**



Some important terms:

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



Disk Scheduling (Cont.)

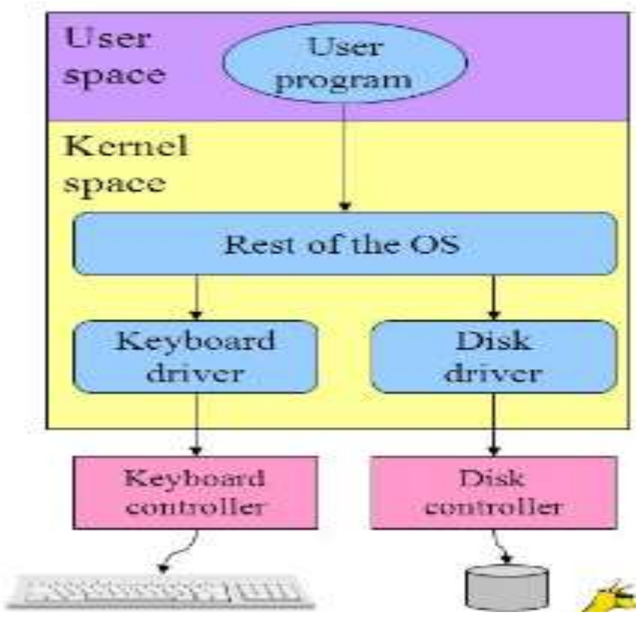
- There are many sources of disk I/O request
  - OS
  - System processes
  - Users processes

Disk Scheduling (Cont.)

- I/O request includes input or output mode, disk address, memory address, number of sectors to transfer
- OS maintains queue of requests, per disk or device
- Idle disk can immediately work on I/O request, busy disk means work must queue
  - Optimization algorithms only make sense when a queue exists

Disk Scheduling (Cont.)

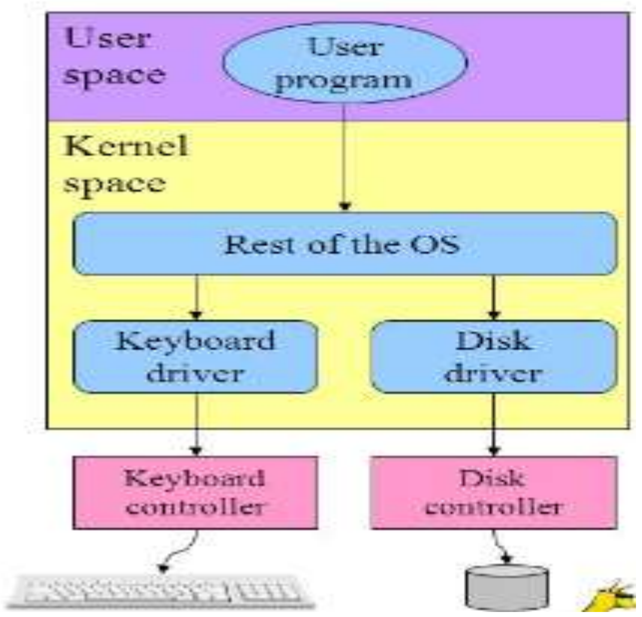
- Drive controllers
  - have small buffers and
  - can manage a queue of I/O requests (of varying "depth")



Disk Scheduling (Cont.)

- Several algorithms exist to schedule the servicing of disk I/O requests

- The analysis is true for one or many platters



Disk Scheduling Algorithms

There are many Disk Scheduling Algorithms-

- FCFS
- SSTF
- SCAN
- C-SCAN
- LOOK
- C-LOOK

Disk Scheduling (Cont.)

- We illustrate scheduling algorithms with a request queue (0-199)
- **Disk Queue with requests for I/O to blocks on cylinders**

98, 183, 37, 122, 14, 124, 65, 67

- **Disk Head is initially at Cylinder 53**

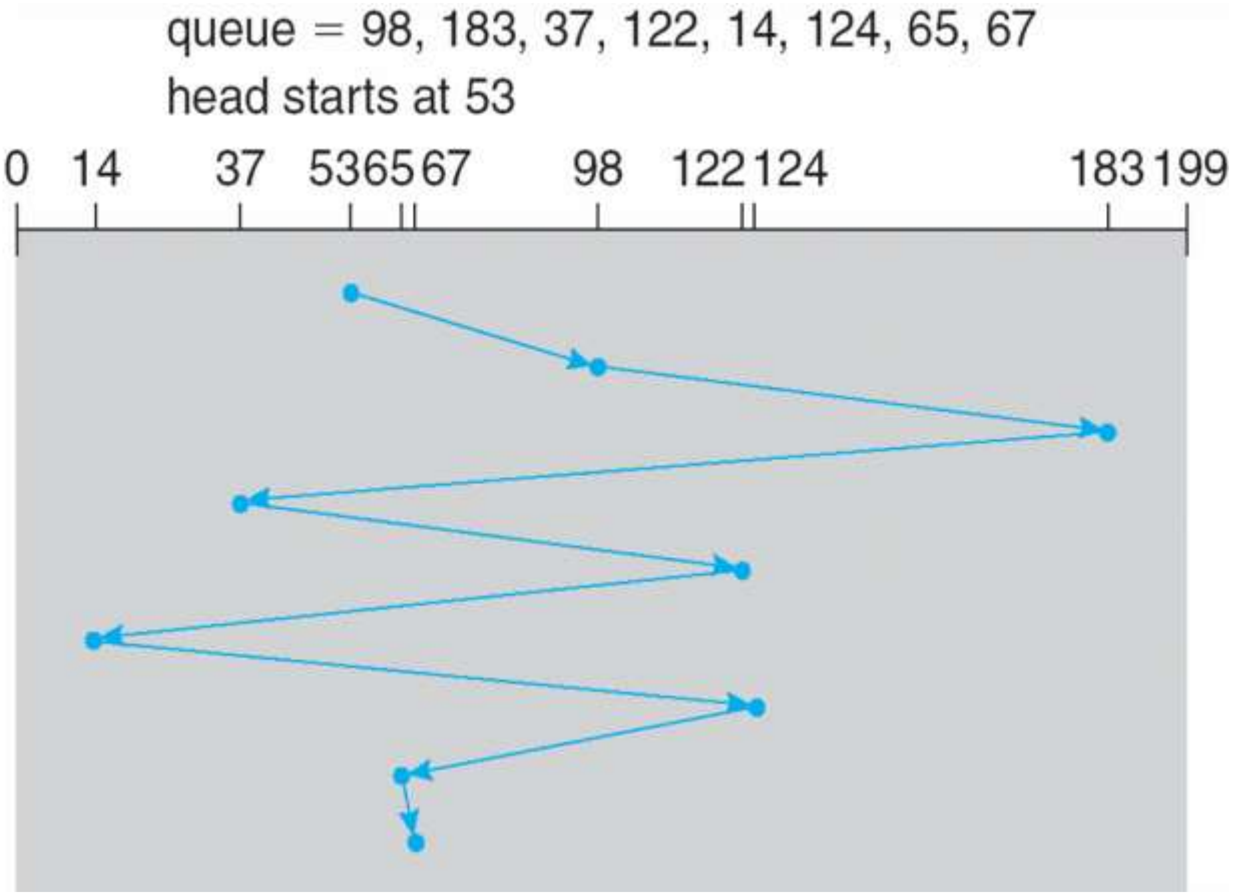
Head pointer 53

FCFS

- Simplest form of disk scheduling
- First-come, first-served (FCFS) algorithm.
- Intrinsically fair
- Does not provide the fastest service

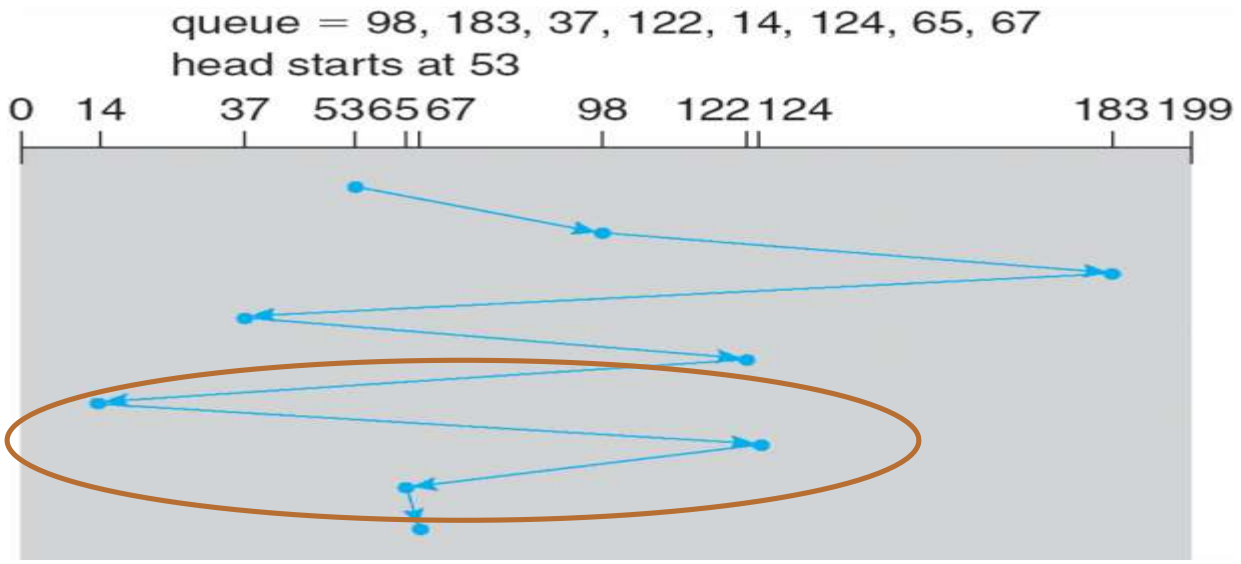
FCFS

Illustration shows total head movement of 640 cylinders



FCFS

- The wild swing from **122 to 14 and then back to 124** illustrates the problem with this schedule.
- If the requests for cylinders **37 and 14** could be serviced together, **before or after the requests for 122 and 124,**
    - the total **head movement** could be **decreased substantially**, and performance could be thereby improved.



FCFS

- Maintain Queue of Requests
- Implemented as FIFO Queue

FCFS

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
- **Unacceptably High Response time**

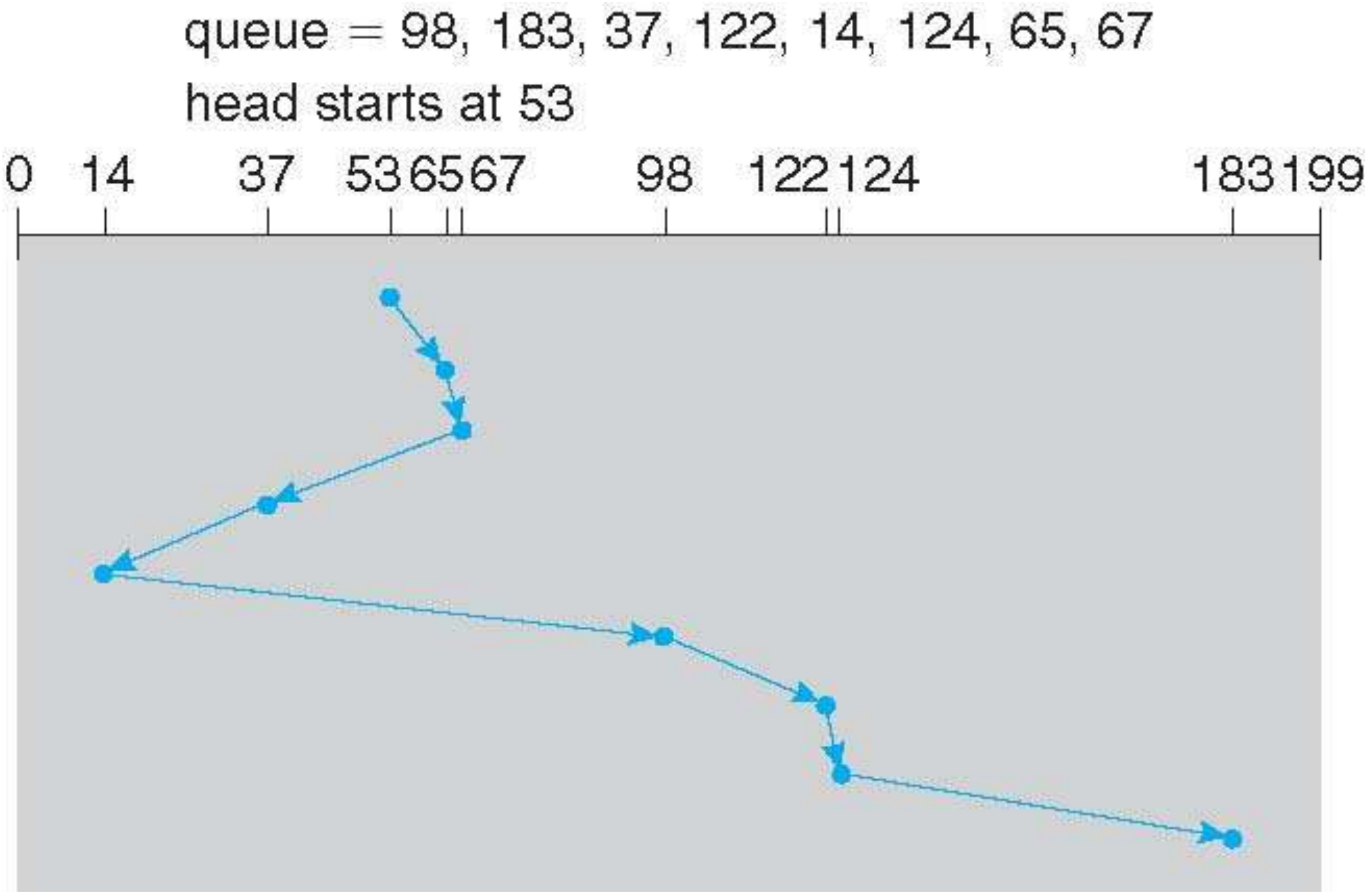
SSTF

- Service all the **requests close to the current head position** before moving the head far to service other

- Shortest Seek Time First selects the **request with the minimum seek time** from the current head position

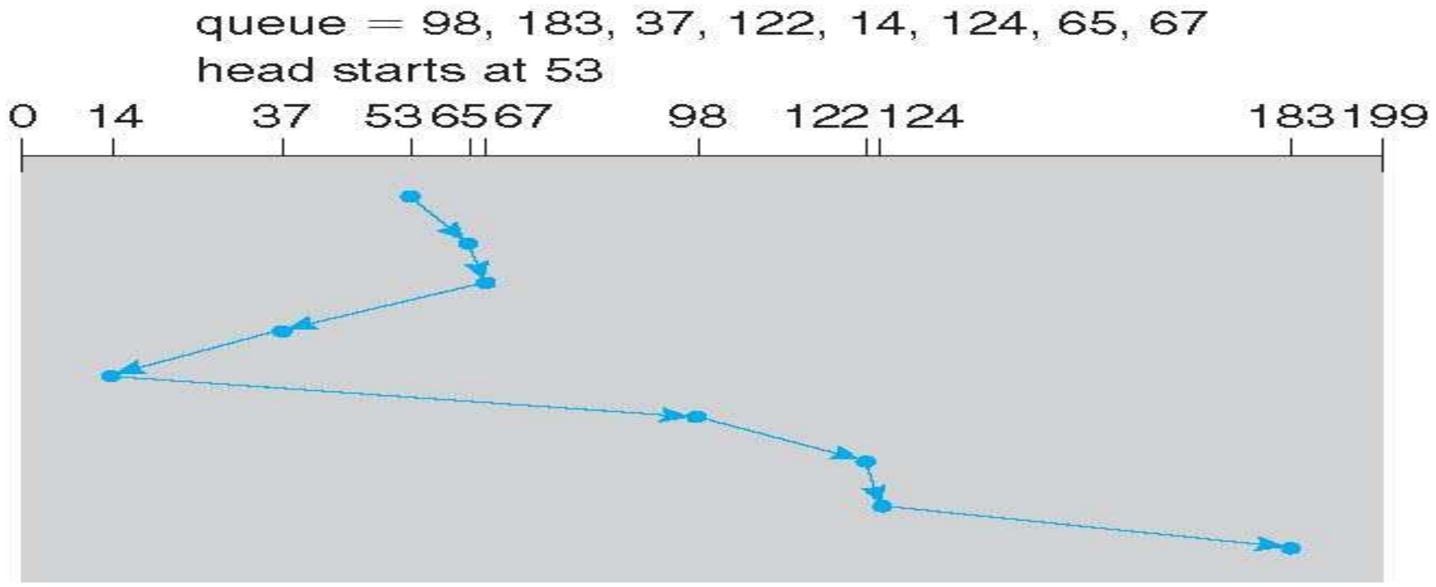
SSTF

- Illustration shows total head movement of 236 cylinders



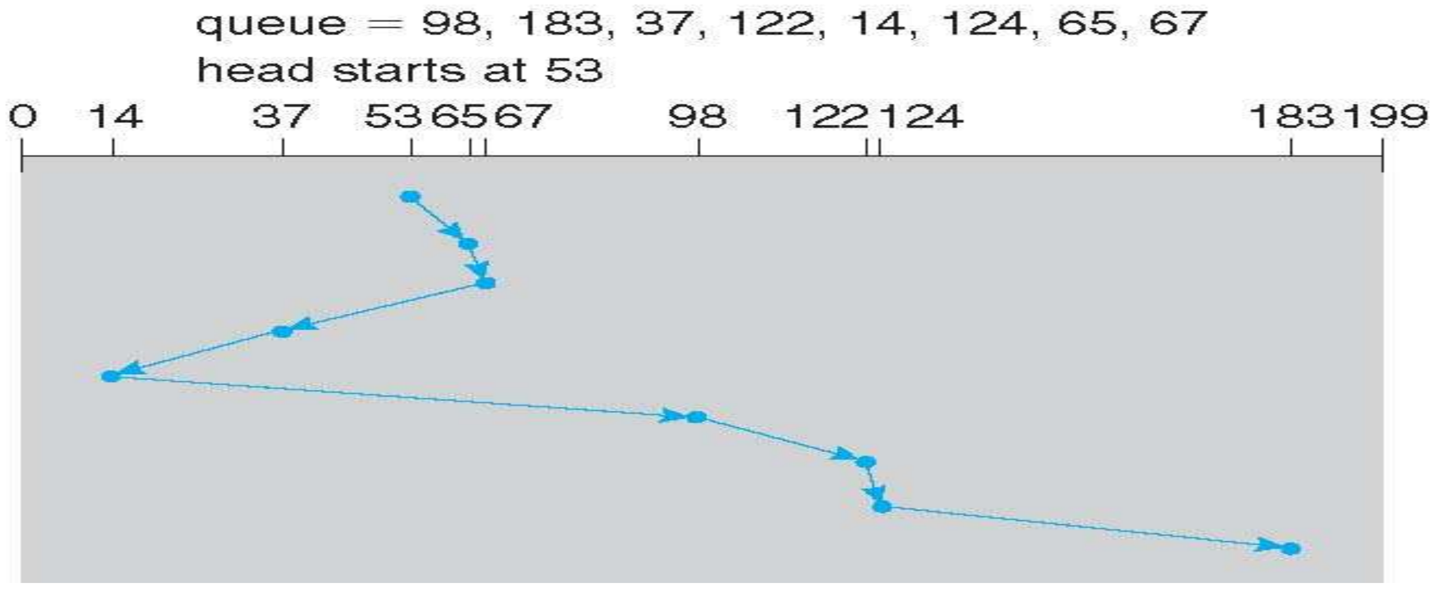
SSTF

- The closest request to the initial head position (53) is at cylinder 65.
- At cylinder 65, the next closest is cylinder 67.



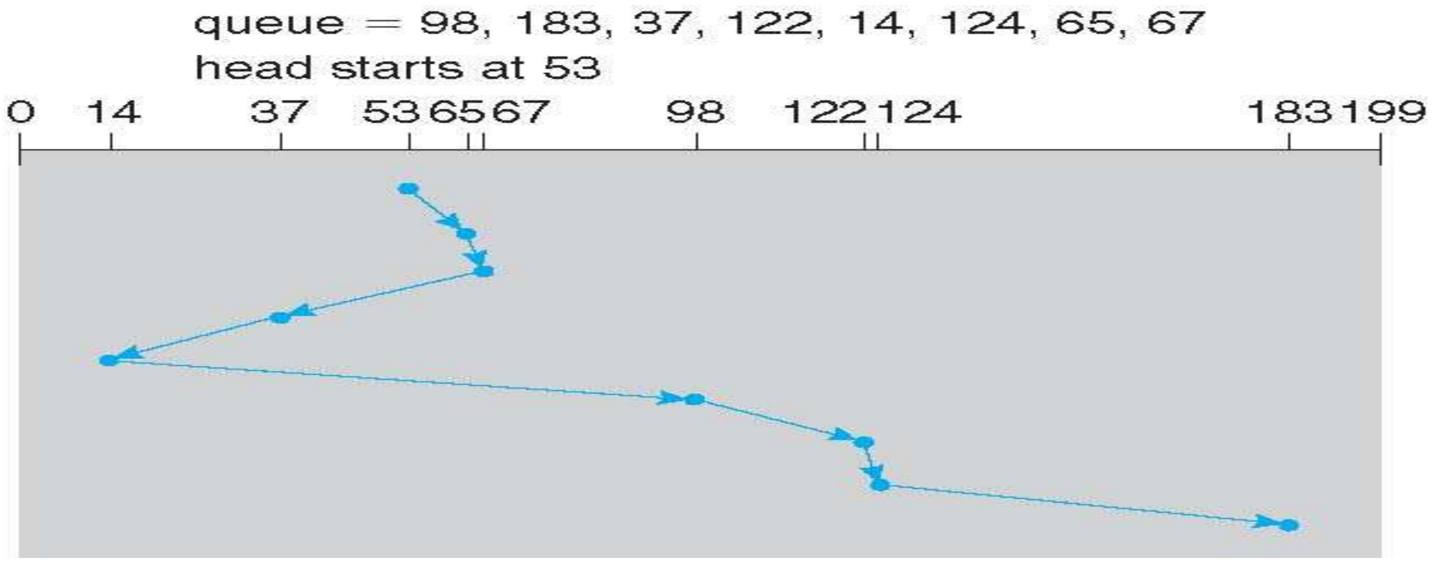
SSTF

- From there, the request at cylinder 37 is closer than the one at 98, so 37 is served next.
  - 67-37=30
  - 98-67=31
- Continuing, we service the request at cylinder 14, then 98, 122, 124, and finally 183



SSTF

- **A total head movement of only 236 cylinders-**
- **In FCFS, 640 cylinders,**
- SSTF-Little more than one-third of the distance needed for FCFS scheduling
- This algorithm gives a substantial improvement in performance.



SSTF

- SSTF scheduling is a form of SJF scheduling; may **cause starvation** of some requests

- How??

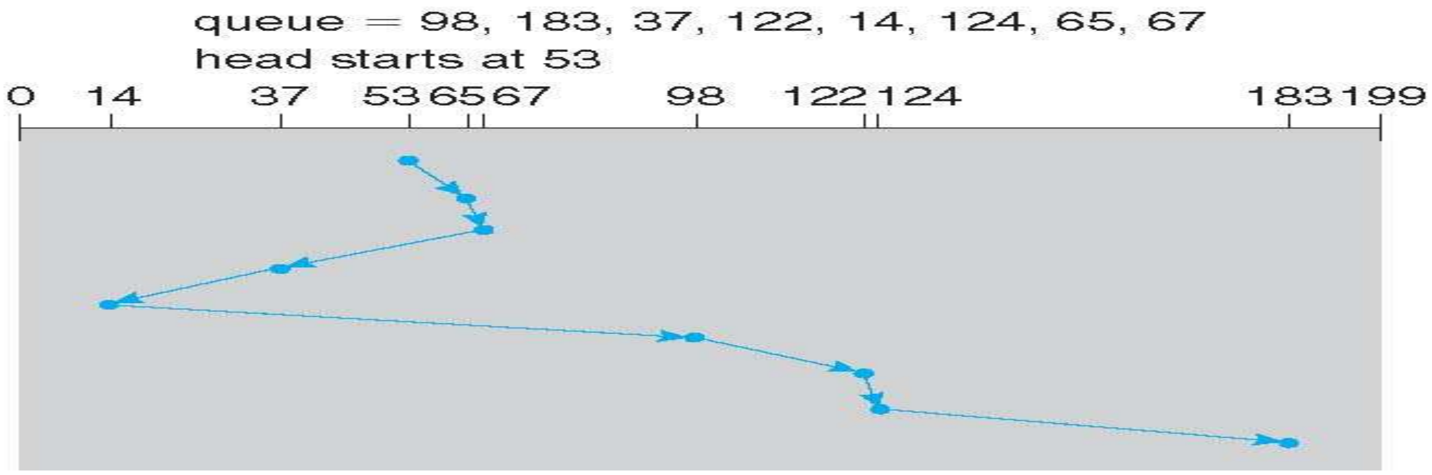
SSTF

- If Requests are for cylinders 14 and 186??
- While the request from 14 is being serviced, a new request near 14 arrives.
  - This new request will be serviced next, **making the request at 186 wait.**
- While this request is being serviced, another request close to 14 could arrive.
  - A continual stream of requests near one another could cause the request for **cylinder 186 to wait indefinitely.**

SSTF

- Substantial improvement over the FCFS algorithm
- **Not optimal**

- We can do better by moving the head
  - from 53 to 37, **even though the latter is not closest**, and
  - then to 14, before turning around to service 65, 67, 98, 122, 124, and 183.
  - This strategy **reduces the total** head movement to 208 cylinders.



SSTF

Advantages:

- Average Response Time decreases
  - **Very Good Average Response time**
- Throughput increases
  - **Excellent Throughput (More seeks are short)**

Disadvantages:

- **Overhead** to calculate **seek time in advance**
- Can **cause Starvation** for a request
- High variance of response time
  - **Intolerable Response time variance**
  - SSTF **favors only some requests**

SCAN

- The disk arm starts at one end of the disk, and moves toward the other end, servicing requests until it gets to the other end of the disk

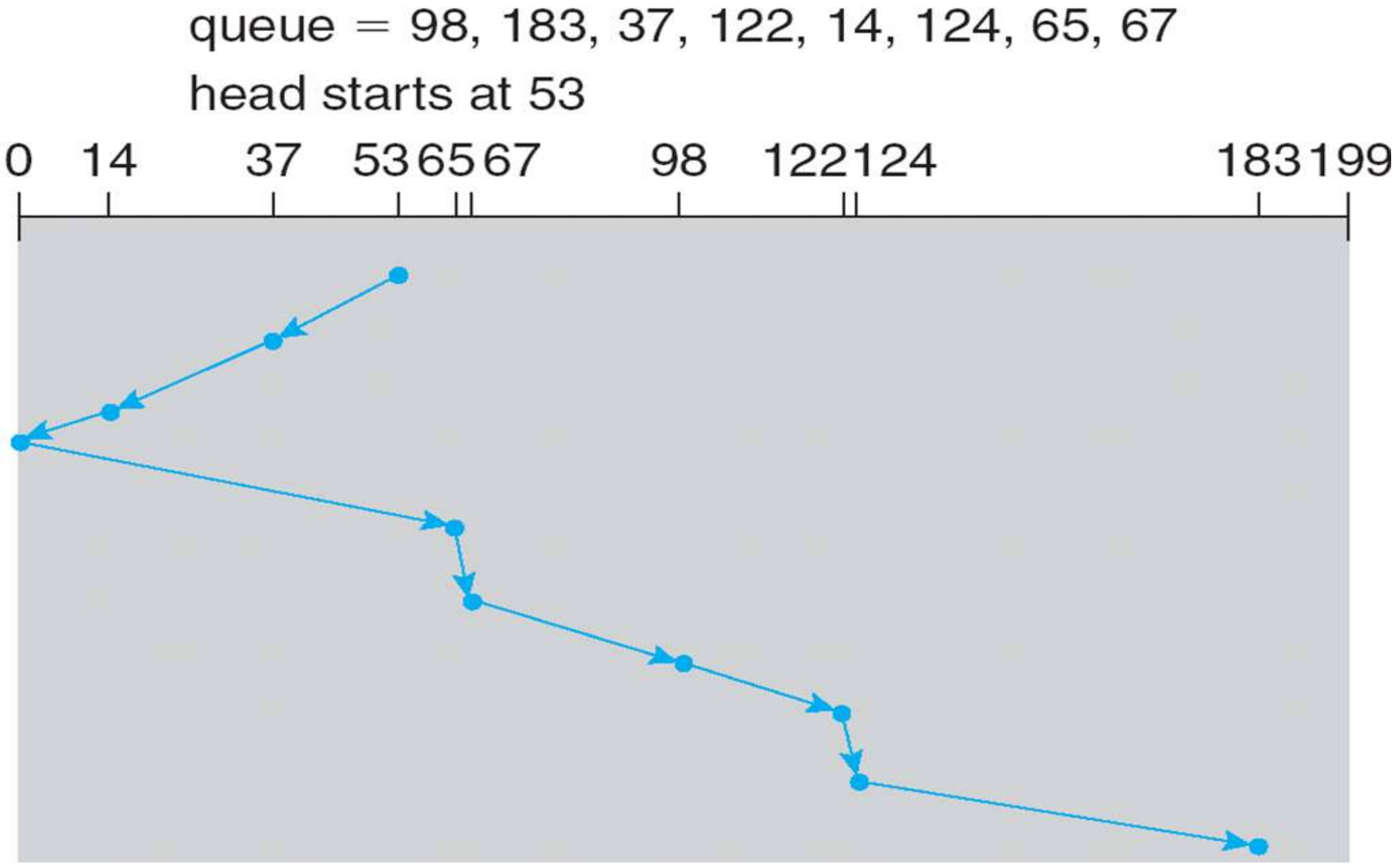
- At the other end, the direction of head movement is reversed and servicing continues.

- The head continuously scans back and forth across the disk.

SCAN

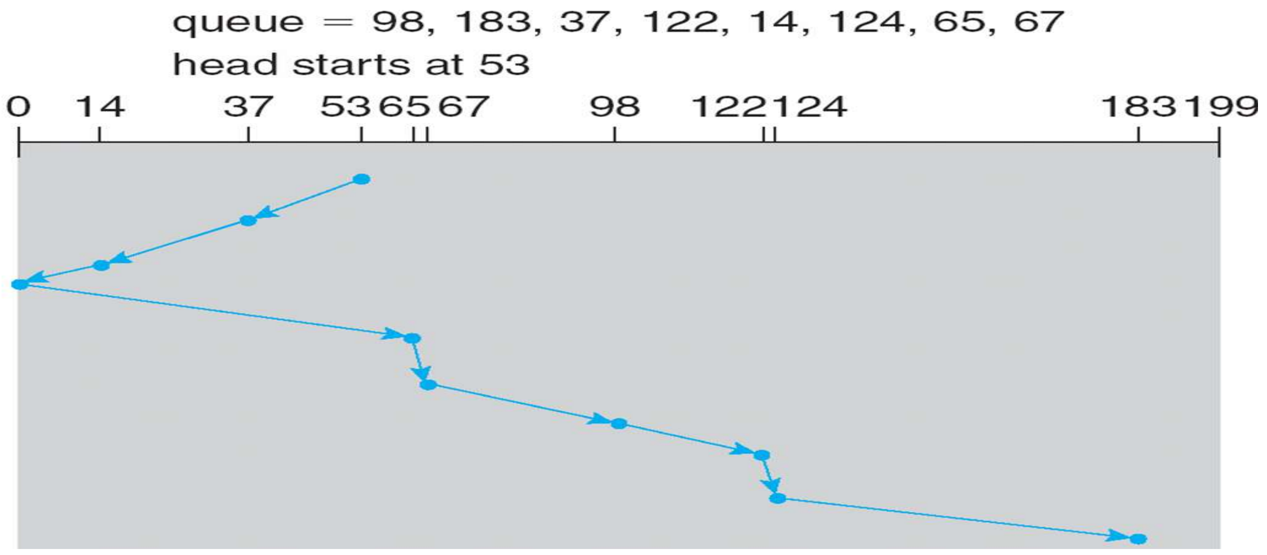
- **SCAN algorithm** Sometimes called the **elevator algorithm**
- Since the disk arm behaves just like an elevator in a building,
  - First servicing all the **requests going up** and
  - then reversing to **service requests the other way.**

SCAN (Cont.)


queue = 98, 183, 37, 122, 14, 124, 65, 67
head starts at 53
0
14
37
53
65
67
98
122
124
183
199

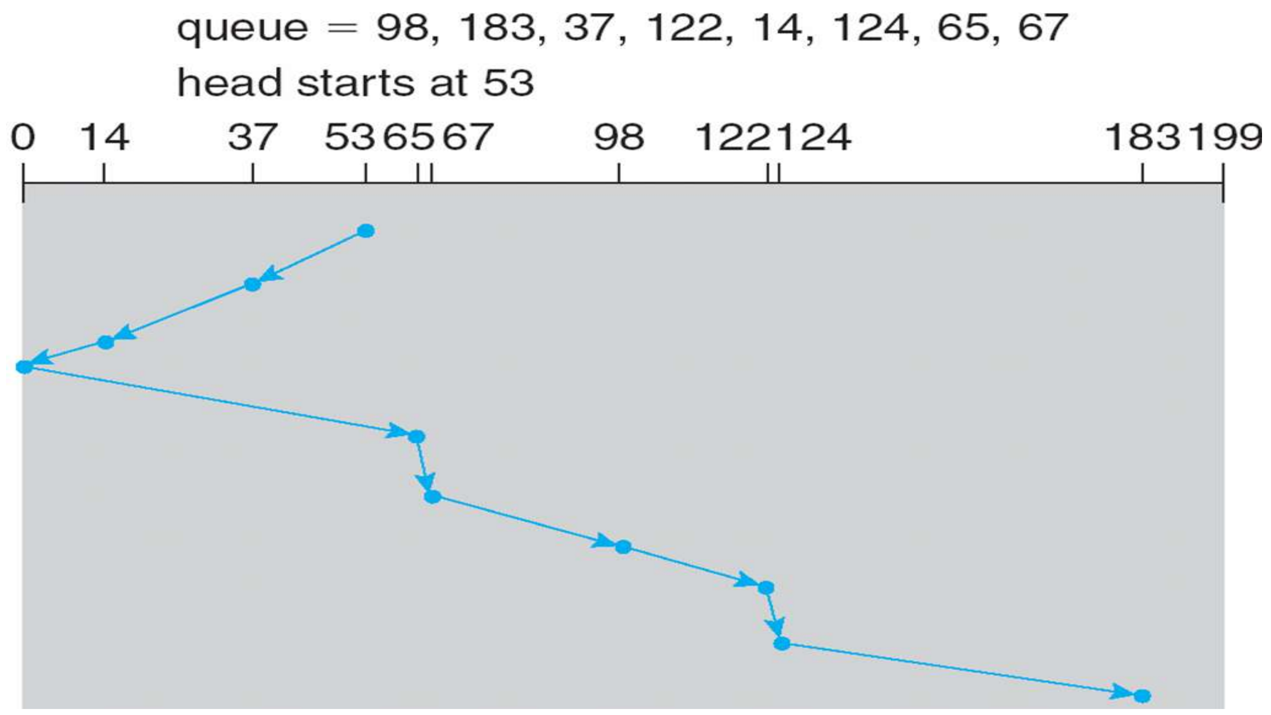
SCAN (Cont.)

- **Need to know the direction of head movement in addition to the head's current position, In Advance**



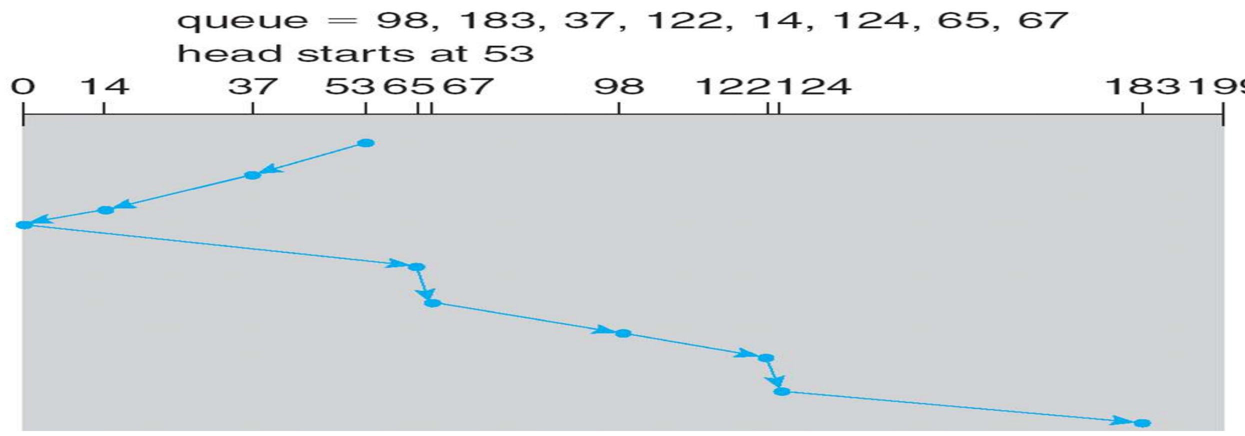
SCAN (Cont.)

- Assuming that the disk arm is moving toward 0 and that the initial head position is again 53,
- The head will next service 37 and then 14.
- At cylinder 0, the arm will reverse and will move toward the other end of the disk, servicing the requests at 65, 67, 98, 122, 124, and 183



SCAN (Cont.)

- If a request arrives in the queue just **in front of the head**,
  - it will be **serviced almost immediately;**
- A request arriving just **behind the head**
  - will have to **wait until the arm moves to the end of the disk, reverses direction**, and comes back.



SCAN

- Illustration shows total head movement of 208 cylinders
- **SSTF, 236 cylinders-**
- **FCFS, 640 cylinders**

SCAN

- But note that if requests are uniformly dense, largest density at other end of disk and those wait the longest
- ??

SCAN

- Assuming a uniform distribution of requests for cylinders,
- When the head reaches one end and reverses direction.
- At this point, **relatively few requests are immediately in front of the head, since these cylinders have recently been serviced.**
- The **heaviest density** of requests is at the **other end of the disk**
- **These requests have also waited the longest so why not go there first?**
- That is the idea of the next algorithm.

SCAN

Advantages:

- High throughput
- Low variance of response time
  - **Response time variance better than SSTF**
- Average response time
  - **Worse than SSTF, better than FCFS**

Disadvantages:

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

C-SCAN

- Circular Scan

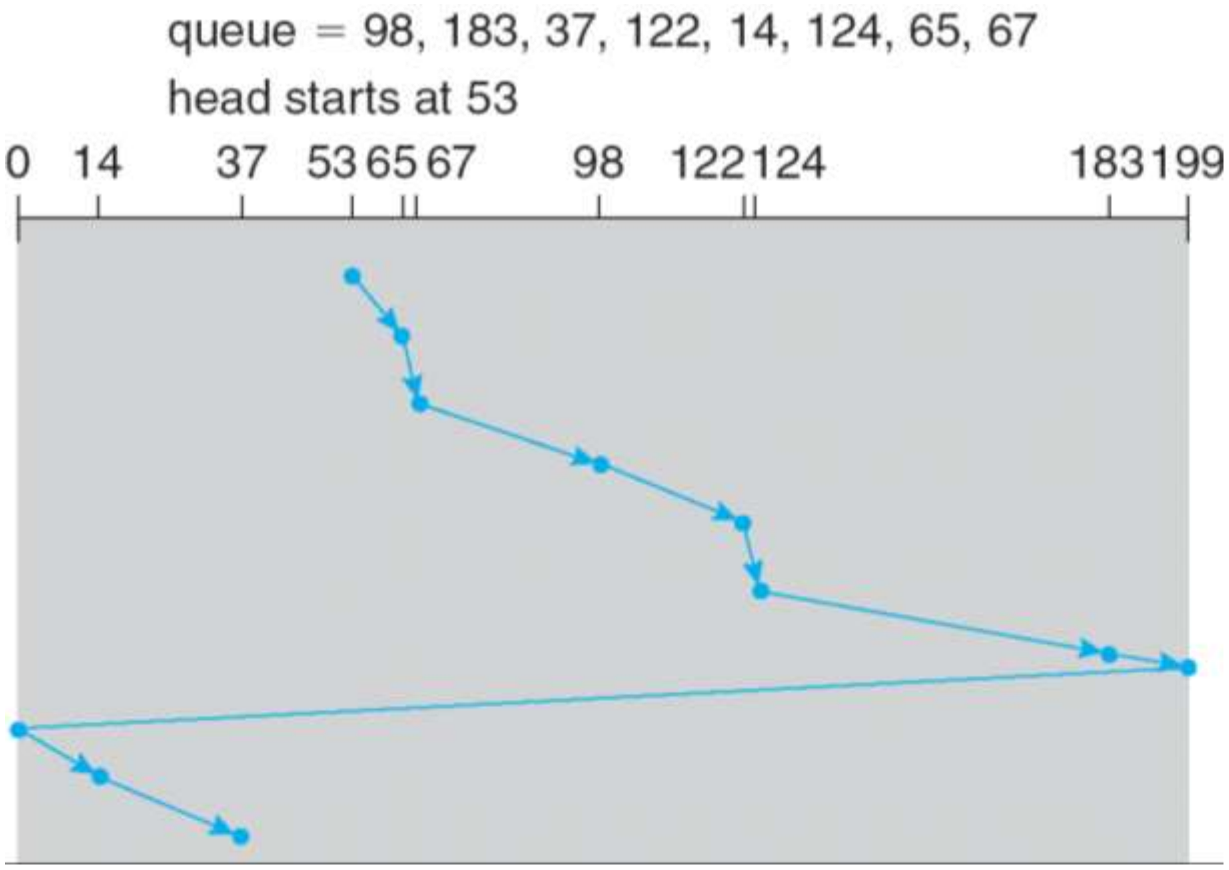
C-SCAN

- The head moves from **one end of the disk to the other**, servicing requests as it goes
  - When it reaches the other end, however, it **immediately returns to the beginning of the disk**, **without servicing any requests on the return trip**

C-SCAN

- Provides a more uniform wait time than SCAN
- Treats the cylinders as a **circular list that wraps around from the last cylinder to the first one**

C-SCAN (Cont.)


queue = 98, 183, 37, 122, 14, 124, 65, 67
head starts at 53
0
14
37
53
65
67
98
122
124
183
199

C-SCAN

Advantages:

- Provides more uniform wait time compared to SCAN

SCAN , C-SCAN

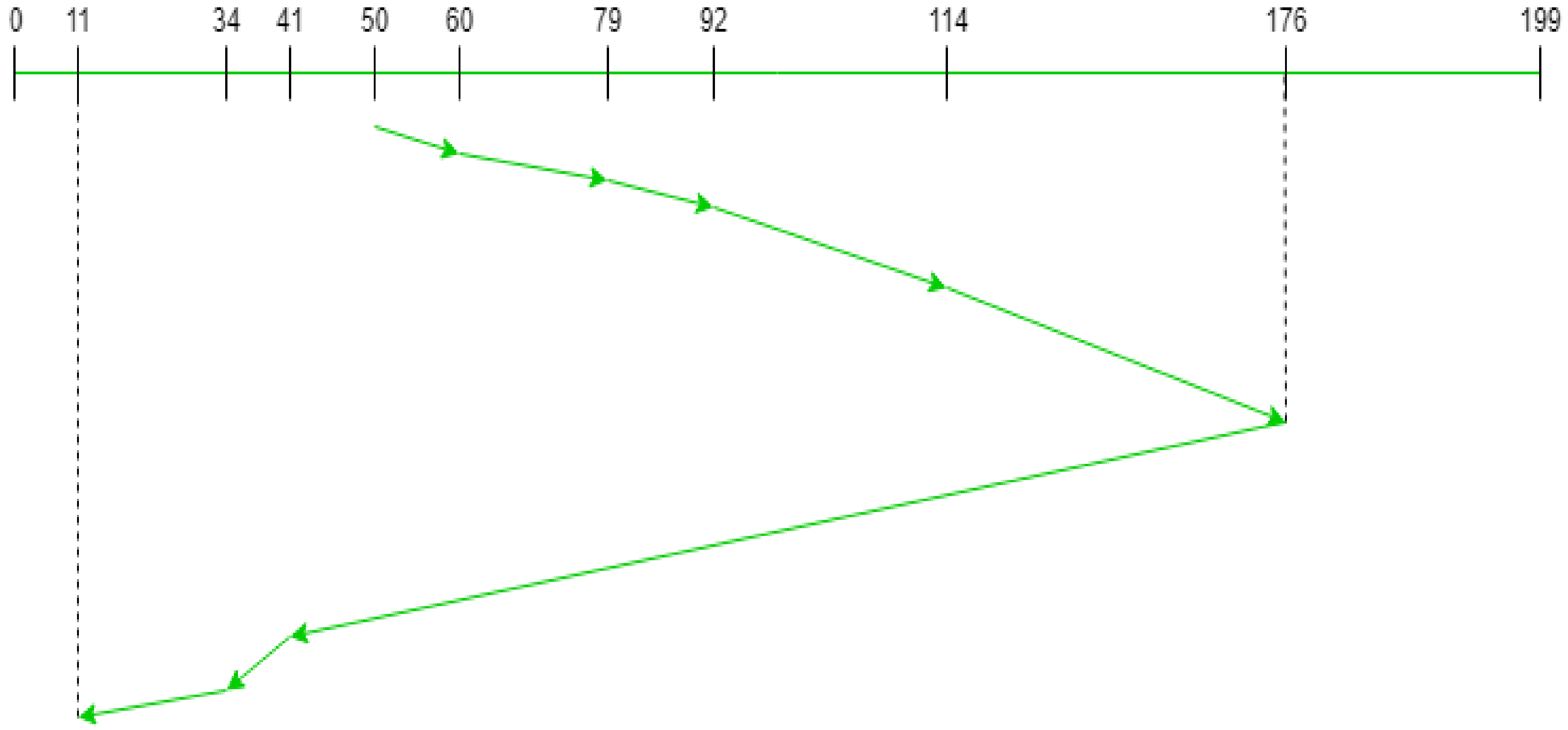
- Both SCAN and C-SCAN move the disk arm across the **full width of the disk**.
- In practice **neither algorithm is often implemented this way.**

LOOK, C-LOOK

- LOOK a version of SCAN
- C-LOOK a version of C-SCAN
- They ***look* for a request before continuing** to move in a given direction
- Arm only goes **as far as the last request in each direction**, then **reverses direction immediately, instead of going all the way to the end of the disk**

LOOK Algorithm

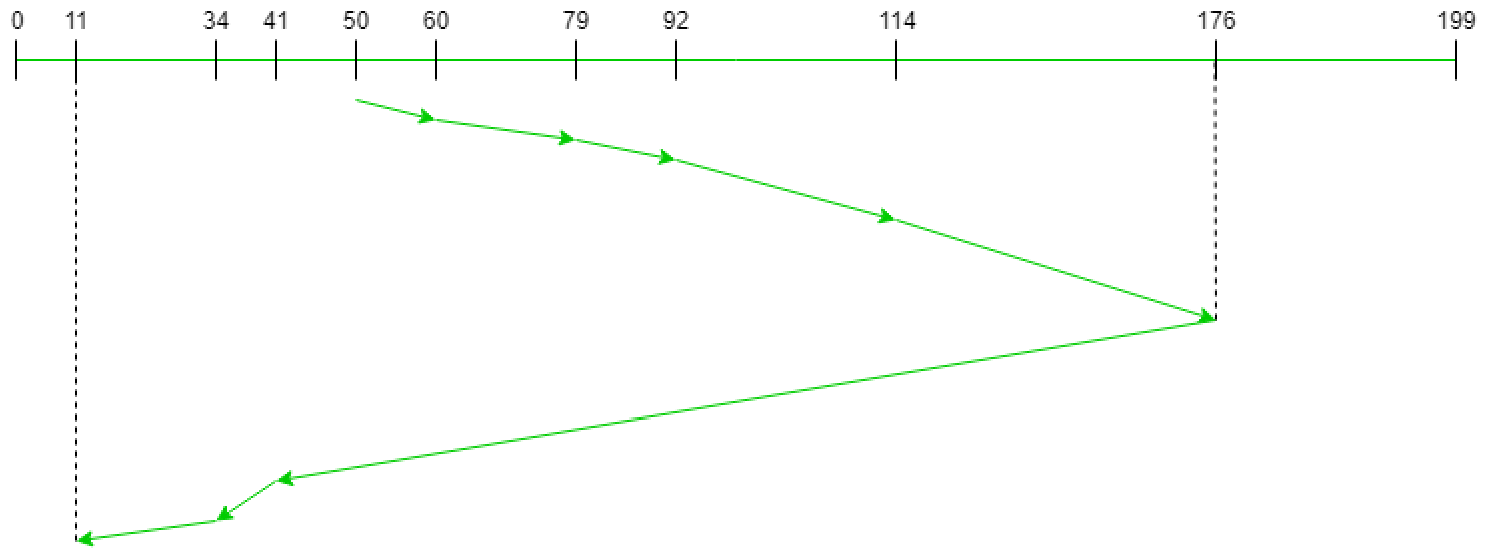
- Request sequence = {176, 79, 34, 60, 92, 11, 41, 114}
- Initial head position = 50
- Direction = right (We are moving from left to right)



- Seek Sequence is 60 79 92 114 176 41 34 11

LOOK Algorithm

- Request sequence = {176, 79, 34, 60, 92, 11, 41, 114}
- Initial head position = 50
- Direction = right (We are moving from left to right)

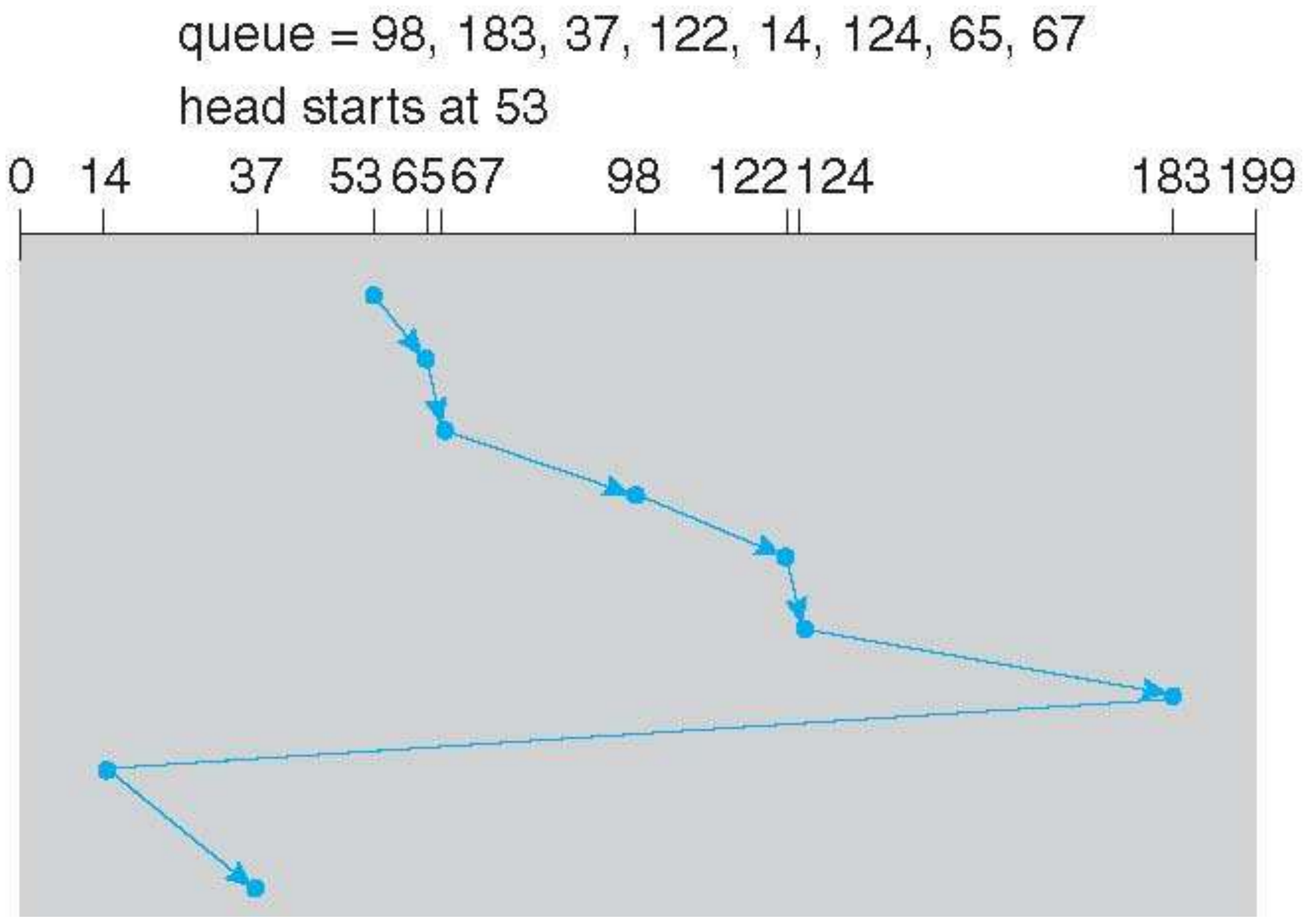


- Seek Sequence is 60 79 92 114 176 41 34 11
- Total seek count= (60-50)+(79-60)+(92-79)+(114-92)+(176-114)+(176-41)+(41-34)+(34-11)=291
- Total number of seek operations = 291

C-LOOK

- Circular Look

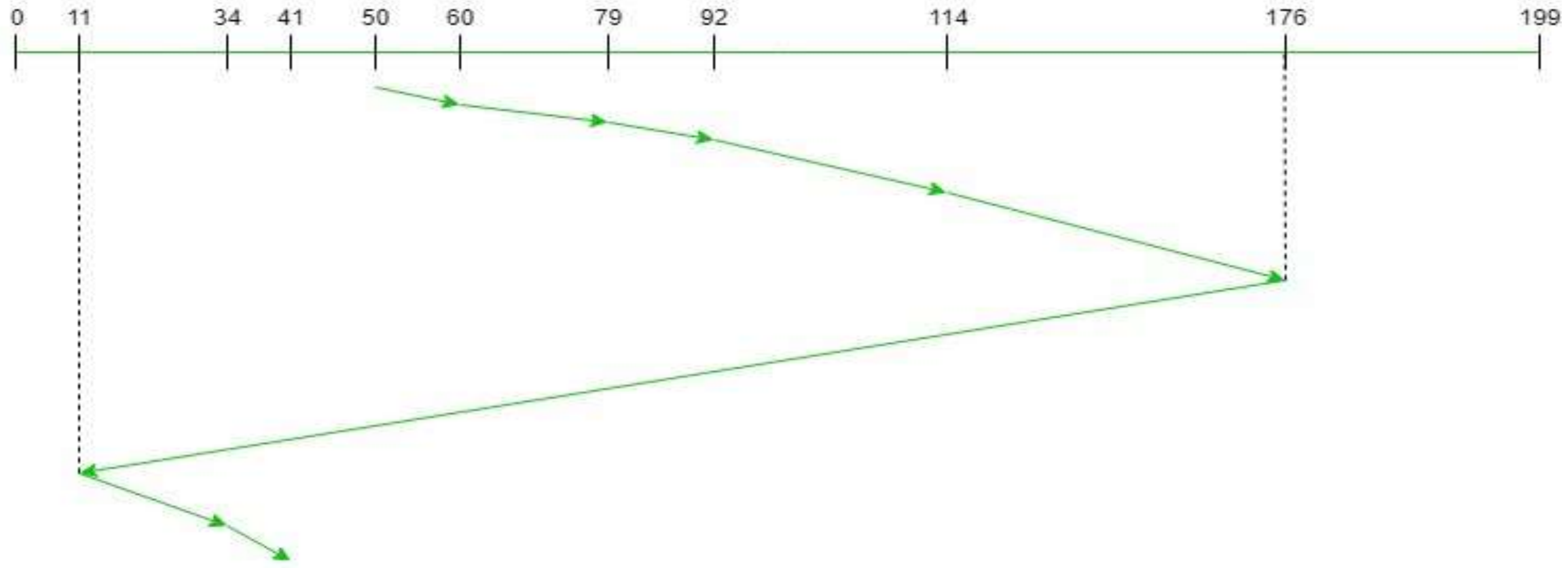
C-LOOK (Cont.)


queue = 98, 183, 37, 122, 14, 124, 65, 67
head starts at 53
0
14
37
53
65
67
98
122
124
183
199

C-LOOK Algorithm

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

- Initial head position = 50
- Direction = right (We are moving from left to right)



- Seek Sequence is 60 79 92 114 176 11 34 41
- The total seek count = (60 – 50) + (79 – 60) + (92 – 79) + (114 – 92) + (176 – 114) + (176 – 11) + (34 – 11) + (41 – 34) = 321
- Total number of seek operations = 321

LOOK,C-LOOK

LOOK-

- Improves Mean Response time and Variance

C-LOOK-

- Very popular

Selecting a Disk-Scheduling Algorithm

- SSTF is common and has a **natural appeal**
- SCAN and C-SCAN perform better for systems that place a **heavy load on the disk**
  - **Less starvation**
- Performance depends on the number and types of requests
- Requests for disk service can be **influenced by the file-allocation method**
  - **And metadata layout**

Selecting a Disk-Scheduling Algorithm

- The disk-scheduling algorithm should be written as a **separate module** of the operating system, **allowing it to be replaced with a different algorithm if necessary**
- **Either SSTF or LOOK is a reasonable choice for the default algorithm**
- What about rotational latency?
  - Difficult for OS to calculate
- How does disk-based queueing effect OS queue ordering efforts?

ISRO | ISRO CS 2018 | Question 68

Disk requests come to a disk driver for cylinders in the order 10, 22, 20, 2, 40, 6, and 38 at a time when the disk drive is reading from cylinder 20. The seek time is 6 ms/cylinder. The total seek time, if the disk arm scheduling algorithms is first-come-first-served is

(A) 360 ms
(B) 850 ms
(C) 900 ms
(D) None of the above

ISRO | ISRO CS 2018 | Question 68

Disk requests come to a disk driver for cylinders in the order 10, 22, 20, 2, 40, 6, and 38 at a time when the disk drive is reading from cylinder 20. The seek time is 6 ms/cylinder. The total seek time, if the disk arm scheduling algorithms is first-come-first-served is

(A) 360 ms
(B) 850 ms
(C) 900 ms
(D) None of the above

Answer: (D)

Explanation: FCFS
Total seek time in FCFS Scheduling when the disk drive is reading from cylinder 20 for cylinders in the order 10, 22, 20, 2, 40, 6, and 38 :

= (10 + 12 + 2 + 18 + 38 + 34 + 32)*6 = 146*6 = 876 ms

As no other option matches the answer, option (D) is correct.

GATE | GATE-CS-2014-(Set-1) | Question 65

Suppose a disk has 201 cylinders, numbered from 0 to 200. At some time the disk arm is at cylinder 100, and there is a queue of disk access requests for cylinders 30, 85, 90, 100, 105, 110, 135 and 145. If Shortest-Seek Time First (SSTF) is being used for scheduling the disk access, the request for cylinder 90 is serviced after servicing _____________ number of requests.

(A) 1
(B) 2
(C) 3
(D) 4

GATE | GATE-CS-2014-(Set-1) | Question 65

Suppose a disk has 201 cylinders, numbered from 0 to 200. At some time the disk arm is at cylinder 100, and there is a queue of disk access requests for cylinders 30, 85, 90, 100, 105, 110, 135 and 145. If Shortest-Seek Time First (SSTF) is being used for scheduling the disk access, the request for cylinder 90 is serviced after servicing ____________ number of requests.

(A) 1
(B) 2
(C) 3
(D) 4

Answer: (C)
Explanation: The disk will service that request first whose cylinder number is closest to its arm. Hence 1st serviced request is for cylinder no 100 (as the arm is itself pointing to it), then 105, then 110, and then the arm comes to service request for cylinder 90. Hence before servicing request for cylinder 90, the disk would had serviced 3 requests.

Hence option C.

For I/O Operations

- **Unit of transfer:**
- Data may be transferred as a stream of bytes or characters(e.g., terminal I/O) or in larger blocks (e.g., disk I/O).

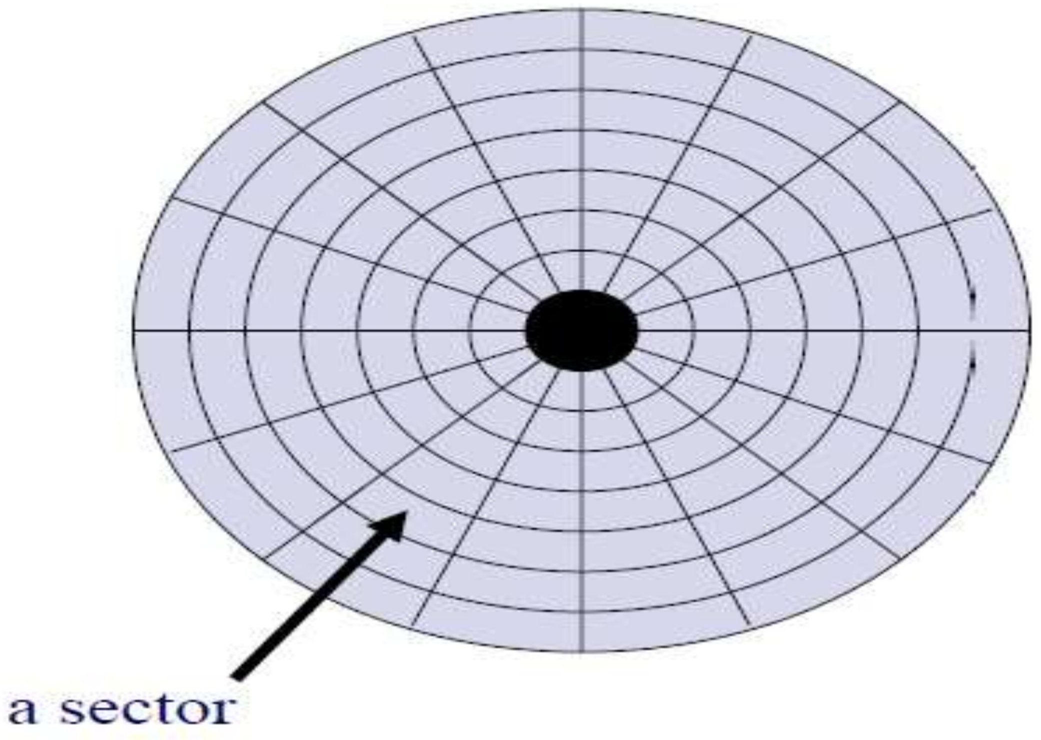
Disk Management

- The operating system is responsible for several other aspects of disk management like
  – disk initialization
  – disk formatting
  – booting from disk, and
  – bad-block recovery

Disk Management

Low-level formatting, or **physical formatting** —

- Dividing a disk into sectors that the disk controller can read and write



Disk Management

Low-level formatting, or **physical formatting** —

- Each sector can hold
- header information,
- plus data,
- plus error correction code (**ECC**)

Disk Management

Low-level formatting, or **physical formatting** —

- Disk controller handles how many bytes of data space to leave between the header and trailer of all sectors.
- It is usually possible to choose among a few sizes, such as 256,512, and 1,024 bytes.
- Usually 512 bytes of data

Disk Management

Low-level formatting, or **physical formatting** —

- Enables the manufacturer to test the disk
- To initialize the mapping from logical block numbers to defect-free sectors on the disk.

Disk Management

Partitioning

- To use a disk to hold files, the operating system still needs to record its own data structures on the disk
  - **Partition** the disk into one or more groups of cylinders, each treated as a logical disk

- The operating system can treat each partition as though it were a separate disk.
  - One partition can hold a copy of the operating system's executable code
  - Another holds user files

Disk Management

Logical formatting

- "making a file system"
- "creation of a file system"
- The OS stores the initial file-system data structures onto the disk.
- These data structures may include
  - maps of free and allocated space (a FAT or inodes)
  - an initial empty directory

Disk Management (Cont.)

Bad Blocks

- Methods such as **sector sparing** used to handle bad blocks

Disk Management (Cont.)

Bad Blocks

- Because disks have moving parts and small tolerances
- As the disk head flies just above the disk surface,
- They are prone to failure

1) If the failure is complete;
   - disk needs to be replaced and its contents restored from backup media to the new disk.
2) More frequently, one or more sectors become defective.

Disk Management (Cont.)

Bad Blocks

- On simple disks, such as some disks with IDE controllers, bad blocks are handled manually.

- More sophisticated disks, used in high-end PCs are smarter about bad-block recovery.
  - The controller maintains a list of bad blocks on the disk.
  - The list is initialized during the low-level formatting at the factory and is updated over the life of the disk.

Disk Management (Cont.)

Bad Blocks

- Low-level formatting also sets aside spare sectors not visible to the operating system.

- 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

GATE | GATE CS 2008 | Question 31

For a magnetic disk with concentric circular tracks, the seek latency is not linearly proportional to the seek distance due to

(A) non-uniform distribution of requests

(B) arm starting and stopping inertia

(C) higher capacity of tracks on the periphery of the platter

(D) use of unfair arm scheduling policies

GATE | GATE CS 2008 | Question 31

For a magnetic disk with concentric circular tracks, the seek latency is not linearly proportional to the seek distance due to

(A) non-uniform distribution of requests

(B) arm starting and stopping inertia

(C) higher capacity of tracks on the periphery of the platter

(D) use of unfair arm scheduling policies

Answer: (B)

Explanation: Whenever head moves from one track to other then its speed and direction changes, which is noting but change in motion or the case of inertia. So answer B