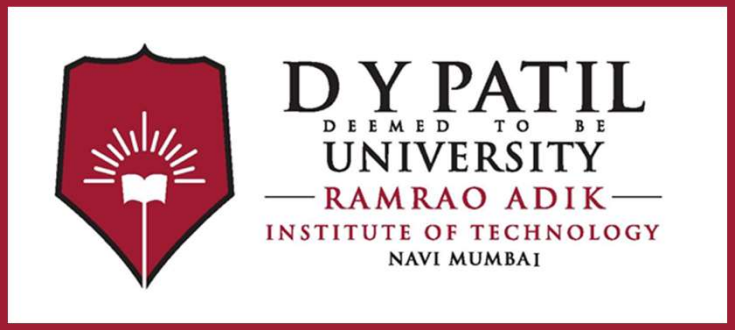


Subject Name: Operating Systems

Unit: 6 Unit Name: Input /Output Management

Faculty Name: Ms. Puja Padiya

Index

Lecture No: 37

Disk Organization and I/O Management

•Since main memory is usually too small to accommodate all the data and programs permanently, the computer system must provide secondary storage to back up main memory.

• Modern computer systems use **disks** as the **primary on-line storage** medium for information (both programs and data).

•The **file system** provides the mechanism for on-line storage of and access to both data and programs residing on the disks.

•A file is a collection of **related information** defined by its creator.

•The files are mapped by the operating system onto physical devices.

•Files are normally organized into directories for ease of use.

The devices that attach to a computer vary in many aspects.

Some devices transfer a character or a block of characters at a time.

Some can be accessed only sequentially, others randomly.

Some transfer data synchronously, others asynchronously.

Some are dedicated, some shared.

They can be read-only or read–write.

They vary greatly in speed.

In many ways, they are also the slowest major component of the computer.

Because devices are a performance bottleneck, another key is to optimize I/O for maximum concurrency.

CHAPTER OBJECTIVES

CHAPTER OBJECTIVES

- To describe the physical structure of secondary storage
- To explain the performance characteristics of mass-storage devices.
- To evaluate disk scheduling algorithms.

PHYSICAL STRUCTURE OF SECONDARY STORAGE DEVICES

Magnetic disks provide the bulk of secondary storage for modern computer systems. Conceptually, disks are relatively simple

Each disk platter has a flat circular shape, like a CD.

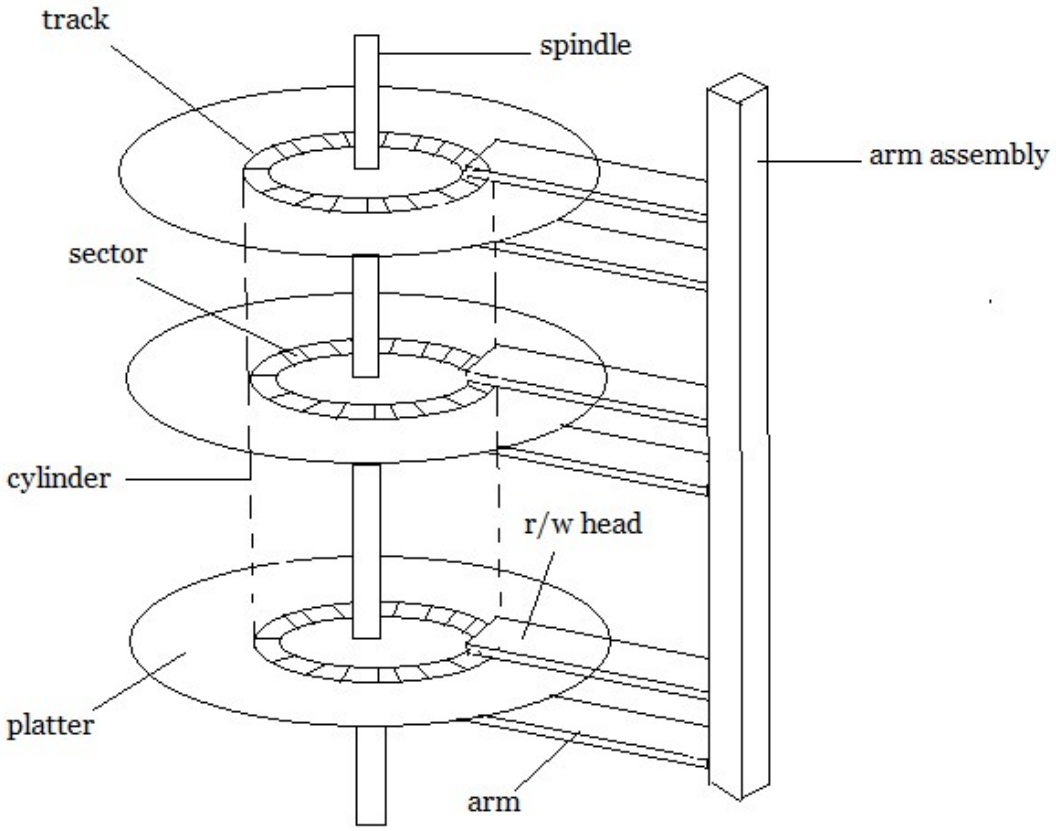
Common platter diameters range from 1.8 to 3.5 inches.

The two surfaces of a platter are covered with a magnetic material.

We store information by recording it magnetically on the platters.

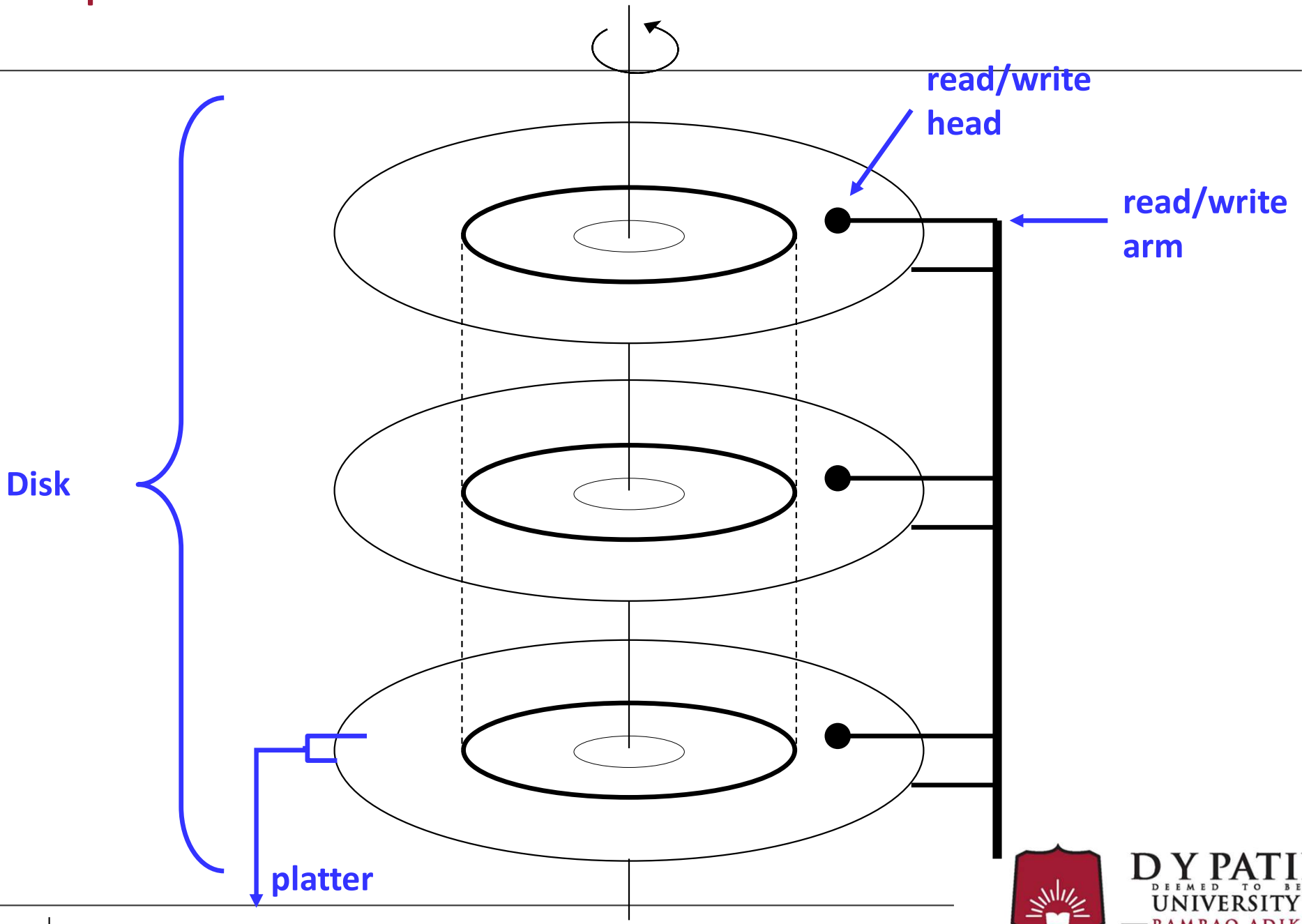
Magnetic Disk Structure

- In modern computers, most of the secondary storage is in the form of magnetic disks. Hence, knowing the structure of a magnetic disk is necessary to understand how the data in the disk is accessed by the computer.



Structure of a magnetic disk

Multi-platter Disk



A read – write head “flies” just above each surface of every platter.

The heads are attached to a disk arm that moves all the heads as a unit.

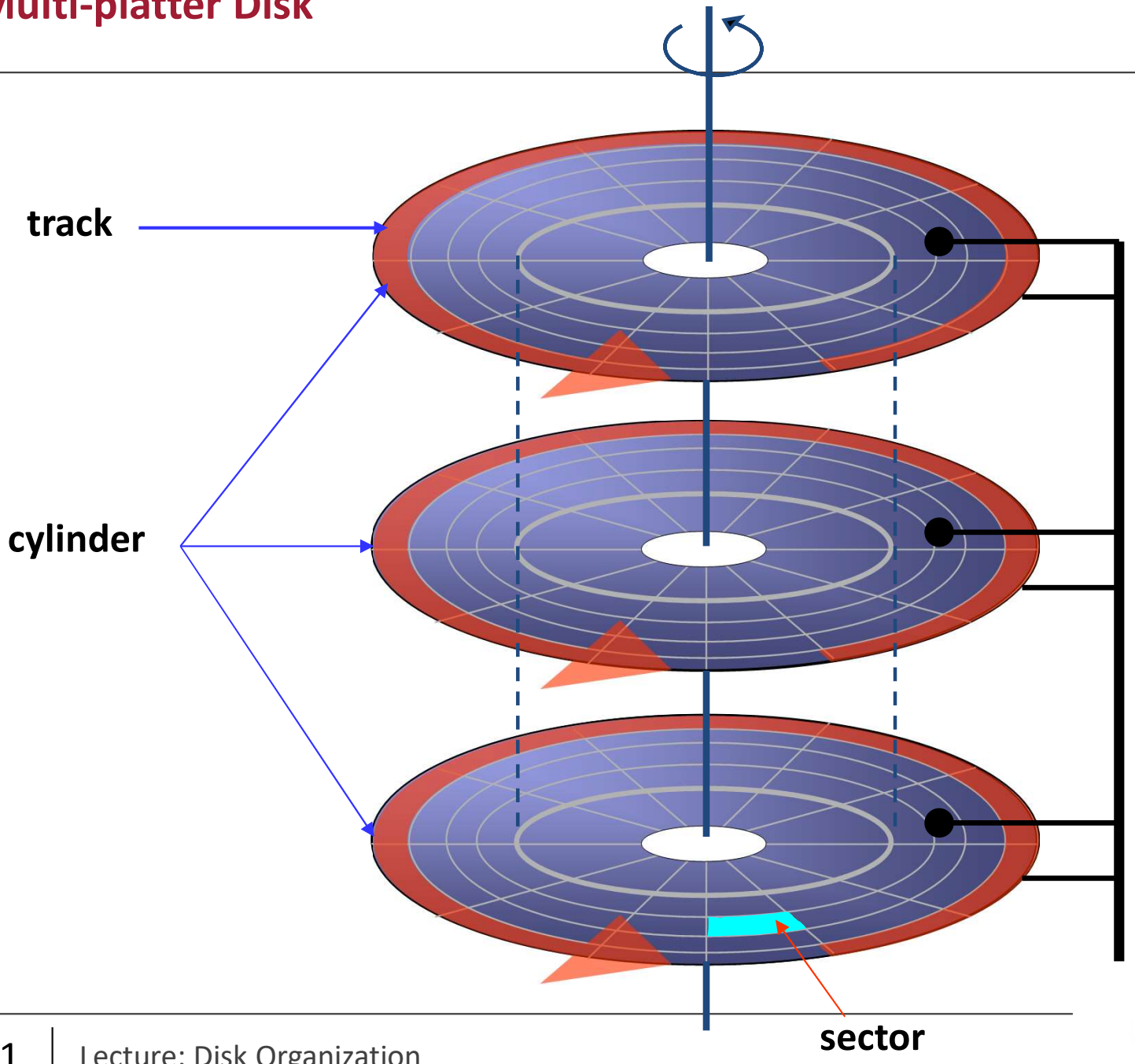
The surface of a platter is logically divided into circular tracks, which are subdivided into sectors.

The set of tracks that are at one arm position makes up a cylinder.

There may be thousands of concentric cylinders in a disk drive, and each track may contain hundreds of sectors.

The storage capacity of common disk drives is measured in gigabytes.

Multi-platter Disk



- When the disk is in use, a drive motor spins it at high speed. Most drives
- rotate 60 to 250 times per second, specified in terms of rotations per minute
- (RPM).
- Common drives spin at 5,400, 7,200, 10,000, and 15,000 RPM.
- Disk speed has two parts.
- **The transfer rate** is the rate at which data flow between the drive and the computer.
- **The positioning time or random-access time**, consists of two parts:
  - The seek time: the time necessary to move the disk arm to the desired cylinder
  - The rotational latency.: the time necessary for the desired sector to rotate to the disk head
- Typical disks can transfer several megabytes of data per second, and they have seek times and rotational latencies of several milliseconds.

Head Crash

- Because the disk head flies on an extremely thin cushion of air (measured
- in microns), there is a danger that the head will make contact with the disk
- surface.
- Although the disk platters are coated with a thin protective layer, the head will sometimes damage the magnetic surface.
- This accident is called a **head crash**.
- A head crash normally **cannot be repaired**; the entire disk must be replaced.
- A disk can be removable, allowing different disks to be mounted as needed.

Controllers

A disk drive is attached to a computer by a set of wires called an I/O

bus.

Several kinds of buses are available, including advanced technology

attachment (ATA), serial ATA (SATA), eSATA, universal serial bus (USB),and

fibre channel (FC).

The data transfers on a bus are carried out by special electronic processors called controllers.

Thehost controller is the controller at the computer end of the bus.

A disk controller is built into each disk drive.

Disk I/O operation

To perform a disk I/O operation, the computer places a command into the host controller, typically using memory-mapped I/O ports

The host controller then sends the command via messages to the disk Controller

The disk controller operates the disk-drive hardware to carry out the command. Disk controllers usually have a built-in cache.

Data transfer at the disk drive happens between the cache and the disk surface, and data transfer to the host, at fast electronic speeds, occurs between the cache and the host controller.

Positioning the Read Write Head

- The actual details of disk I/O operation depend on the computer system, the operating system, and the nature of the I/O channel and disk controller hardware.

- When the disk drive is operating, the disk is rotating at constant speed. To read or write, the head must be positioned at the desired track and at the beginning of the desired sector on that track. Track selection involves moving the head in a movable head system or electronically selecting one head on a fixed-head system. On a movable-head system, the time it takes to position the head at the track is known as seek time.

Disk Performance Parameters

- The actual details of disk I/O operation depend on the computer system, the operating system, and the nature of the I/O channel and disk controller hardware.

- When the disk drive is operating, the disk is rotating at constant speed.To read or write, the head must be positioned at the desired track and at the beginning of the desired sector on that track. Track selection involves moving the head in a movablehead system or electronically selecting one head on a fixed-head system. On a movable-head system, the time it takes to position the head at the track is known as seek time.

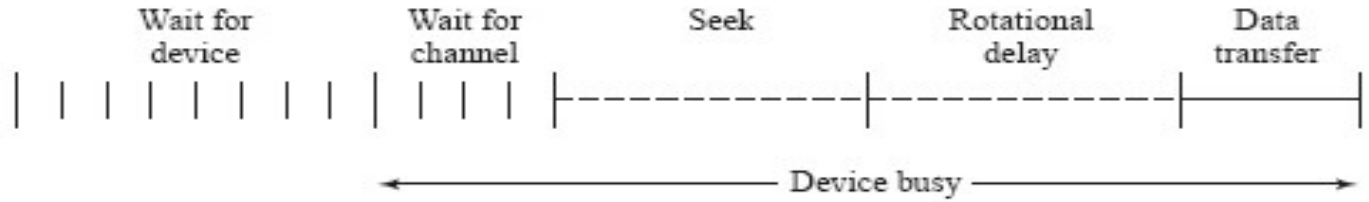
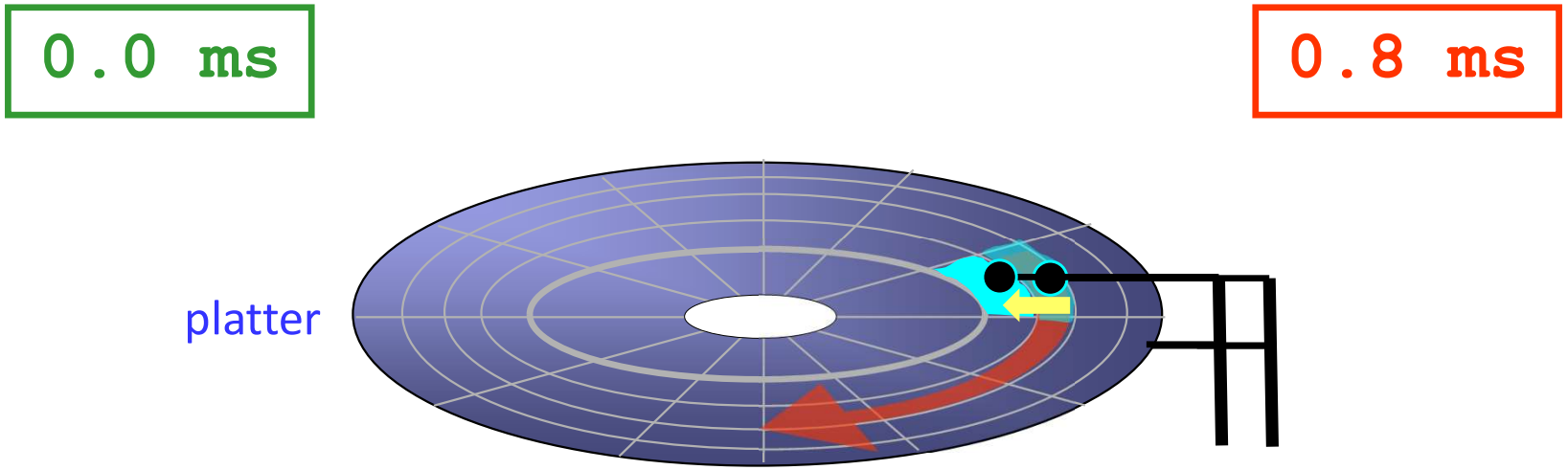


Figure 11.6 Timing of a Disk I/O Transfer

Disk Performance Parameters

- ***Access Time*** is the sum of:
  - ***Seek time:*** The time it takes to position the head at the desired track
  - ***Rotational delay*** or ***rotational latency:*** The time its takes for the beginning of the sector to reach the head
- ***Transfer Time*** is the time taken to transfer the data.

Seek Time

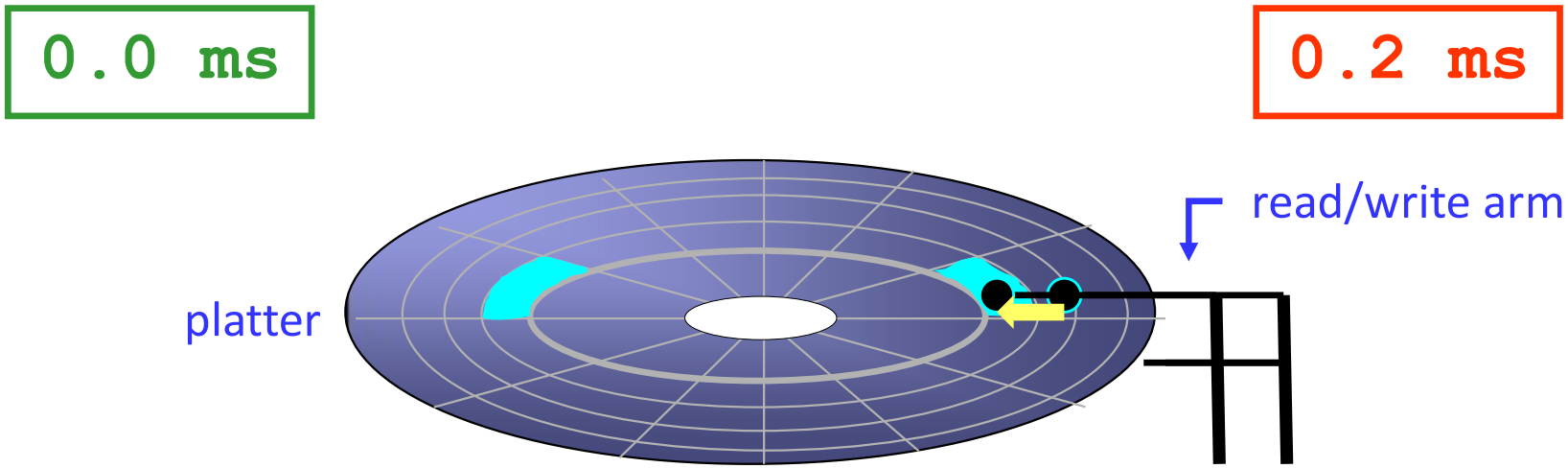


- measures the amount of time required for the read/write heads to move between tracks over the surface of the platters

$$\text{Average Seek Time} = \frac{sum\ of\ the\ time\ of\ all\ possible\ seeks}{\#\ of\ all\ posssible\ seeks}$$

Rotational delay

- After the read/write arm has picked the right track, the time it takes for the requested sector (on that track) to come under the read/write head



Other measurements

- **Avg. seek time** = $$\frac{sum\ of\ the\ time\ of\ all\ possible\ seeks}{\#\ of\ all\ posssible\ seeks}$$

- **Avg. Rotation latency / rotational delay** = latency halfway around the disk

- **Transfer time** = the time it takes to transfer a block of bits, typically a sector, under the read/write head
  - *Function of the size, rotational speed, recording density of a track, speed of electronics connecting disk to computer*

- **Controller Overhead time** = the overhead the controller imposes in performing I/O access

Example taken from: Computer architecture by J. Hennessy and D. Patterson

What is avg. time to read or write (access time) a 512-byte sector for a typical disk?

Avg. seek time (ST) = 5ms

Transfer time (TT) = 40 MB/sec

Rotational delay (RD) = 10,000 RPM

Controller overhead (CO) = 0.1 ms

Assume the disk is idle and so no queuing delay

dat = ST + RD + TT + CO

= 5ms + (0.5/10,000 rpm) + (0.5KB/40.0 MB/sec) + 0.1 ms

= 5.0 + 3.0 +0.013 + 0.1 = **8.11 ms**

Assuming the measured seek time is 33% of the calculated average. So we obtain:

dat = 1.67 ms + 3.0 ms + 0.013 ms + 0.1 ms = 4.783 ms

Note that 0.013/4.783 = 0.3% is the disk transferring data in this example

Lecture No: 38

Disk Scheduling algorithm: FCFS, SSTF, SCAN

Disk Scheduling

- **Access Time:** seek time + rotational delay + transfer time + controller overhead
- **Transfer Time:** time to copy the block to and from the disk
- **Disk Bandwidth:** total number of bytes transferred, divided by the total time between the first request for service and the completion of the last transfer
- **Seek Time:** the for the disk arm to move the heads to the cylinder containing the desired sector.
- **Rotational Delay:** time waiting for the disk to rotate the desired sector to the disk head

Disk Scheduling Policies

- Consider the typical situation in a multiprogramming environment, in which the operating system maintains a queue of requests for each I/O device.
- So, for a single disk, there will be a number of I/O requests (reads and writes) from various processes in the queue.
- If we selected items from the queue in random order, then we can expect that the tracks to be visited will occur randomly, giving poor performance.
- This random scheduling is useful as a benchmark against which to evaluate other techniques.

Purpose of Disk Scheduling

- **Disk Head Scheduler:** After the completion of the current operation, the *disk head scheduler* examines the current contents of the queue and decides which request to be served next.
- The Disk Head Scheduler must consider **3 important factors**
- *overall performance* of the disk
- .*fairness* in treating processes
- *cost* of executing scheduling algorithm

The main purpose of disk scheduling algorithm is to select a disk request from the queue of IO requests and decide the schedule when this request will be processed.

Goal of Disk Scheduling Algorithm

- Fairness
- High throughout
- Minimal traveling head time

Disk Scheduling Algorithms

- The list of various disks scheduling algorithm is given below. Each algorithm is carrying some advantages and disadvantages. The limitation of each algorithm leads to the evolution of a new algorithm.

1. **FCFS scheduling algorithm**
2. **SSTF (shortest seek time first) algorithm**
3. **SCAN scheduling**
4. **C-SCAN scheduling**
5. **LOOK Scheduling**
6. **C-LOOK scheduling**

First-In-First-Out

- The simplest form of scheduling is first-in-first-out (FIFO) scheduling, which processes items from the queue in sequential order. This strategy has the advantage of being fair, because every request is honored and the requests are honored in the order received.

- With FIFO, if there are only a few processes that require access and if many of the requests are to clustered file sectors, then we can hope for good performance. However, this technique will often approximate random scheduling in performance, if there are many processes competing for the disk.

FCFS Disk Scheduling Algorithm-

As the name suggests, this algorithm entertains requests in the order they arrive in the disk queue.
It is the simplest disk scheduling algorithm.

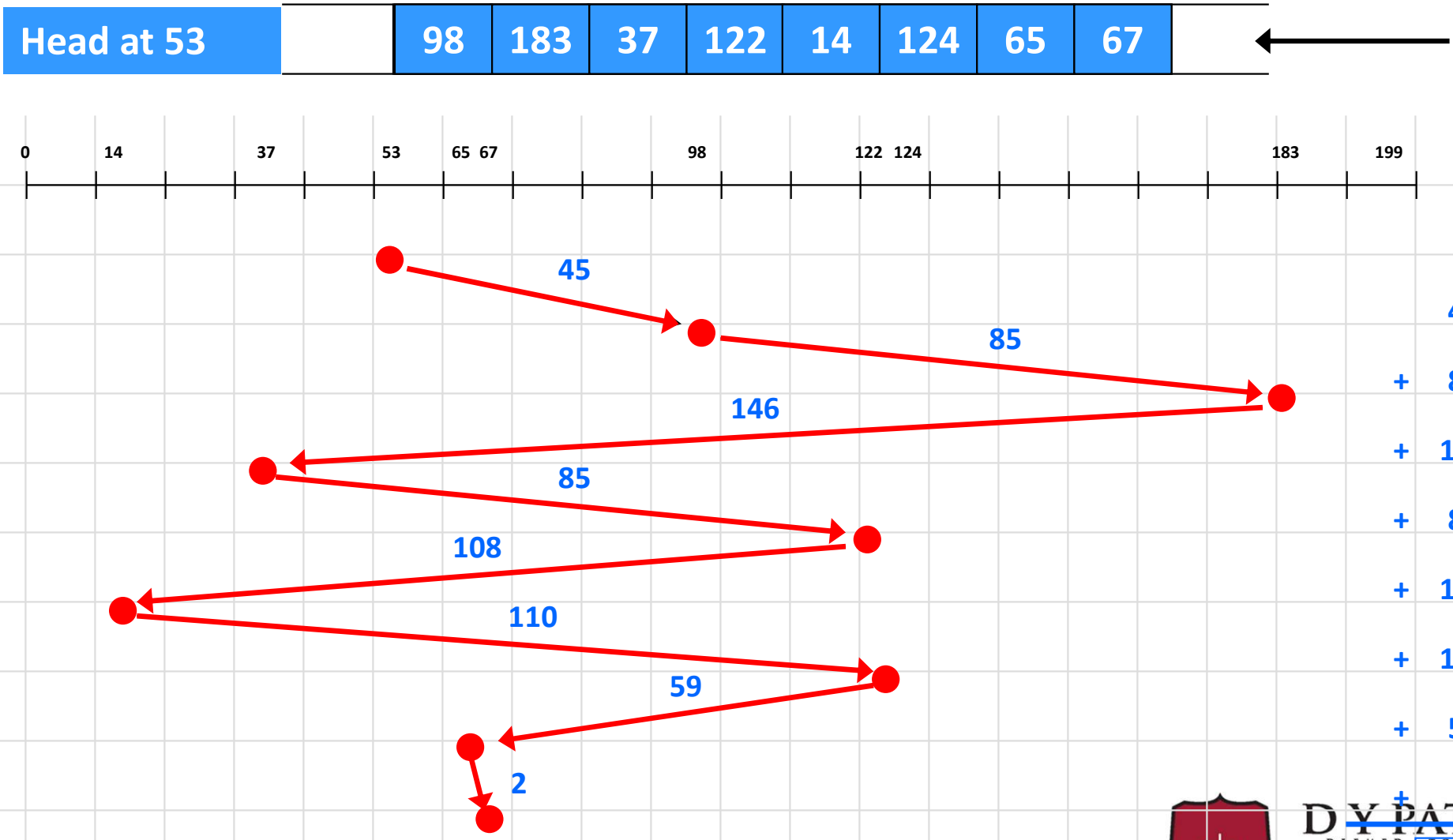
Advantages-

It is simple, easy to understand and implement.
It does not cause starvation to any request.

Disadvantages-

It results in increased total seek time.
It is inefficient.

Disk Scheduling FIFO



FCFS

Consider a disk queue with requests for I/O to blocks on cylinders 98, 183, 41, 122, 14, 124, 65, 67. The FCFS scheduling algorithm is used. The head is initially at cylinder number 53. The cylinders are numbered from 0 to 199. The total head movement (in number of cylinders) incurred while servicing these requests is_____

Total head movements incurred while servicing these requests
= (98 – 53) + (183 – 98) + (183 – 41) + (122 – 41) + (122 – 14) + (124 – 14) + (124 – 65) + (67 – 65)
= 45 + 85 + 142 + 81 + 108 + 110 + 59 + 2
=632

SSTF Disk Scheduling Algorithm-

SSTF stands for **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.

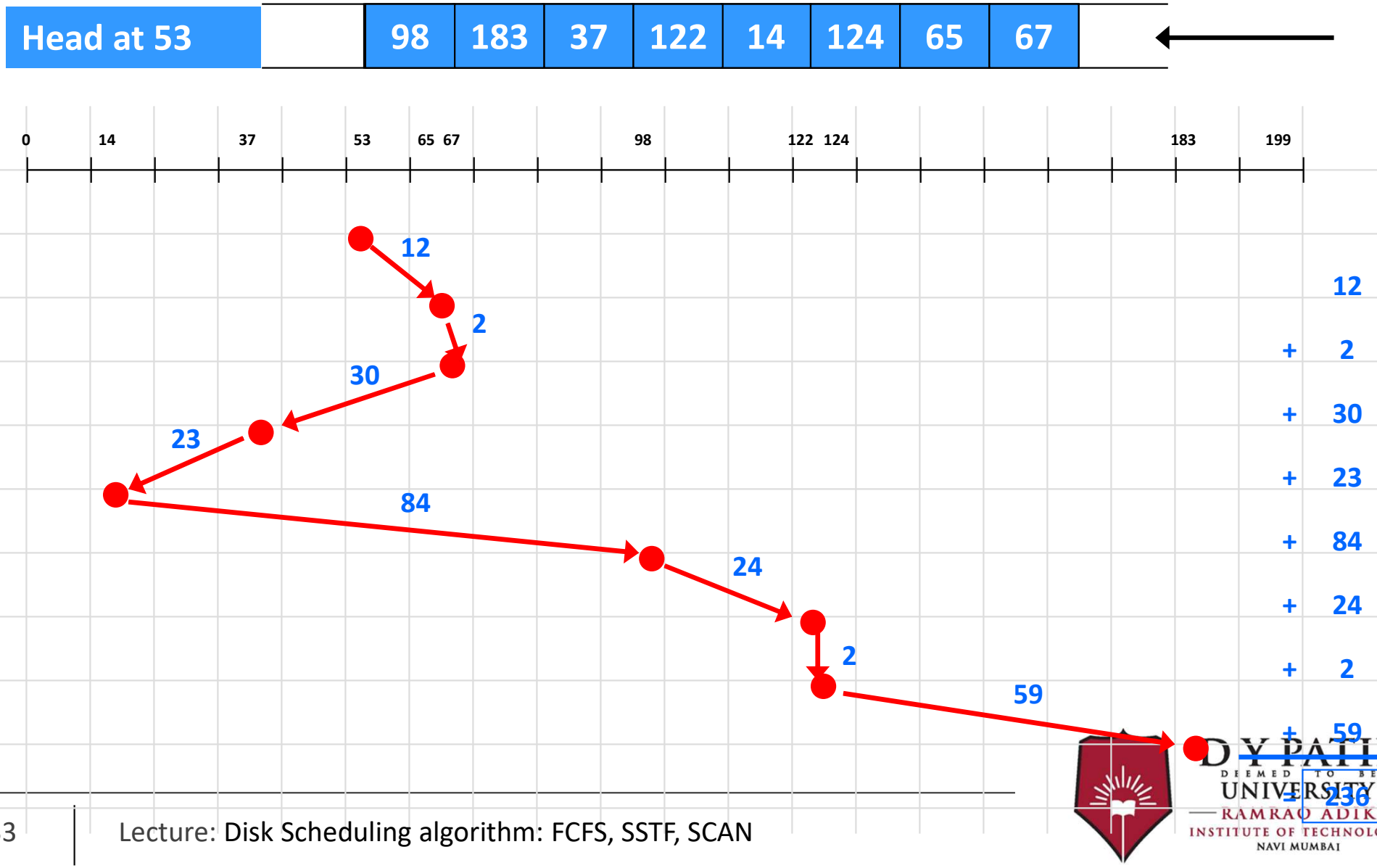
Advantages-

It reduces the total seek time as compared to FCFS.
It provides increased throughput.
It provides less average response time and waiting time.

Disadvantages-

There is an overhead of finding out the closest request.
The requests which are far from the head might starve for the CPU.
It provides high variance in response time and waiting time.
Switching the direction of head frequently slows down the algorithm.

Disk Scheduling SSTF



Disk Scheduling SSTF

- 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. The cylinders are numbered from 0 to 199. The total head movement (in number of cylinders) incurred while servicing these requests is _____.

- Total head movements incurred while servicing these requests
- = (65 – 53) + (67 – 65) + (67 – 41) + (41 – 14) + (98 – 14) + (122 – 98) + (124 – 122) + (183 – 124)
- = 12 + 2 + 26 + 27 + 84 + 24 + 2 + 59
- = 236

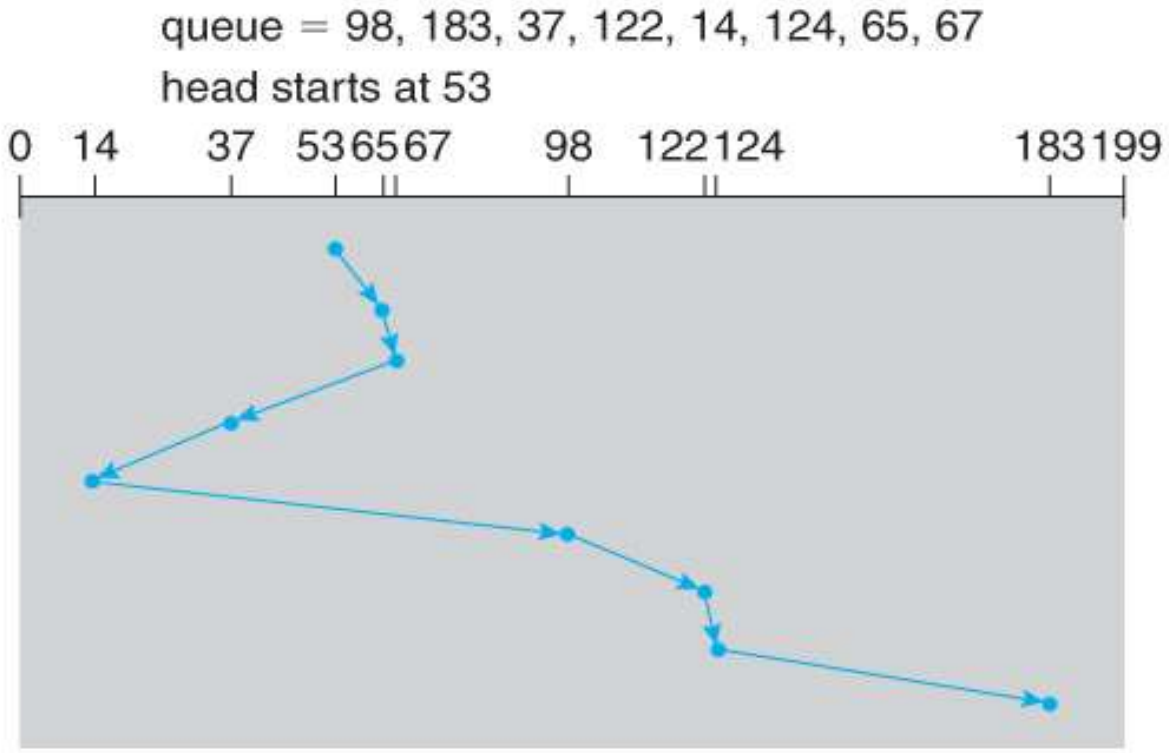
Shortest Service Time First

- The SSTF policy is to select the disk I/O request that requires the least movement of the disk arm from its current position. Thus, we always choose to incur the minimum seek time.

- The first track accessed is 98, because this is the closest requested track to the starting position.

Secondary Storage- Disk Scheduling - SSTF

- ***Shortest Seek Time First*** scheduling is more efficient, but may lead to starvation if a constant stream of requests arrives for the same general area of the disk.
- SSTF **reduces the total head movement** to 236 cylinders, down from 640 required for the same set of requests under FCFS.



SCAN Disk Scheduling Algorithm-

- As the name suggests, this algorithm scans all the cylinders of the disk back and forth.
- Head starts from one end of the disk and move towards the other end servicing all the requests in between.
- After reaching the other end, head reverses its direction and move towards the starting end servicing all the requests in between.
- The same process repeats.
-

NOTE-

SCAN Algorithm is also called as **Elevator Algorithm.**
This is because its working resembles the working of an elevator.

Secondary Storage- Disk Scheduling - SCAN

- Under the **SCAN** algorithm, If a request arrives **just ahead** of the moving head then it **will be processed** right away, but if it arrives just **after the head** has passed, then it will **have to wait** for the head to pass going the other way on the return trip. This leads to a fairly wide variation in access times which can be improved upon.

- Consider, **for example**, when the head reaches the high end of the disk: Requests with high cylinder numbers just missed the passing head, which means they are all fairly recent requests, whereas requests with low numbers may have been waiting for a much longer time. Making the return scan from high to low then ends up accessing recent requests first and making older requests wait that much longer.

SCAN

- With SCAN, the arm is required to move in one direction only, satisfying all outstanding requests en route, until it reaches the last track in that direction or until there are no more requests in that direction.This latter refinement is sometimes referred to as the LOOK policy. The service direction is then reversed and the scan proceeds in the opposite direction, again picking up all requests in order.

- As can be seen, the SCAN policy behaves almost identically with the SSTF policy. Indeed, if we had assumed that the arm was moving in the direction of lower track numbers at the beginning of the example, then the scheduling pattern would have been identical for SSTF and SCAN.

- With SCAN, if the expected time for a scan from inner track to outer track is *t*, *then the expected service interval for sectors at the* periphery is 2*t*

SCAN Disk Scheduling Algorithm-

Advantages-

It is simple, easy to understand and implement.

It does not lead to starvation.

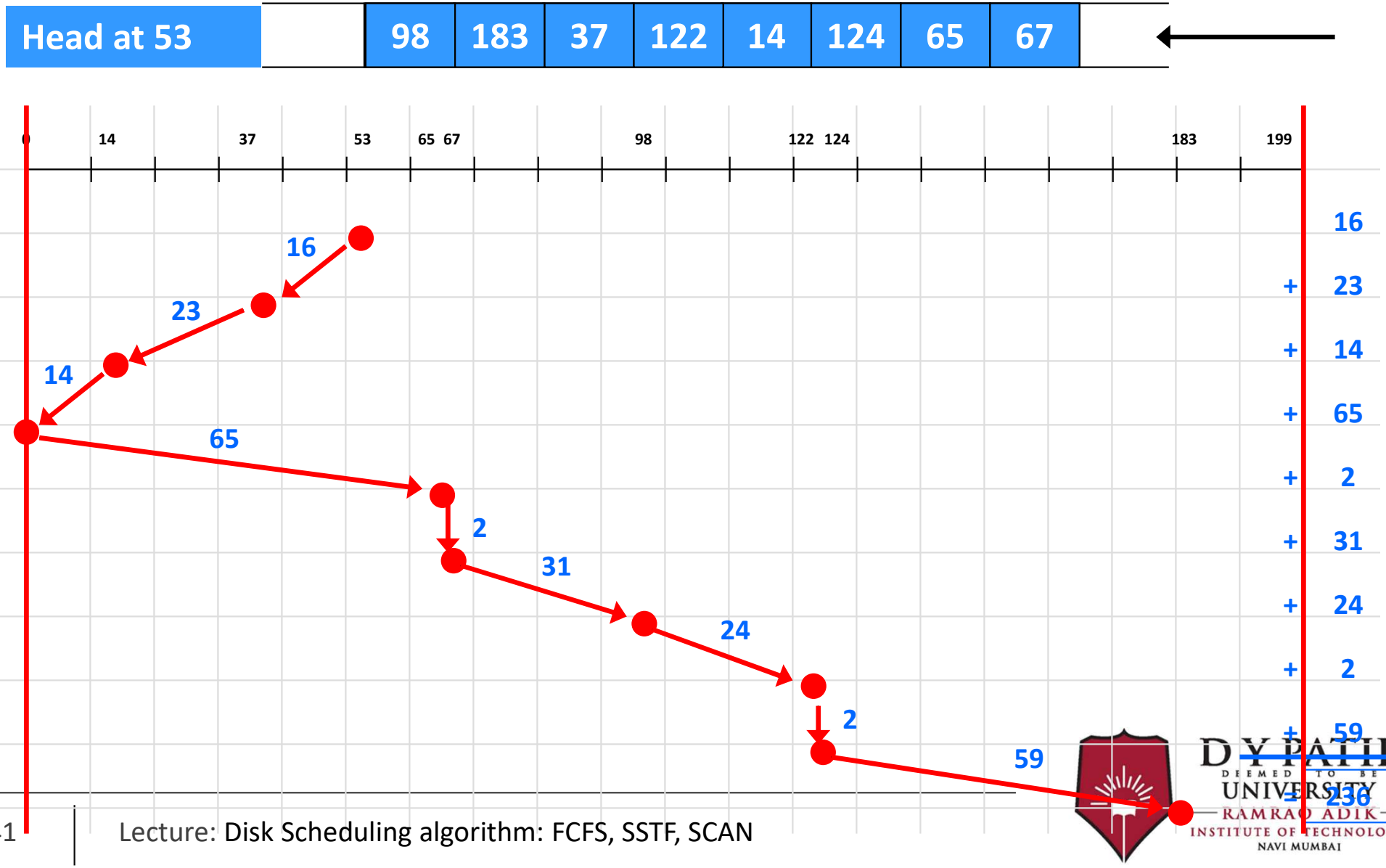
It provides low variance in response time and waiting time.

Disadvantages-

It causes long waiting time for the cylinders just visited by the head.

It causes the head to move till the end of the disk even if there are no requests to be serviced.

Disk Scheduling SCAN



SCAN

- Consider a disk queue with requests for I/O to blocks on cylinders 98, 183, 41, 122, 14, 124, 65, 67. The SCAN scheduling algorithm is used. The head is initially at cylinder number 53 moving towards larger cylinder numbers on its servicing pass. The cylinders are numbered from 0 to 199. The total head movement (in number of cylinders) incurred while servicing these requests is _____.

Total head movements incurred while servicing these requests
= (65 – 53) + (67 – 65) + (98 – 67) + (122 – 98) + (124 – 122) + (183 – 124) + (199 – 183) + (199 – 41) + (41 – 14)

= 12 + 2 + 31 + 24 + 2 + 59 + 16 + 158 + 27
= 331
Alternatively,
Total head movements incurred while servicing these requests
= (199 – 53) + (199 – 14)
= 146 + 185
- = 331

Secondary Storage- Disk Scheduling - SCAN

- The ***SCAN*** algorithm, a.k.a. the ***elevator*** algorithm moves back and forth from one end of the disk to the other, similarly to an elevator processing requests in a tall building.

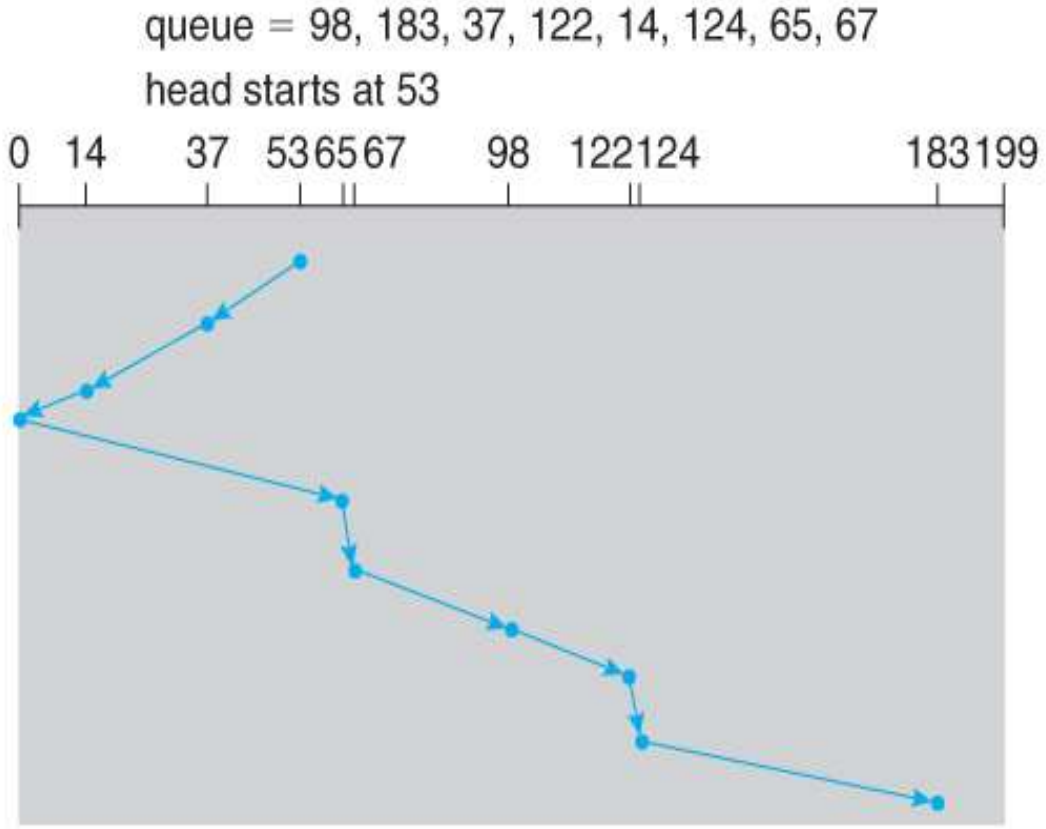


Figure : SCAN disk scheduling

Secondary Storage- Selection of a Disk-Scheduling Algorithm

- With very **low loads all algorithms are equal**, as there will normally only be 1 request to process at a time.
- For slightly **larger loads**, **SSTF** offers better performance than **FCFS**, but may lead to starvation when loads become heavy enough. For busier systems, **SCAN** and **LOOK** algorithms eliminate starvation problems.
- Some improvement to overall file system access times can be made by **intelligent placement of directory** and/or inode information. If those structures are placed in **the middle of the disk** instead of at the beginning of the disk, then maximum distance from those structures to data blocks is reduced to only **½ of the disk** size.

.

Lecture No: 39

Disk Scheduling algorithm: CSCAN, LOOK, CLOOK.

C-SCAN Disk Scheduling Algorithm-

- Circular-SCAN Algorithm is an improved version of the **SCAN Algorithm**.
- Head starts from one end of the disk and move towards the other end servicing all the requests in between.
- After reaching the other end, head reverses its direction.
- It then returns to the starting end without servicing any request in between.
- The same process repeats.

C-SCAN Disk Scheduling Algorithm-

Advantages-

The waiting time for the cylinders just visited by the head is reduced as compared to the SCAN Algorithm.

It provides uniform waiting time.

It provides better response time.

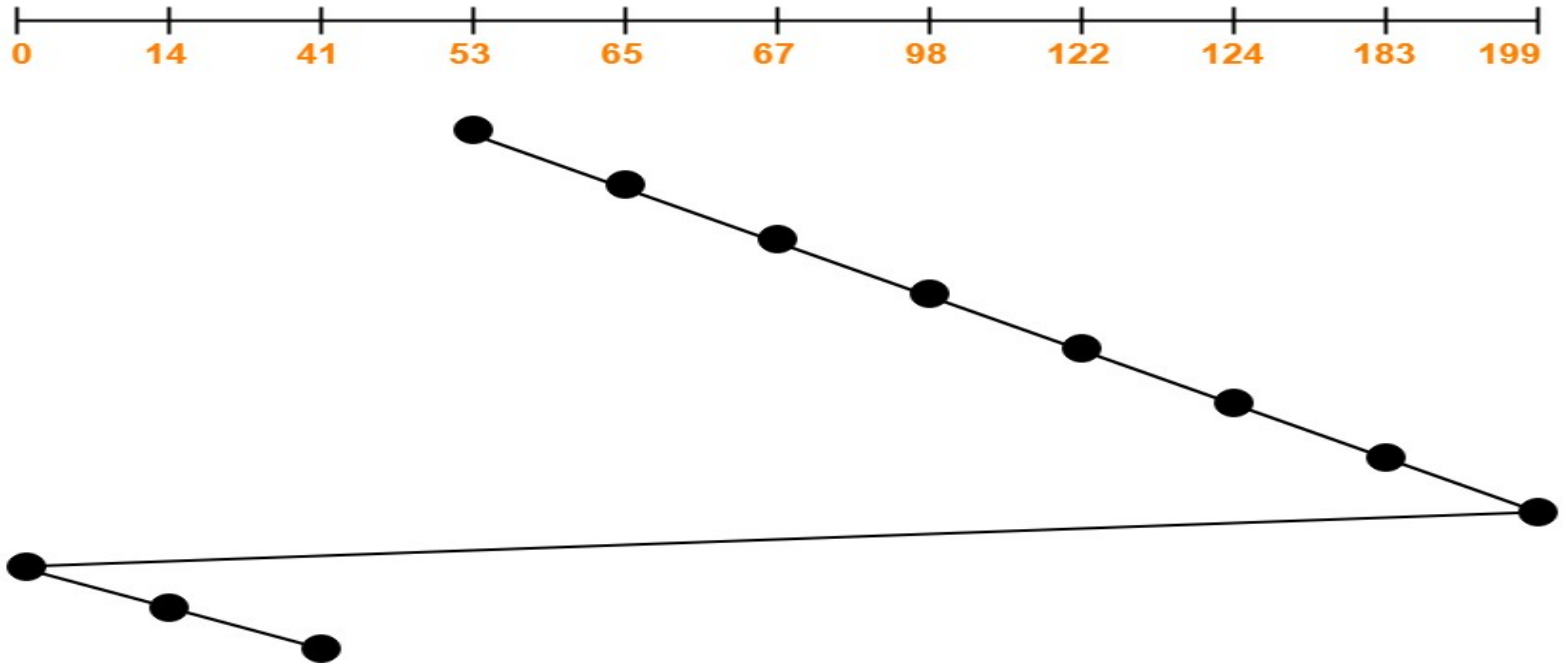
Disadvantages-

It causes more seek movements as compared to SCAN Algorithm.

It causes the head to move till the end of the disk even if there are no requests to be serviced.

C-SCAN Disk Scheduling Algorithm-

- **Problem-**
- Consider a disk queue with requests for I/O to blocks on cylinders 98, 183, 41, 122, 14, 124, 65, 67. The C-SCAN scheduling algorithm is used. The head is initially at cylinder number 53 moving towards larger cylinder numbers on its servicing pass. The cylinders are numbered from 0 to 199. The total head movement (in number of cylinders) incurred while servicing these requests is

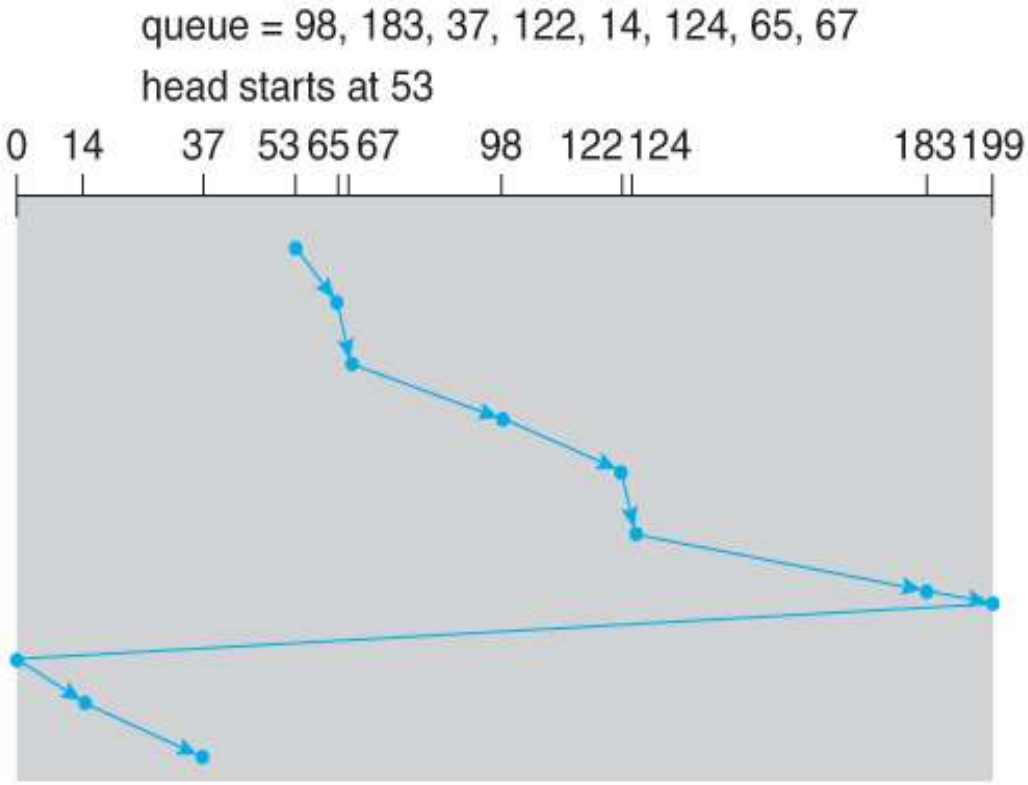


- Total head movements incurred while servicing these requests
- = (65 – 53) + (67 – 65) + (98 – 67) + (122 – 98) + (124 – 122) + (183 – 124) + (199 – 183) + (199 – 0) + (14 – 0) + (41 – 14)
- = 12 + 2 + 31 + 24 + 2 + 59 + 16 + 199 + 14 + 27
- = 386

- **Alternatively**,
- Total head movements incurred while servicing these requests
- = (199 – 53) + (199 – 0) + (41 – 0)
- = 146 + 199 + 41

Secondary Storage- Disk Scheduling – C-SCAN

- The ***Circular-SCAN*** algorithm **improves** upon **SCAN** by treating all requests in a **circular queue fashion** - Once the head reaches the end of the disk, it returns to the other end **without processing any requests**, and then starts again from the beginning of the

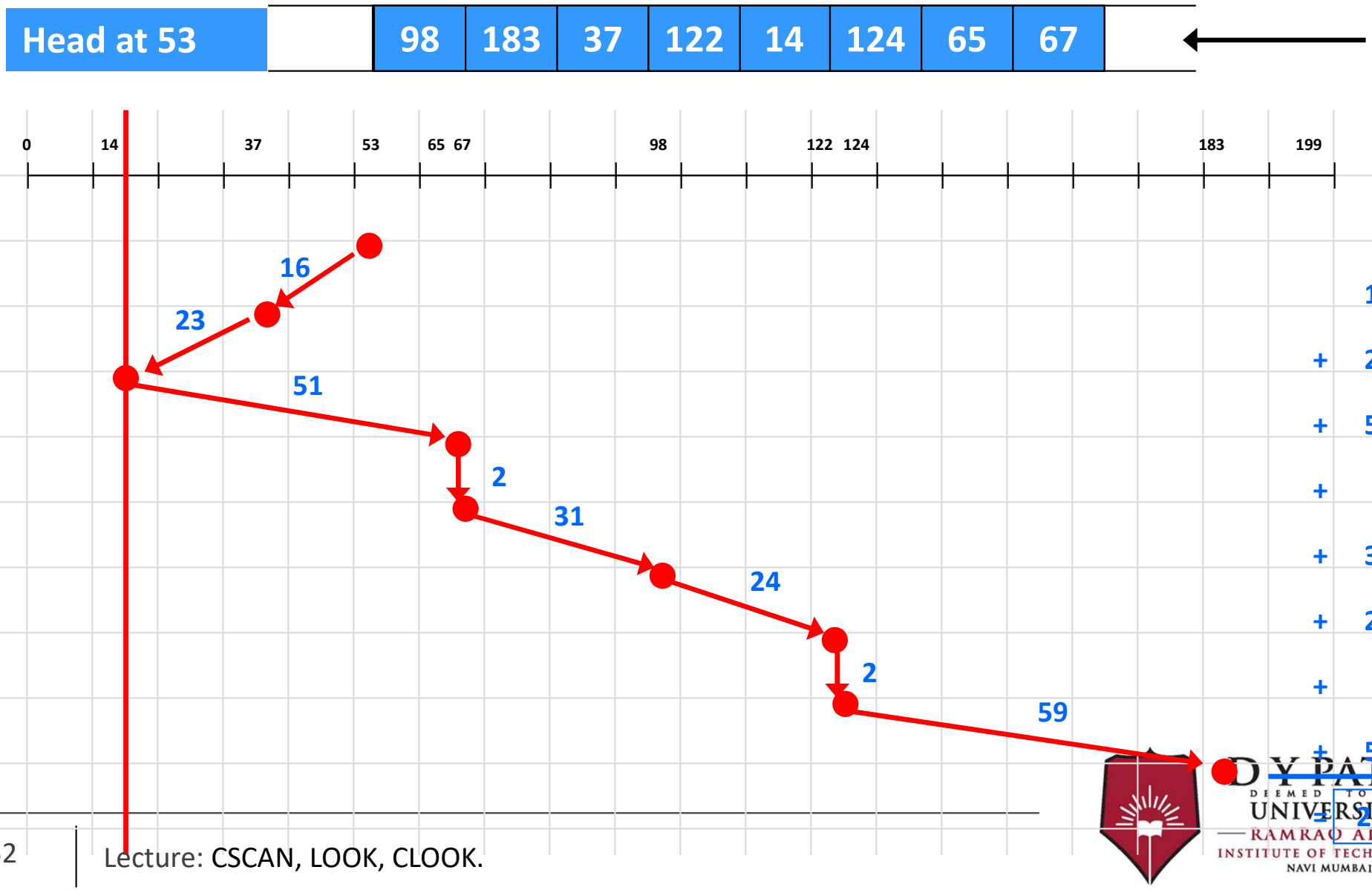


- Figure : C-SCAN disk scheduling

LOOK Disk Scheduling Algorithm-

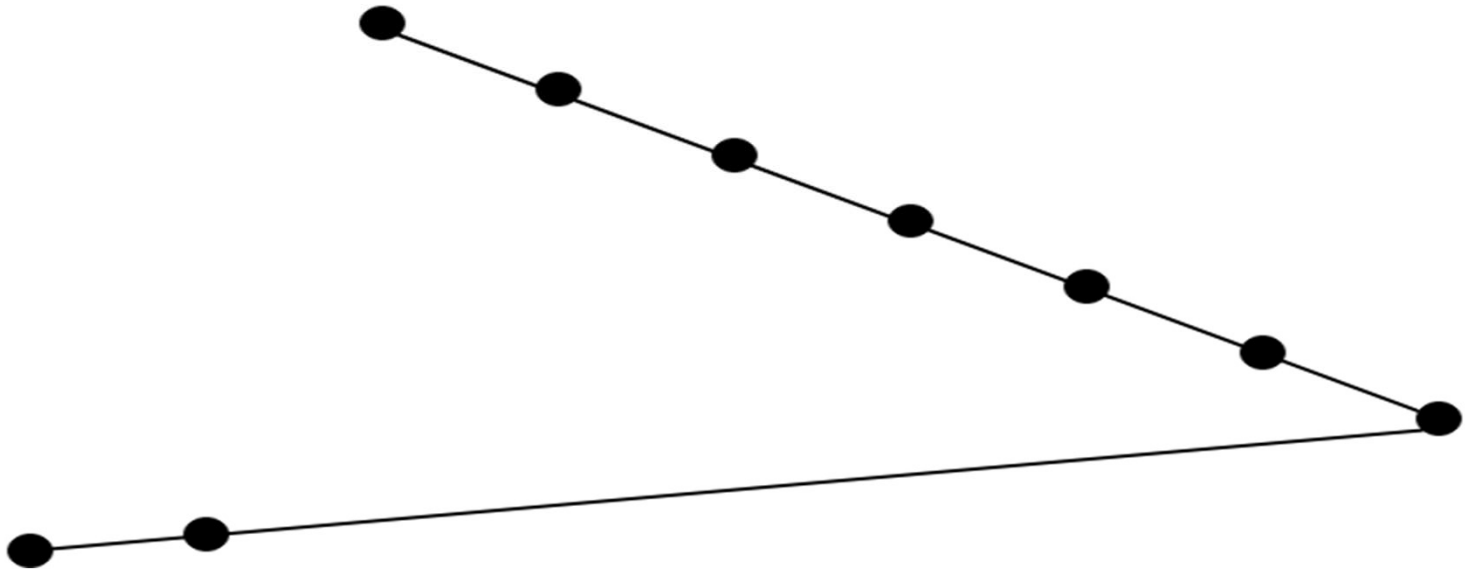
-
- LOOK Algorithm is an improved version of the **SCAN Algorithm**.
- Head starts from the first request at one end of the disk and moves towards the last request at the other end servicing all the requests in between.
- After reaching the last request at the other end, head reverses its direction.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

Disk Scheduling LOOK



- **Problem-**
-
- Consider a disk queue with requests for I/O to blocks on cylinders 98, 183, 41, 122, 14, 124, 65, 67. The LOOK scheduling algorithm is used. The head is initially

-



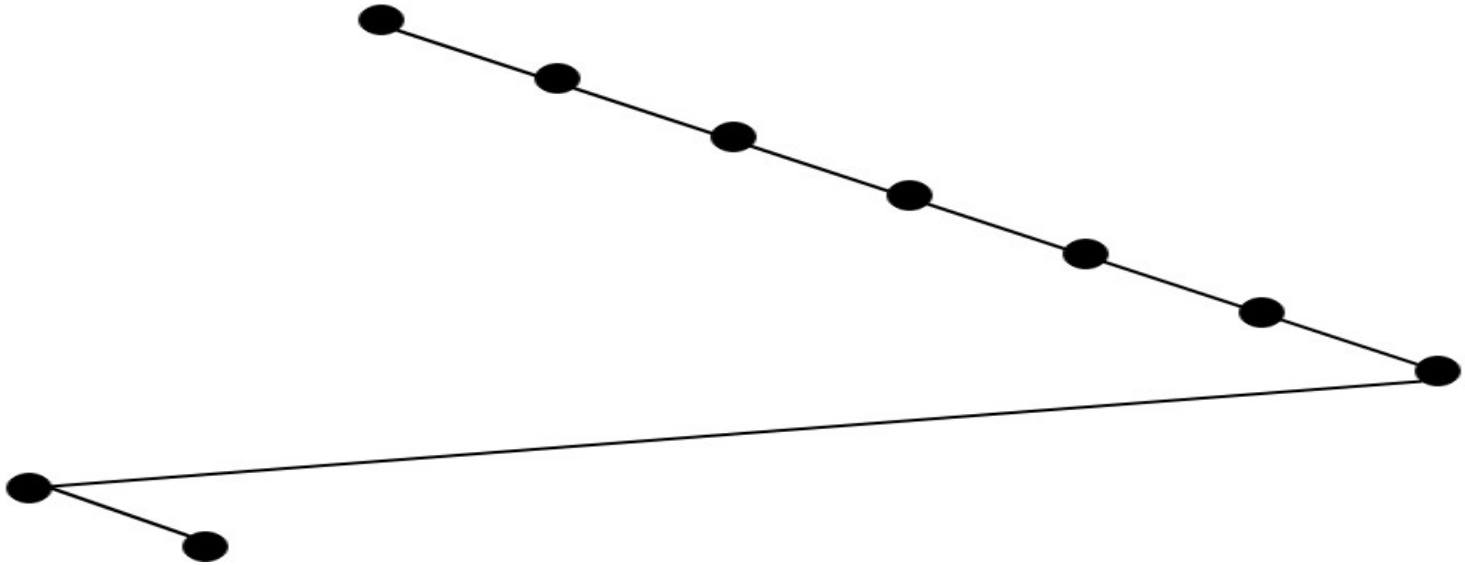
- Total head movements incurred while servicing these requests
- = (65 – 53) + (67 – 65) + (98 – 67) + (122 – 98) + (124 – 122) + (183 – 124) + (183 – 41) + (41 – 14)
- = 12 + 2 + 31 + 24 + 2 + 59 + 142 + 27
- = 299
- **Alternatively**,
- Total head movements incurred while servicing these requests
- = (183 – 53) + (183 – 14)
- = 130 + 169
- = 299

C-LOOK Disk Scheduling Algorithm-

-
- Circular-LOOK Algorithm is an improved version of the **LOOK 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 without servicing any request in between.
- The same process repeats.

- Consider a disk queue with requests for I/O to blocks on cylinders 98, 183, 41, 122, 14, 124, 65, 67. The C-LOOK scheduling algorithm is used. The head is initially at cylinder number 53 moving towards larger cylinder numbers on its servicing pass. The cylinders are numbered from 0 to 199. The total head

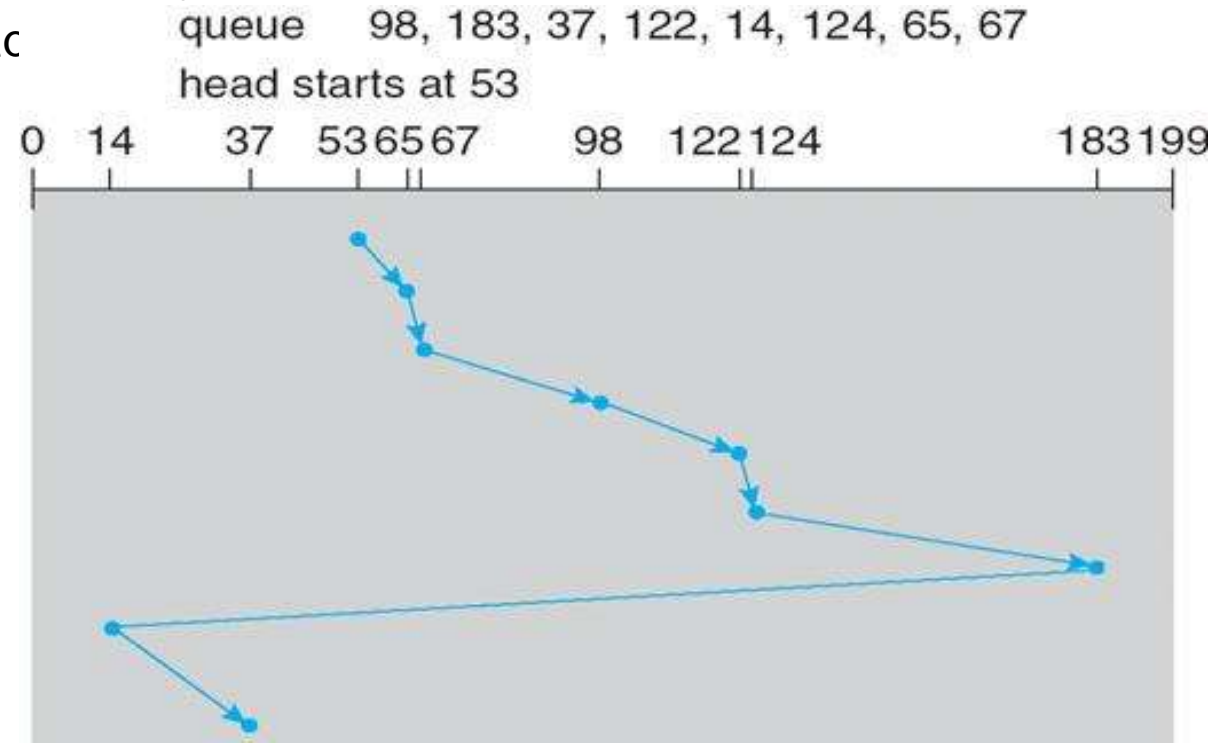
-



- Total head movements incurred while servicing these requests
- = (65 – 53) + (67 – 65) + (98 – 67) + (122 – 98) + (124 – 122) + (183 – 124) + (183 – 14) + (41 – 14)
- = 12 + 2 + 31 + 24 + 2 + 59 + 169 + 27
- = 326
-
- **Alternatively**,
- Total head movements incurred while servicing these requests
- = (183 – 53) + (183 – 14) + (41 – 14)
- = 130 + 169 + 27
- = 326

Secondary Storage- Disk Scheduling – C-LOOK

- ***C-LOOK*** scheduling **improves** upon **C-SCAN** by **looking ahead** at the queue of pending requests, and not moving the heads any farther towards the end of the disk than is necessary.
- The follc



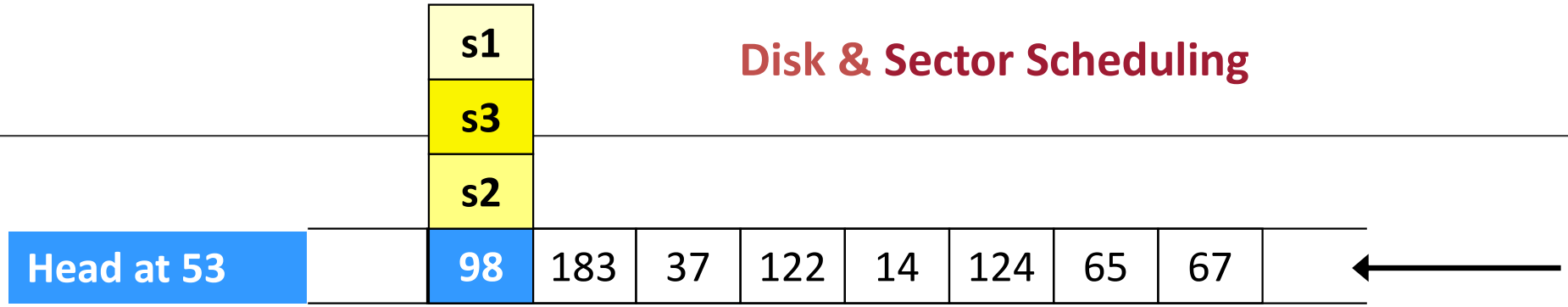
- Figure : C-LOOK disk scheduling

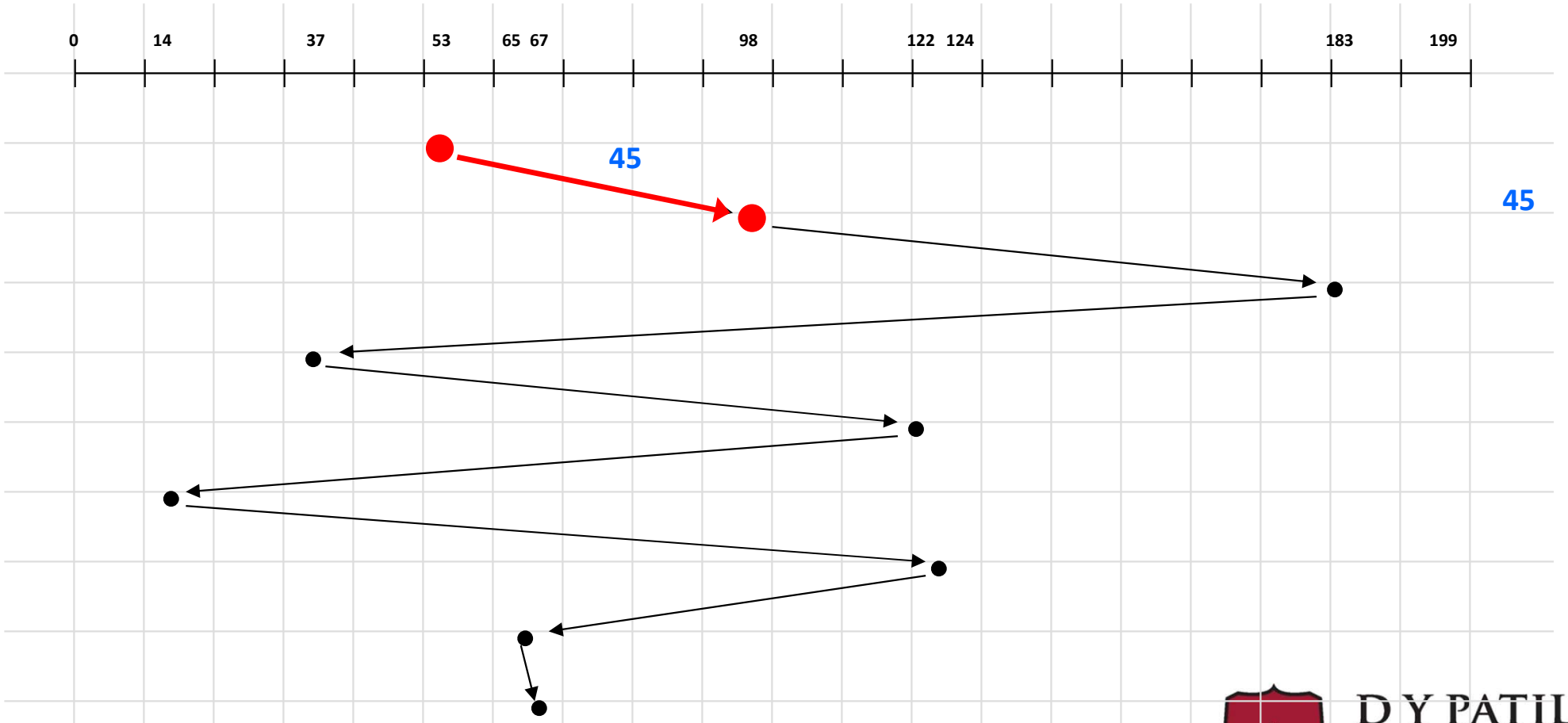
Practice problem

Suppose that a disk drive has 5000 cylinders, numbered 0 to 4999. The drive is currently serving a request at cylinder 143, and the previous request was at cylinder 125. The queue of pending requests in FIFO is ordered as 80, 1470, 913, 1777, 948, 1022, 1750,130. What is the total distance that the disk arm moves for following by applying following algorithms? 10

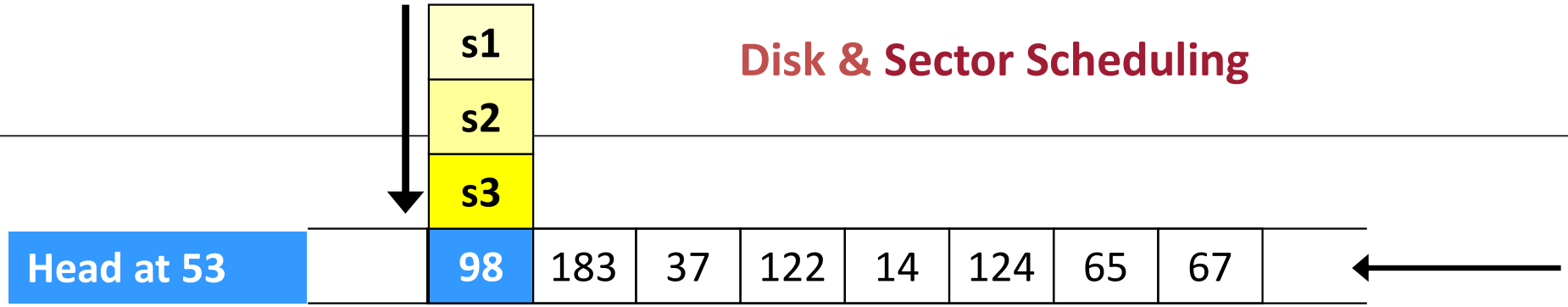
1. FCFS 2. SSTF 3. LOOK 4. SCAN

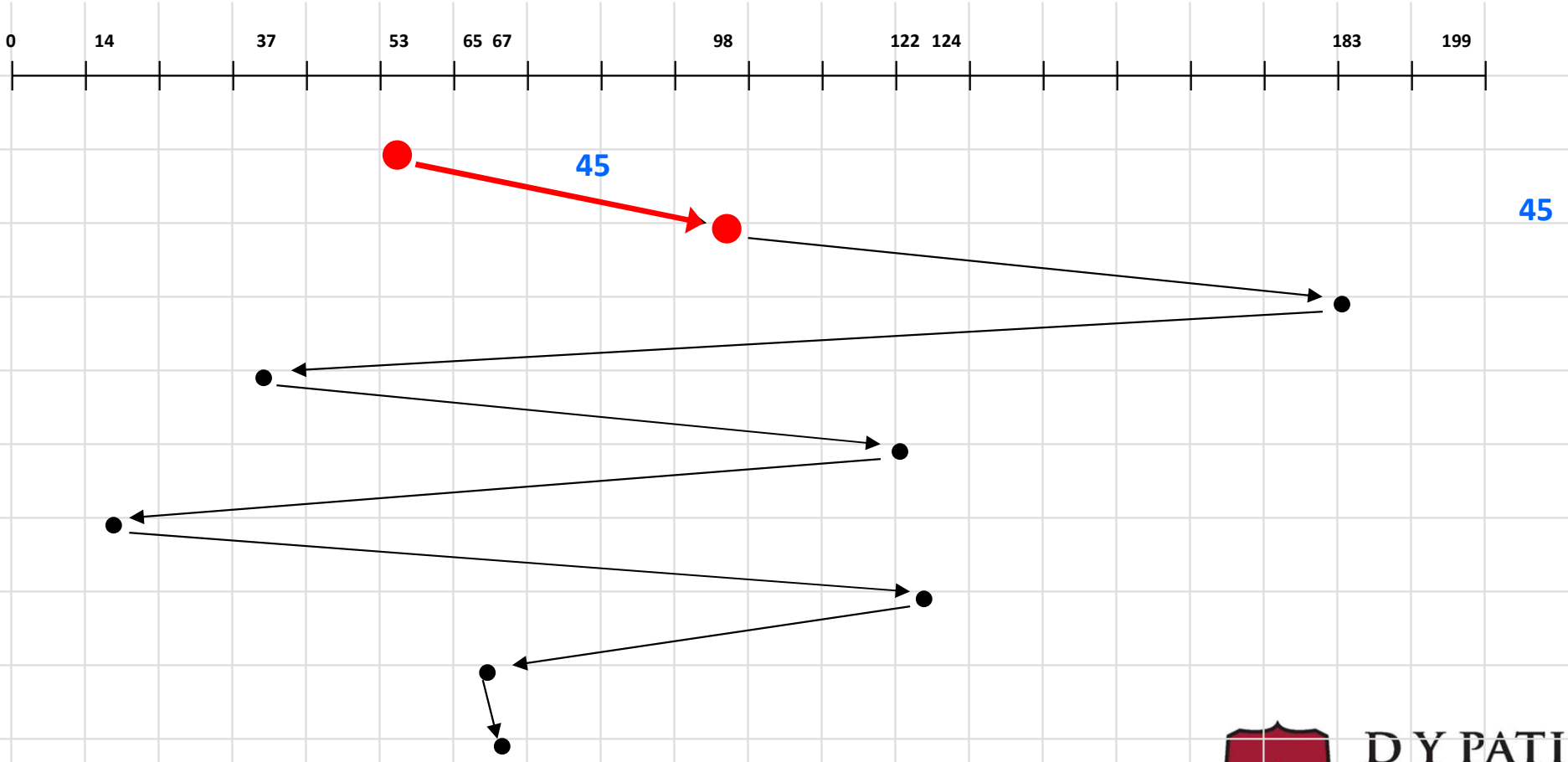
Disk & Sector Scheduling



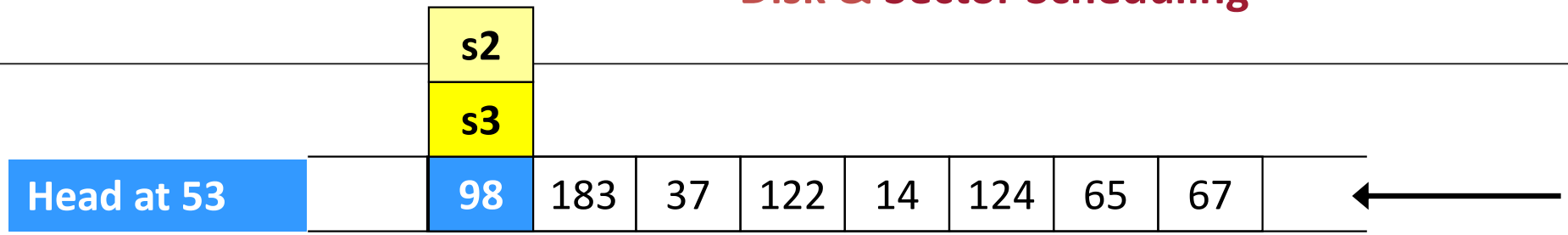


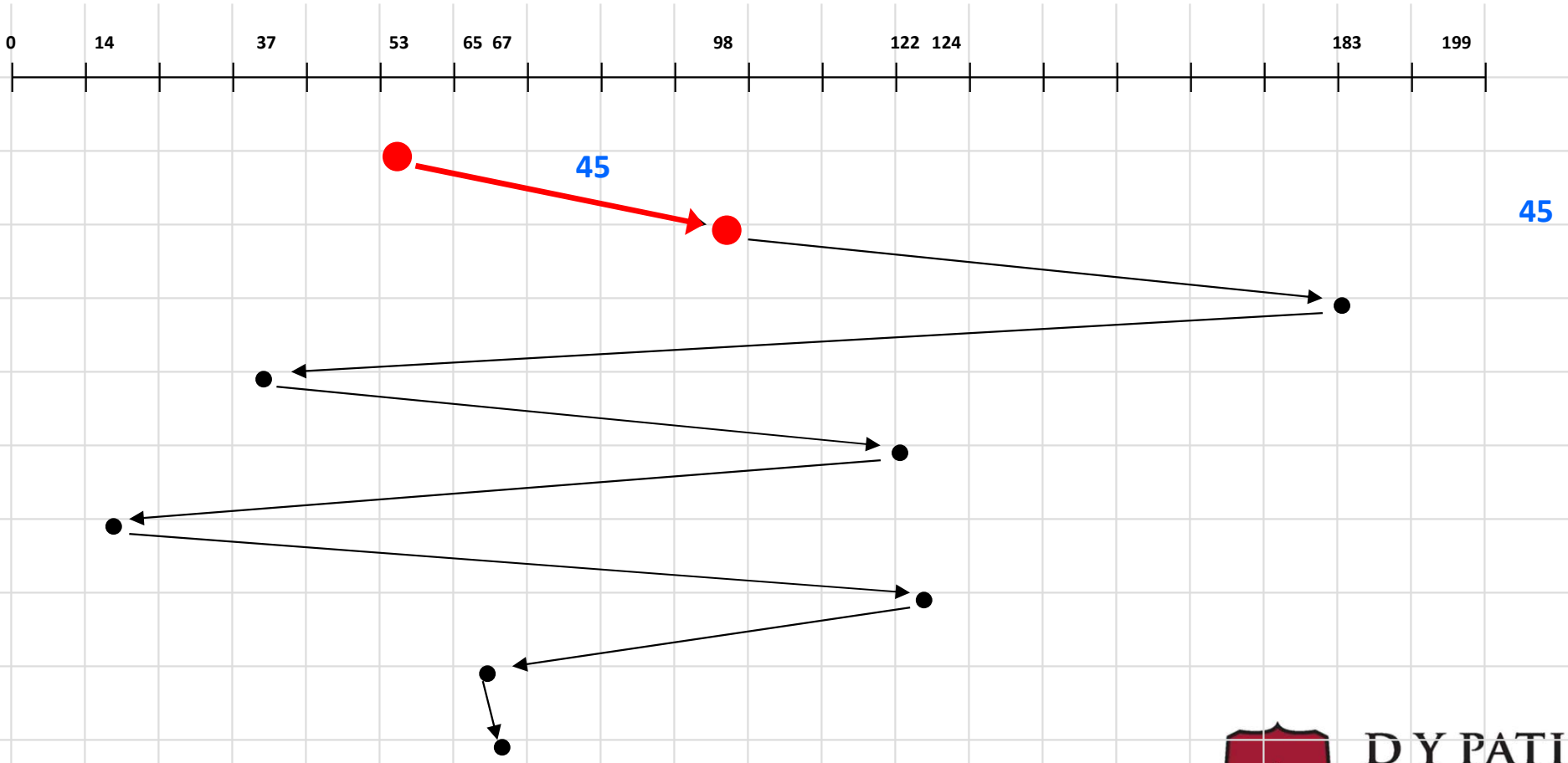
Disk & Sector Scheduling



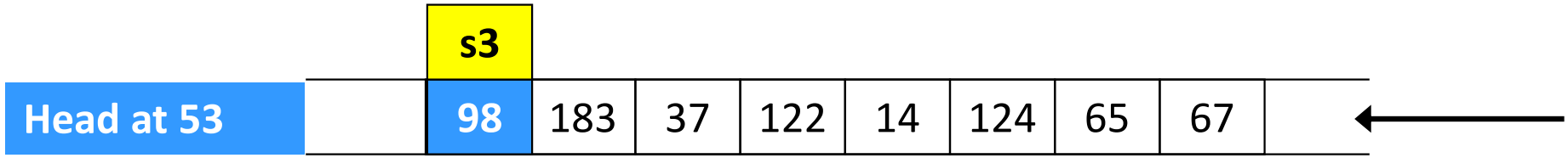


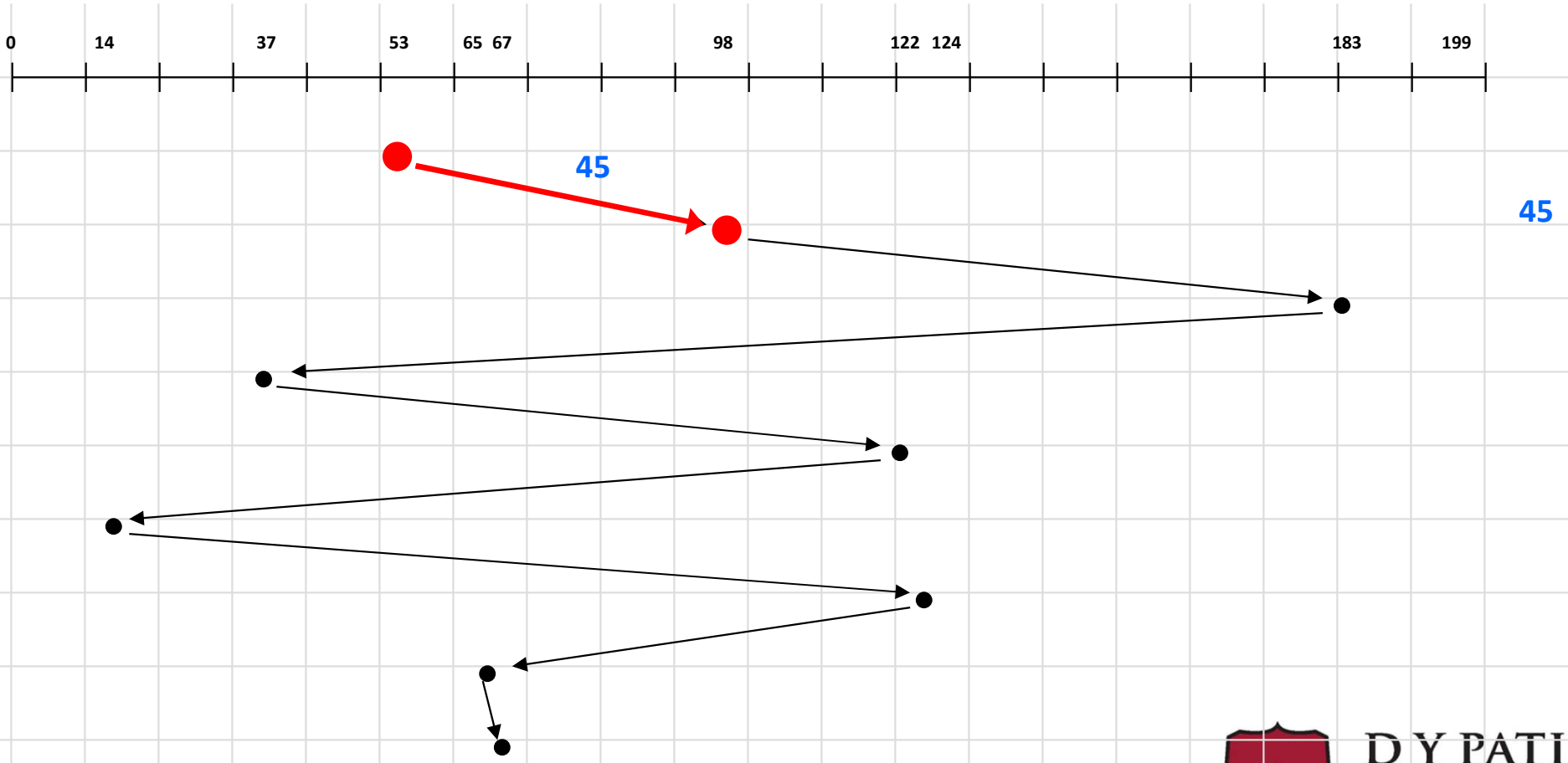
Disk & Sector Scheduling



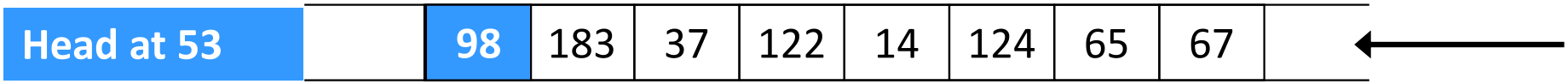


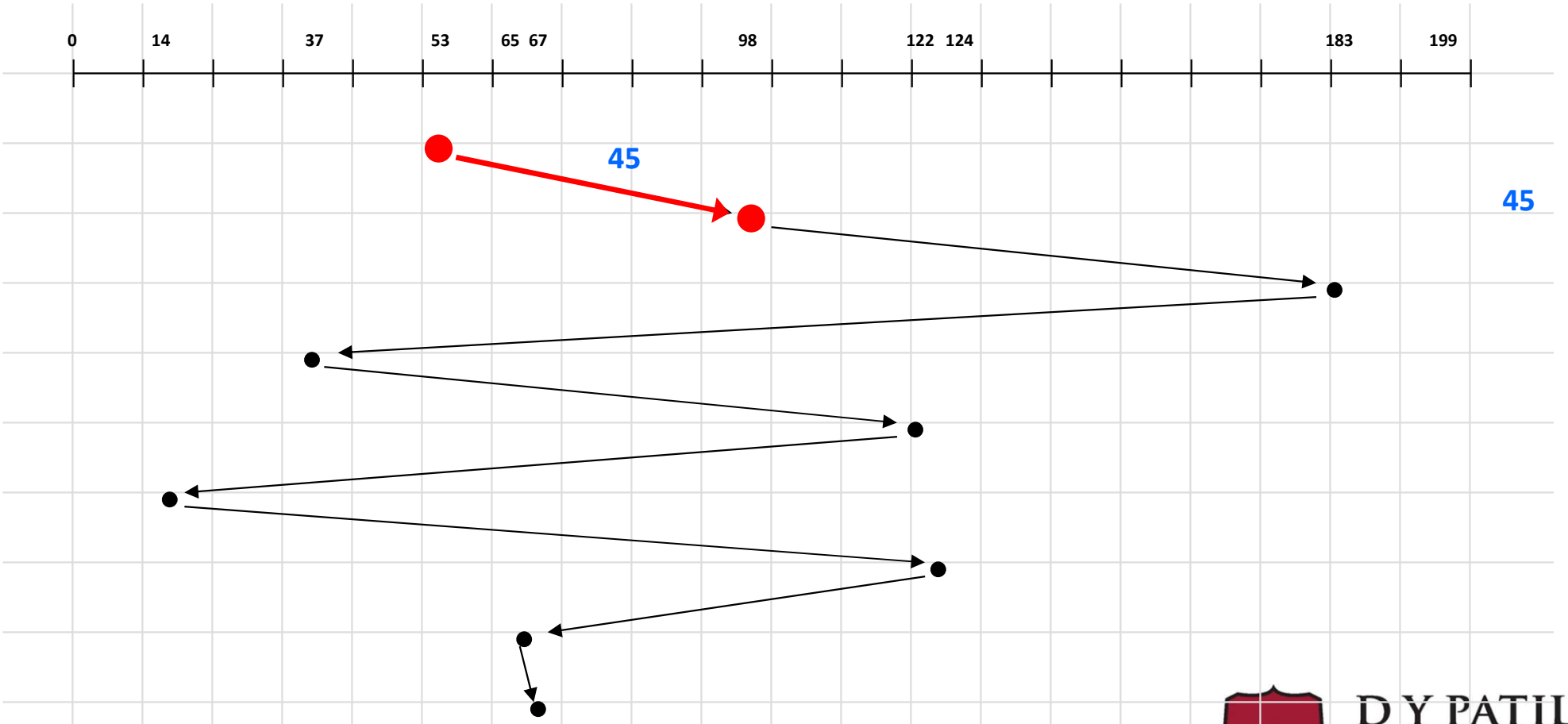
Disk & Sector Scheduling



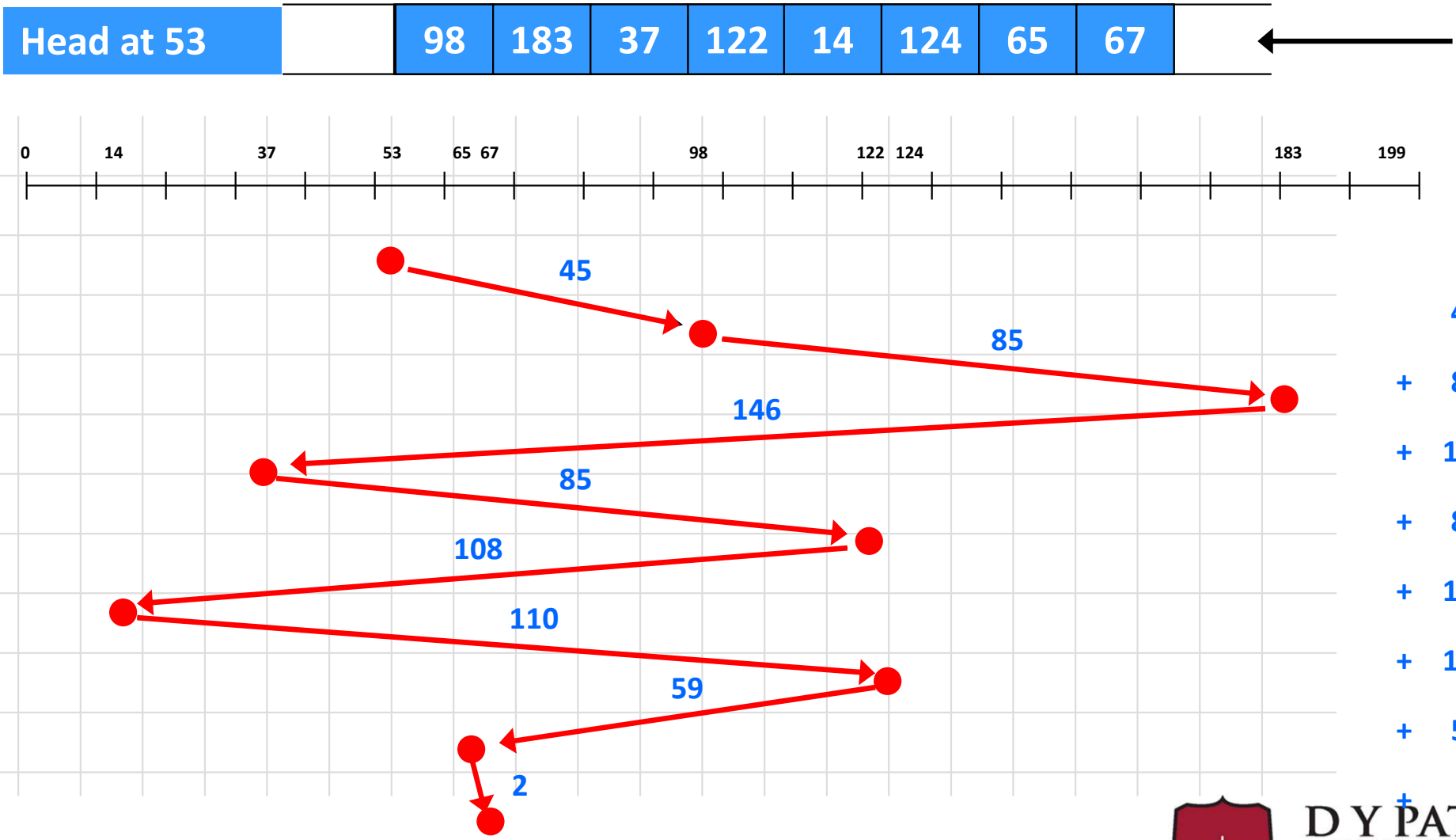


Disk & Sector Scheduling





Disk & Sector Scheduling



Secondary Storage- Selection of a Disk-Scheduling Algorithm

- On modern disks the **rotational latency** can be almost as significant as the **seek time**, however it is not within the OS control to account for that, because modern disks do not reveal their internal sector mapping schemes.
  - Some disk manufacturers provide for disk scheduling algorithms **directly on their disk controllers**, so that if a series of requests are sent from the computer to the controller then those requests can be processed in an **optimal order**.
  - Unfortunately there are **some considerations** that the OS must take into account that are **beyond the abilities of the on-board disk-scheduling algorithms**, such as priorities of some requests over others, or the need to process certain requests in a particular order. For this reason **OSes may elect** to spoon-feed requests to the disk controller one at a time in certain situations.

Table 11.2 Comparison of Disk Scheduling Algorithms

(a) FIFO (starting at track 100)		(b) SSTF (starting at track 100)		(c) SCAN (starting at track 100, in the direction of increasing track number)		(d) C-SCAN (starting at track 100, in the direction of increasing track number)	
Next track accessed	Number of tracks traversed	Next track accessed	Number of tracks traversed	Next track accessed	Number of tracks traversed	Next track accessed	Number of tracks traversed
55	45	90	10	150	50	150	50
58	3	58	32	160	10	160	10
39	19	55	3	184	24	184	24
18	21	39	16	90	94	18	166
90	72	38	1	58	32	38	20
160	70	18	20	55	3	39	1
150	10	150	132	39	16	55	16
38	112	160	10	38	1	58	3
184	146	184	24	18	20	90	32
Average seek length	55.3	Average seek length	27.5	Average seek length	27.8	Average seek length	35.8

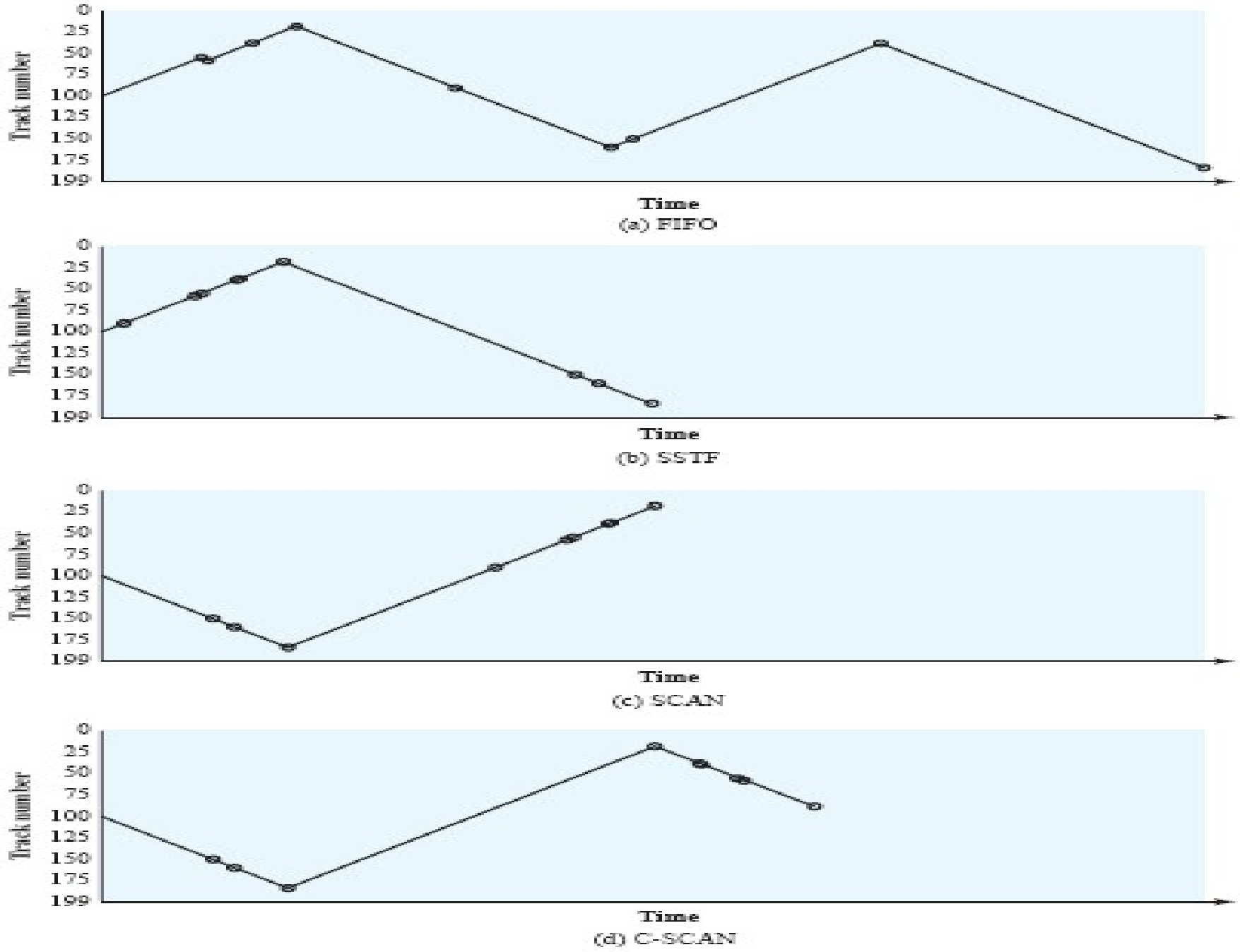


Figure 11.7 Comparison of Disk Scheduling Algorithms (see Table 11.3)

Table 11.3 Disk Scheduling Algorithms

Name	Description	Remarks
Selection according to requestor		
RSS	Random scheduling	For analysis and simulation
FIFO	First in first out	Fairest of them all
PRI	Priority by process	Control outside of disk queue management
LIFO	Last in first out	Maximize locality and resource utilization
Selection according to requested item		
SSTF	Shortest service time first	High utilization, small queues
SCAN	Back and forth over disk	Better service distribution
C-SCAN	One way with fast return	Lower service variability
N-step-SCAN	SCAN of N records at a time	Service guarantee
FSCAN	N-step-SCAN with N = queue size at beginning of SCAN cycle	Load sensitive

Previous Year Questions

b. Compare the following Disk scheduling algorithms using appropriate example- SSTF, FCFS, SCAN, C-SCAN, LOOK [10]

(b) Explain techniques of disk scheduling. 10

B) Explain the different disk performance parameters. 10

(a) Explain various disc scheduling algorithms. Explain the criteria for selecting the best disc scheduling algorithm. 10

Previous Year Questions

(b) The requested tracks in the order received are - 54, 57, 40, 20, 80, 120, 150, 45, 180. Apply the following disk scheduling algorithm starting track at 90. 10

(1) FCFS (ii) SSTF (iii) C-SCAN

Previous Year Questions

- Explain disk scheduling algorithms. [10M, May2018]
- Compare disk scheduling algorithms. [10M, Dec2018]
- Explain disk scheduling methods. [10M, Dec2019]

- Suppose that a disk drive has 5000 cylinders, numbered 0 to 4999. The drive is currently serving a request at cylinder 143, and the previous request was at cylinder 125. The queue of pending requests in FIFO is ordered as 80, 1470, 913, 1777, 948, 1022, 1750,130. What is the total distance that the disk arm moves for following by applying following algorithms? 1. FCFS 2. SSTF 3. LOOK 4. SCAN [10M, May2019]

Summary

- In this chapter we successfully studied basic concepts of I/O Management in detail along with that we have seen different Disk Scheduling Algorithms.

Lecture 39: Disk Management

Disk Management

Low-level formatting, or **physical formatting** — Dividing a disk into sectors that the disk controller can read and write

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
- **Logical formatting** or "making a file system"
- To increase efficiency most file systems group blocks into **clusters**
  - Disk I/O done in blocks
  - File I/O done in clusters

Boot block initializes system

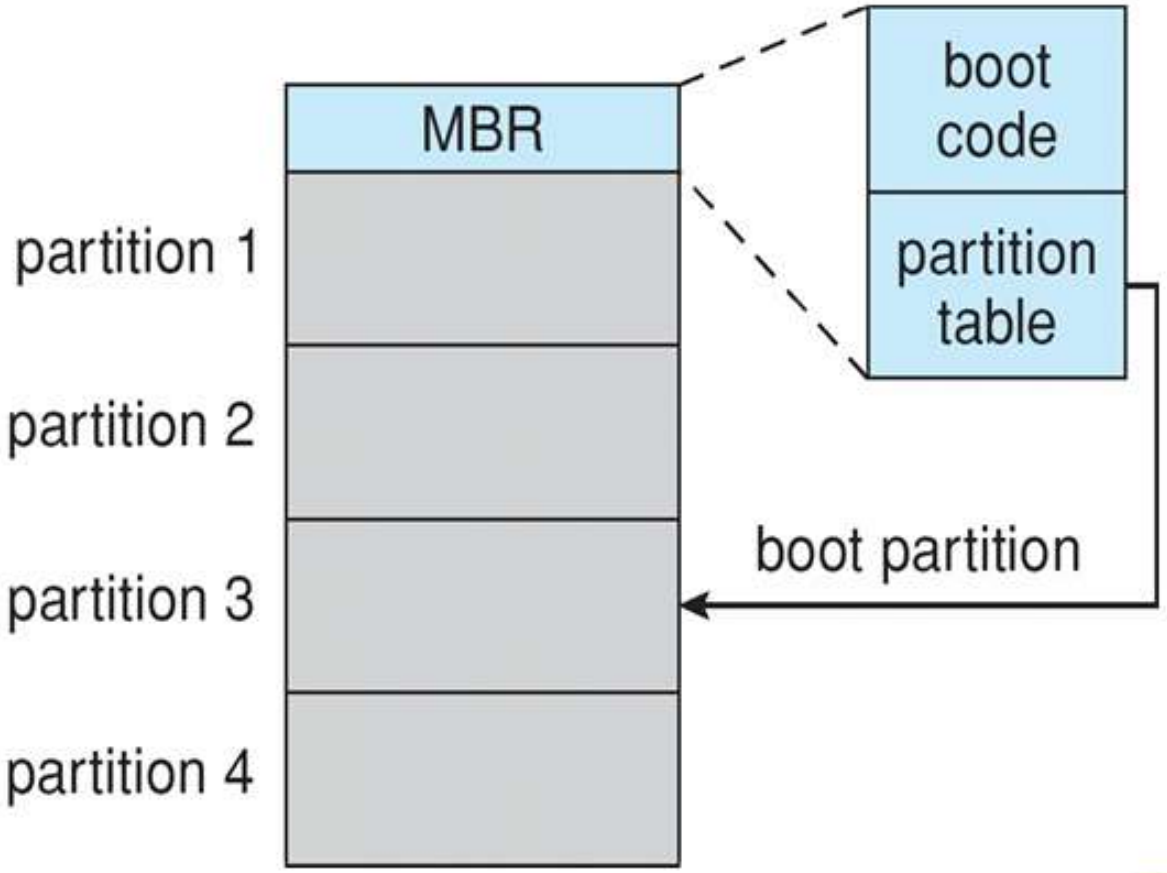
- The bootstrap is stored in ROM
- **Bootstrap loader** program

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

- The bootstrap is stored in ROM
- **Bootstrap loader** program

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

Booting from a Disk in Windows 2000



Swap-Space Management

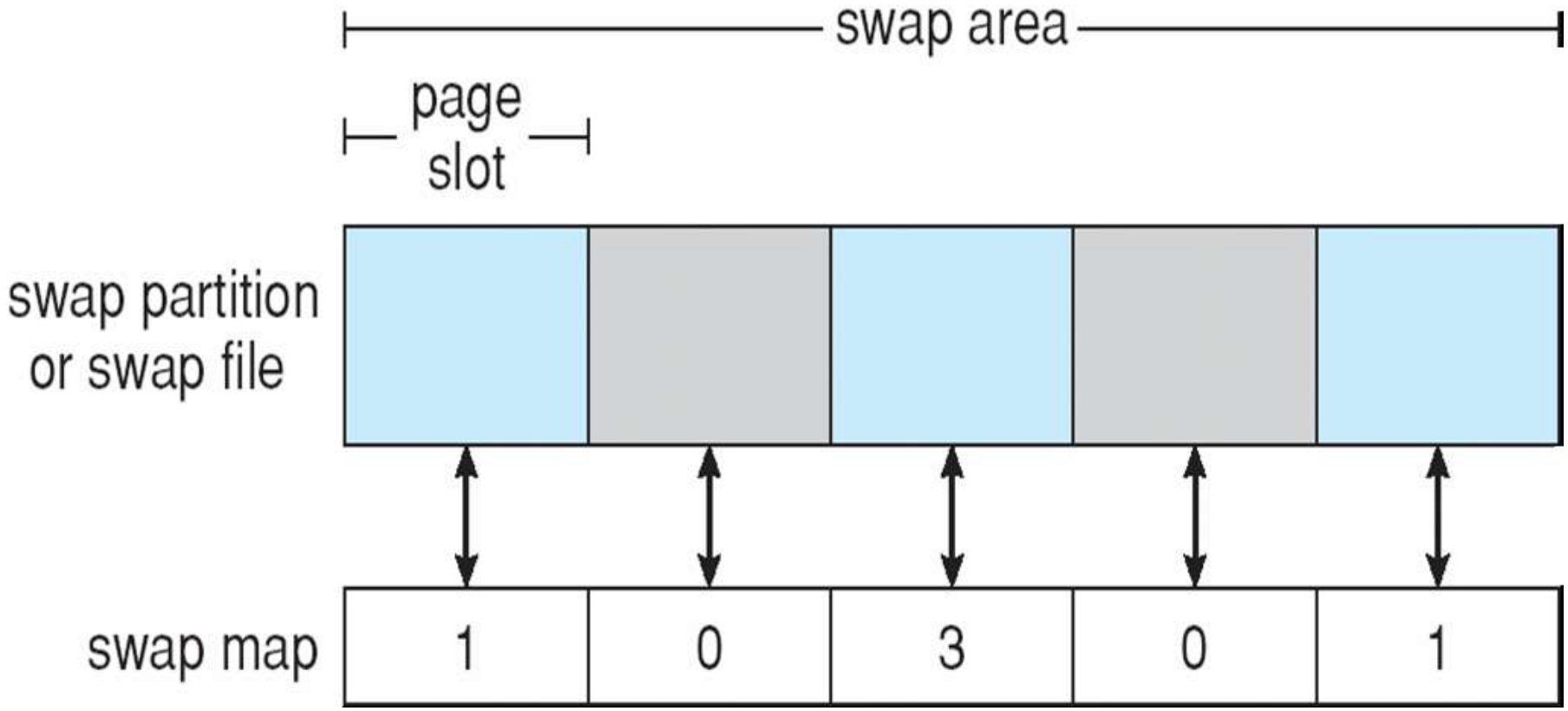
Swap-space — Virtual memory uses disk space as an extension of main memory

Swap-space can be carved out of the normal file system, or, more commonly, it can be in a separate disk partition

Swap-space management

- 4.3BSD allocates swap space when process starts; holds text segment (the program) and data segment
- Kernel uses **swap maps** to track swap-space use
- Solaris 2 allocates swap space only when a page is forced out physical memory, not when the virtual memorof y page is first created

Data Structures for Swapping on Linux Systems



RAID Structure

RAID – multiple disk drives provides reliability via **redundancy**
Increases the **mean time to failure**
Frequently combined with **NVRAM** to improve write performance
RAID is arranged into six different levels

RAID(Con.)

Several improvements in disk-use techniques involve the use of multiple disks working cooperatively

Disk **striping** uses a group of disks as one storage unit

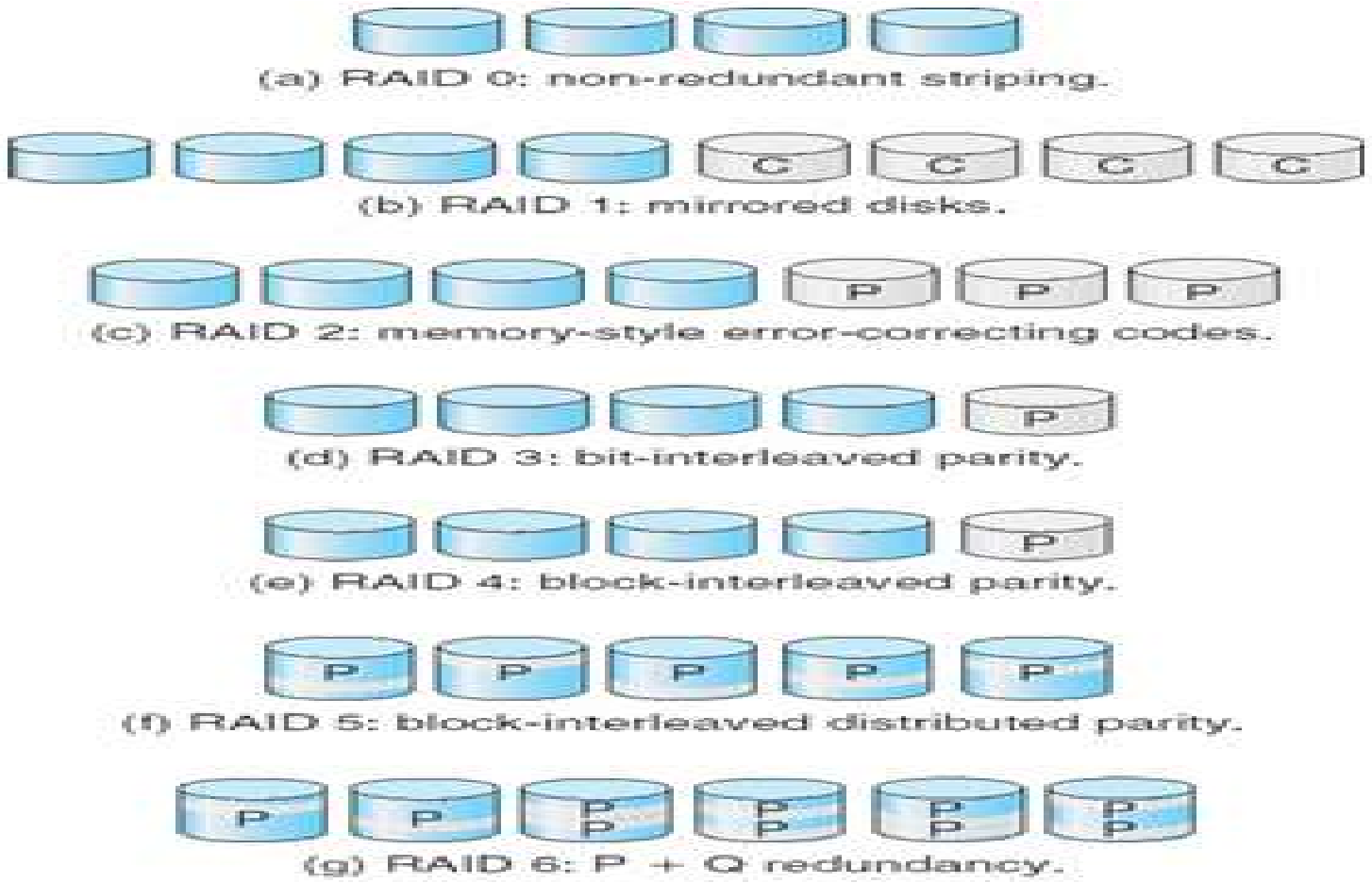
RAID schemes improve performance and improve the reliability of the storage system by storing redundant data

- **Mirroring** or **shadowing** (**RAID 1**) keeps duplicate of each disk
- Striped mirrors (**RAID 1+0**) or mirrored stripes (**RAID 0+1**) provides high performance and high reliability
- **Block interleaved parity** (**RAID 4, 5, 6**) uses much less redundancy

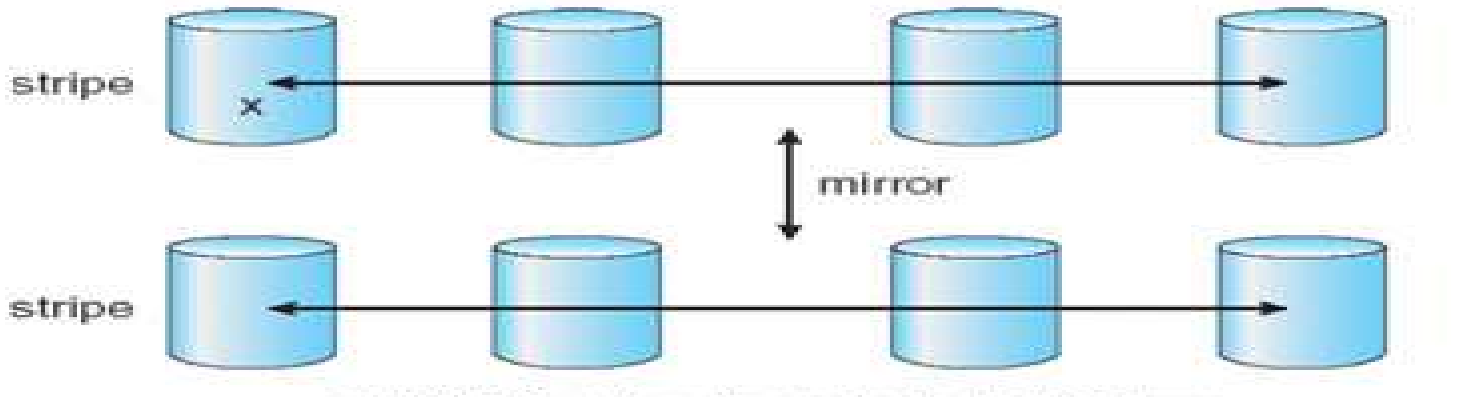
RAID within a storage array can still fail if the array fails, so automatic **replication** of the data between arrays is common

Frequently, a small number of **hot-spare** disks are left unallocated, automatically replacing a failed disk and having data rebuilt onto them

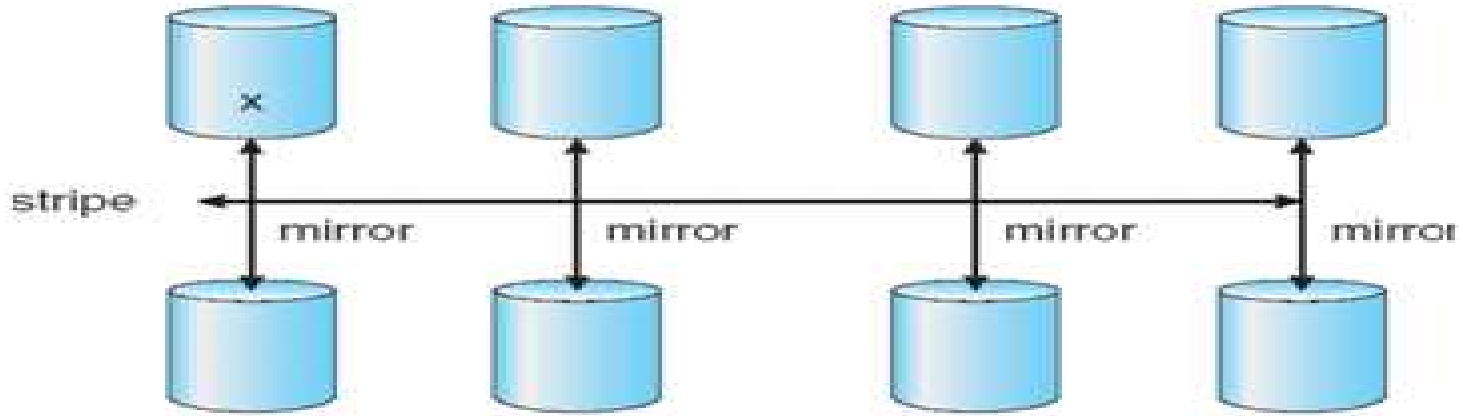
RAID level



RAID (0 + 1) and (1 + 0)



a) RAID 0 + 1 with a single disk failure.



b) RAID 1 + 0 with a single disk failure.

Extensions

RAID alone does not prevent or detect data corruption or other errors, just disk failures

Solaris ZFS adds **checksums** of all data and metadata

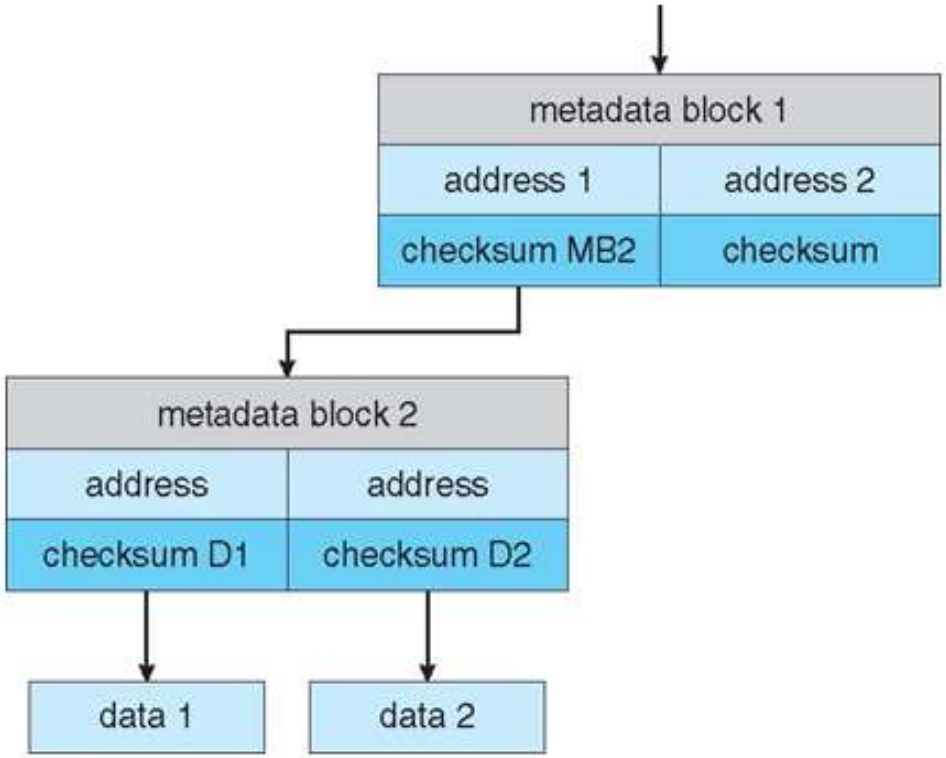
Checksums kept with pointer to object, to detect if object is the right one and whether it changed

Can detect and correct data and metadata corruption

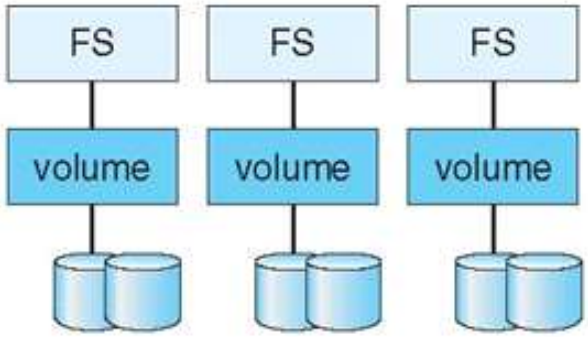
ZFS also removes volumes, partitions

- Disks allocated in **pools**
- Filesystems with a pool share that pool, use and release space like “malloc” and “free” memory allocate / release calls

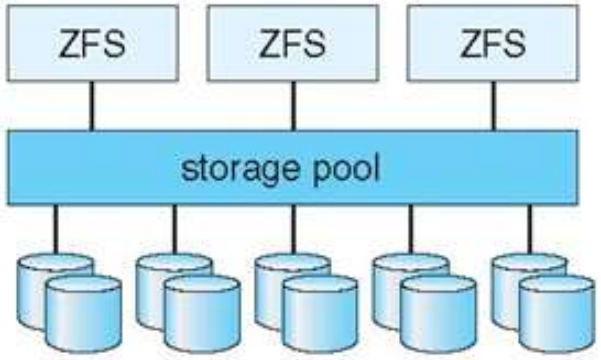
ZFS Checksums All Metadata and Data



Traditional and Pooled Storage



(a) Traditional volumes and file systems.



(b) ZFS and pooled storage.

Stable-Storage Implementation

Write-ahead log scheme requires stable storage

To implement stable storage:

- Replicate information on more than one nonvolatile storage media with independent failure modes
- Update information in a controlled manner to ensure that we can recover the stable data after any failure during data transfer or recovery

Tertiary Storage Devices

Low cost is the defining characteristic of tertiary storage

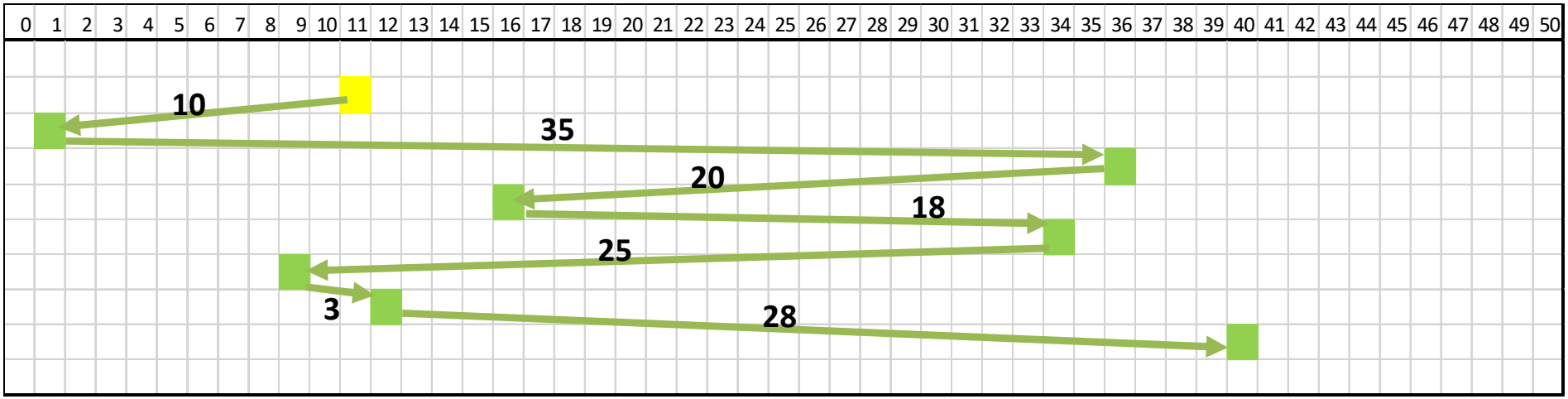
Generally, tertiary storage is built using **removable media**

Common examples of removable media are floppy disks and CD-ROMs; other types are available

University of Mumbai
Examination June 2021
Examinations Commencing from 1st June 2021
Program: Computer Engineering
Curriculum Scheme: Rev 2019 "C" Scheme
Examination: SE Semester IV
Course Code: CSC404 and Course Name: Operating System
Time: 2 hour **Max. Marks: 80**
Exam Date: 08th June 2021

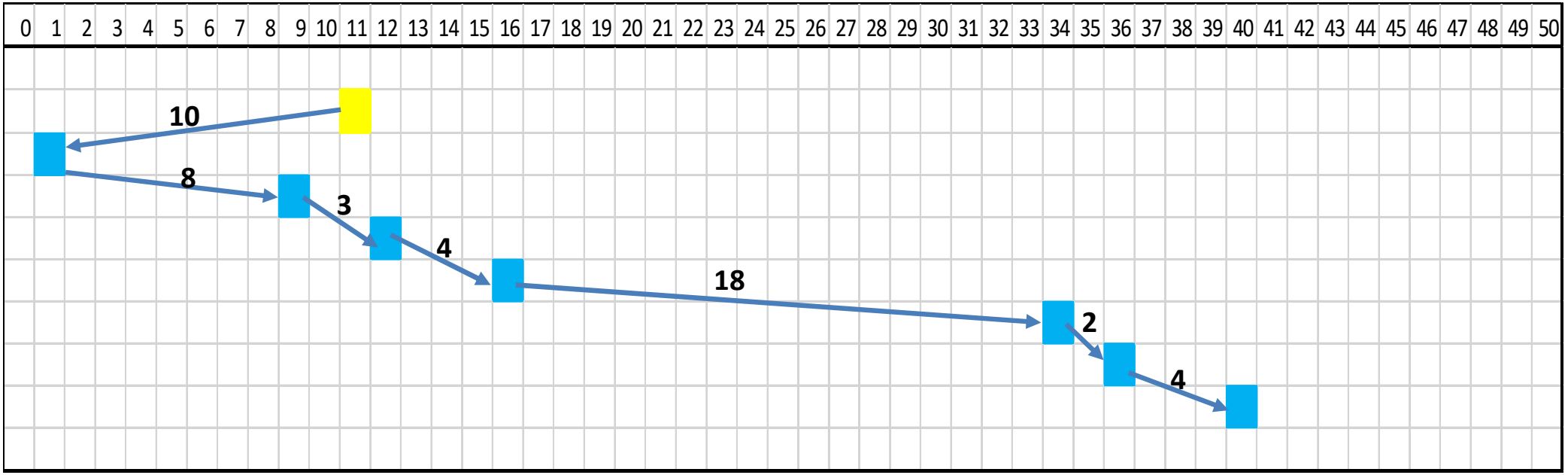
10 Marks	Consider a disk with 51(0 to 50) cylinders. While the seek to cylinder 11 is in progress, the request comes for the following cylinders, in the order 1, 36, 16, 34, 9, 12 and 40. The arm moves in an increasing number of cylinders. What is the total distance the arm moves to complete pending requests using FCFS and LOOK algorithms?

FCFS



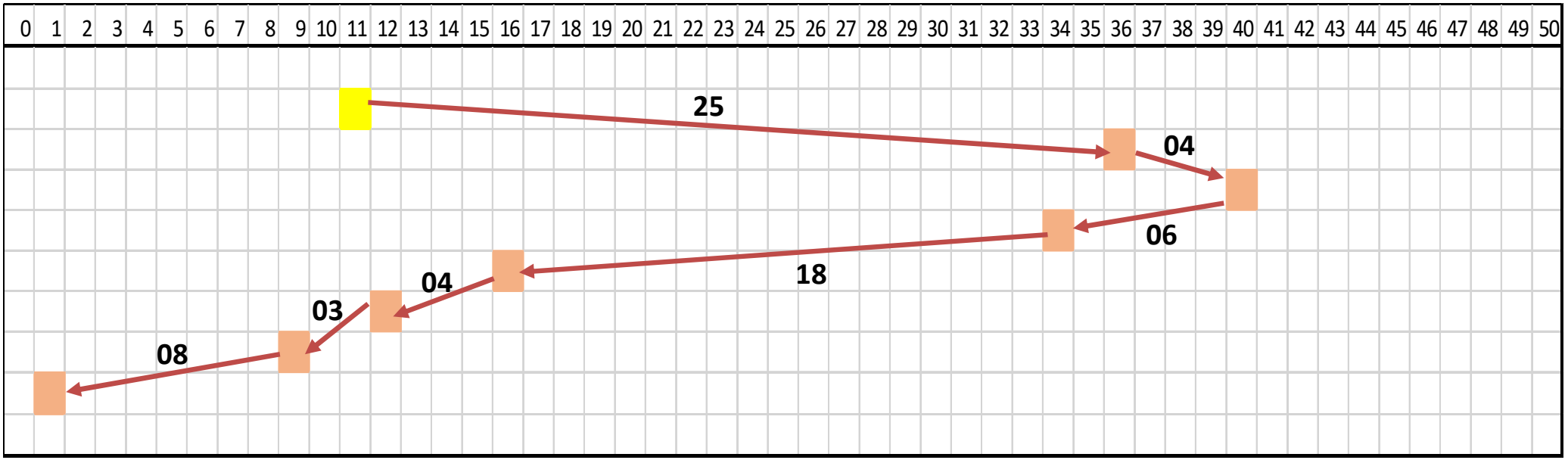
FCFS	SEEK-Time
11	
1	10
36	35
16	20
34	18
9	25
12	3
40	28
Tot	139

LOOK-Solution 1



LOOK-Soln1	
Initial Position	
11	SEEK-Time
11	
1	10
9	8
12	3
16	4
34	18
36	2
40	4
Tot Seek Time	49

LOOK Solution2



LOOK-Soln2	
Initial Position	SEEK-Time
11	
11	
36	25
40	4
34	6
16	18
12	4
9	3
1	8
Tot Seek Time	68

Thank You

D Y PATIL
DEEMED TO BE
UNIVERSITY
RAMRAO ADIK
INSTITUTE OF TECHNOLOGY
NAVI MUMBAI