OPERATING SYSTEMS

Storage Management - 1

Nitin V Pujari
Faculty, Computer Science
Dean - IQAC, PES University

OPERATING SYSTEMS

Course Syllabus - Unit 4

Unit 4 : Storage Management

Mass-Storage Structure - Mass-Storage overview, Disk Scheduling, Swap-Space Management, RAID structure. File System Interface - file organization/structure and access methods, directories, sharing. File System Implementation/Internals: File control Block (inode), partitions & mounting, Allocation methods. Case Study: Linux/Windows File Systems

OPERATING SYSTEMS

Course Outline

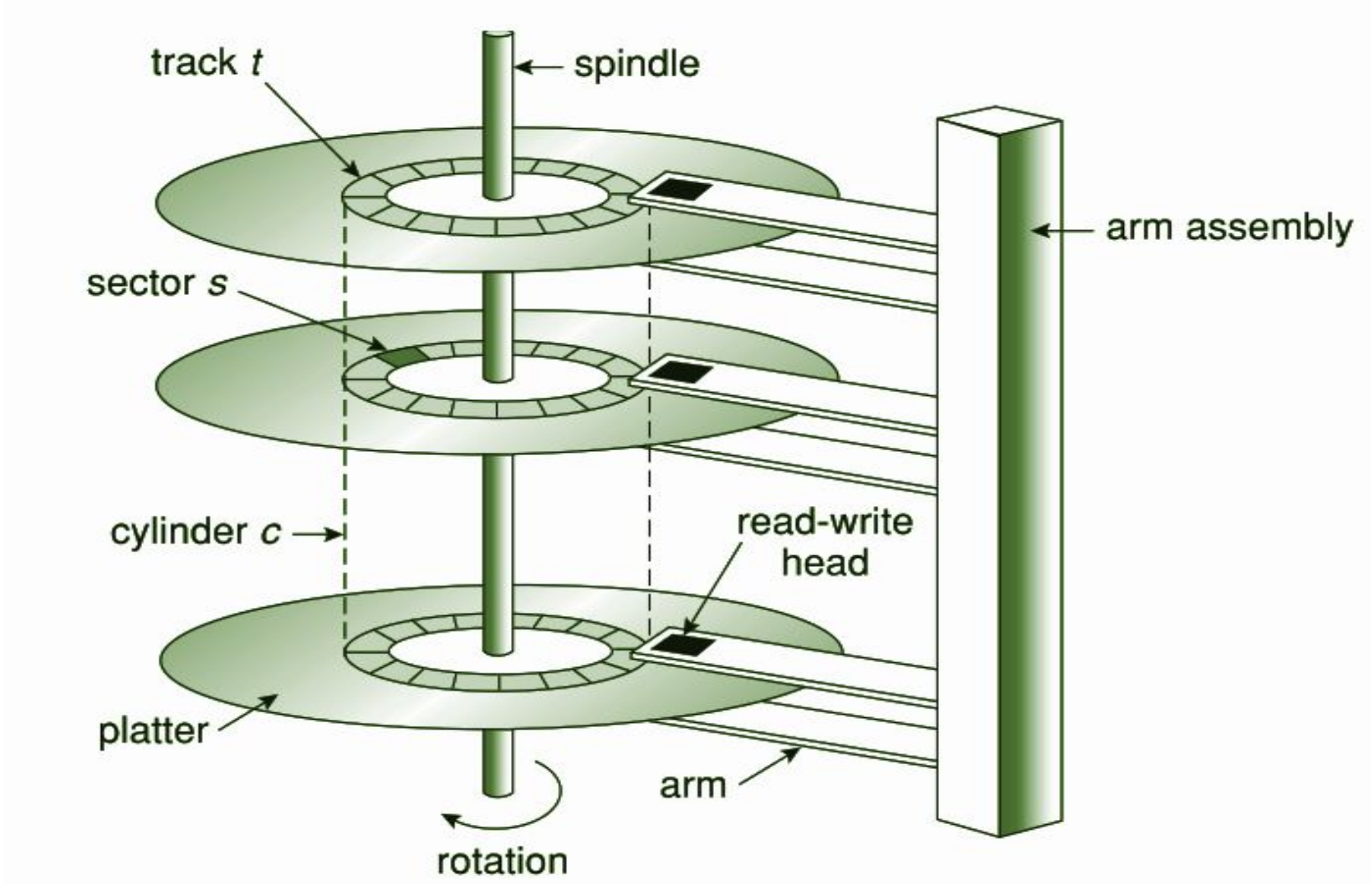
37	12.1	Mass-Storage Structure - Mass-Storage overview	12	82.1
38	12.4	Disk Scheduling	12	
39	12.4	Disk Scheduling	12	
40	12.6-12.7	Swap-Space Management, RAID structure	12	
41	10.1-10.2	File Concept, Access Methods	10	
42	10.3	Directory and Disk Structure	10	
43	10.4-10.6	File-System Mounting, File Sharing, Protection	10	
44	11.1-11.3	File-System Structure, File-System Implementation, Directory Implementation	11	
45	11.4	Allocation methods	11	
46	16.7,11.8	Case Study: Linux/Windows File Systems	11,16	

- # Overview of Mass Storage Structure

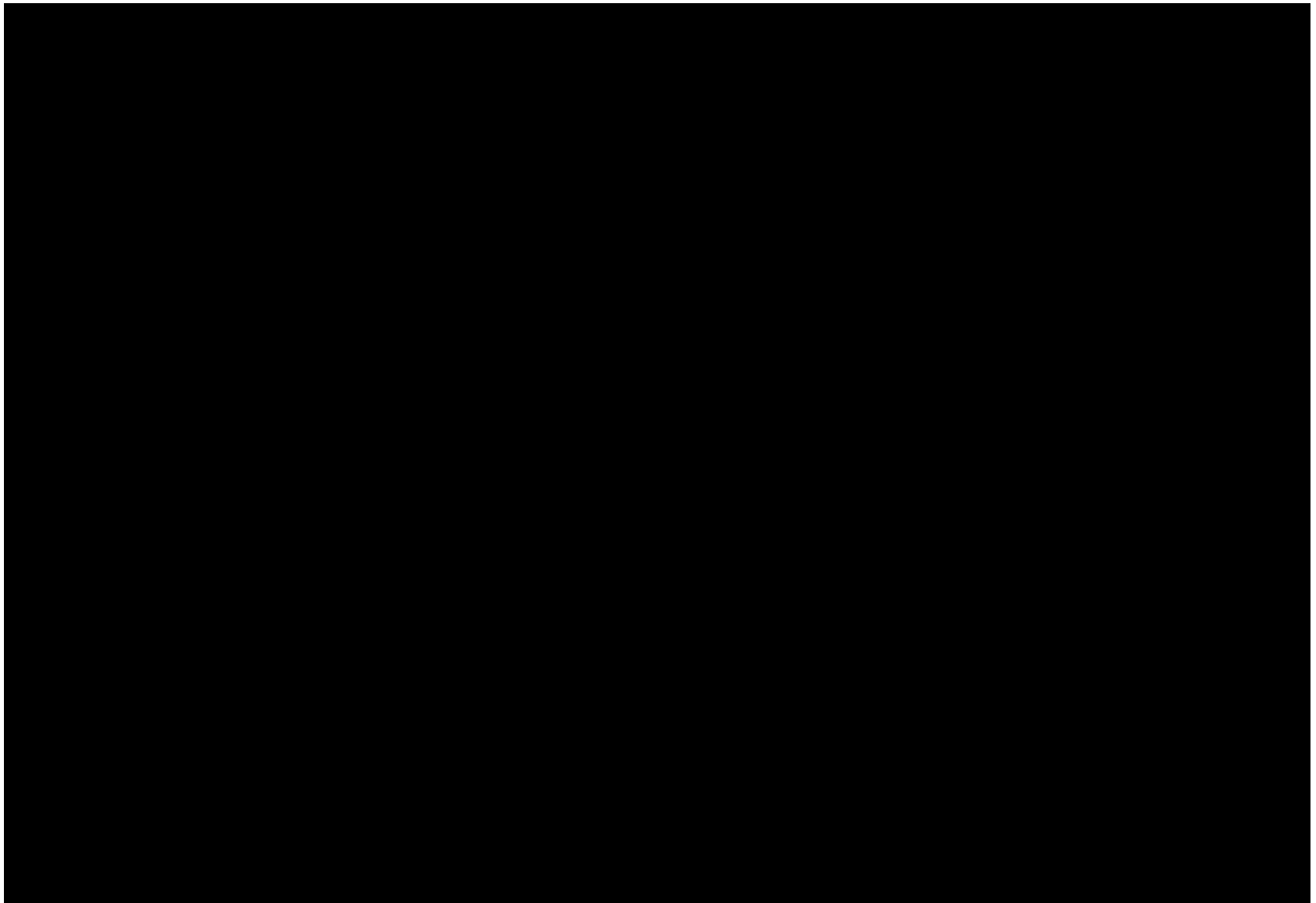
Overview of Mass Storage Structure

- Magnetic disks provide bulk of secondary storage of modern computers
- Drives rotate at 60 to 250 times per second i.e ranging from 3,600 RPM to 15,000 RPM
- Currently Hard drives have been engineered with spin rates as low as 1,200 RPM and as high as 15K RPM.
- Today's most common RPM rates, in both laptop and desktop PCs, are between 5,400 and 7,200 RPM
- Given two identically designed hard drives with the same **areal** densities, a 7,200 RPM drive will deliver data about 33% faster than the 5,400 RPM drive.
- Consequently, this specification is important when evaluating the expected performance of a hard drive or when comparing different HDD models.
- **The word "areal" refers to an area, which is an expanse of space or a region of land.**

Overview of Mass Storage Structure



Overview of Mass Storage Structure



Overview of Mass Storage Structure

- Magnetic disks provide bulk of secondary storage of modern computers
  - Drives rotate at 60 to 250 times per second
  - Transfer rate is rate at which data flow between drive and computer
  - Positioning time **(random-access time)** is time to move disk arm to desired cylinder **(seek time)** and time for desired sector to rotate under the disk head **(rotational latency)**
  - Head crash results from disk head making contact with the disk surface

Overview of Mass Storage Structure

- In general, higher RPM means equates to superior hard drive performance, but it also has a couple of drawbacks.
- In order to achieve higher rotational speeds, hard drives must draw more power to overcome increased wind resistance.
- This means higher RPM drives will put more strain on a power supply, which can add up when several high RPM drives are installed in one computer.
- Average disk drives rotate at 5400 or 7200 RPM.
- High-end drives which rotate at 10,000 or 15,000 RPM create so much wind resistance that disk memory capacity must often be compromised by shrinking the diameter of the platter in order to create a drive which runs effectively and does not draw too much power.
- Therefore, drives with very high RPM may not be economical as data storage

Overview of Mass Storage Structure

- The processing and multitasking speed of a computer is **more often limited by inadequate RAM or a slow processor, than hard drive RPM.**

- High-RPM hard drives are often used by hard core gamers and technologists that like to have computers which operate as fast as technology will allow

- For normal users a hard drive average RPM and high storage capacity should be satisfactory.

Overview of Mass Storage Structure

- Disks can be removable
- Drive attached to computer via I/O bus
- Host controller in computer uses bus to talk to disk controller built into drive or storage array
- Buses vary, including EIDE, ATA, SATA, USB, Fibre Channel, SCSI, SAS, Firewire

Overview of Mass Storage Structure

- **EIDE** : Enhanced Integrated Drive Electronics
- **ATA** : Advanced Technology Attachment
- **SATA** : Serial Advanced Technology Attachment
- **eSATA**: external Serial Advanced Technology Attachment
- **PATA** : Parallel Advanced Technology Attachment
- **USB** : Universal Serial Bus
- **Fibre Channel**
- **SCSI** : Small Computer System Interface
- **SAS** : Serial Attached SCSI
- **Firewire**

Overview of Mass Storage Structure - Additional Input

- More About **SATA**

- Serial ATA is able to move data up and down at the same time, instead of moving in the old parallel pattern. There's several versions available, with the newest 3.0 version theoretically able to move 6 Gigabytes per Second

- SATA allows hot-swapping of drives. This is handy in a server environment where machines have to be dependable

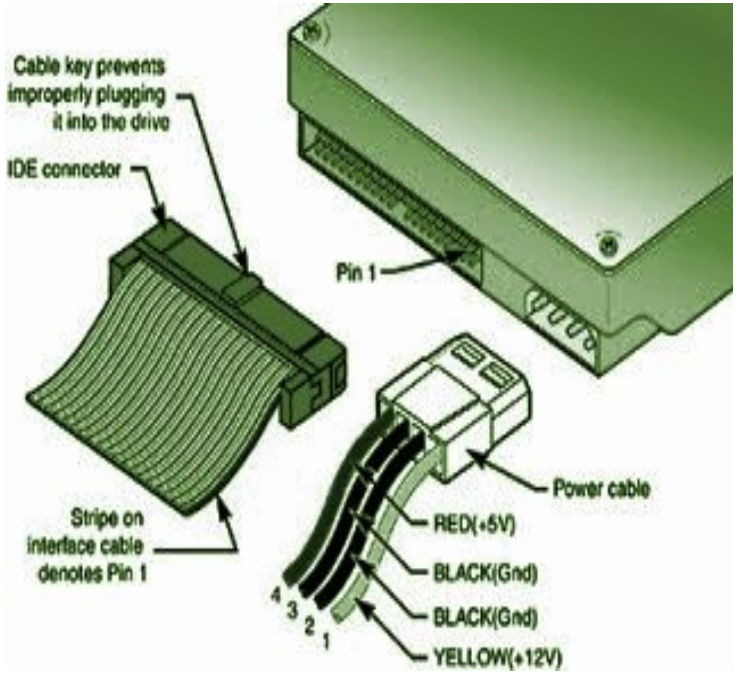
- SATA 1 came out in 2002, and 3.1 is currently being used on the new solid state drives.

    - **SATA1** was rated at about **150 MB**/Sec.
    - **SATA2** was rated at about **300 MB**/sec
    - **SATA3** was rated at about **600 MB**/sec.

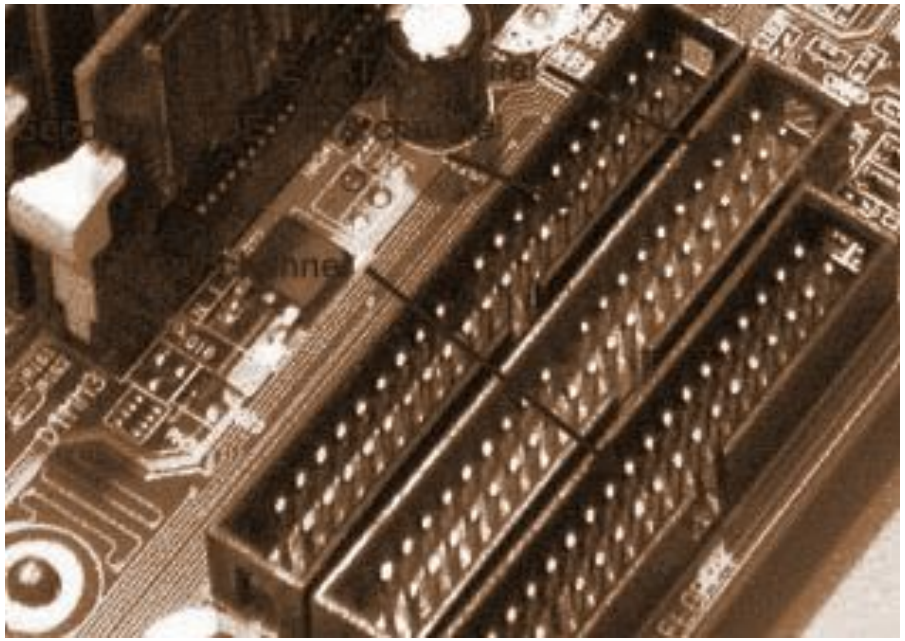
Overview of Mass Storage Structure

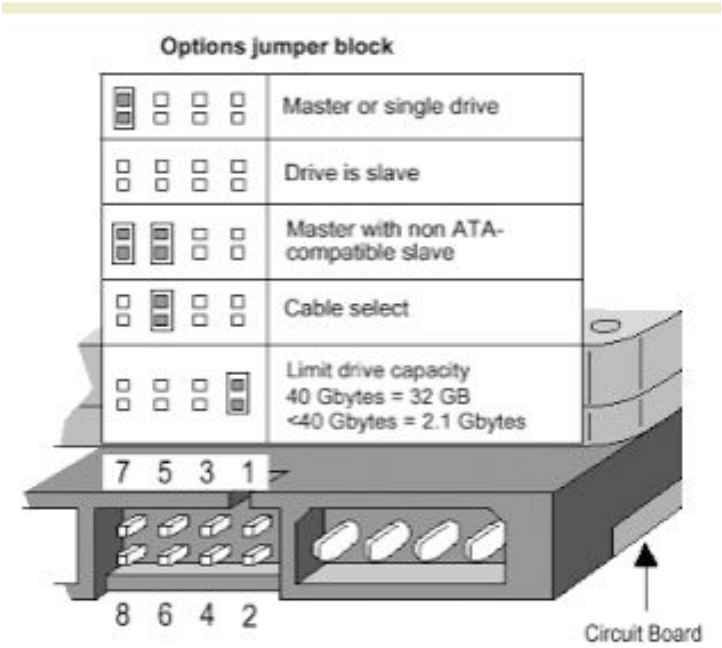
- Hard drives are rated by several important features.
- **Interface**: Serial or Parallel, internal and external. Data speeds vary.
- **Spindle Speed**: The rotation speed of magnetic drive platters
- **Storage Capacity**: Capacity available on new drives is constantly changing, but the price per unit of storage is dropping on magnetic drives. Solid state drives cost more per unit of capacity, but this is changing too.
- **Cache/Buffer size**: Buffers improve performance of drives by allowing a little leeway while data is being transferred. Cost goes up with buffer size. Some Hybrid drive setups use the Solid State portion as a large buffer to the magnetic drive.
- **Seek Time**: How fast is the data found and retrieved ? Cost goes up with performance. Solid State Drives have greatly improved seek time over magnetic drives.

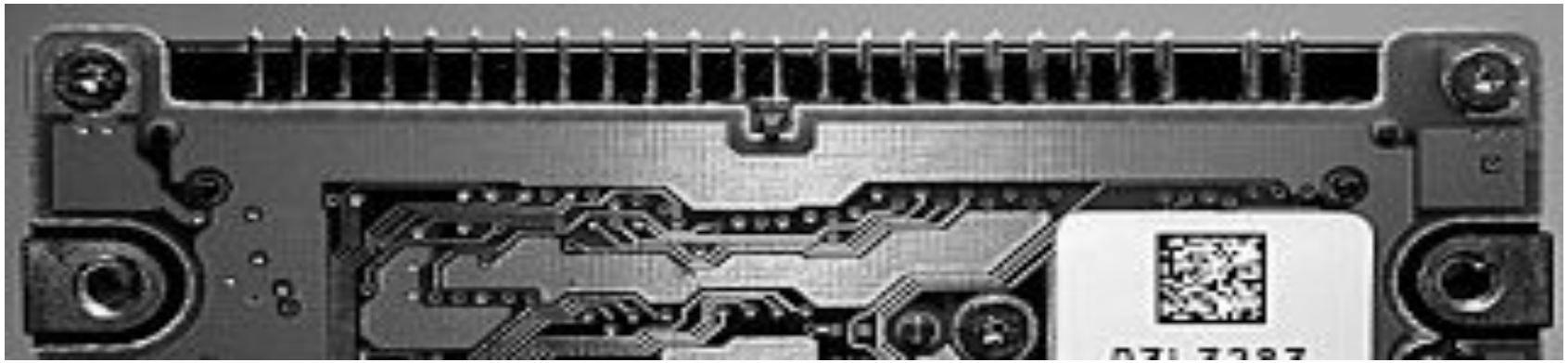
Overview of Storage Connectors - Additional Input



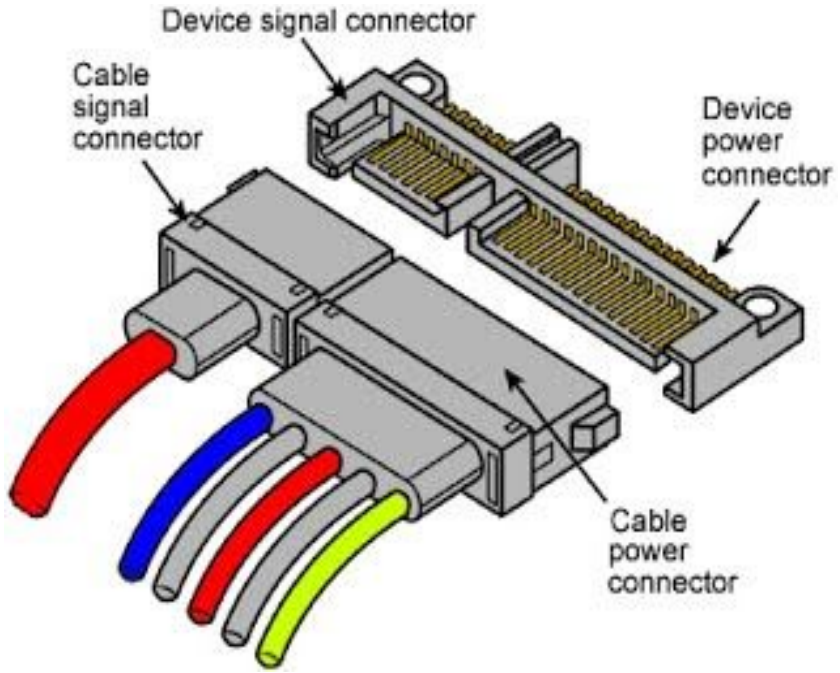
Overview of Storage Connectors - Additional Input

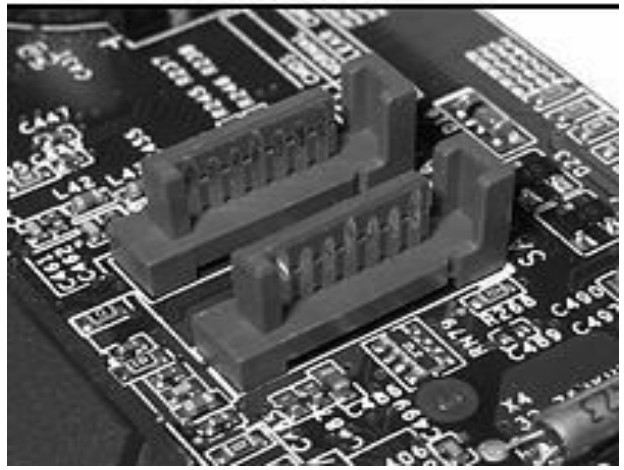


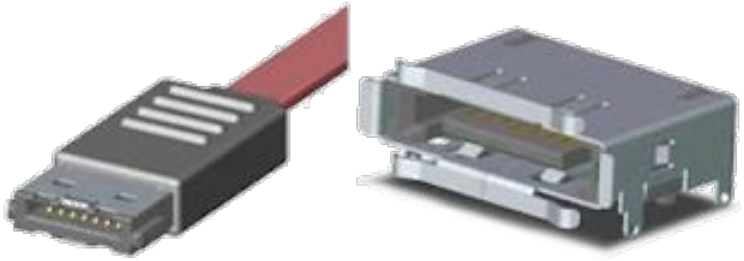


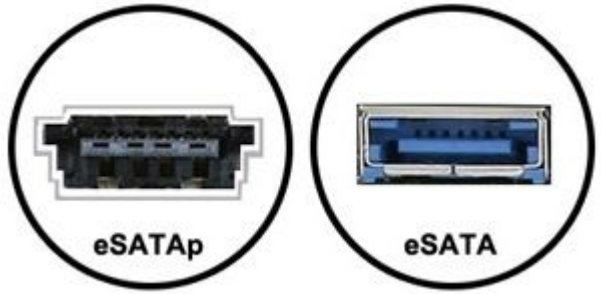


Overview of Storage Connectors - Additional Input









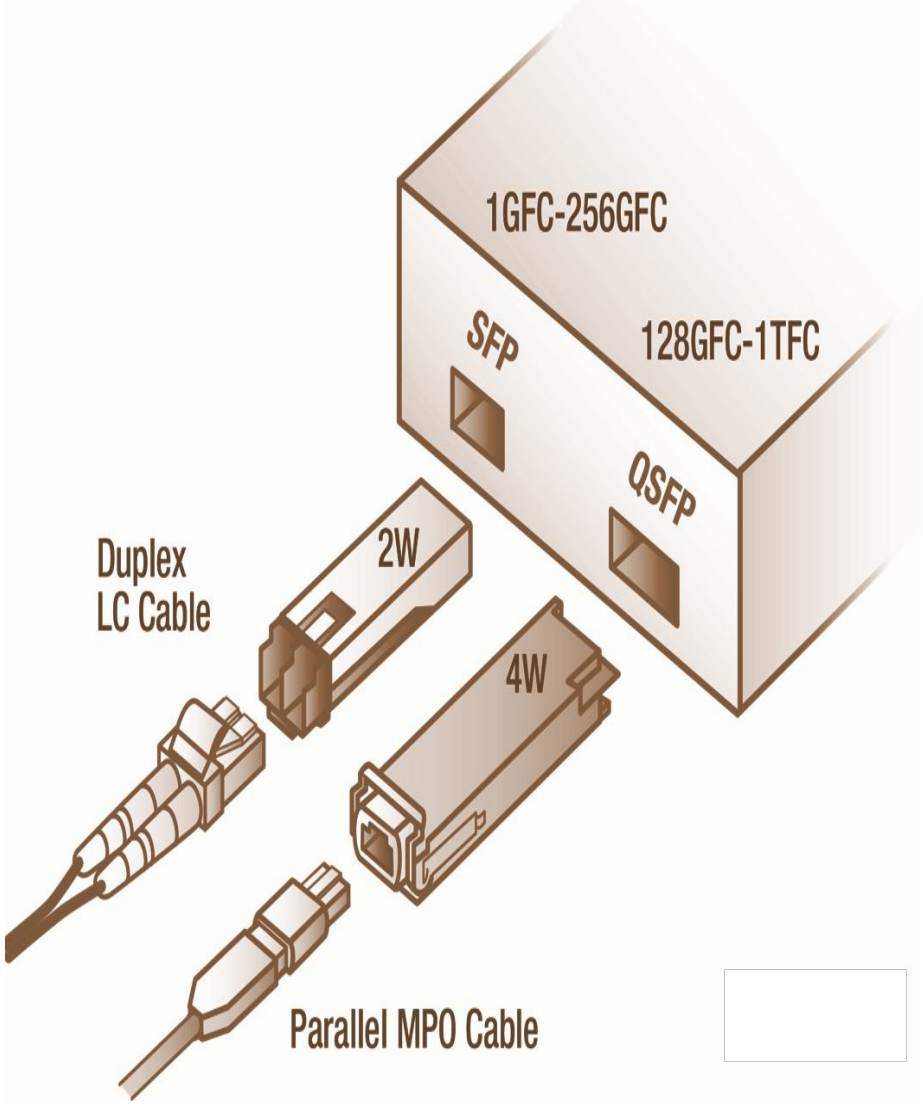
Overview of Storage Connectors - Additional Input

- **Fibre Channel** is a technology for transmitting data between computer devices at data rates of up to 4 Gbps (and 10 Gbps in the near future).
- **Fibre Channel** is especially suited for connecting computer servers to shared storage devices and for interconnecting storage controllers and drives.
- Since **Fibre Channel** is three times as fast, it has begun to replace the Small Computer System Interface (SCSI) as the transmission interface between servers and clustered storage devices.
- **Fibre Channel** is more flexible; devices can be as far as ten kilometers (about six miles) apart if optical fiber is used as the physical medium.
- Optical fiber is not required for shorter distances, however, because Fibre Channel also works using coaxial cable and ordinary telephone twisted pair.
- **Fibre Channel** technology is used with server storage networks

Overview of Storage Connectors - Additional Input

- **Fibre Channel** offers point-to-point, switched, and loop interfaces.
- It is designed to interoperate with SCSI, the Internet Protocol (IP) and other protocols, but has been criticized for its lack of compatibility - primarily because (like in the early days of SCSI technology) manufacturers sometimes interpret specifications differently and vary their implementations.
- Standards for **Fibre Channel** are specified by the Fibre Channel Physical and Signalling standard, and the ANSI X3.230-1994, which is also ISO 14165-1. (Rouse, 2005)
- **Why have the buffer ? Why not just go from the disk straight to the memory?**
- **Speed matching.** The disk supplies data at a fixed rate, which might exceed the rate the memory can accept it. In particular the memory might be busy servicing a request from the processor or from another DMA controller.

Overview of Storage Connectors - Additional Input



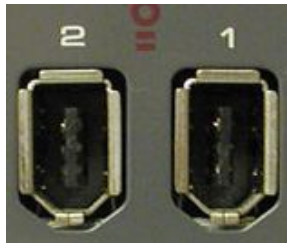
- **SFP:** The **S**mall **F**orm-factor **P**luggable transceiver
- **QSFP:** The **Q**uad **S**mall **F**orm-factor **P**luggable transceiver
- **LC :** **L**ucent **C**onnector
- **GFC :** **G**eneric **F**low **C**ontrol
- **TFC:** The **T**erabit **F**ibre **C**hannel
- **MPO:** **M**ulti-fibre **P**ush **O**n

Overview of Storage Connectors - Additional Input

- **FireWire** is a method of transferring information between digital devices, especially audio and video equipment.

- Also known as IEEE 1394, **FireWire** is fast => the latest version achieves speeds up to 800 Mbps. At some time in the future, that number is expected to jump to an unbelievable 3.2 Gbps when manufacturers overhaul the current **FireWire** cables.

- You can connect up to **63 devices** to a **FireWire** bus. Windows operating systems (98 and later) and Mac OS (8.6 and later) both support it.



- Overview of Mass Storage Structure

Topic Outline

- Disk Scheduling
- First Come First Serve Disk Scheduling - FCFS
- Shortest Seek Time First Disk Scheduling - SSTF

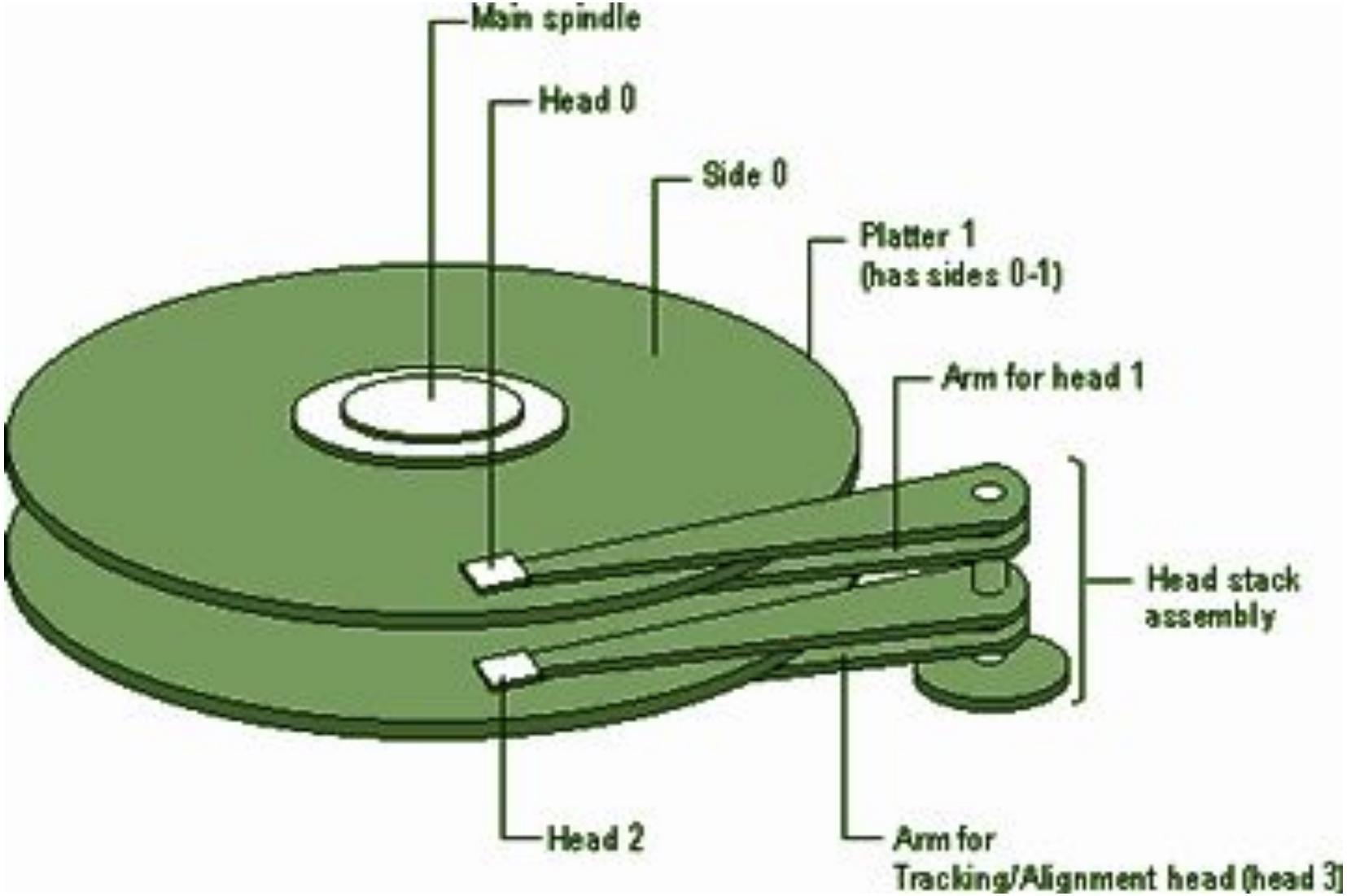
Disk Scheduling

- 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 is correlated to seek distance
- 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
- There are many sources of disk I/O request
  - OS
  - System processes
  - Users processes

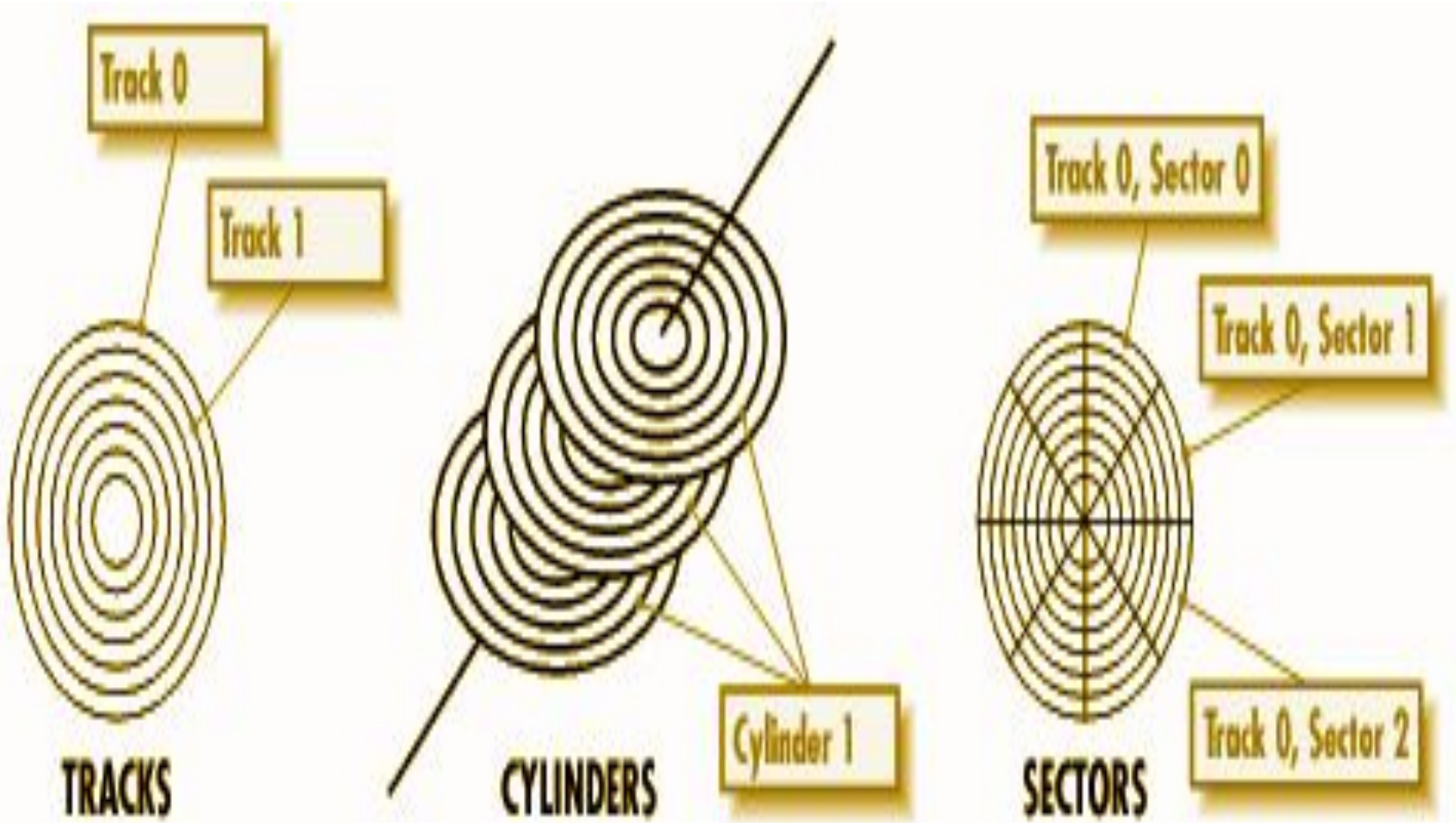
Disk Scheduling

- 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
- Note that drive controllers have small buffers and can manage a queue of I/O requests (of varying “depth”)
- Several algorithms exist to schedule the servicing of disk I/O requests
- The analysis is true for one or many platters
- Seek Distance is in terms of **Cylinders**

Disk Scheduling



Disk Scheduling



Disk Scheduling

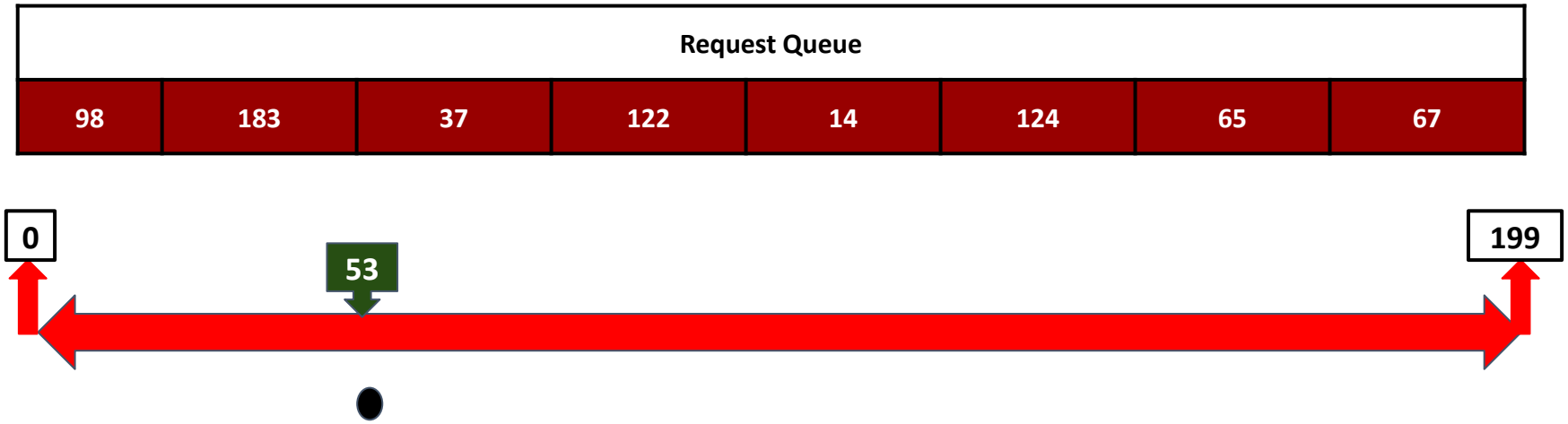
- We illustrate scheduling algorithms with a request queue (0-199)

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

    - Head pointer 53

First Come First Serve - FCFS Disk Scheduling

- Request Queue => 0 .. 199
- 98, 183, 37, 122, 14, 124, 65, 67
- Head pointer initially @ 53 currently pointing @98



Seek Distance = 0

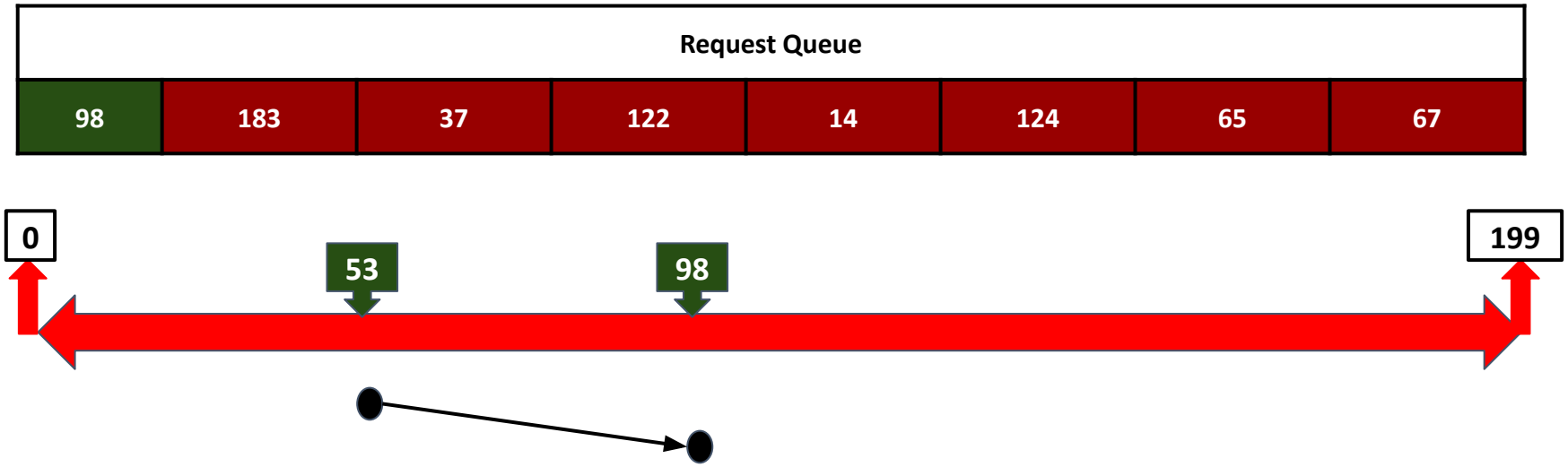
Seek Distance = Seek Distance + abs (Current Cylinder Number - New Cylinder Number)

Seek Distance = 0 + abs()

Seek Distance =

First Come First Serve - FCFS Disk Scheduling

- Request Queue => 0 .. 199
- 98, 183, 37, 122, 14, 124, 65, 67



Seek Distance = 0

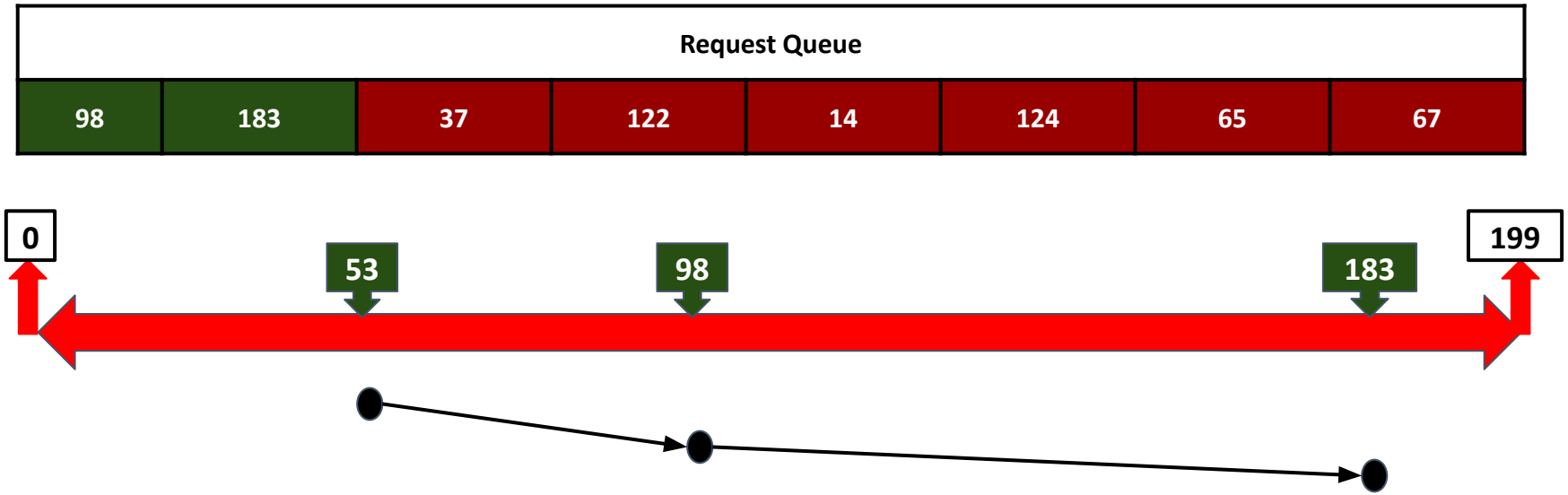
Seek Distance = Seek Distance + abs (Current Cylinder Number - New Cylinder Number)

Seek Distance = 0 + abs(53 - 98)

Seek Distance = 45

First Come First Serve - FCFS Disk Scheduling

- Request Queue => 0 .. 199
- 98, 183, 37, 122, 14, 124, 65, 67

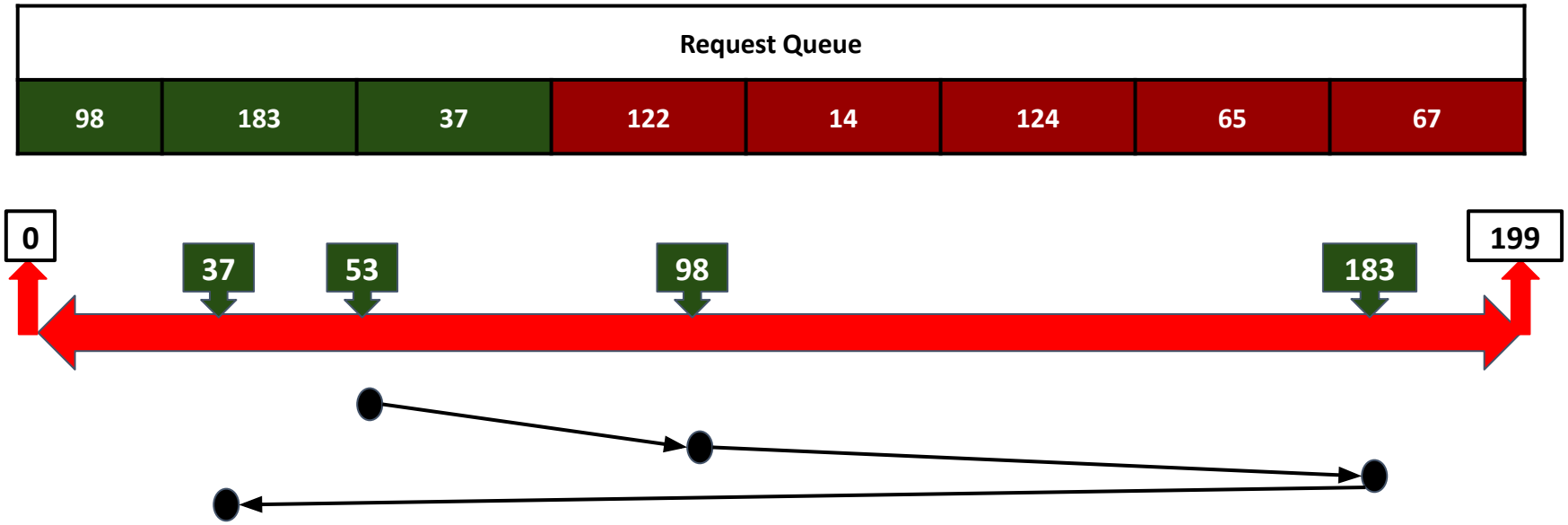


Seek Distance = 45

Seek Distance = Seek Distance + abs (Current Cylinder Number - New Cylinder Number)
Seek Distance = 45 + abs(98 - 183)
Seek Distance = 130

First Come First Serve - FCFS Disk Scheduling

- Request Queue => 0 .. 199
- 98, 183, 37, 122, 14, 124, 65, 67



Seek Distance = 130

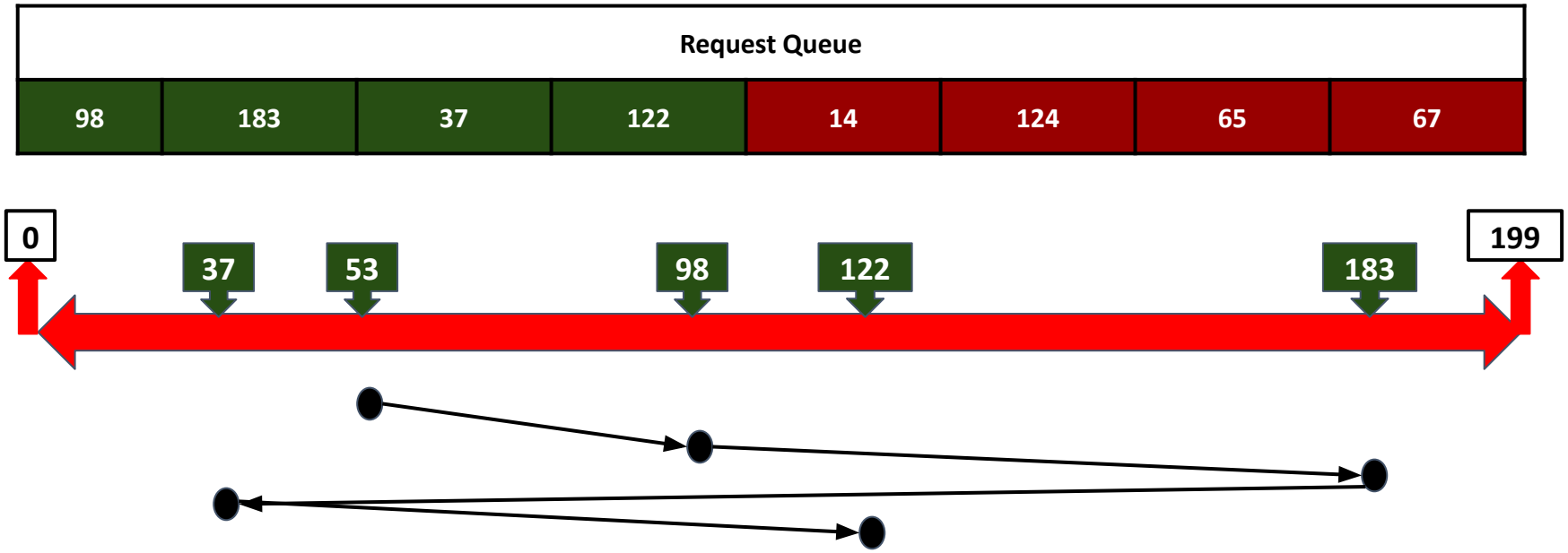
Seek Distance = Seek Distance + abs (Current Cylinder Number - New Cylinder Number)

Seek Distance = 130 + abs(183-37)

Seek Distance = 276

First Come First Serve - FCFS Disk Scheduling

- Request Queue => 0 .. 199
- 98, 183, 37, 122, 14, 124, 65, 67



Seek Distance = 276

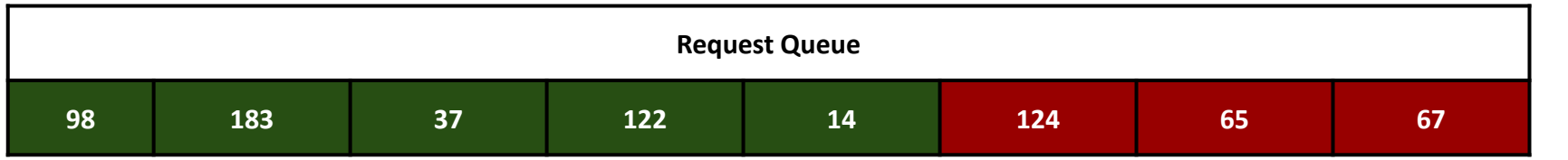
Seek Distance = Seek Distance + abs (Current Cylinder Number - New Cylinder Number)

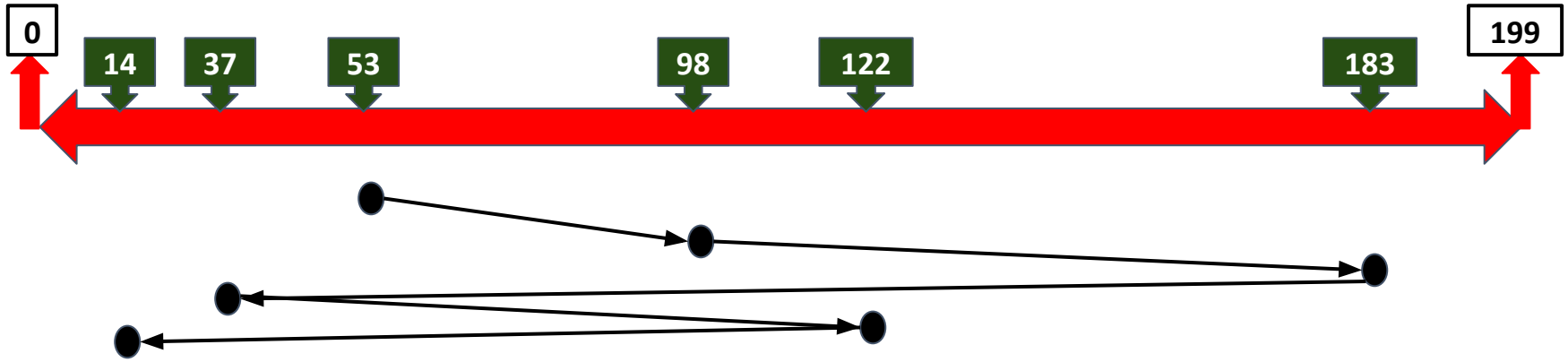
Seek Distance = 276 + abs(37-122)

Seek Distance = 361

First Come First Serve - FCFS Disk Scheduling

- Request Queue => 0 .. 199
- 98, 183, 37, 122, 14, 124, 65, 67



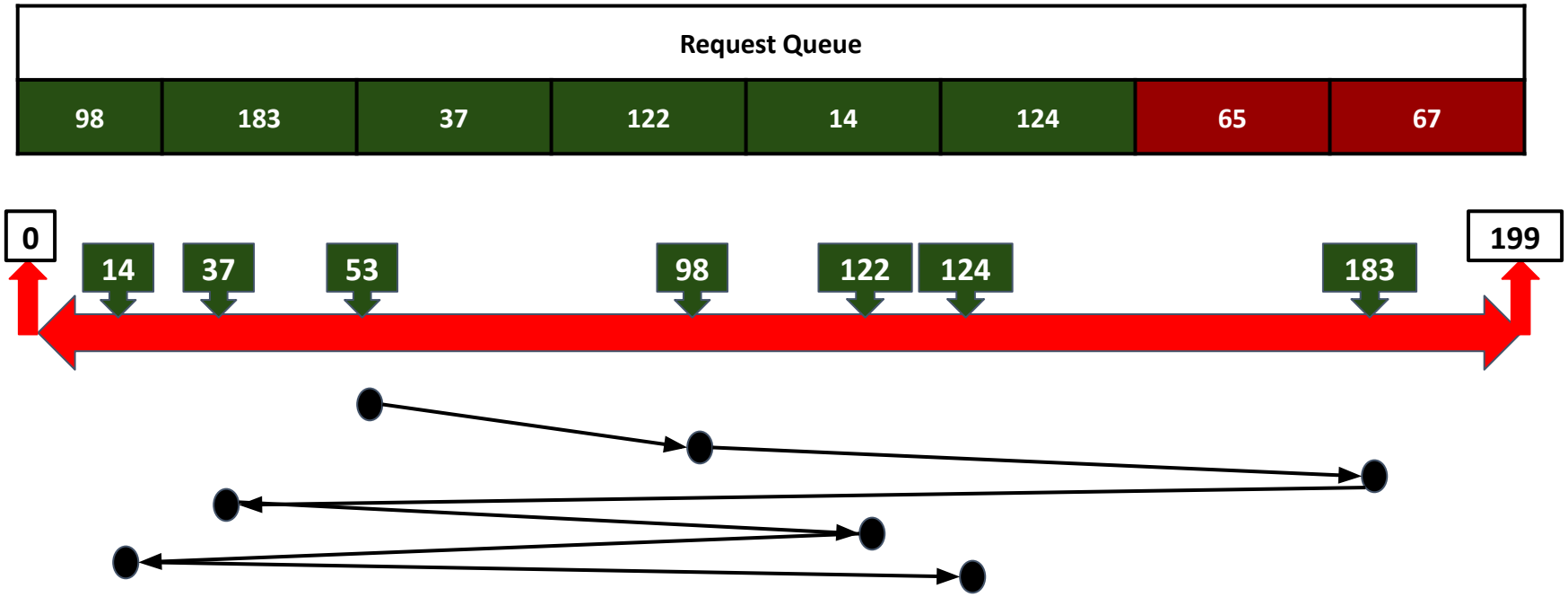


Seek Distance = 361

Seek Distance = Seek Distance + abs (Current Cylinder Number - New Cylinder Number)
Seek Distance = 361 + abs(122-14)
Seek Distance = 469

First Come First Serve - FCFS Disk Scheduling

- Request Queue => 0 .. 199
- 98, 183, 37, 122, 14, 124, 65, 67



Seek Distance = 469

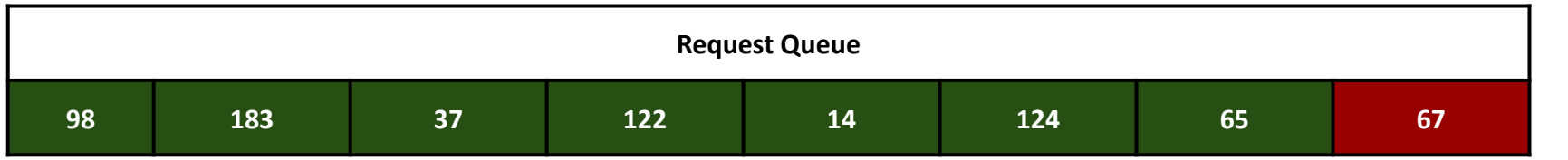
Seek Distance = Seek Distance + abs (Current Cylinder Number - New Cylinder Number)

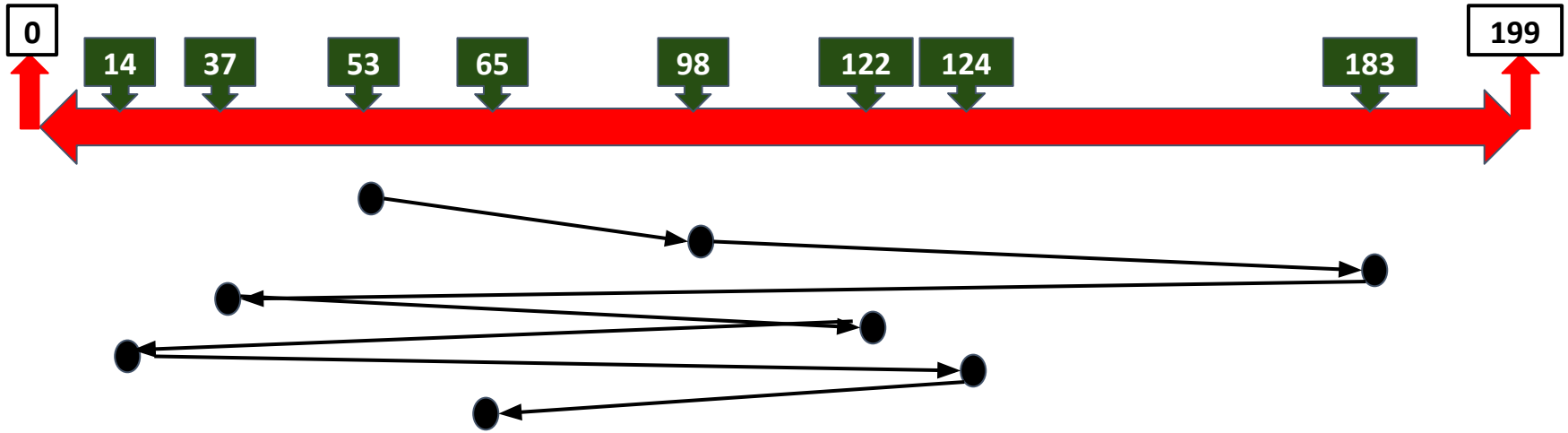
Seek Distance = 469 + abs(14-124)

Seek Distance = 579

First Come First Serve - FCFS Disk Scheduling

- Request Queue => 0 .. 199
- 98, 183, 37, 122, 14, 124, 65, 67





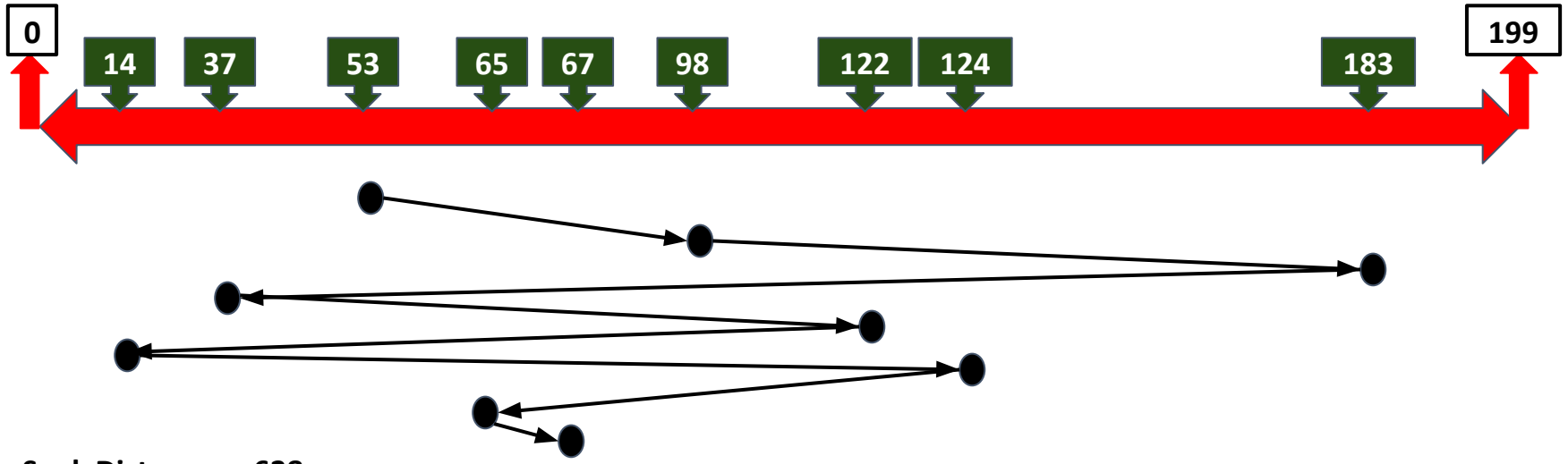
Seek Distance = 579

Seek Distance = Seek Distance + abs (Current Cylinder Number - New Cylinder Number)
Seek Distance = 579 + abs(124-65)
Seek Distance = 638

First Come First Serve - FCFS Disk Scheduling

- Request Queue => 0 .. 199
- 98, 183, 37, 122, 14, 124, 65, 67

Request Queue							
98	183	37	122	14	124	65	67



Final Seek Count in terms of Cylinders

640

Seek Distance = 638

Seek Distance = Seek Distance + abs (Current Cylinder Number - New Cylinder Number)
Seek Distance = 638 + abs(65-67)
Seek Distance = 640

Shortest Seek Time First - SSTF Disk Scheduling

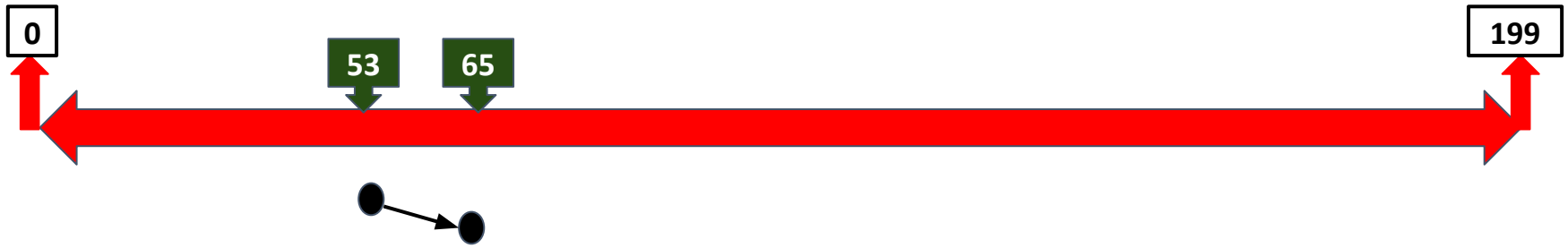
- Shortest Seek Time First selects the request with the minimum seek time from the current head position

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

Shortest Seek Time First - SSTF Disk Scheduling

- Request Queue => 0 .. 199
- 98, 183, 37, 122, 14, 124, 65, 67
- Head pointer initially @ 53

	Request							
Distance from Current Head Position @ 53	98	183	37	122	14	124	65	67
	45	130	16	69	39	71	12	14



Seek Distance = 0

Seek Distance = Seek Distance + abs (Current Cylinder Number - New Cylinder Number)

Seek Distance =0 + abs(53-65)

Seek Distance = 12

Shortest Seek Time First - SSTF Disk Scheduling

- Request Queue => 0 .. 199
- 98, 183, 37, 122, 14, 124, 65, 67
- Head pointer initially @ 53

	Request							
Distance from Current Head Position @ 65	98	183	37	122	14	124	65	67
	33	118	28	57	51	59	S	2

Seek Distance = 12

Seek Distance = Seek Distance + abs (Current Cylinder Number - New Cylinder Number)
Seek Distance =12 + abs(65-67)
Seek Distance = 14

Shortest Seek Time First - SSTF Disk Scheduling

- Request Queue => 0 .. 199
- 98, 183, 37, 122, 14, 124, 65, 67
- Head pointer initially @ 53

	Request							
Distance from Current Head Position @ 67	98	183	37	122	14	124	65	67
	31	116	30	55	53	57	S	S

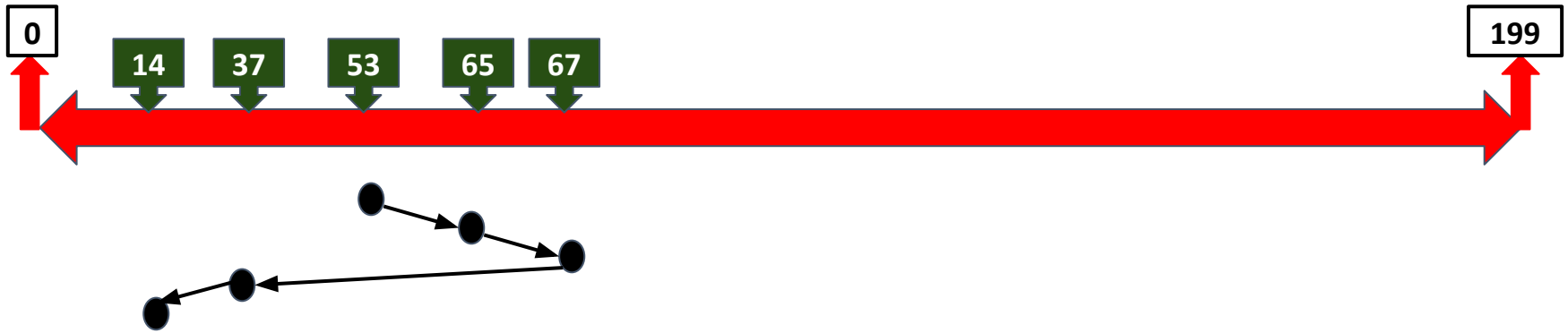
Seek Distance = 14

Seek Distance = Seek Distance + abs (Current Cylinder Number - New Cylinder Number)
Seek Distance =14 + abs(67-37)
Seek Distance = 44

Shortest Seek Time First - SSTF Disk Scheduling

- Request Queue => 0 .. 199
- 98, 183, 37, 122, 14, 124, 65, 67
- Head pointer initially @ 53

	Request							
Distance from Current Head Position @ 37	98	183	37	122	14	124	65	67
	61	146	S	85	23	87	S	S



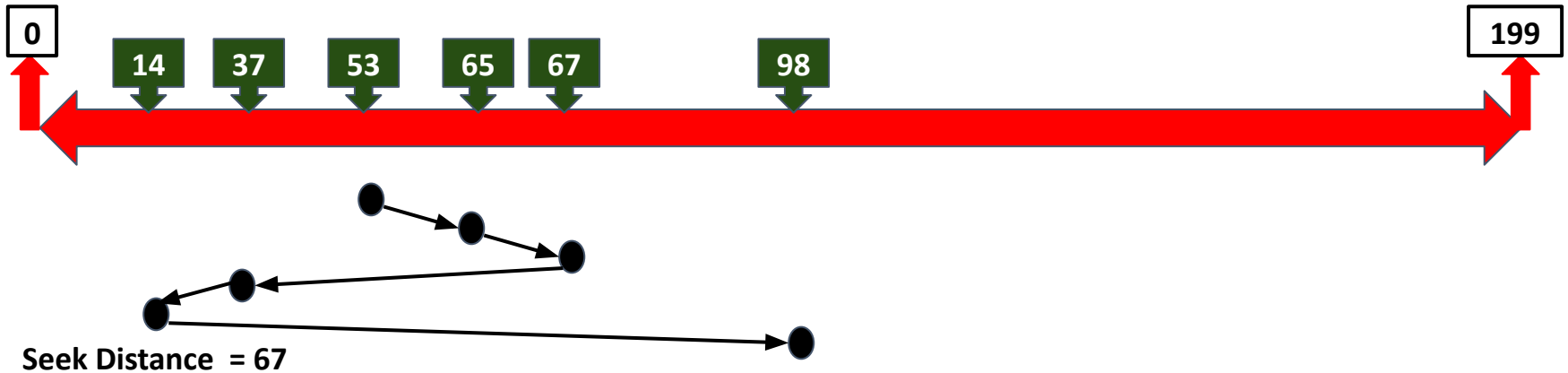
Seek Distance = 44

Seek Distance = Seek Distance + abs (Current Cylinder Number - New Cylinder Number)
Seek Distance =44 + abs(37-14)
Seek Distance = 67

Shortest Seek Time First - SSTF Disk Scheduling

- Request Queue => 0 .. 199
- 98, 183, 37, 122, 14, 124, 65, 67
- Head pointer initially @ 53

	Request							
Distance from Current Head Position @ 14	98	183	37	122	14	124	65	67
	84	169	S	108	S	110	S	S



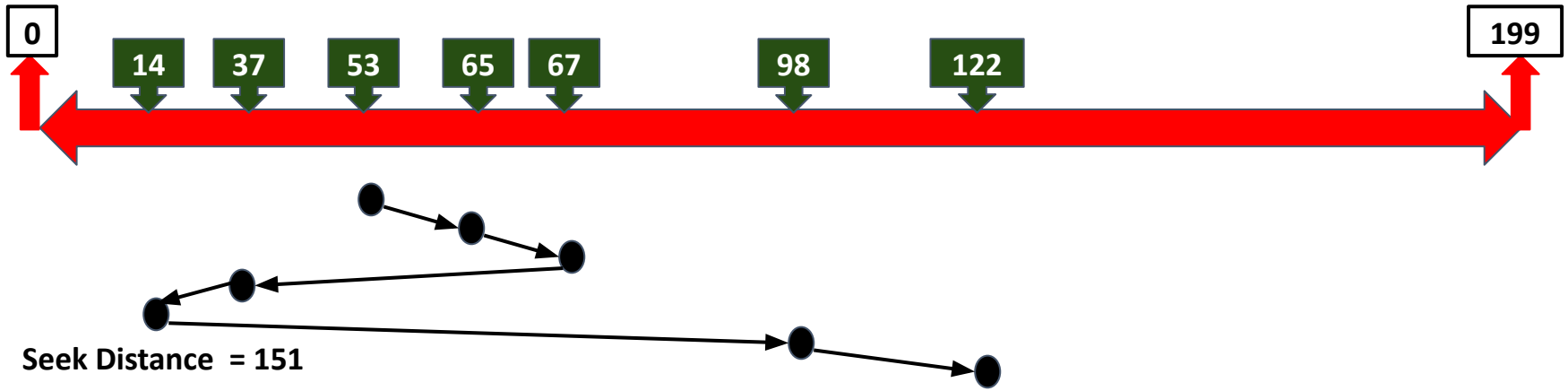
Seek Distance = 67

Seek Distance = Seek Distance + abs (Current Cylinder Number - New Cylinder Number)
Seek Distance =67 + abs(14-98)
Seek Distance = 151

Shortest Seek Time First - SSTF Disk Scheduling

- Request Queue => 0 .. 199
- 98, 183, 37, 122, 14, 124, 65, 67
- Head pointer initially @ 53

	Request							
Distance from Current Head Position @ 98	98	183	37	122	14	124	65	67
	S	85	S	24	S	26	S	S



Seek Distance = 151

Seek Distance = Seek Distance + abs (Current Cylinder Number - New Cylinder Number)

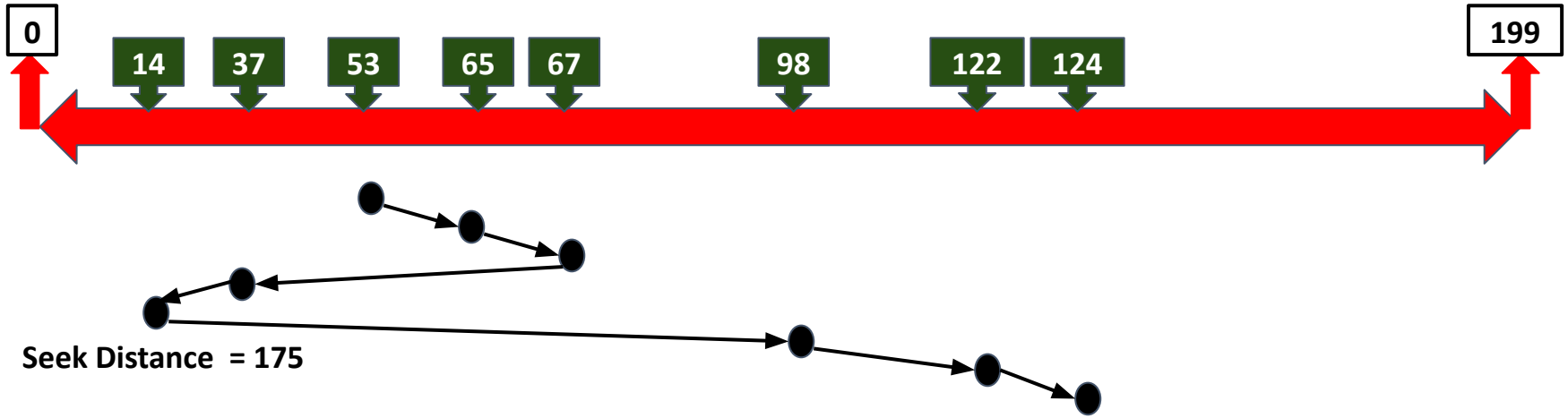
Seek Distance =151 + abs(98-122)

Seek Distance = 175

Shortest Seek Time First - SSTF Disk Scheduling

- Request Queue => 0 .. 199
- 98, 183, 37, 122, 14, 124, 65, 67
- Head pointer initially @ 53

	Request							
Distance from Current Head Position @ 122	98	183	37	122	14	124	65	67
	S	61	S	S	S	2	S	S



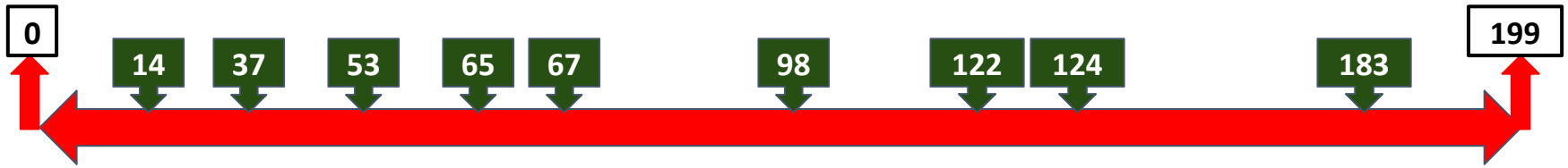
Seek Distance = 175

Seek Distance = Seek Distance + abs (Current Cylinder Number - New Cylinder Number)
Seek Distance =175 + abs(122-124)
Seek Distance = 177

Shortest Seek Time First - SSTF Disk Scheduling

- Request Queue => 0 .. 199
- 98, 183, 37, 122, 14, 124, 65, 67
- Head pointer initially @ 53

	Request							
Distance from Current Head Position @ 124	98	183	37	122	14	124	65	67
	S	59	S	S	S	S	S	S



Seek Distance = 177

Seek Distance = Seek Distance + abs (Current Cylinder Number - New Cylinder Number)
Seek Distance =177 + abs(124-183)
Seek Distance = 236

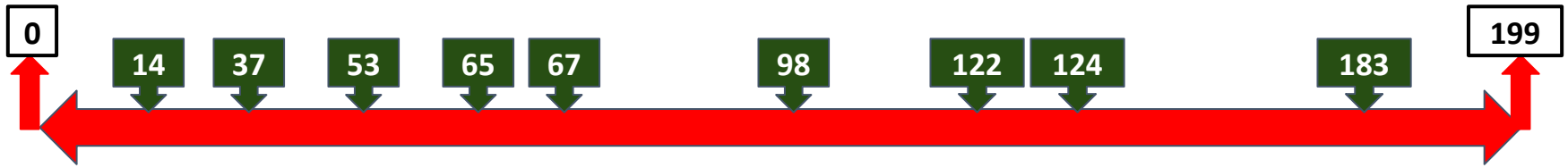
Shortest Seek Time First - SSTF Disk Scheduling

- Request Queue => 0 .. 199
- 98, 183, 37, 122, 14, 124, 65, 67
- Head pointer initially @ 53

	Request							
Distance from Current Head Position @ 183	98	183	37	122	14	124	65	67
	S	S	S	S	S	S	S	S

Final Seek Count in terms of Cylinders

236



Seek Distance = 177

Seek Distance = Seek Distance + abs (Current Cylinder Number - New Cylinder Number)
Seek Distance =177 + abs(124-183)
Seek Distance = 236

Topics Uncovered Today

- Disk Scheduling
- First Come First Serve Disk Scheduling - FCFS
- Shortest Seek Time First Disk Scheduling - SSTF

SCAN Disk Scheduling

- 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, where the head movement is reversed and servicing continues.

- SCAN algorithm is sometimes also called the **elevator algorithm**

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

SCAN Disk Scheduling

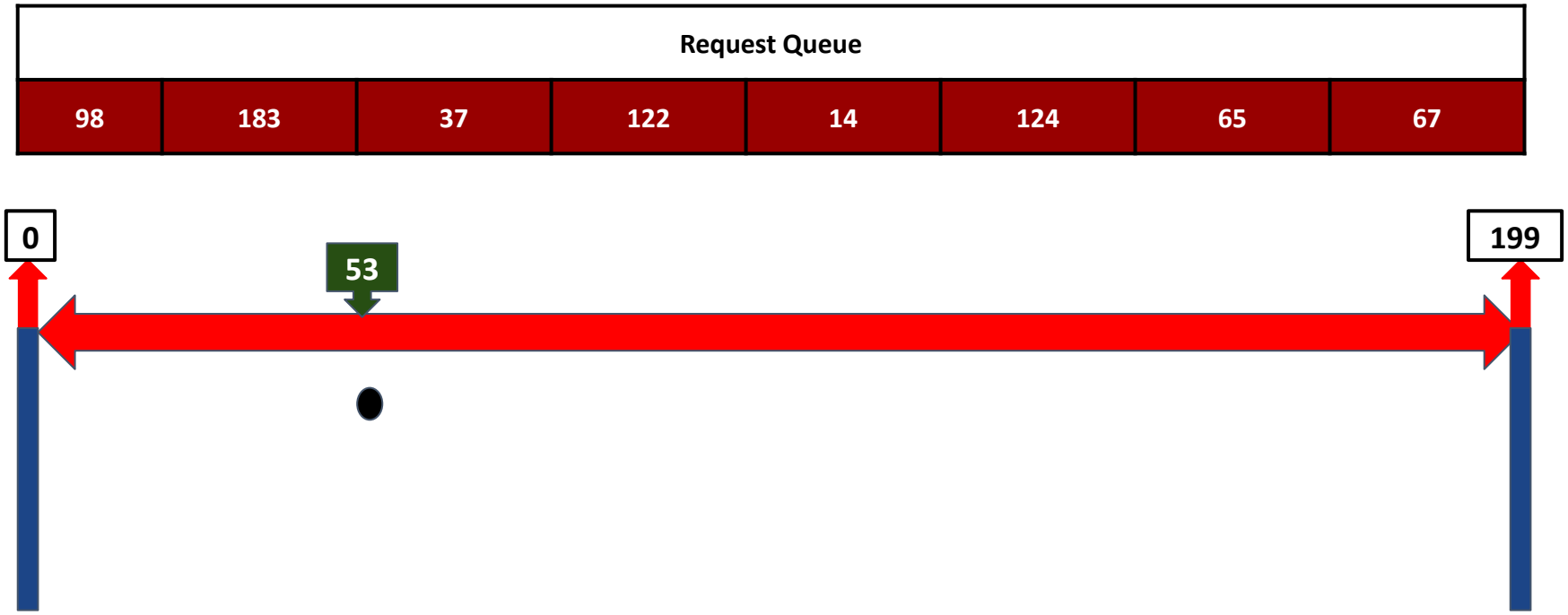
- We illustrate scheduling algorithms with a request queue (0-199)

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

    - Head pointer 53

SCAN Disk Scheduling

- Request Queue => 0 .. 199
- 98, 183, 37, 122, 14, 124, 65, 67
- Head pointer initially @ 53

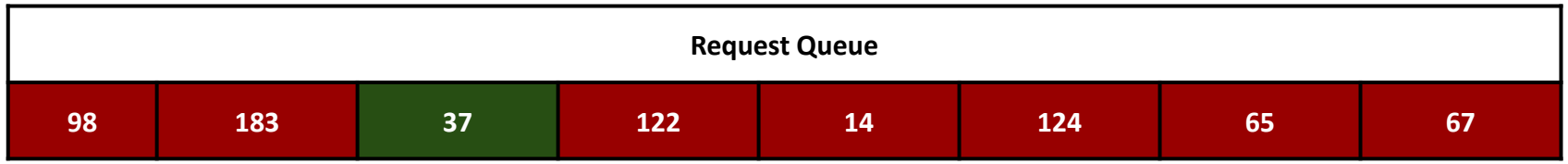


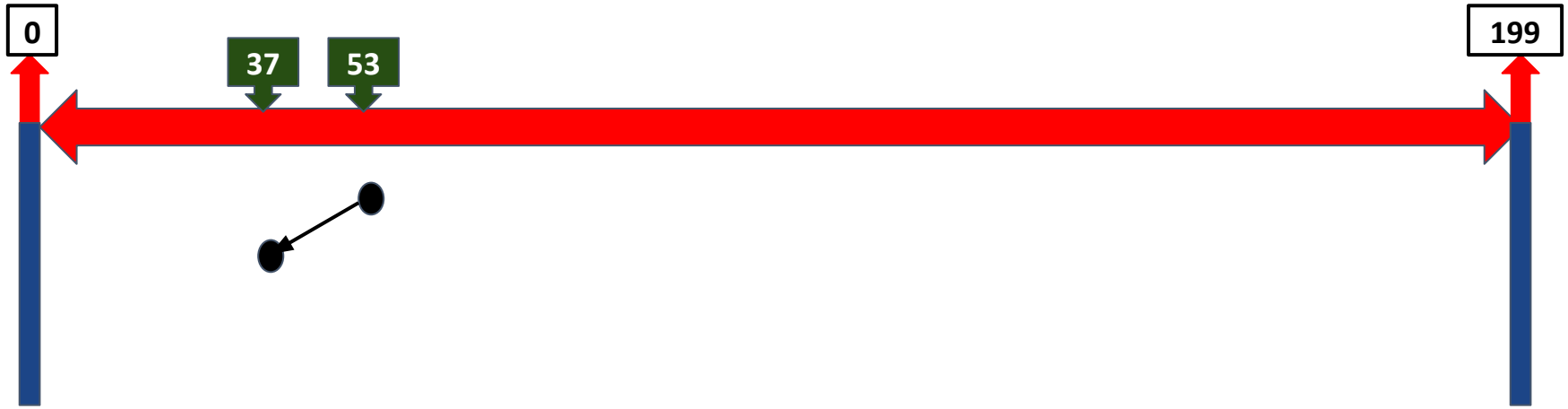
Seek Distance = 0

Seek Distance = Seek Distance + abs (Current Cylinder Number - New Cylinder Number)
Seek Distance = 0 + abs()
Seek Distance = 0

SCAN Disk Scheduling

- Request Queue => 0 .. 199
- 98, 183, 37, 122, 14, 124, 65, 67
- Head pointer initially @ 53



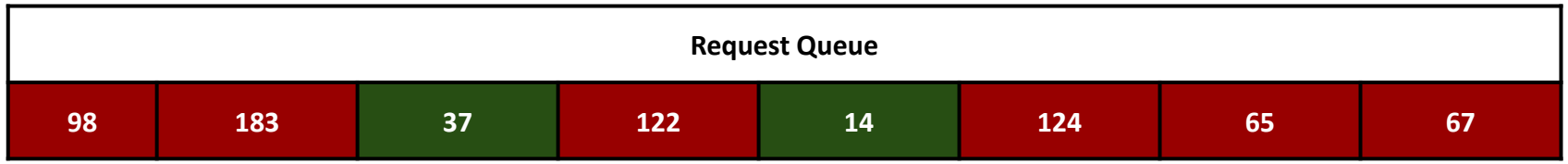


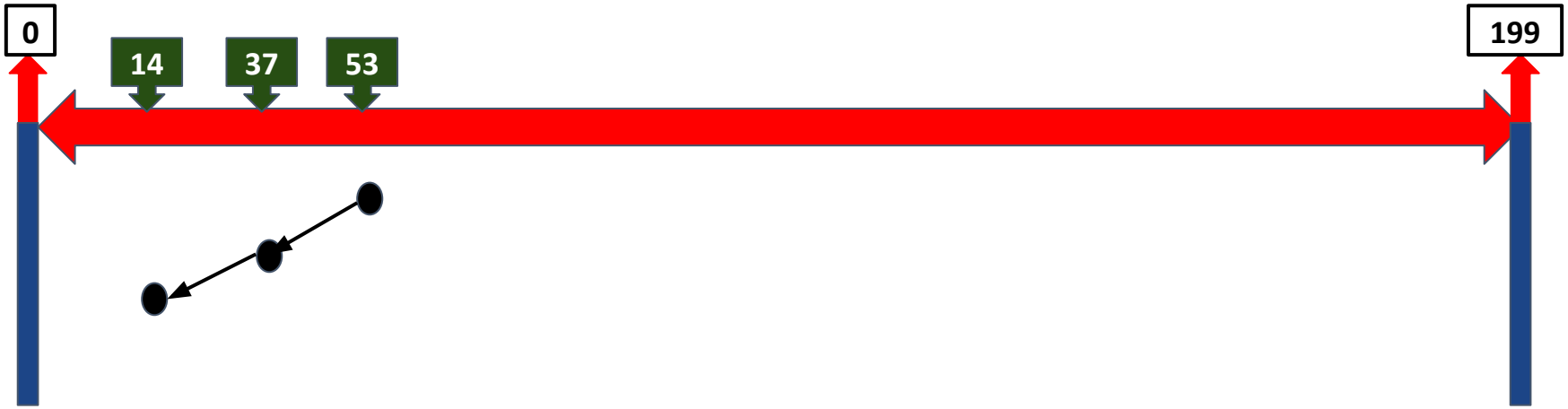
Seek Distance = 0

Seek Distance = Seek Distance + abs (Current Cylinder Number - New Cylinder Number)
Seek Distance = 0 + abs(53-37)
Seek Distance = 16

SCAN Disk Scheduling

- Request Queue => 0 .. 199
- 98, 183, 37, 122, 14, 124, 65, 67
- Head pointer initially @ 53



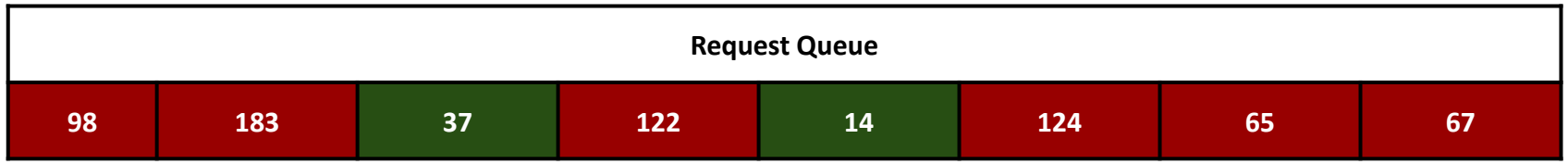


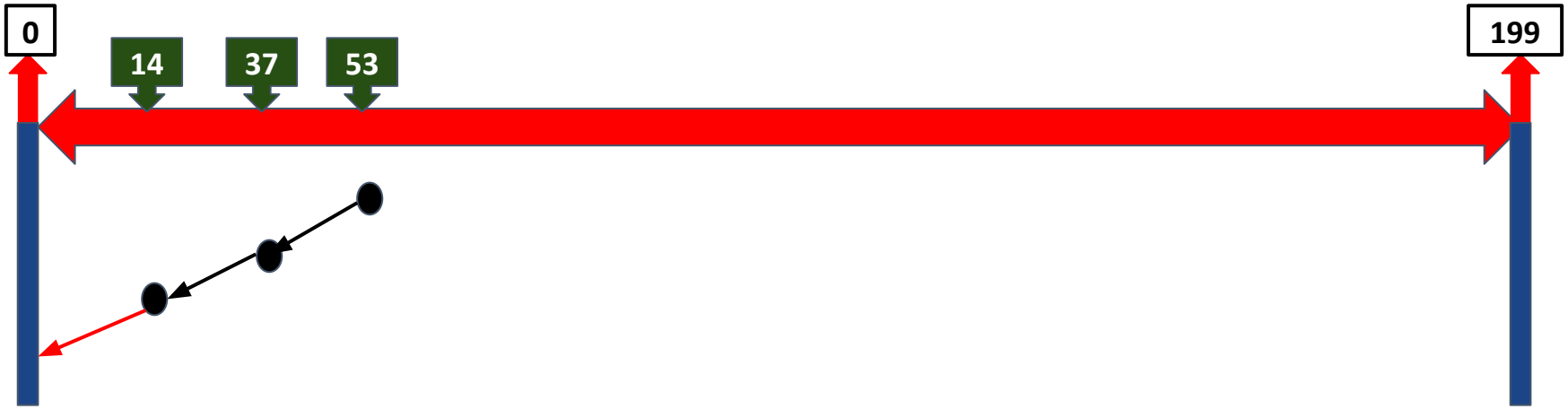
Seek Distance = 16

Seek Distance = Seek Distance + abs (Current Cylinder Number - New Cylinder Number)
Seek Distance = 16 + abs(37-14)
Seek Distance = 39

SCAN Disk Scheduling

- Request Queue => 0 .. 199
- 98, 183, 37, 122, 14, 124, 65, 67
- Head pointer initially @ 53



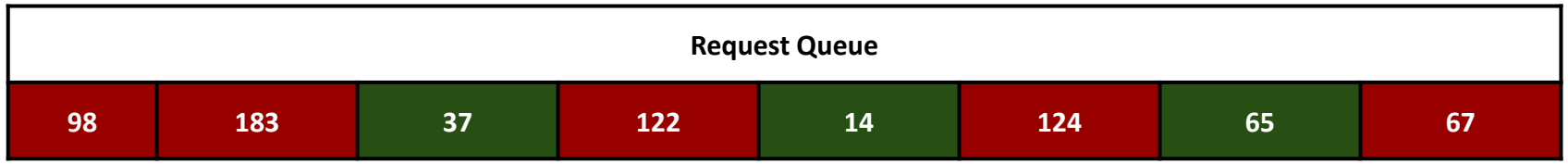


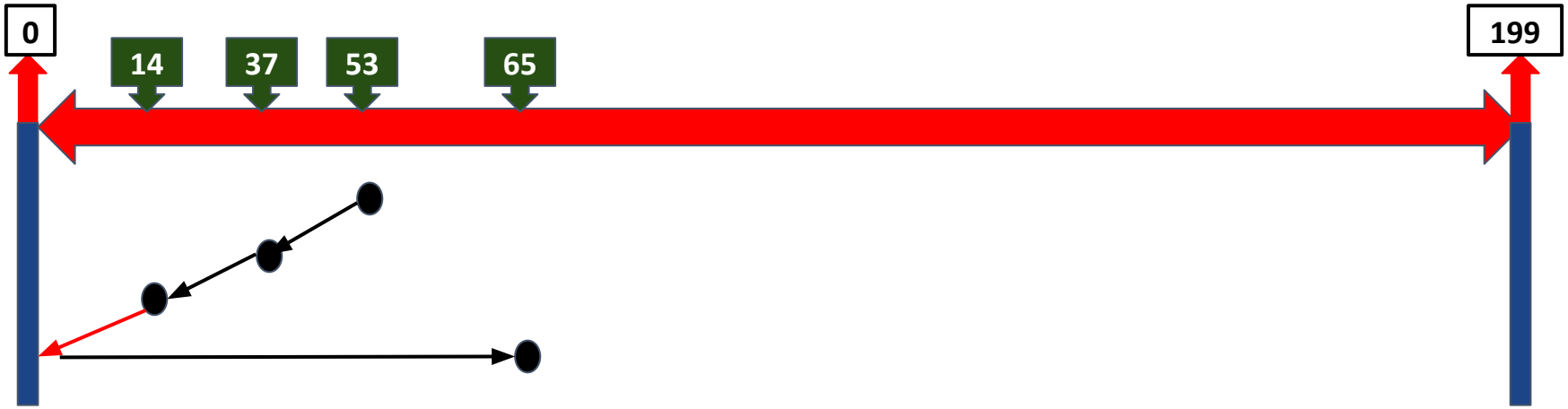
Seek Distance = 39

Seek Distance = Seek Distance + abs (Current Cylinder Number - New Cylinder Number)
Seek Distance = 39 + abs(14-0)
Seek Distance = 53

SCAN Disk Scheduling

- Request Queue => 0 .. 199
- 98, 183, 37, 122, 14, 124, 65, 67
- Head pointer initially @ 53



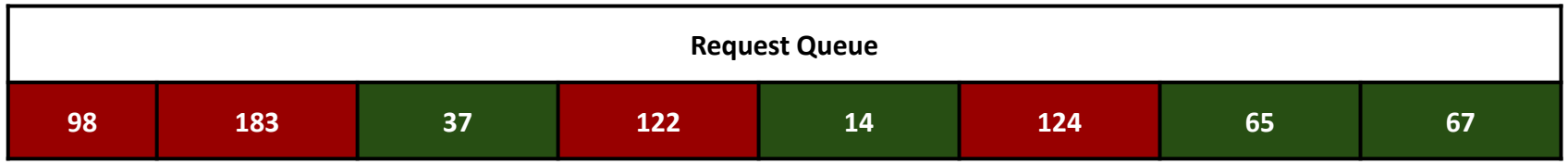


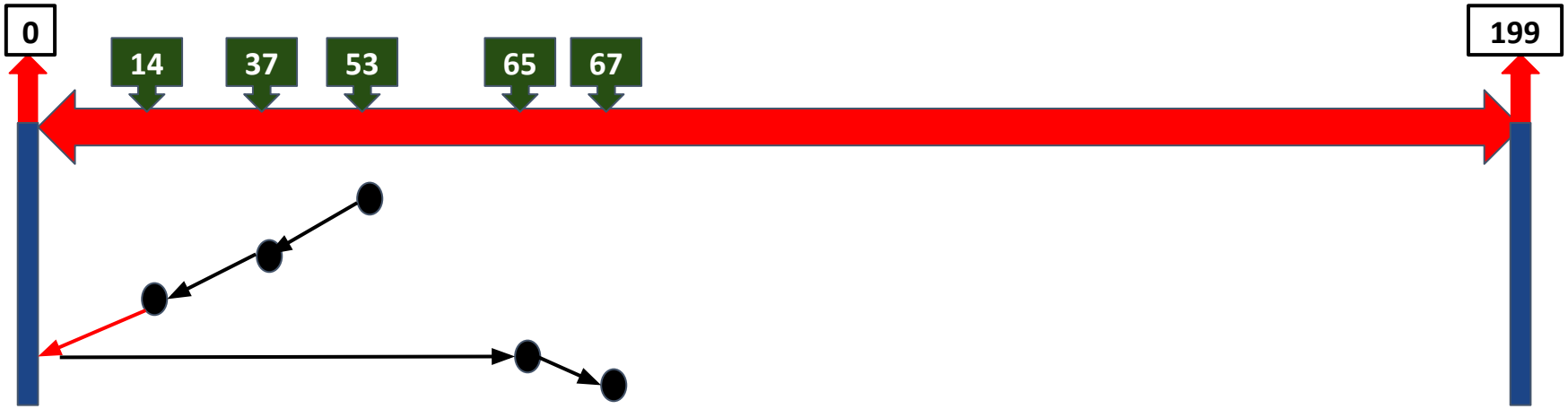
Seek Distance = 53

Seek Distance = Seek Distance + abs (Current Cylinder Number - New Cylinder Number)
Seek Distance = 53 + abs(0-65)
Seek Distance = 118

SCAN Disk Scheduling

- Request Queue => 0 .. 199
- 98, 183, 37, 122, 14, 124, 65, 67
- Head pointer initially @ 53



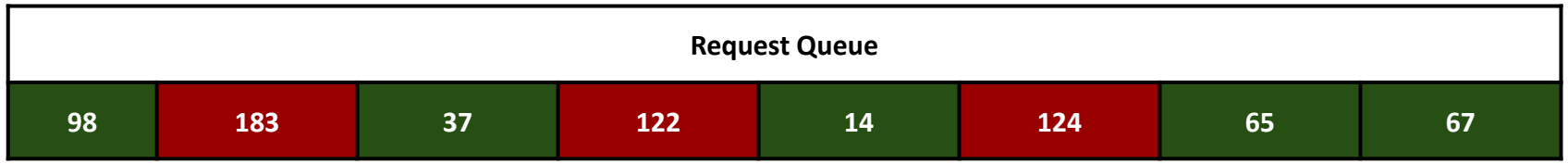


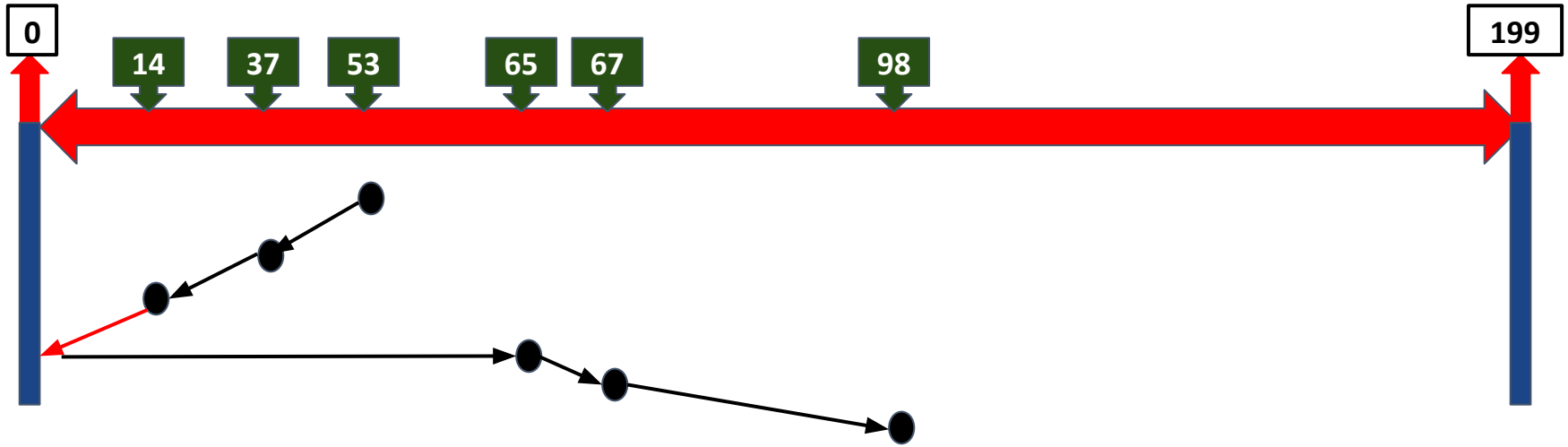
Seek Distance = 118

Seek Distance = Seek Distance + abs (Current Cylinder Number - New Cylinder Number)
Seek Distance = 118 + abs(65-67)
Seek Distance = 120

SCAN Disk Scheduling

- Request Queue => 0 .. 199
- 98, 183, 37, 122, 14, 124, 65, 67
- Head pointer initially @ 53



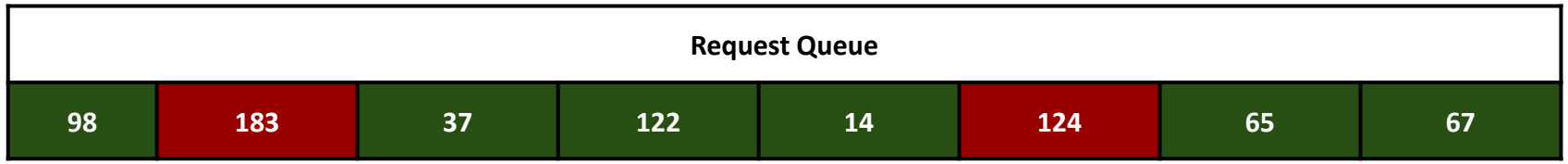


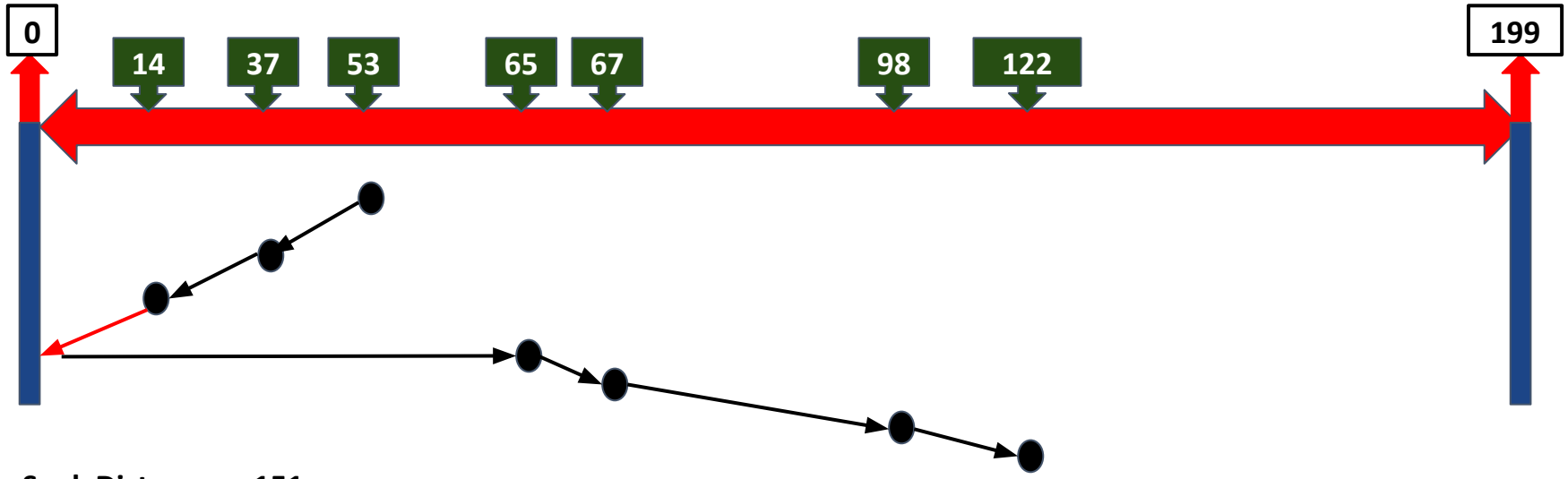
Seek Distance = 120

Seek Distance = Seek Distance + abs (Current Cylinder Number - New Cylinder Number)
Seek Distance = 120 + abs(67-98)
Seek Distance = 151

SCAN Disk Scheduling

- Request Queue => 0 .. 199
- 98, 183, 37, 122, 14, 124, 65, 67
- Head pointer initially @ 53



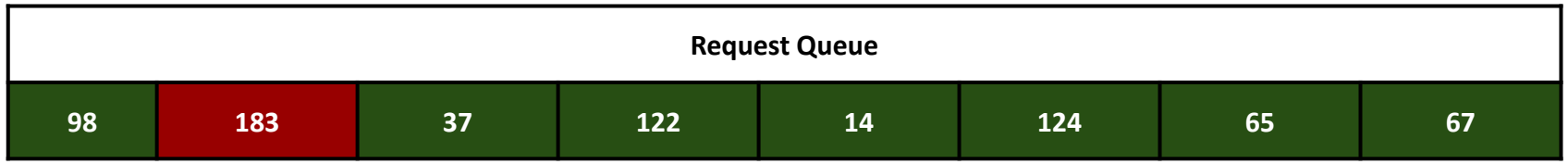


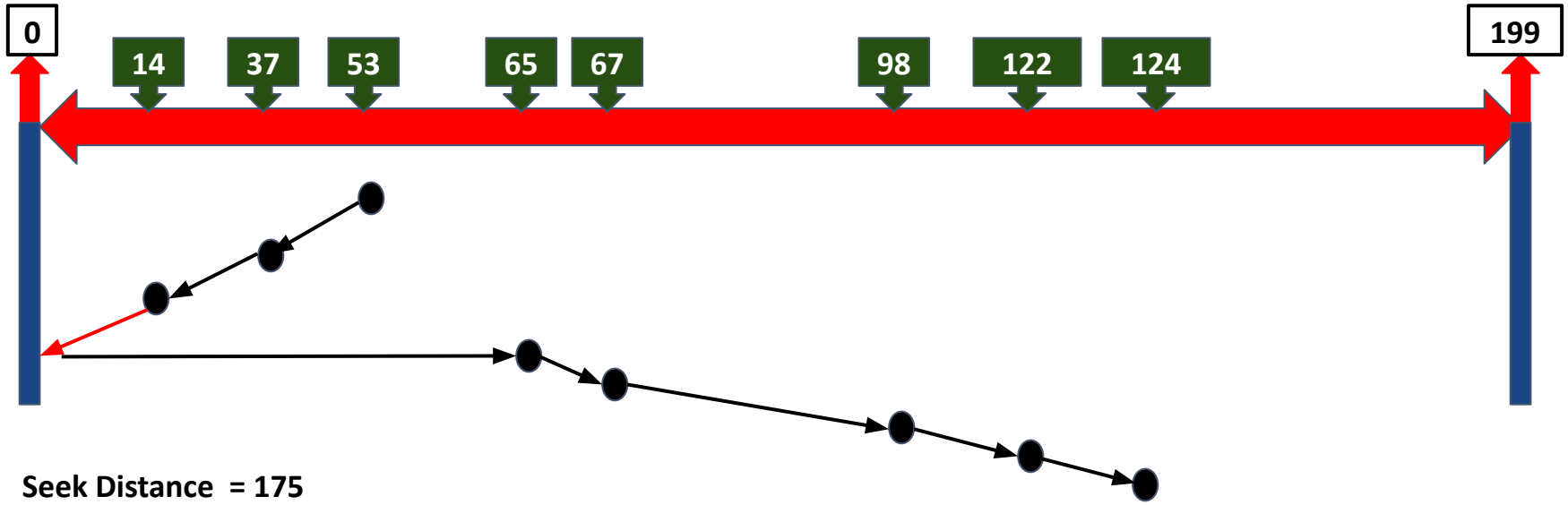
Seek Distance = 151

Seek Distance = Seek Distance + abs (Current Cylinder Number - New Cylinder Number)
Seek Distance = 151 + abs(98-122)
Seek Distance = 175

SCAN Disk Scheduling

- Request Queue => 0 .. 199
- 98, 183, 37, 122, 14, 124, 65, 67
- Head pointer initially @ 53





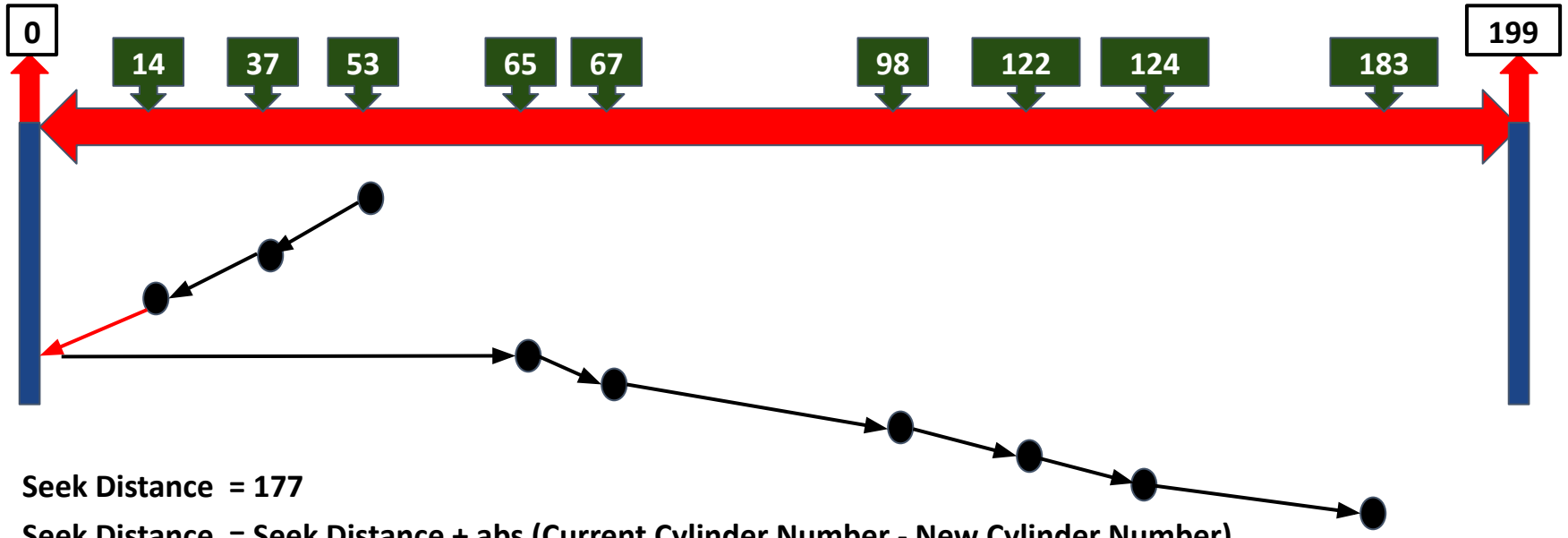
Seek Distance = 175

Seek Distance = Seek Distance + abs (Current Cylinder Number - New Cylinder Number)
Seek Distance = 175 + abs(122-124)
Seek Distance = 177

SCAN Disk Scheduling

- Request Queue => 0 .. 199
- 98, 183, 37, 122, 14, 124, 65, 67
- Head pointer initially @ 53

Request Queue							
98	183	37	122	14	124	65	67



Final Seek Count in terms of Cylinders

236

Seek Distance = 177

Seek Distance = Seek Distance + abs (Current Cylinder Number - New Cylinder Number)

Seek Distance = 177 + abs(124-183)

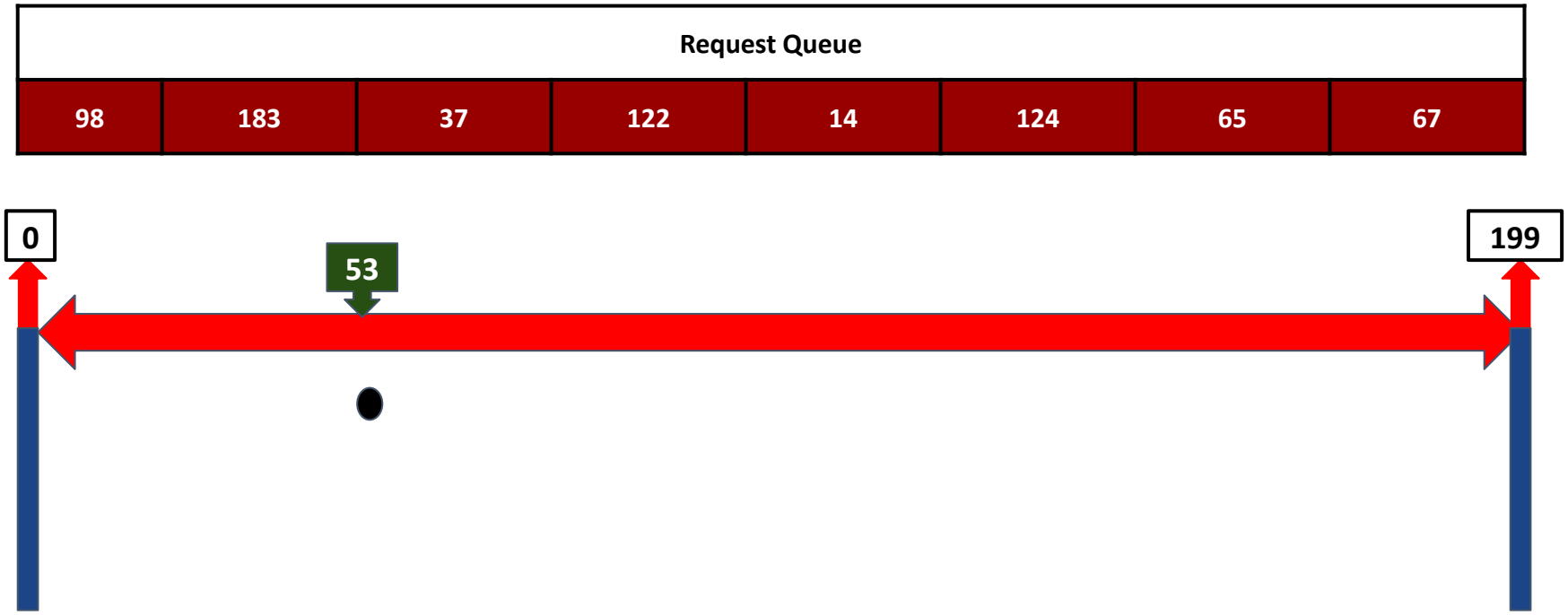
Seek Distance = 236

C - SCAN Disk Scheduling

- Provides a more uniform wait time than 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
- Treats the cylinders as a circular list that wraps around from the last cylinder to the first one
- Total number of cylinders ?

C - SCAN Disk Scheduling

- Request Queue => 0 .. 199
- 98, 183, 37, 122, 14, 124, 65, 67
- Head pointer initially @ 53

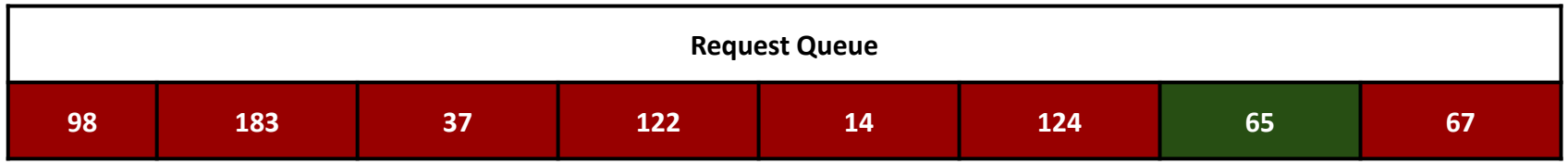


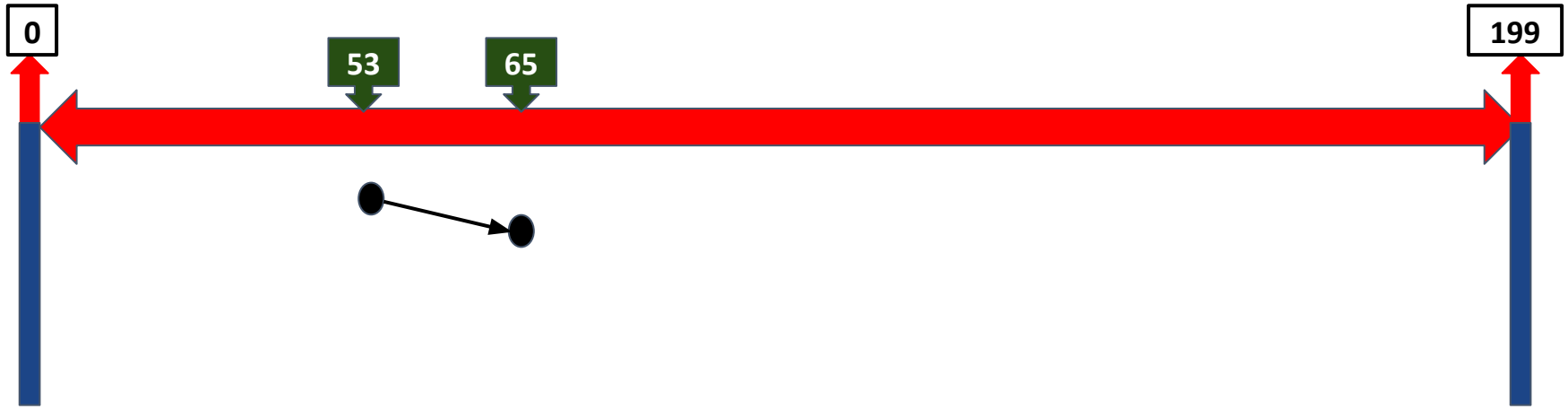
Seek Distance = 0

Seek Distance = Seek Distance + abs (Current Cylinder Number - New Cylinder Number)
Seek Distance = 0 + abs(0)
Seek Distance = 0

C - SCAN Disk Scheduling

- Request Queue => 0 .. 199
- 98, 183, 37, 122, 14, 124, 65, 67
- Head pointer initially @ 53



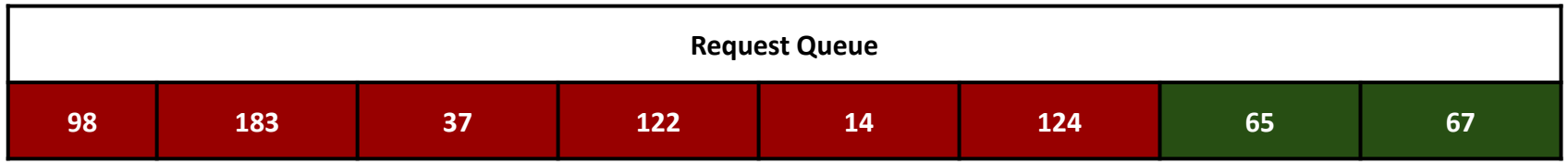


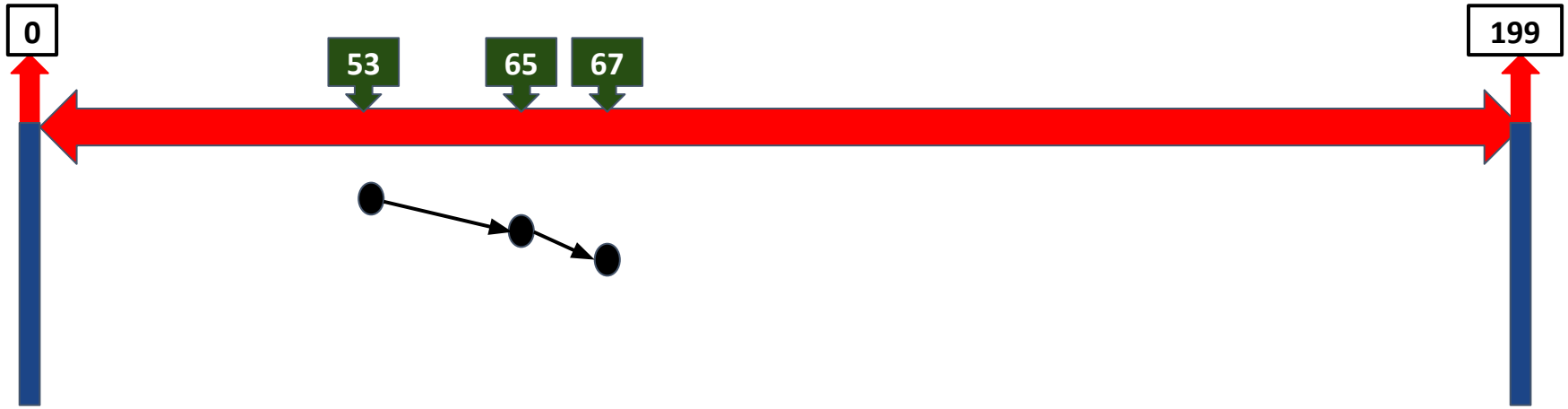
Seek Distance = 0

Seek Distance = Seek Distance + abs (Current Cylinder Number - New Cylinder Number)
Seek Distance = 0 + abs(53-65)
Seek Distance = 12

C - SCAN Disk Scheduling

- Request Queue => 0 .. 199
- 98, 183, 37, 122, 14, 124, 65, 67
- Head pointer initially @ 53



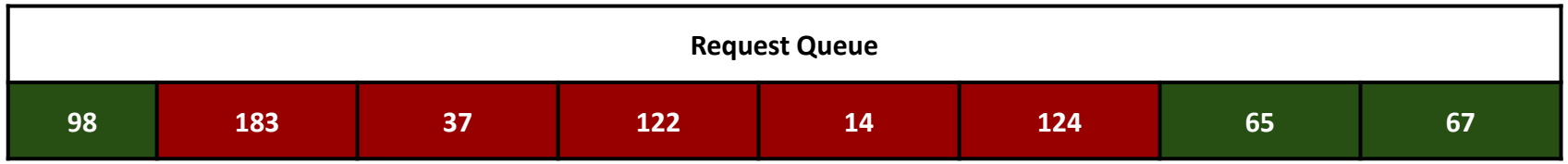


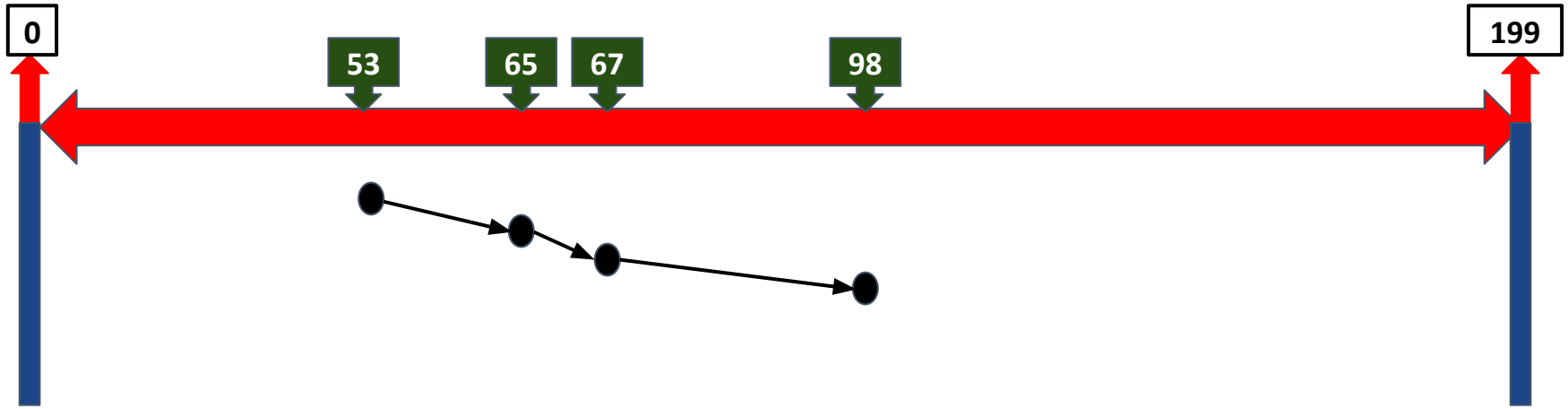
Seek Distance = 12

Seek Distance = Seek Distance + abs (Current Cylinder Number - New Cylinder Number)
Seek Distance = 12 + abs(65-67)
Seek Distance = 14

C - SCAN Disk Scheduling

- Request Queue => 0 .. 199
- 98, 183, 37, 122, 14, 124, 65, 67
- Head pointer initially @ 53



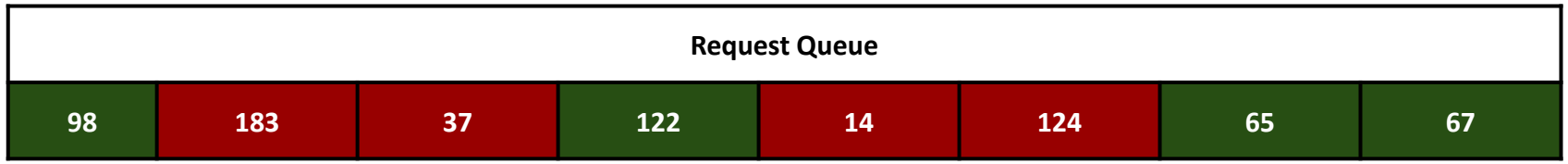


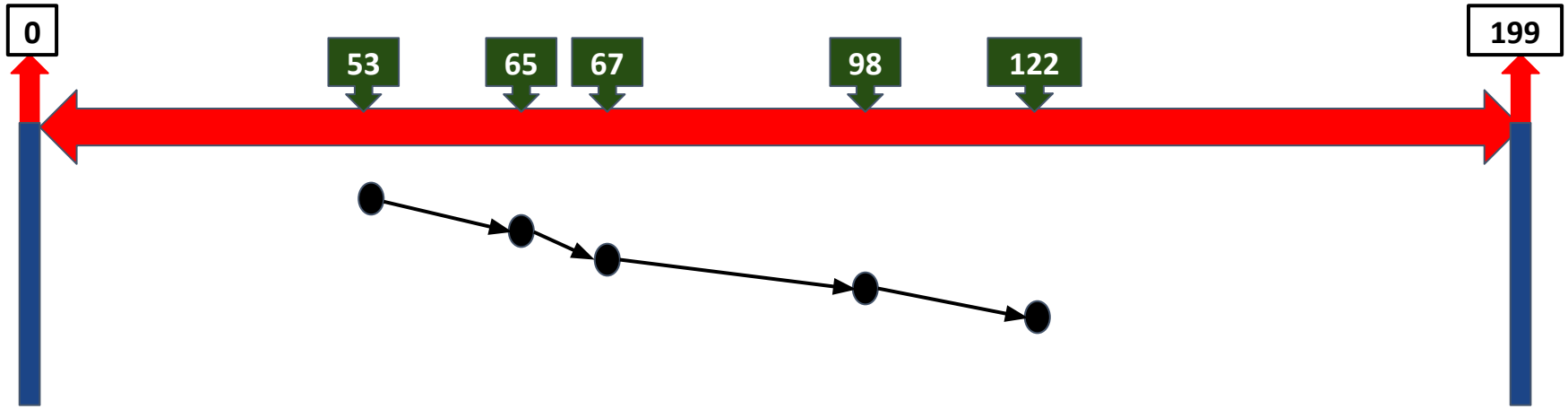
Seek Distance = 14

Seek Distance = Seek Distance + abs (Current Cylinder Number - New Cylinder Number)
Seek Distance = 14 + abs(67-98)
Seek Distance = 45

C - SCAN Disk Scheduling

- Request Queue => 0 .. 199
- 98, 183, 37, 122, 14, 124, 65, 67
- Head pointer initially @ 53



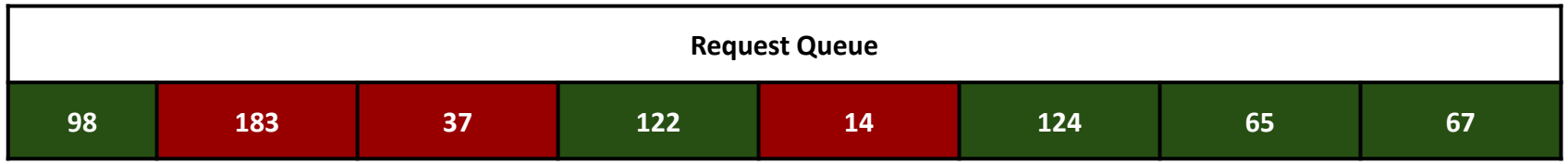


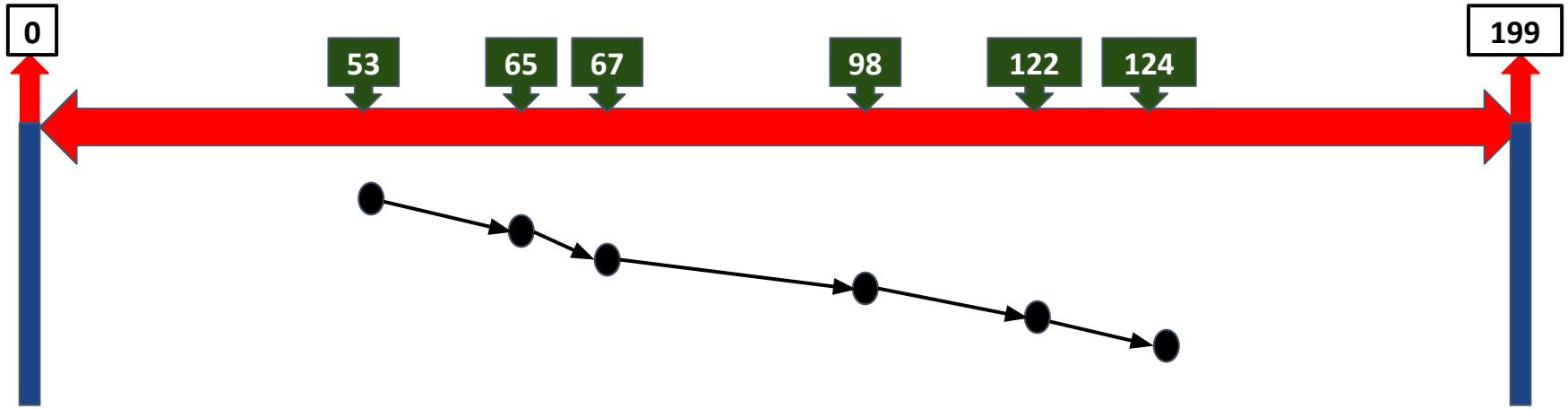
Seek Distance = 45

Seek Distance = Seek Distance + abs (Current Cylinder Number - New Cylinder Number)
Seek Distance = 45 + abs(98-122)
Seek Distance = 69

C - SCAN Disk Scheduling

- Request Queue => 0 .. 199
- 98, 183, 37, 122, 14, 124, 65, 67
- Head pointer initially @ 53



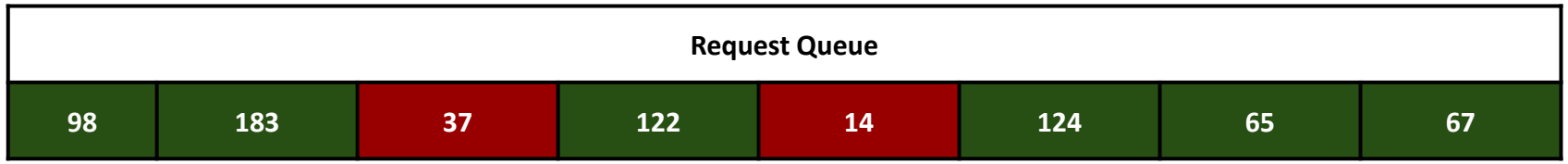


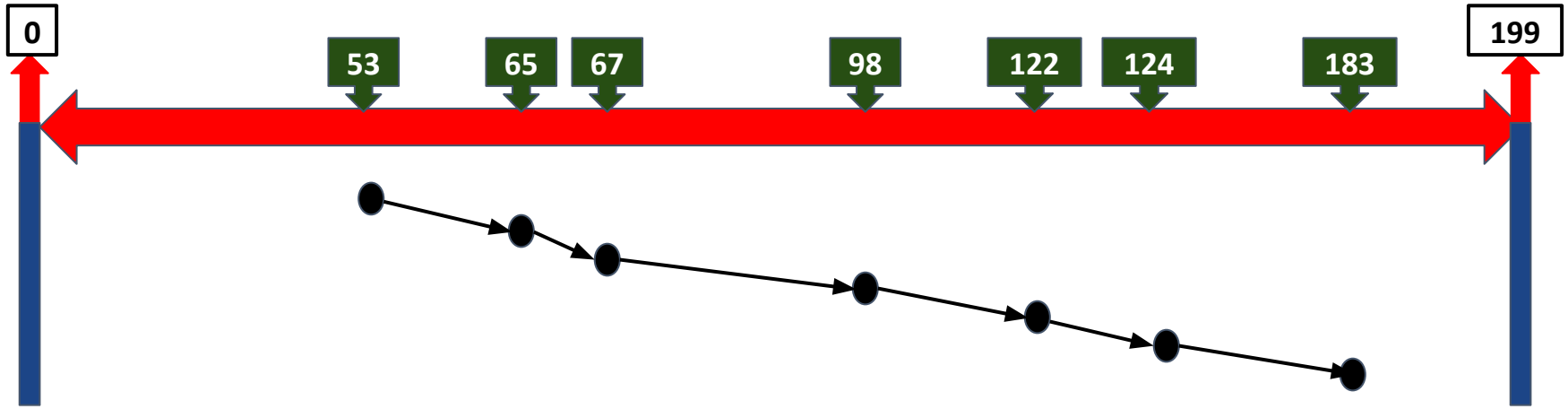
Seek Distance = 69

Seek Distance = Seek Distance + abs (Current Cylinder Number - New Cylinder Number)
Seek Distance = 69 + abs(122-124)
Seek Distance = 71

C - SCAN Disk Scheduling

- Request Queue => 0 .. 199
- 98, 183, 37, 122, 14, 124, 65, 67
- Head pointer initially @ 53



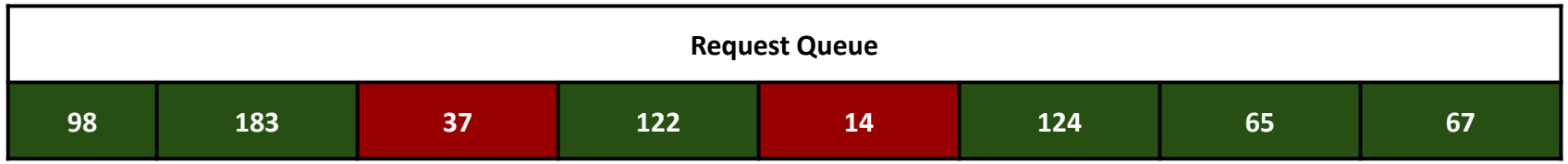


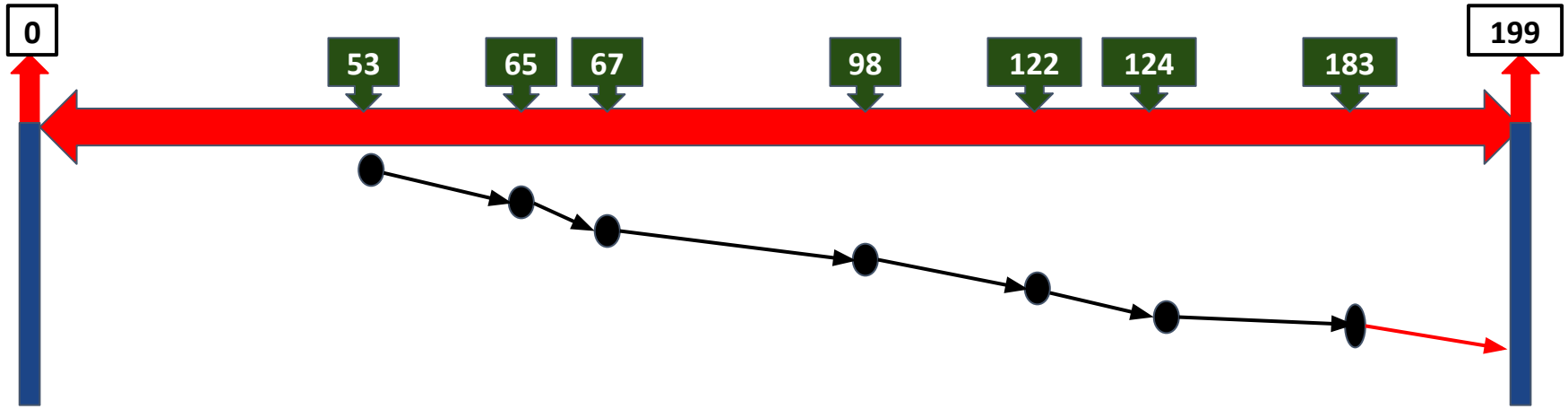
Seek Distance = 71

Seek Distance = Seek Distance + abs (Current Cylinder Number - New Cylinder Number)
Seek Distance = 69 + abs(124-183)
Seek Distance = 130

C - SCAN Disk Scheduling

- Request Queue => 0 .. 199
- 98, 183, 37, 122, 14, 124, 65, 67
- Head pointer initially @ 53



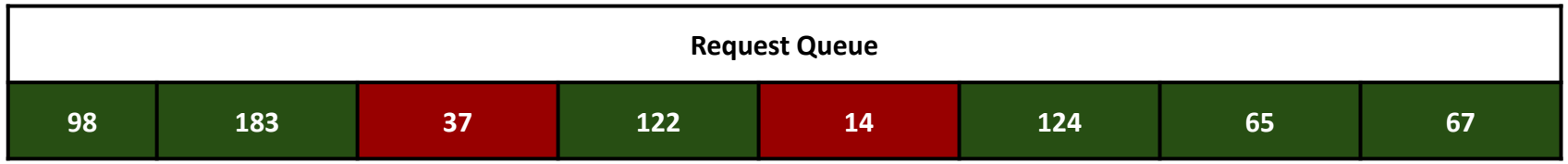


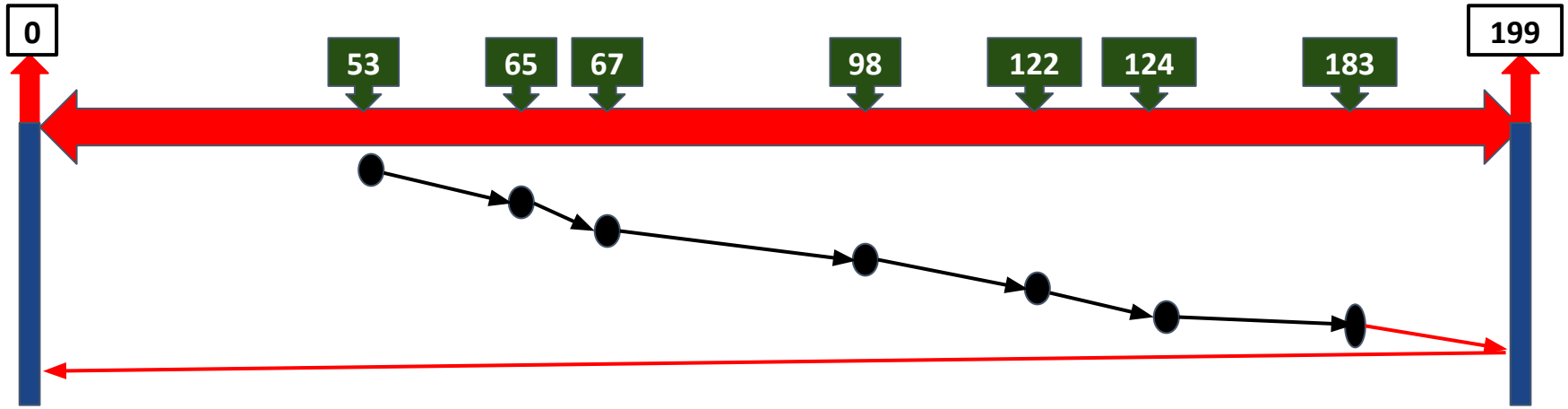
Seek Distance = 130

Seek Distance = Seek Distance + abs (Current Cylinder Number - New Cylinder Number)
Seek Distance = 130 + abs(183-199)
Seek Distance = 146

C - SCAN Disk Scheduling

- Request Queue => 0 .. 199
- 98, 183, 37, 122, 14, 124, 65, 67
- Head pointer initially @ 53



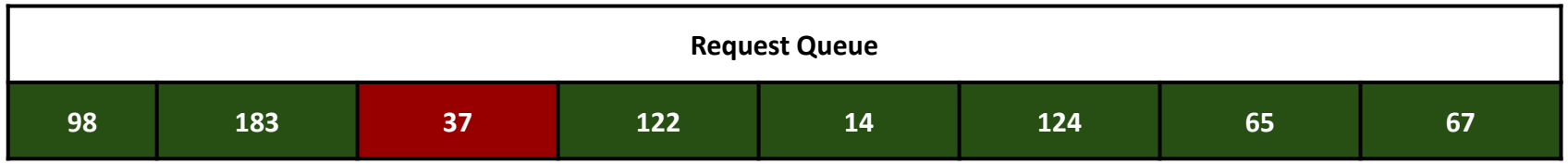


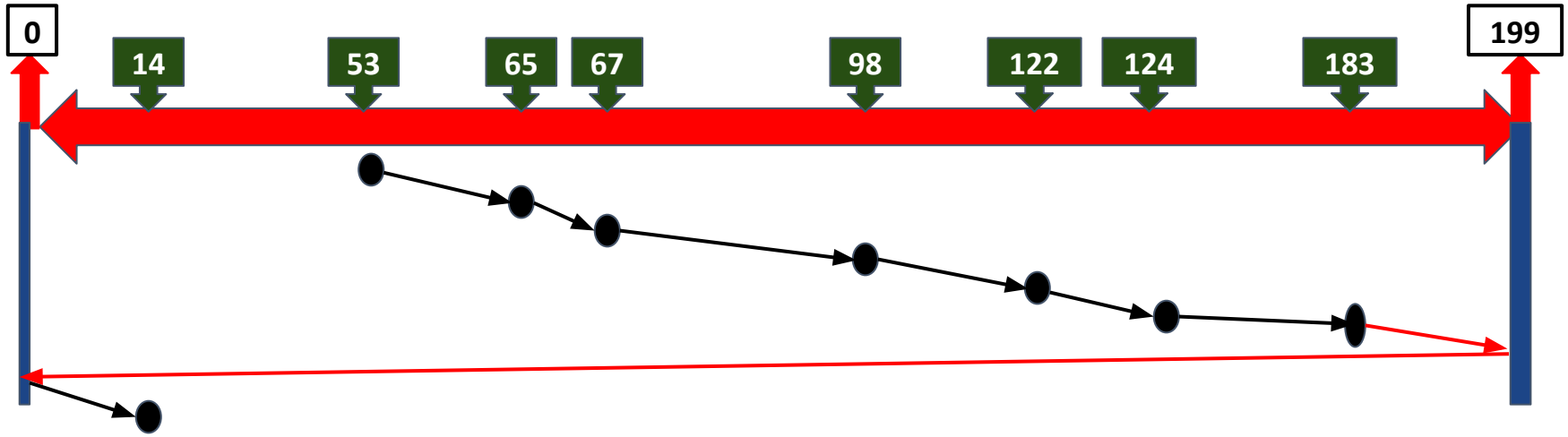
Seek Distance = 146

Seek Distance = Seek Distance + abs (Current Cylinder Number - New Cylinder Number)
Seek Distance = 146 + abs(199-0)
Seek Distance = 345

C - SCAN Disk Scheduling

- Request Queue => 0 .. 199
- 98, 183, 37, 122, 14, 124, 65, 67
- Head pointer initially @ 53





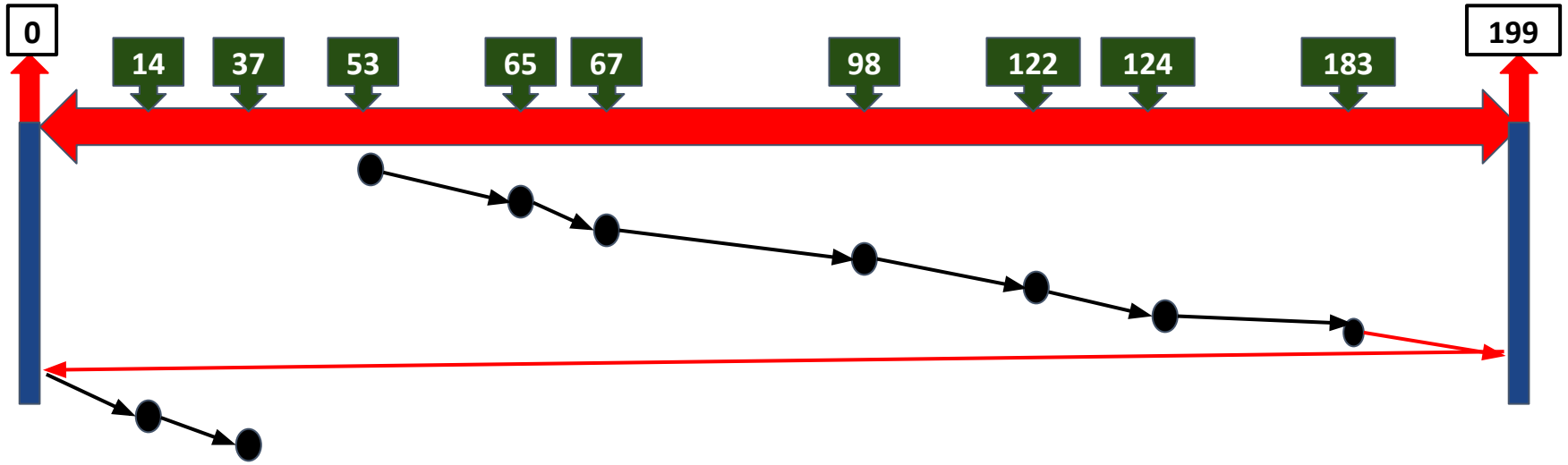
Seek Distance = 345

Seek Distance = Seek Distance + abs (Current Cylinder Number - New Cylinder Number)
Seek Distance = 345 + abs(0-14)
Seek Distance = 359

C - SCAN Disk Scheduling

- Request Queue => 0 .. 199
- 98, 183, 37, 122, 14, 124, 65, 67
- Head pointer initially @ 53

Request Queue							
98	183	37	122	14	124	65	67



Final Seek Count in terms of Cylinders

382

Seek Distance = 359

Seek Distance = Seek Distance + abs (Current Cylinder Number - New Cylinder Number)

Seek Distance = 359 + abs(14-37)

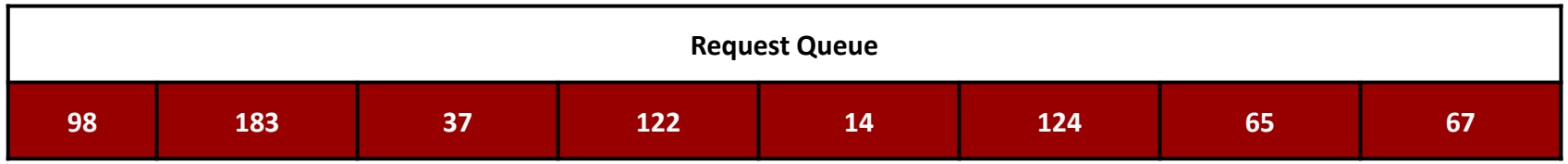
Seek Distance = 382

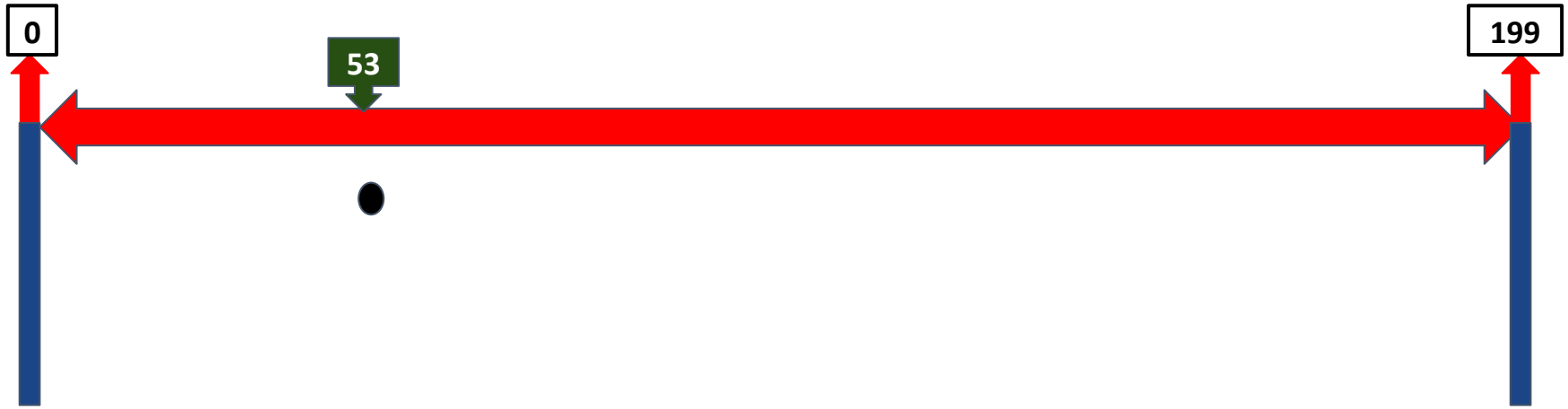
LOOK Disk Scheduling

- LOOK is the advanced version of SCAN (elevator) disk scheduling algorithm which gives slightly better seek time
- The LOOK algorithm services request similarly as SCAN algorithm meanwhile it also “looks” ahead as if there are more tracks that are needed to be serviced in the same direction.
- If there are no pending requests in the moving direction the head reverses the direction and start servicing requests in the opposite direction.
- The main reason behind the better performance of LOOK algorithm in comparison to SCAN is because in this algorithm the head is not allowed to move till the end of the disk.

LOOK Disk Scheduling

- Request Queue => 0 .. 199
- 98, 183, 37, 122, 14, 124, 65, 67
- Head pointer initially @ 53



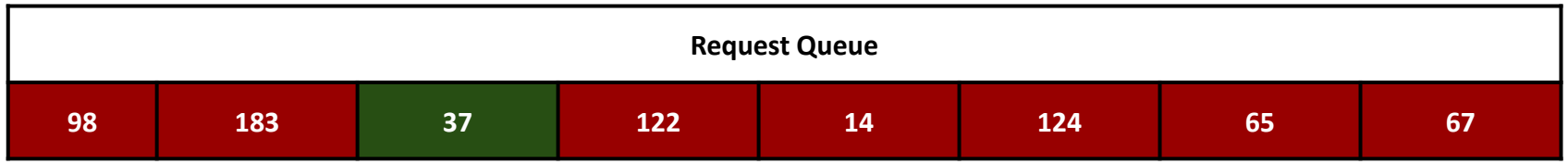


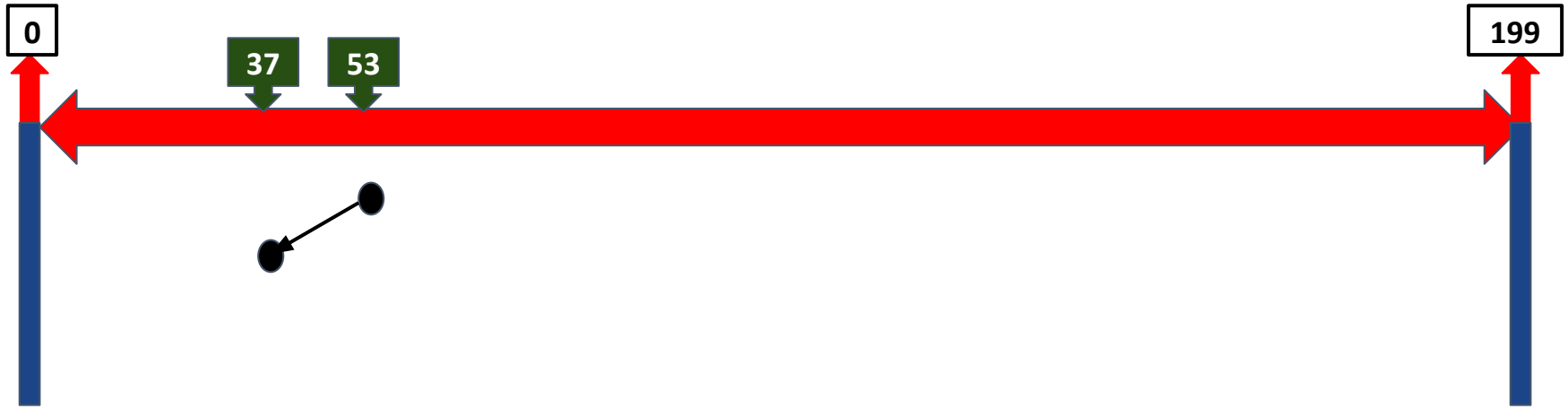
Seek Distance = 0

Seek Distance = Seek Distance + abs (Current Cylinder Number - New Cylinder Number)
Seek Distance = 0 + abs()
Seek Distance = 0

LOOK Disk Scheduling

- Request Queue => 0 .. 199
- 98, 183, 37, 122, 14, 124, 65, 67
- Head pointer initially @ 53



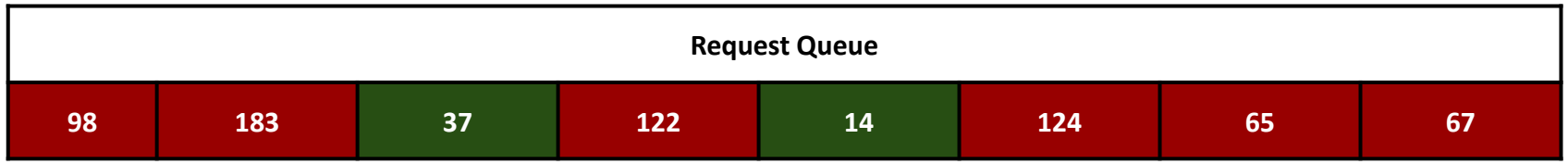


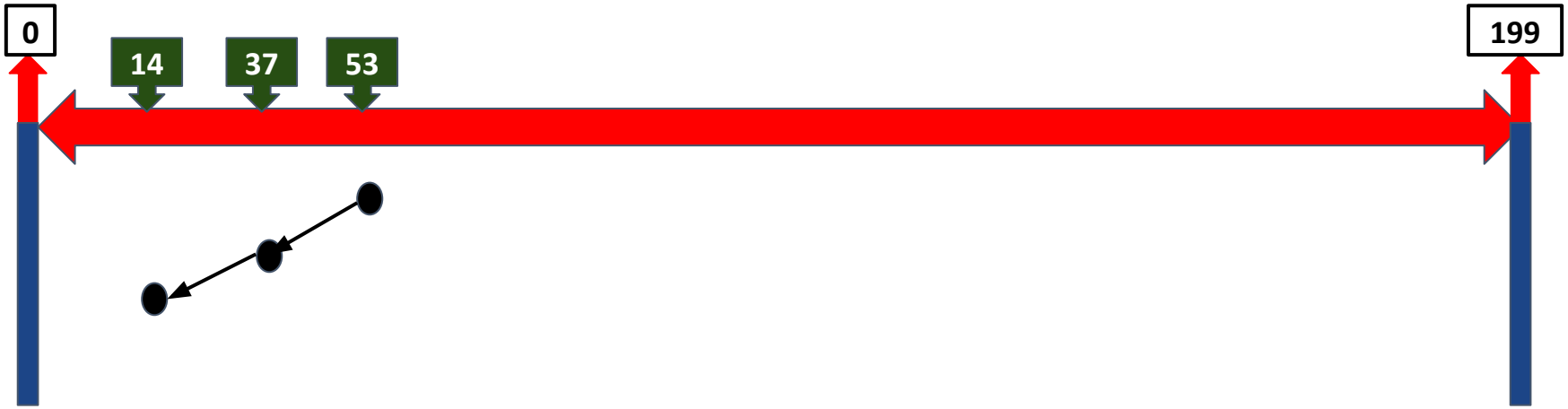
Seek Distance = 0

Seek Distance = Seek Distance + abs (Current Cylinder Number - New Cylinder Number)
Seek Distance = 0 + abs(53-37)
Seek Distance = 16

LOOK Disk Scheduling

- Request Queue => 0 .. 199
- 98, 183, 37, 122, 14, 124, 65, 67
- Head pointer initially @ 53



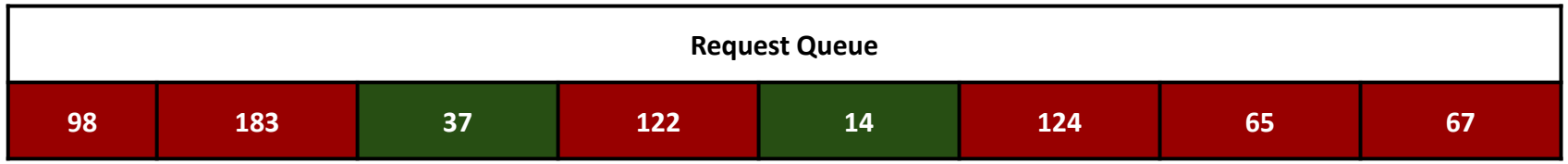


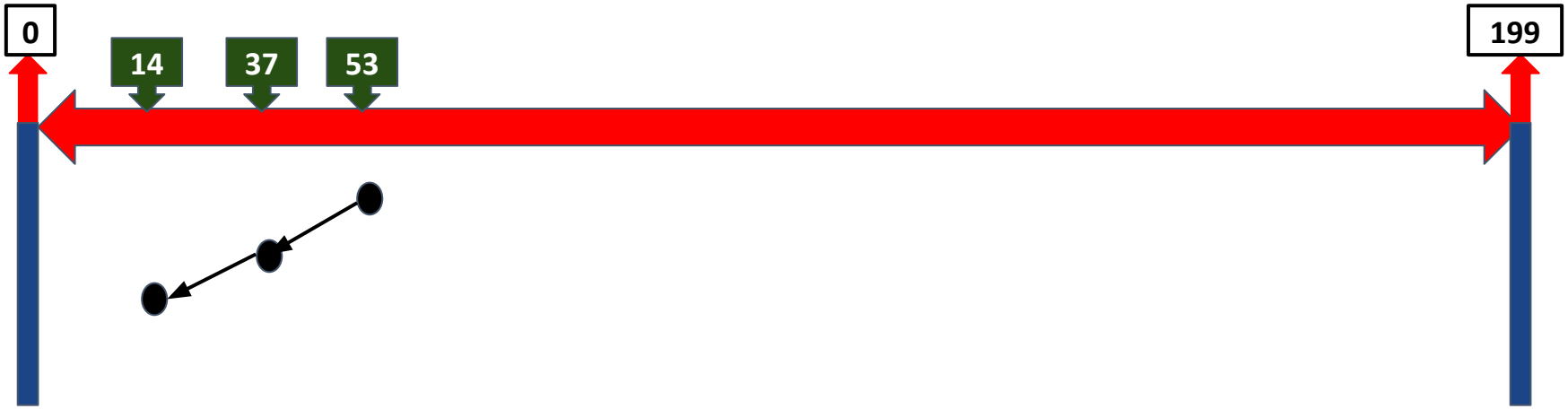
Seek Distance = 16

Seek Distance = Seek Distance + abs (Current Cylinder Number - New Cylinder Number)
Seek Distance = 16 + abs(37-14)
Seek Distance = 39

LOOK Disk Scheduling

- Request Queue => 0 .. 199
- 98, 183, 37, 122, 14, 124, 65, 67
- Head pointer initially @ 53



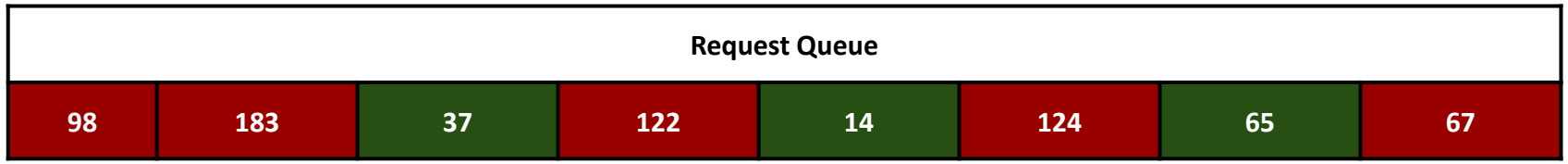


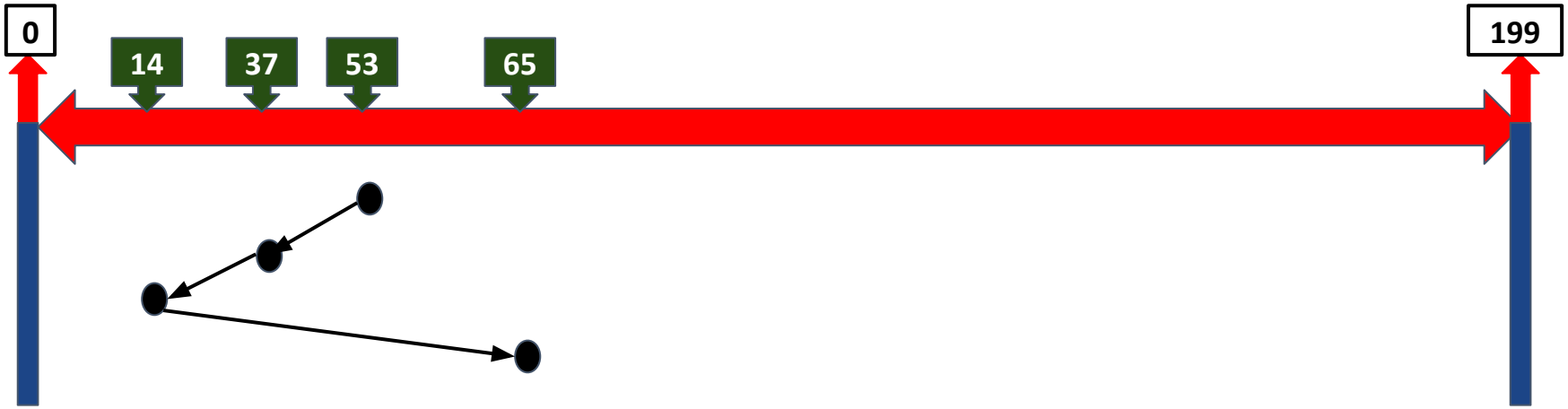
Seek Distance = 39

Seek Distance = Seek Distance + abs (Current Cylinder Number - New Cylinder Number)
Seek Distance = 39 + abs(0-0)
Seek Distance = 39

LOOK Disk Scheduling

- Request Queue => 0 .. 199
- 98, 183, 37, 122, 14, 124, 65, 67
- Head pointer initially @ 53



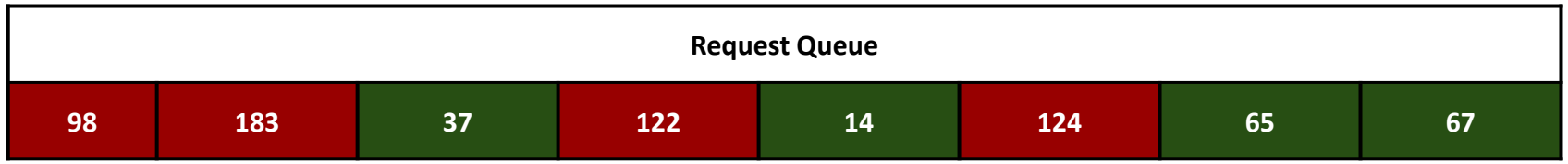


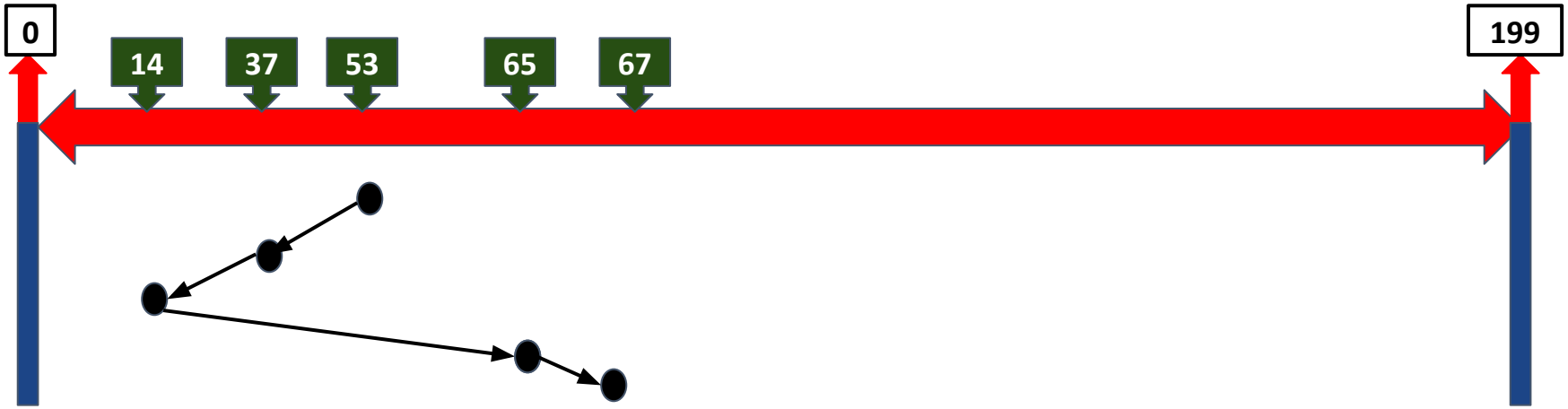
Seek Distance = 39

Seek Distance = Seek Distance + abs (Current Cylinder Number - New Cylinder Number)
Seek Distance = 39 + abs(14-65)
Seek Distance = 90

LOOK Disk Scheduling

- Request Queue => 0 .. 199
- 98, 183, 37, 122, 14, 124, 65, 67
- Head pointer initially @ 53



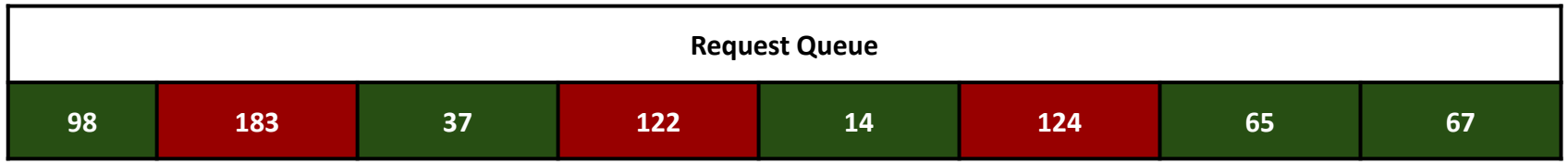


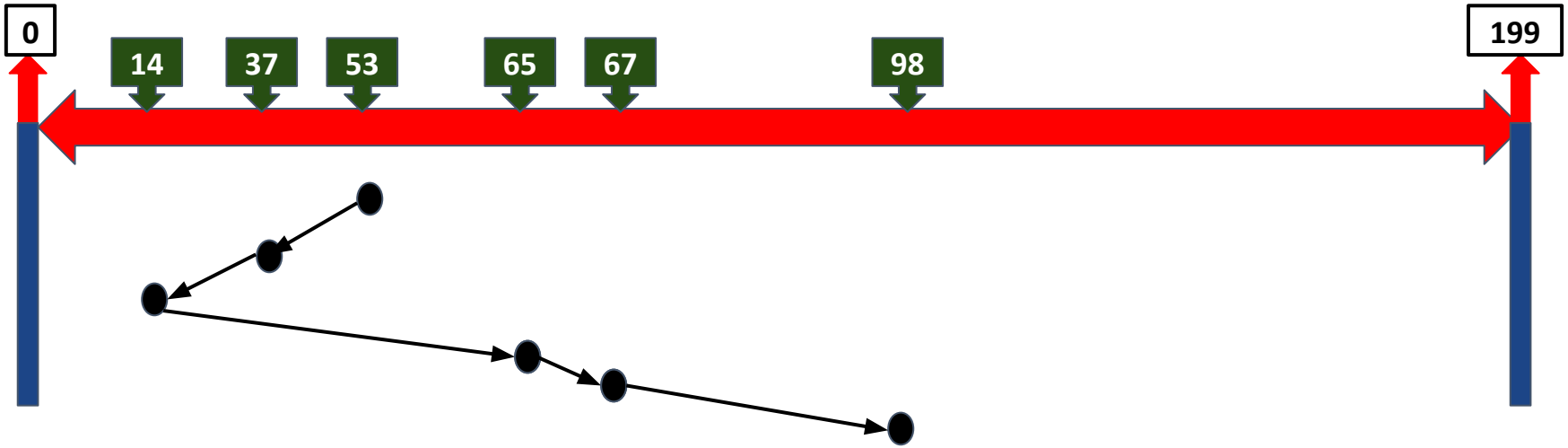
Seek Distance = 90

Seek Distance = Seek Distance + abs (Current Cylinder Number - New Cylinder Number)
Seek Distance = 90 + abs(65-67)
Seek Distance = 92

LOOK Disk Scheduling

- Request Queue => 0 .. 199
- 98, 183, 37, 122, 14, 124, 65, 67
- Head pointer initially @ 53



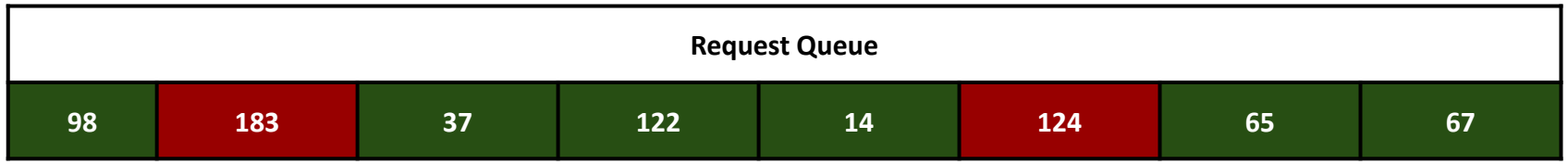


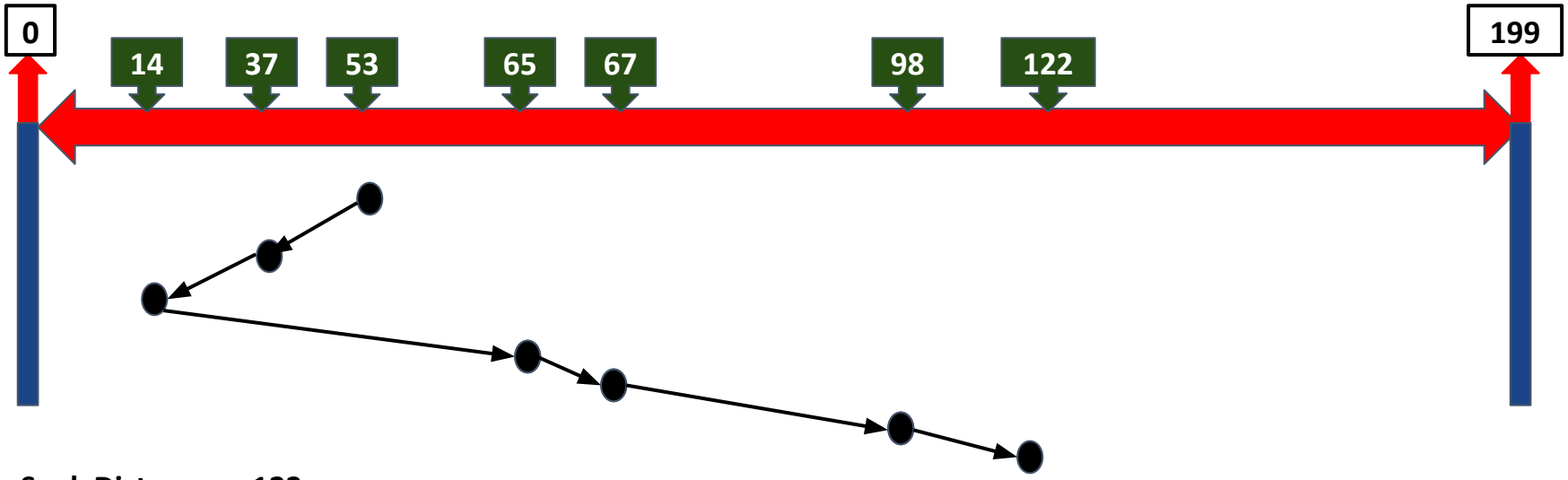
Seek Distance = 92

Seek Distance = Seek Distance + abs (Current Cylinder Number - New Cylinder Number)
Seek Distance = 92 + abs(67-98)
Seek Distance = 123

LOOK Disk Scheduling

- Request Queue => 0 .. 199
- 98, 183, 37, 122, 14, 124, 65, 67
- Head pointer initially @ 53



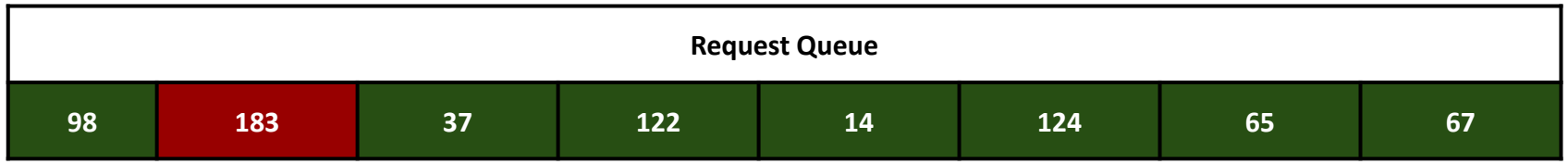


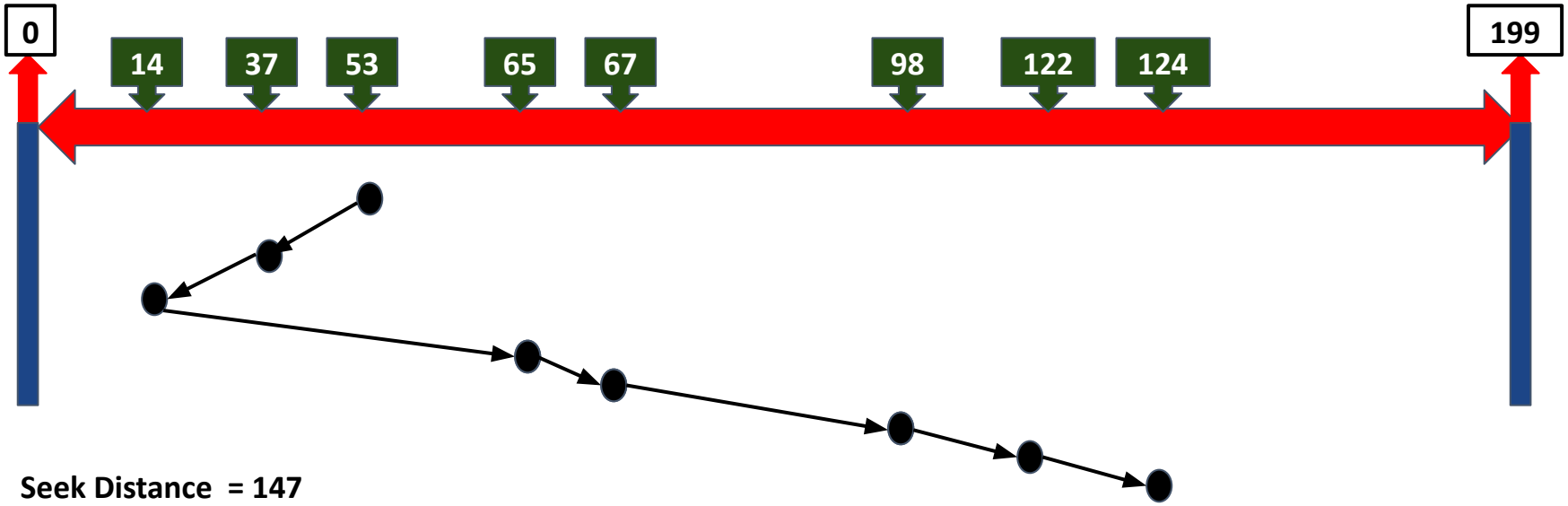
Seek Distance = 123

Seek Distance = Seek Distance + abs (Current Cylinder Number - New Cylinder Number)
Seek Distance = 123 + abs(98-122)
Seek Distance = 147

LOOK Disk Scheduling

- Request Queue => 0 .. 199
- 98, 183, 37, 122, 14, 124, 65, 67
- Head pointer initially @ 53





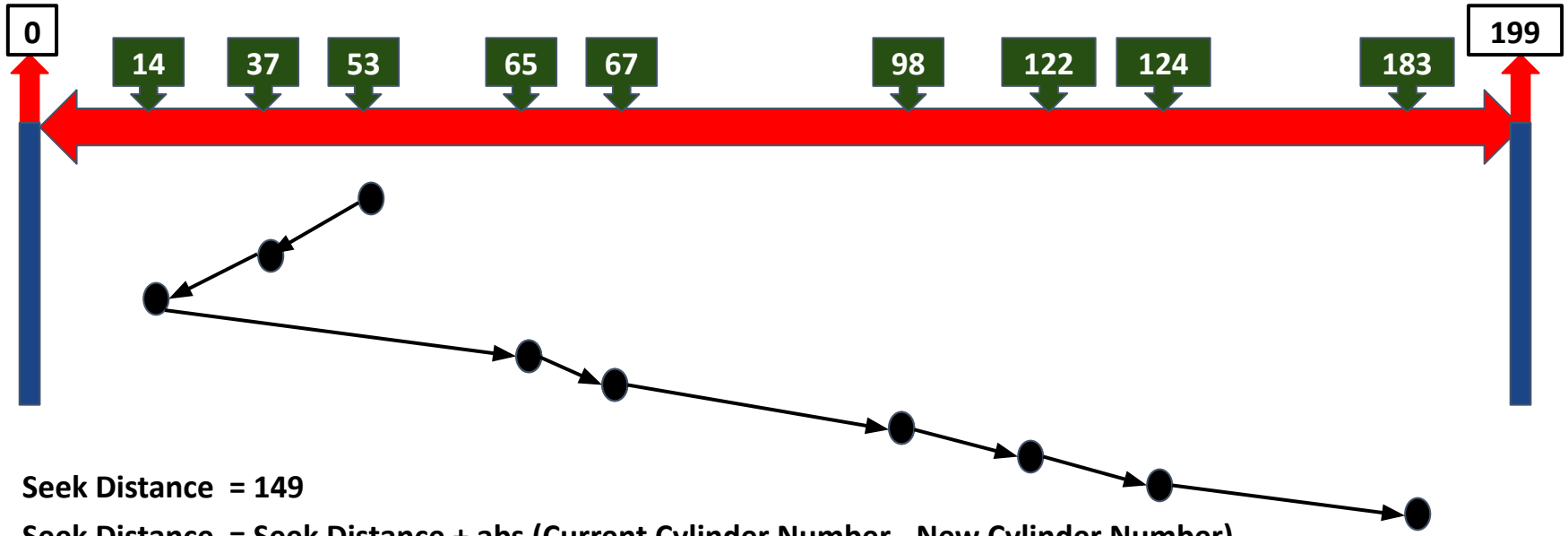
Seek Distance = 147

Seek Distance = Seek Distance + abs (Current Cylinder Number - New Cylinder Number)
Seek Distance = 147 + abs(122-124)
Seek Distance = 149

LOOK Disk Scheduling

- Request Queue => 0 .. 199
- 98, 183, 37, 122, 14, 124, 65, 67
- Head pointer initially @ 53

Request Queue							
98	183	37	122	14	124	65	67



Final Seek Count in terms of Cylinders

208

Seek Distance = 149

Seek Distance = Seek Distance + abs (Current Cylinder Number - New Cylinder Number)

Seek Distance = 149 + abs(124-183)

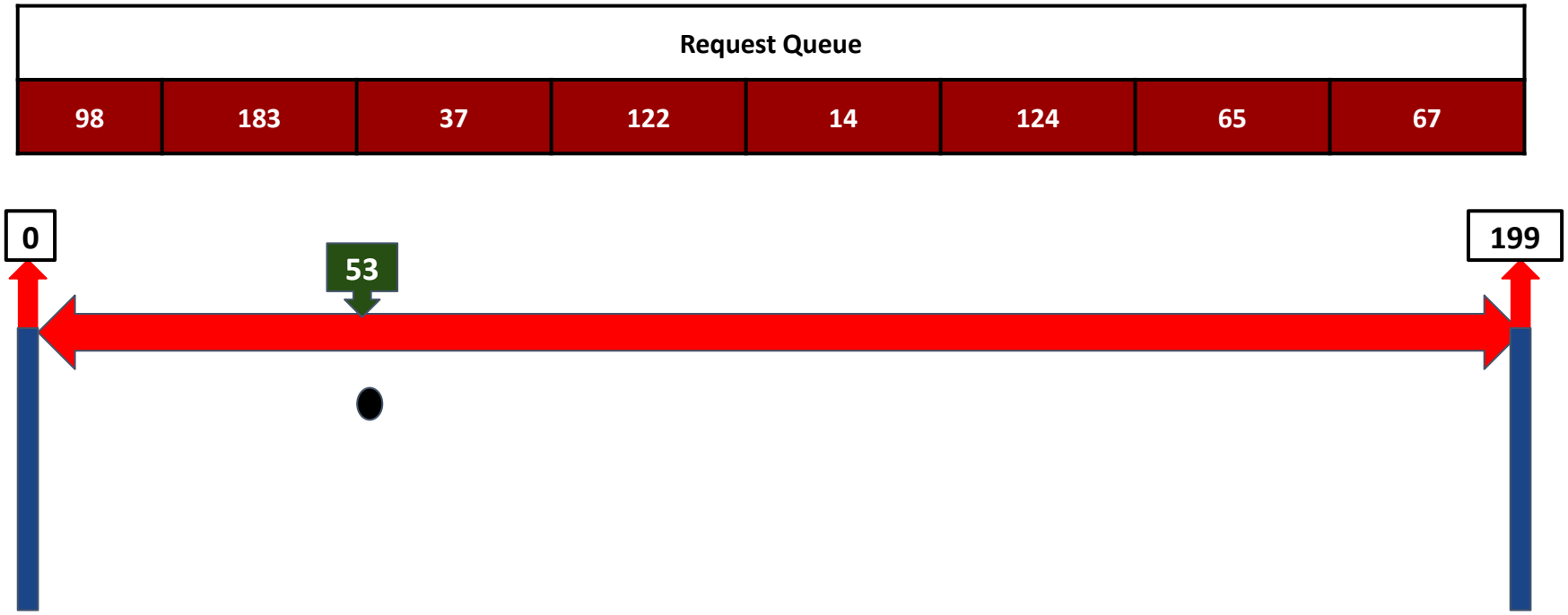
Seek Distance = 208

C - LOOK Disk Scheduling

- LOOK a version of SCAN, C-LOOK a version of C- SCAN
- Arm only goes as far as the last request in each direction, then reverses direction immediately, without first going all the way to the end of the disk
- In this algorithm, the head services requests only in one direction(either left or right) until all the requests in this direction are not serviced and then jumps back to the farthest request on the other direction and service the remaining requests which gives a better uniform servicing as well as avoids wasting seek time for going till the end of the disk.
- Total number of cylinders ?

C - LOOK Disk Scheduling

- Request Queue => 0 .. 199
- 98, 183, 37, 122, 14, 124, 65, 67
- Head pointer initially @ 53

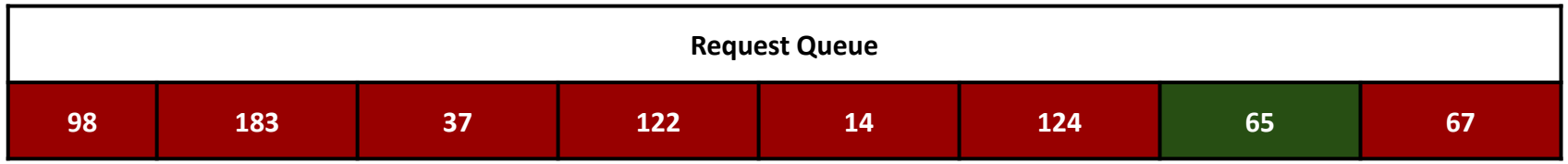


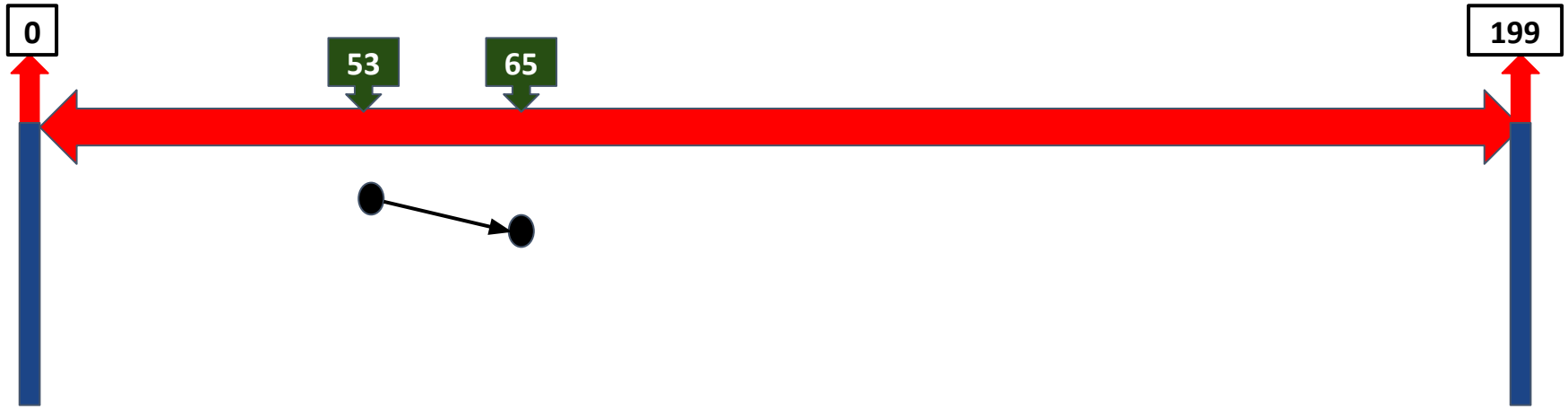
Seek Distance = 0

Seek Distance = Seek Distance + abs (Current Cylinder Number - New Cylinder Number)
Seek Distance = 0 + abs(0)
Seek Distance = 0

C - LOOK Disk Scheduling

- Request Queue => 0 .. 199
- 98, 183, 37, 122, 14, 124, 65, 67
- Head pointer initially @ 53



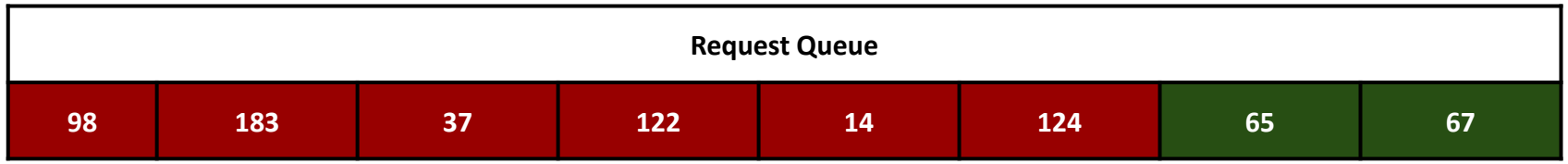


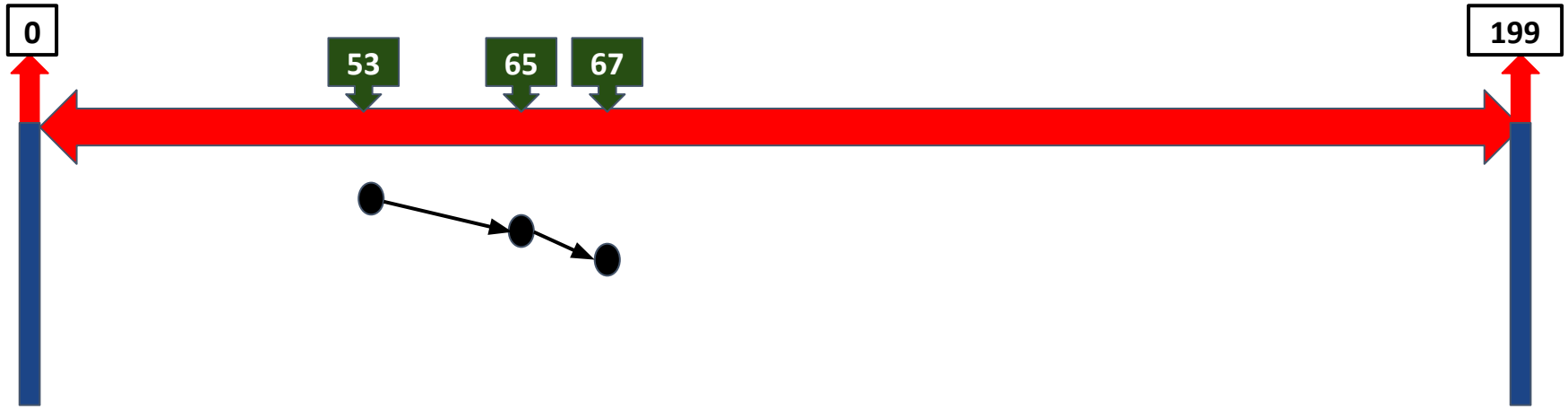
Seek Distance = 0

Seek Distance = Seek Distance + abs (Current Cylinder Number - New Cylinder Number)
Seek Distance = 0 + abs(53-65)
Seek Distance = 12

C - LOOK Disk Scheduling

- Request Queue => 0 .. 199
- 98, 183, 37, 122, 14, 124, 65, 67
- Head pointer initially @ 53



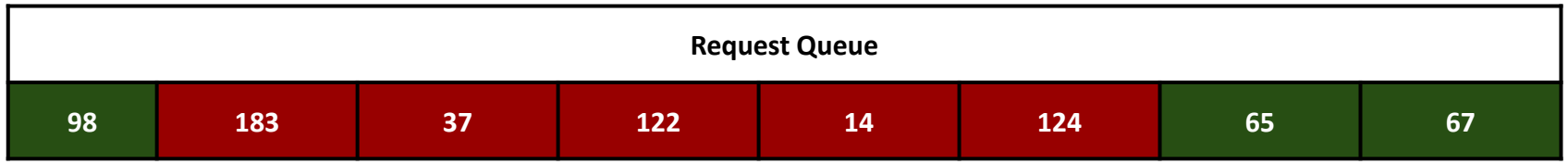


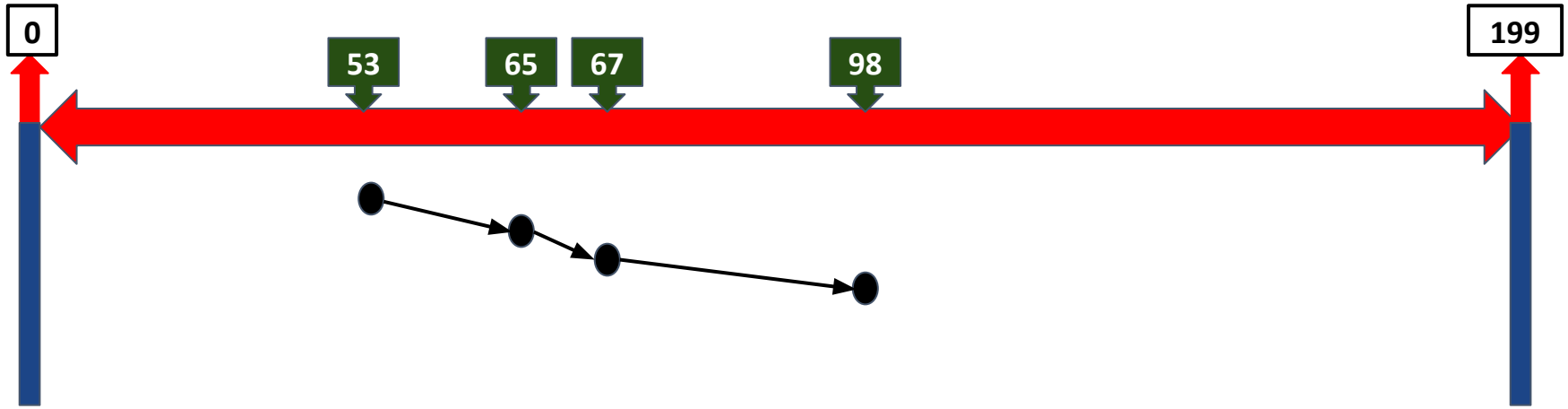
Seek Distance = 12

Seek Distance = Seek Distance + abs (Current Cylinder Number - New Cylinder Number)
Seek Distance = 12 + abs(65-67)
Seek Distance = 14

C - LOOK Disk Scheduling

- Request Queue => 0 .. 199
- 98, 183, 37, 122, 14, 124, 65, 67
- Head pointer initially @ 53



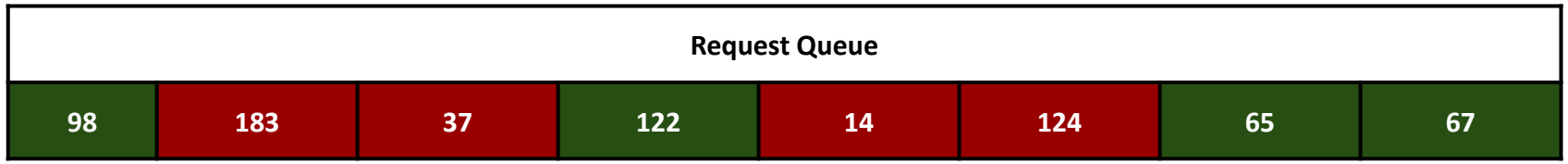


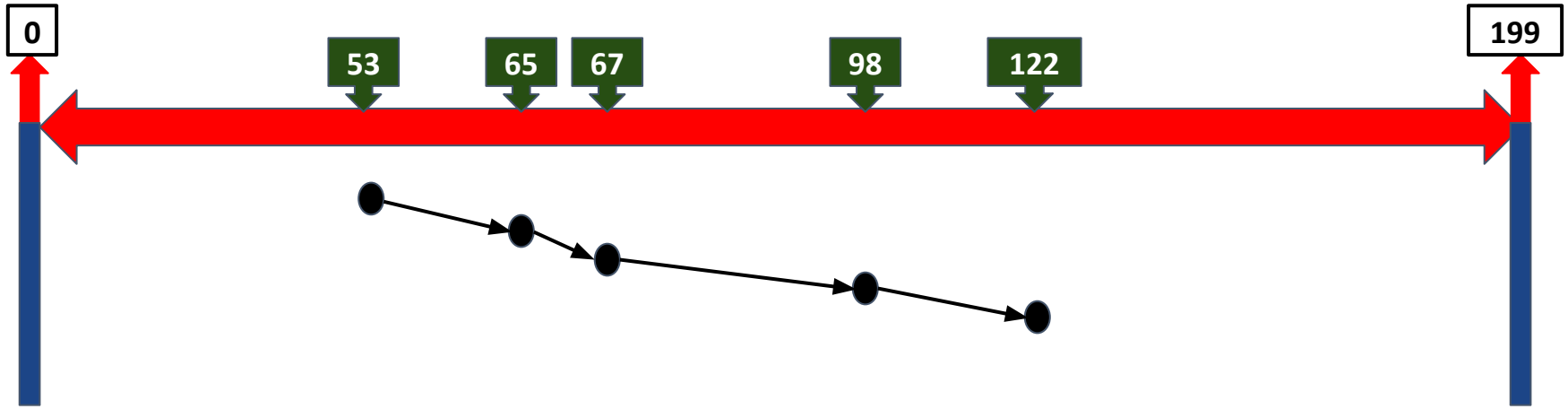
Seek Distance = 14

Seek Distance = Seek Distance + abs (Current Cylinder Number - New Cylinder Number)
Seek Distance = 14 + abs(67-98)
Seek Distance = 45

C - LOOK Disk Scheduling

- Request Queue => 0 .. 199
- 98, 183, 37, 122, 14, 124, 65, 67
- Head pointer initially @ 53



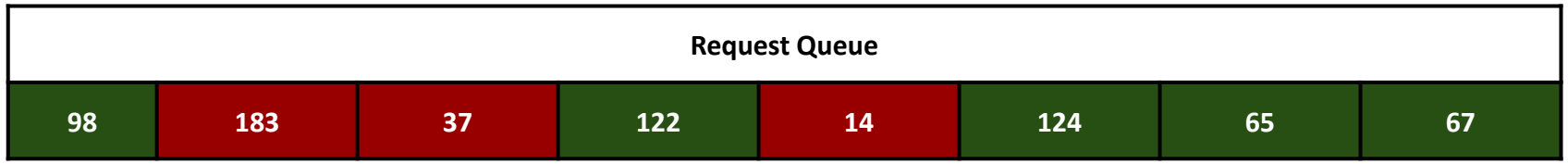


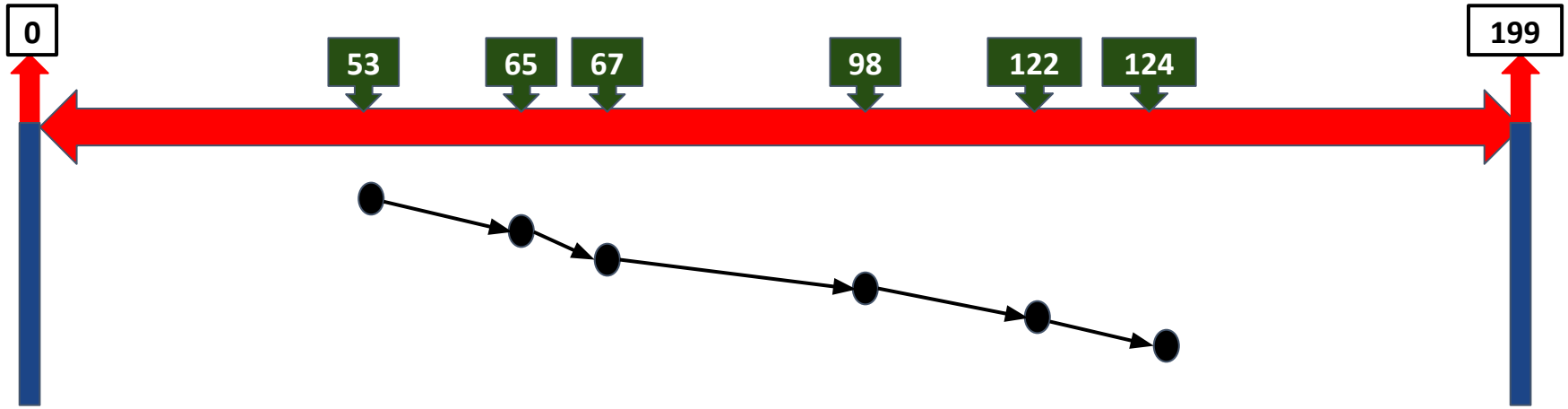
Seek Distance = 45

Seek Distance = Seek Distance + abs (Current Cylinder Number - New Cylinder Number)
Seek Distance = 45 + abs(98-122)
Seek Distance = 69

C - LOOK Disk Scheduling

- Request Queue => 0 .. 199
- 98, 183, 37, 122, 14, 124, 65, 67
- Head pointer initially @ 53



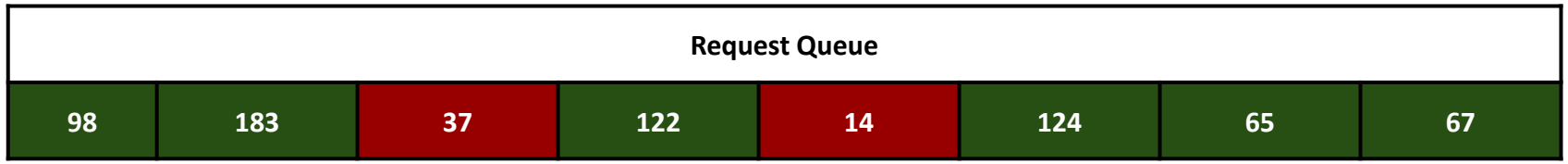


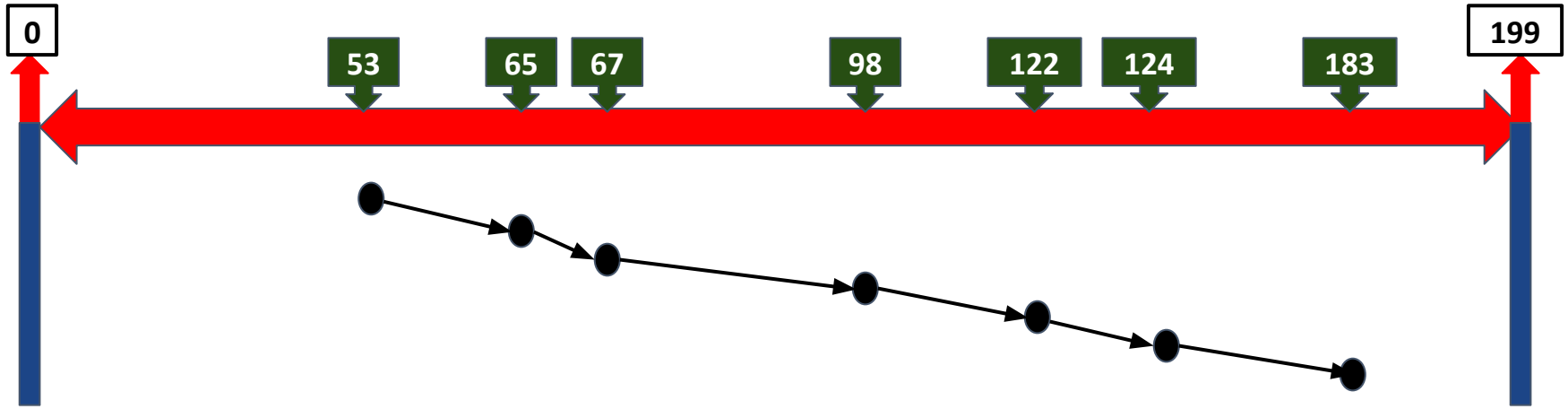
Seek Distance = 69

Seek Distance = Seek Distance + abs (Current Cylinder Number - New Cylinder Number)
Seek Distance = 69 + abs(122-124)
Seek Distance = 71

C - LOOK Disk Scheduling

- Request Queue => 0 .. 199
- 98, 183, 37, 122, 14, 124, 65, 67
- Head pointer initially @ 53



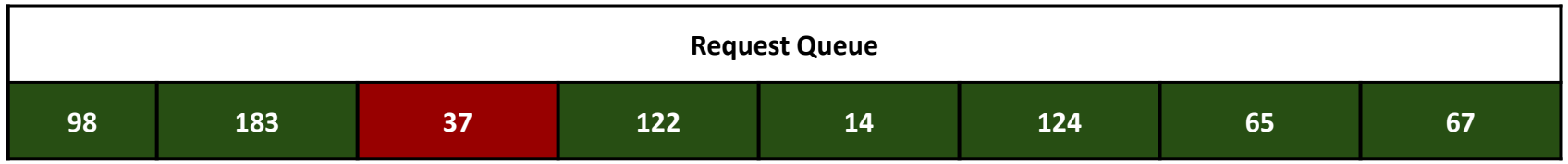


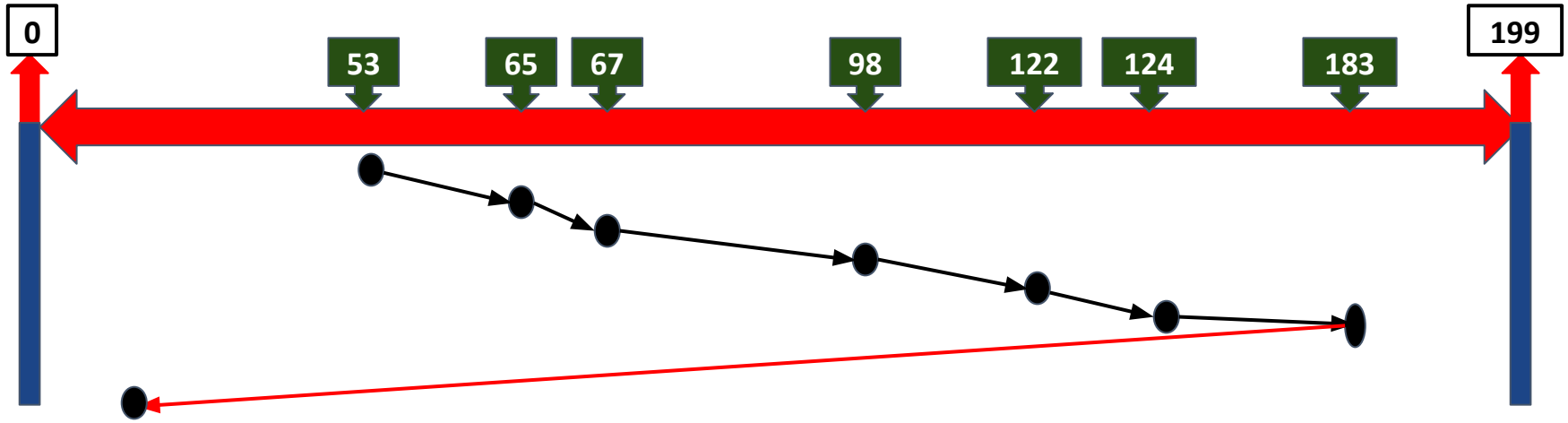
Seek Distance = 71

Seek Distance = Seek Distance + abs (Current Cylinder Number - New Cylinder Number)
Seek Distance = 69 + abs(124-183)
Seek Distance = 130

C - LOOK Disk Scheduling

- Request Queue => 0 .. 199
- 98, 183, 37, 122, 14, 124, 65, 67
- Head pointer initially @ 53



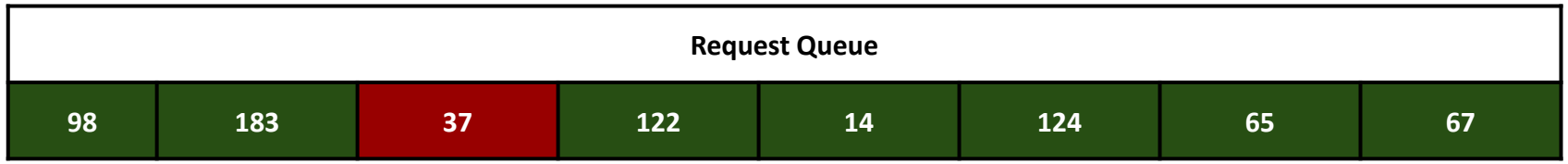


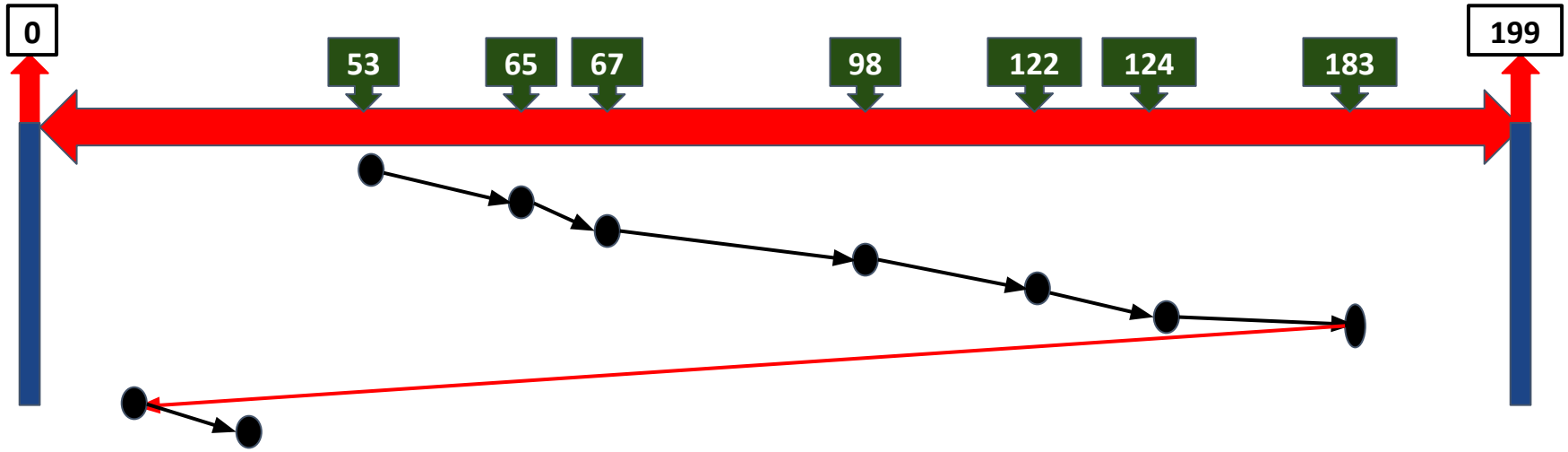
Seek Distance = 130

Seek Distance = Seek Distance + abs (Current Cylinder Number - New Cylinder Number)
Seek Distance = 130 + abs(183-14)
Seek Distance = 299

C - LOOK Disk Scheduling

- Request Queue => 0 .. 199
- 98, 183, 37, 122, 14, 124, 65, 67
- Head pointer initially @ 53





Seek Distance = 299

Seek Distance = Seek Distance + abs (Current Cylinder Number - New Cylinder Number)
Seek Distance = 299 + abs(14-37)
Seek Distance = 322

Disk Scheduling Performance - Example

- Request Queue => 0 .. 199
- 98, 183, 37, 122, 14, 124, 65, 67
- Head pointer initially @ 53

FCFS	SSTF	SCAN	C-SCAN	LOOK	C-LOOK
640	236	236	382	208	322

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
- 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?

Topics Uncovered Today

- SCAN Disk Scheduling
- C-SCAN Disk Scheduling
- LOOK Disc Scheduling
- C-LOOK Disc Scheduling

THANK YOU

Nitin V Pujari
Faculty, Computer Science
Dean - IQAC, PES University

nitin.pujari@pes.edu

For Course Deliverables by the Anchor Faculty click on www.pesuacademy.com