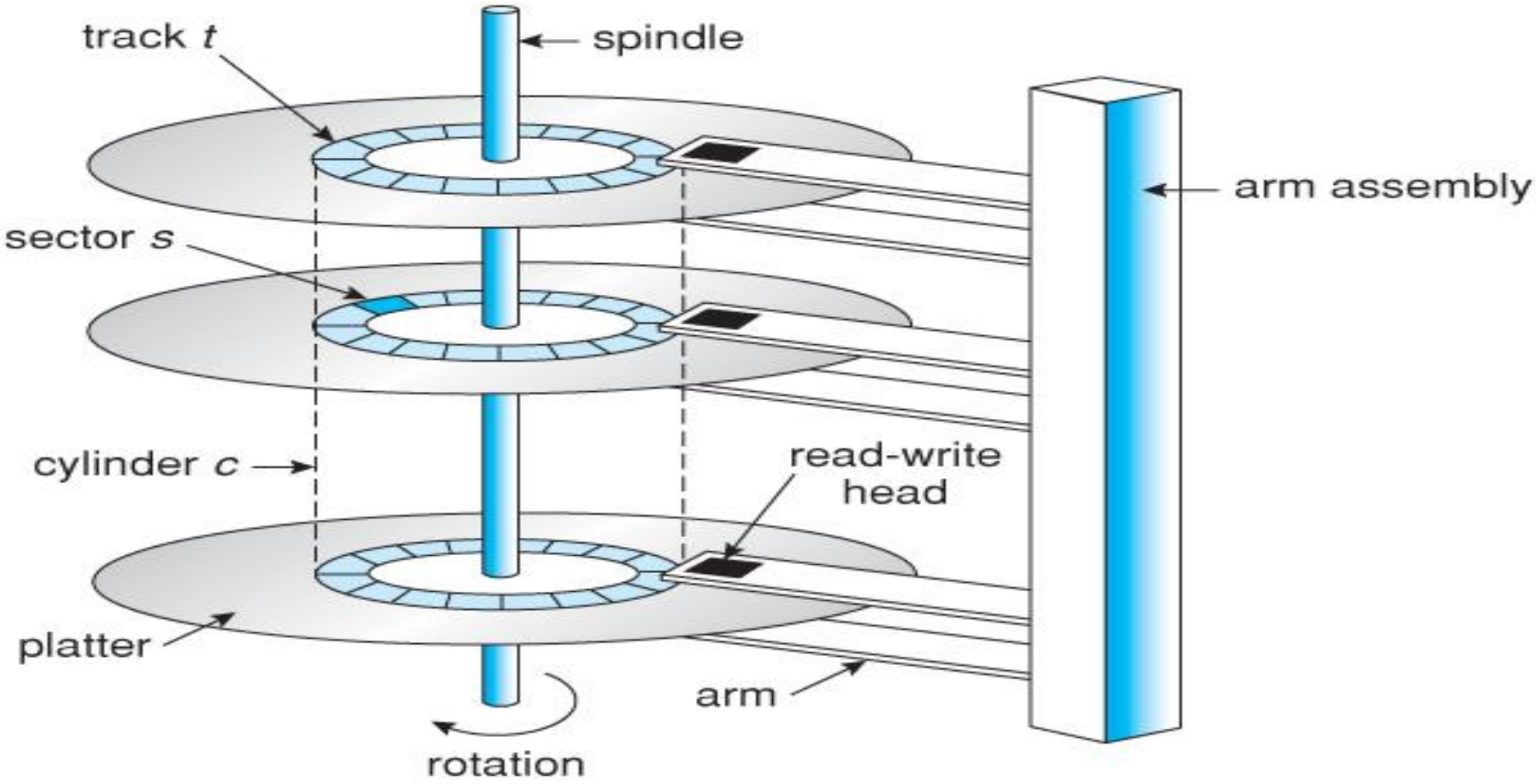
Operating System

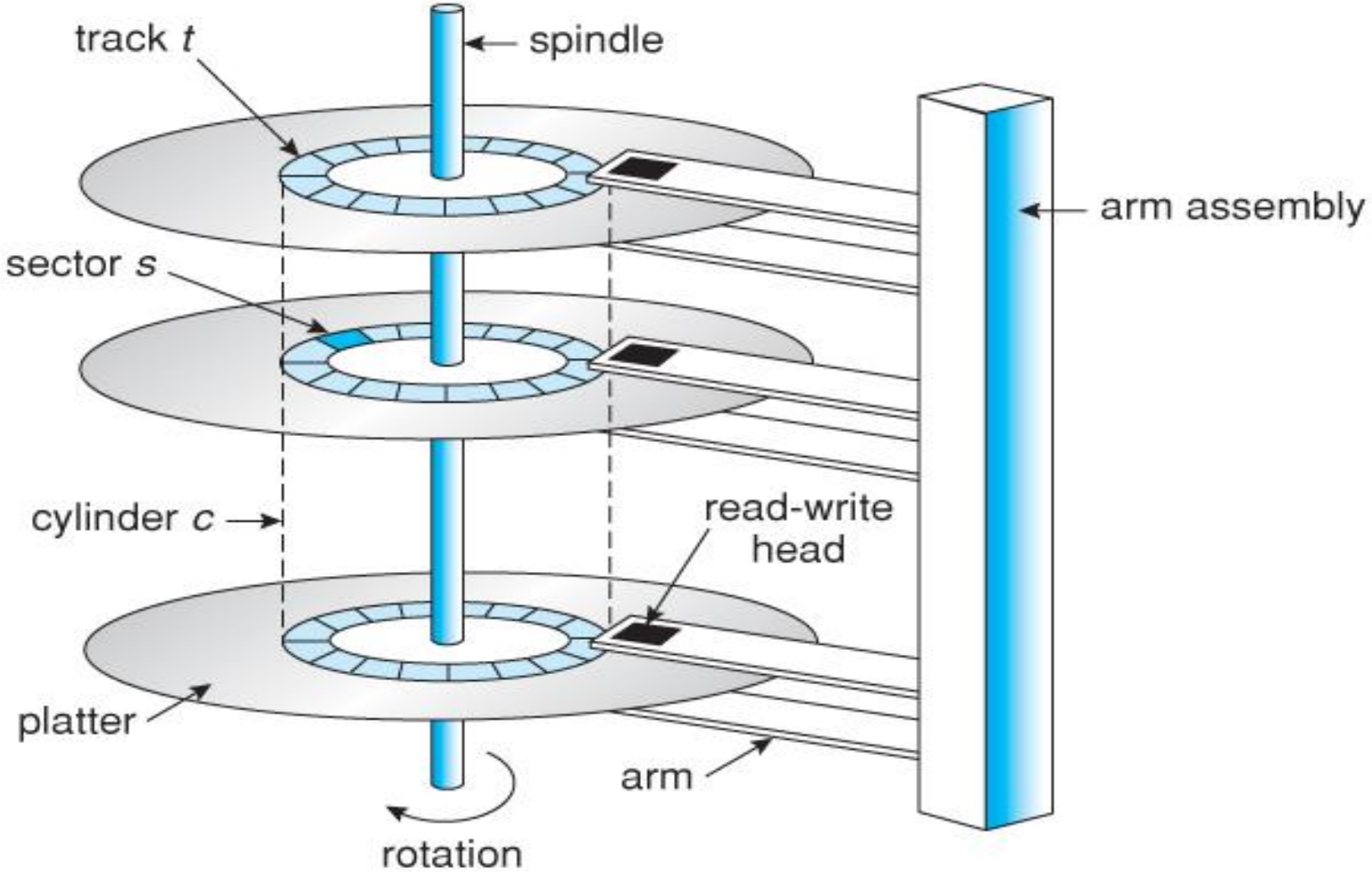
Unit – 8

Disk Management

Mayank Mishra
School of Electronics Engineering
KIIT-Deemed to be University

Hard Disk Architecture





Platter

↓

Surface

↓

Track

↓

Sectors

↓

Data

- Traditional magnetic disks have the following basic structure:
  - One or more ***platters*** in the form of disks covered with magnetic media. ***Hard disk*** platters are made of rigid metal, while "***floppy***" disks are made of more flexible plastic.
  - Each platter has two working ***surfaces.*** Older hard disk drives would sometimes not use the very top or bottom surface of a stack of platters, as these surfaces were more susceptible to potential damage.
  - Each working surface is divided into a number of concentric rings called ***tracks.*** The collection of all tracks that are the same distance from the edge of the platter, (i.e. all tracks immediately above one another as shown in the previous diagram) is called a ***cylinder***.
  - Each track is further divided into ***sectors,*** traditionally containing 512 bytes of data each, although some modern disks occasionally use larger sector sizes.

- The data on a hard drive is read by read-write ***heads.*** The standard configuration (shown below) uses one head per surface, each on a separate ***arm***, and controlled by a common ***arm assembly*** which moves all heads simultaneously from one cylinder to another. (Other configurations, including independent read-write heads, may speed up disk access, but involve serious technical difficulties.)

- The **storage capacity of a traditional disk drive** is equal to *the number of heads (i.e. the number of working surfaces), times the number of tracks per surface, times the number of sectors per track, times the number of bytes per sector*.

  - **Example:** 8 x 2 x 256(tracks) x 512 (sectors) x 512 KB (data)
    - ✓ Total number of sectors = 8 x 2 x 256(tracks) x 512 (sectors)
    - ✓ Total disk size = 8 x 2 x 256(tracks) x 512 (sectors) x 512 KB (data)
      = 1 TB
    - ✓ No. of bits required to represent disk size = 40

Disk Access Time

- **Seek Time :** Time taken by read / head to reach desired track.
- **Rotation Time:** Time taken for one full rotation (360 degree)
- **Rotational Latency:** Time taken to reach the desired sector (Half of the Rotation Time)
- **Transfer Time: (**Data to be Transfer) / (Transfer Rate)
  - Transfer rate- per second data how much we can transfer.
  - Transfer rate = (No. of heads/ surfaces) **x** Capacity of one track **x** Number of rotation per second
- **Disk access time : Seek Time + Rotation Time + Transfer Time + Controller Time (if given in question) + Queue Time (if given in question)**

- **Example:** Consider a hard disk with:
  - 4 surfaces
  - 64 tracks/surface
  - 128 sectors/track
  - 256 bytes/sector

  What is the capacity of the hard disk? The disk is rotating at 3600 RPM, what is the data transfer rate?

- **Example:** Consider a hard disk with:
  - 4 surfaces
  - 64 tracks/surface
  - 128 sectors/track
  - 256 bytes/sector

What is the capacity of the hard disk? The disk is rotating at 3600 RPM, what is the data transfer rate?

- ***Disk capacity = surfaces * tracks/surface * sectors/track * bytes/sector***

  *Disk capacity = 4 * 64 * 128 * 256*

  Disk capacity = 8 MB

- *60 sec -> 3600 rotations*

  1 sec -> 60 rotations

  Data transfer rate = number of rotations per second * track capacity * number of surfaces

  (since 1 R-W head is used for each surface)

  *Data transfer rate = 60 * 128 * 256 * 4*

  Data transfer rate = 7.5 MB/sec

Disk Scheduling Algorithms

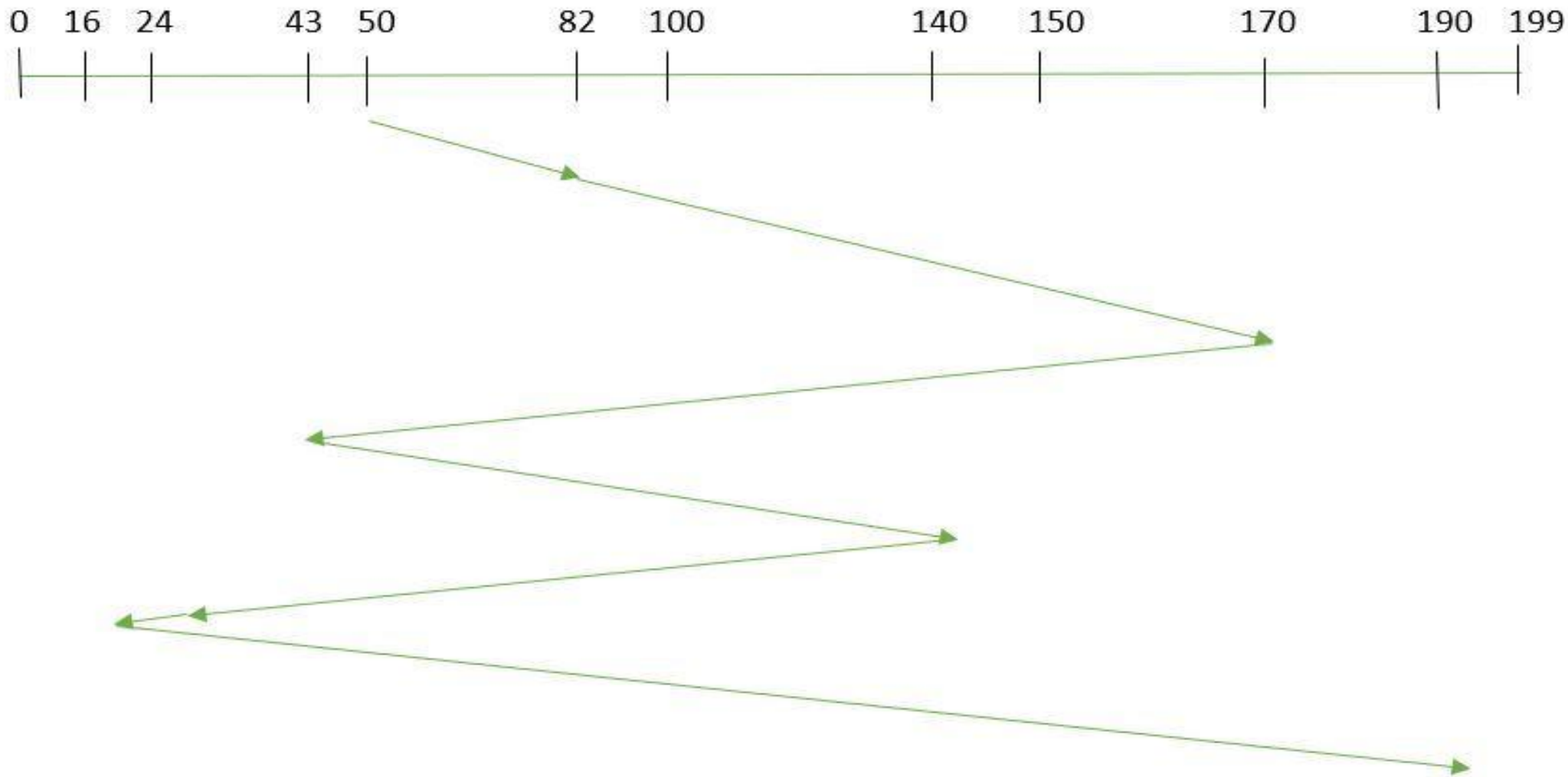
- Disk scheduling algorithms are crucial in managing how data is read from and written to a computer’s hard disk.
- These algorithms help determine the order in which disk read and write requests are processed, significantly impacting the speed and efficiency of data access.
- The goal is to minimize the seek time.
- Common disk scheduling methods include First-Come, First-Served (FCFS), Shortest Seek Time First (SSTF), SCAN, C-SCAN, LOOK, and C-LOOK.

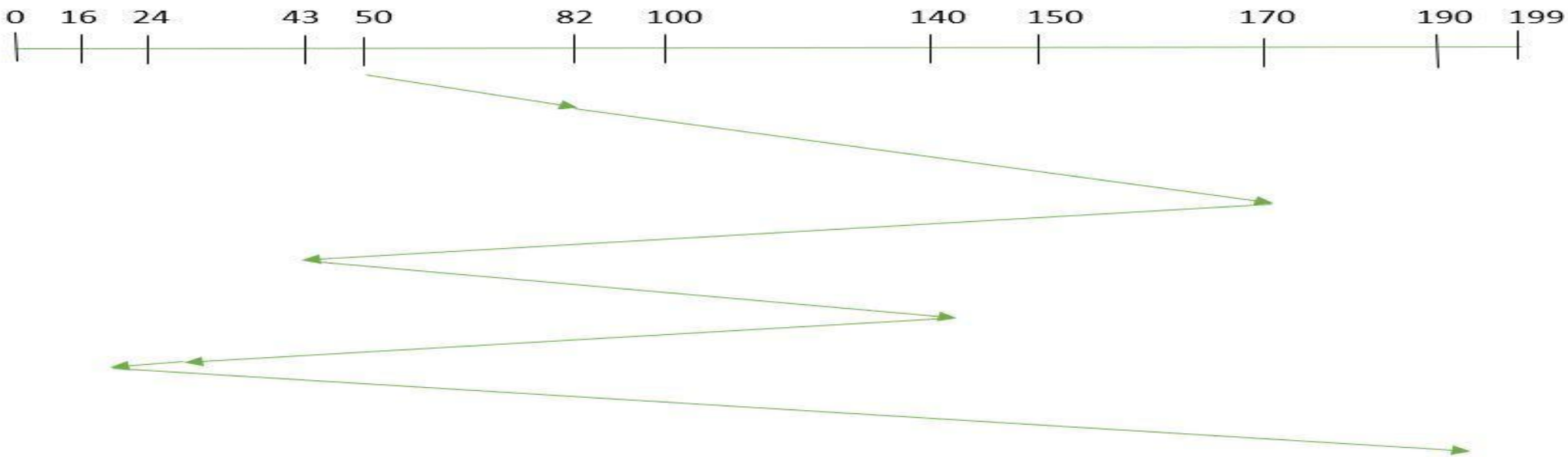
Disk Scheduling Algorithms

➢ First Come First Serve (FCFS)

- ❑ FCFS is the simplest of all Disk Scheduling Algorithms. In FCFS, the requests are addressed in the order they arrive in the disk queue.

- ❑ Example: Suppose the order of request is- (82,170,43,140,24,16,190), and current position of Read/Write head is: 50 . Calculate the total overhead movement (total distance covered by the disk arm)


0
16
24
43
50
82
100
140
150
170
190
199



So, total overhead movement (total distance covered by the disk arm) =

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

Advantages of FCFS

- Every request gets a fair chance
- No indefinite postponement

Disadvantages of FCFS

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

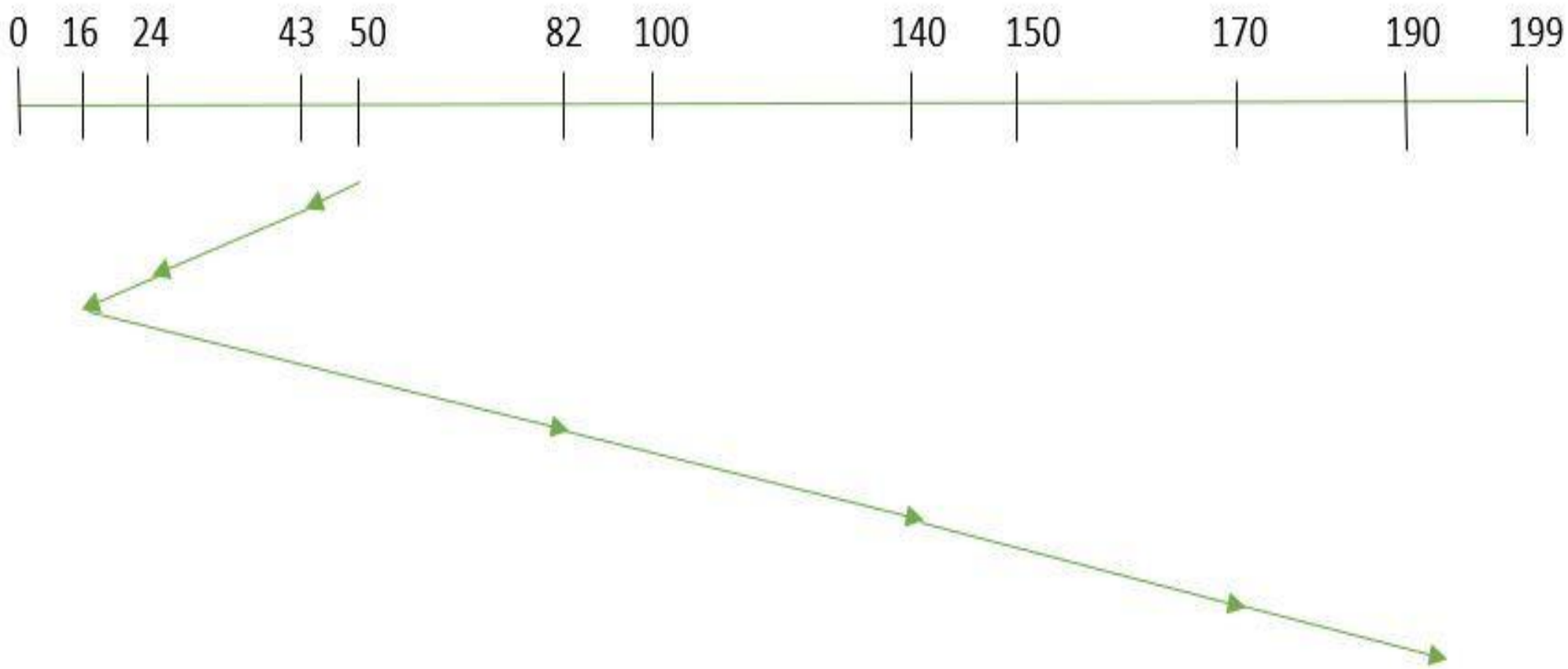
➢ Shortest Seek Time First (SSTF)

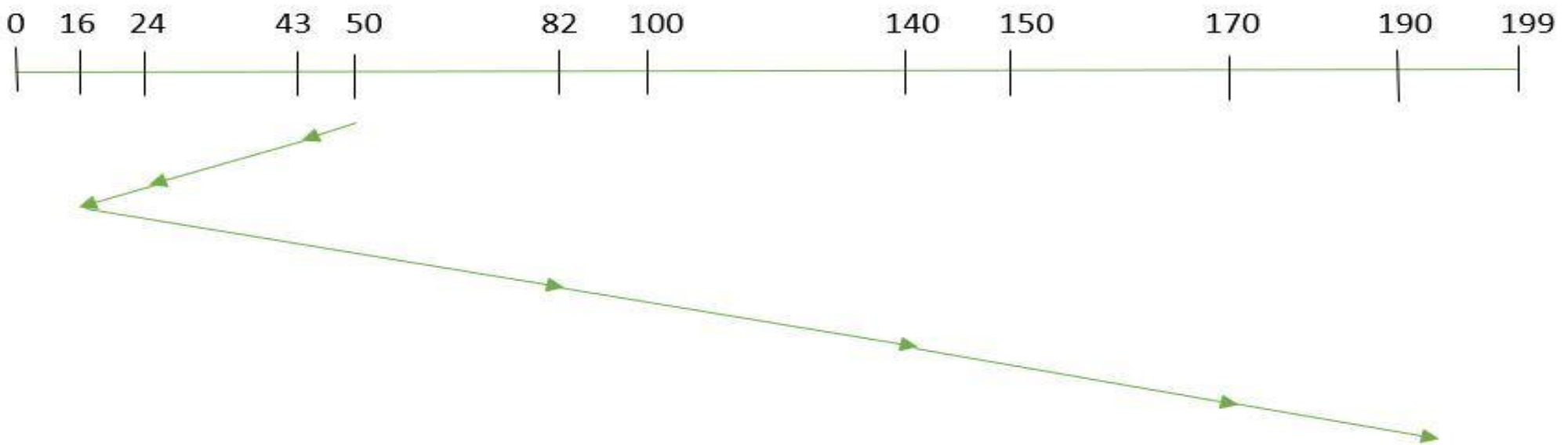
- In SSTF (Shortest Seek Time First), requests having the shortest seek time are executed first. So, the seek time of every request is calculated in advance in the queue and then they are scheduled according to their calculated seek time.

- As a result, the request near the disk arm will get executed first. SSTF is certainly an improvement over FCFS as it decreases the average response time and increases the throughput of the system.

- Example: Suppose the order of request is- (82,170,43,140,24,16,190) and current position of Read/Write head is: 50. Calculate the total overhead movement (total distance covered by the disk arm)

The order of request is- **82,170,43,140,24,16,190**





total overhead movement (total distance covered by the disk arm) =

(50-43)+(43-24)+(24-16)+(82-16)+(140-82)+(170-140)+(190-170) =208

Advantages of SSTF

- The average Response Time decreases
- Throughput increases

Disadvantages of SSTF

- Overhead to calculate seek time in advance
- Can cause Starvation for a request if it has a higher seek time as compared to incoming requests

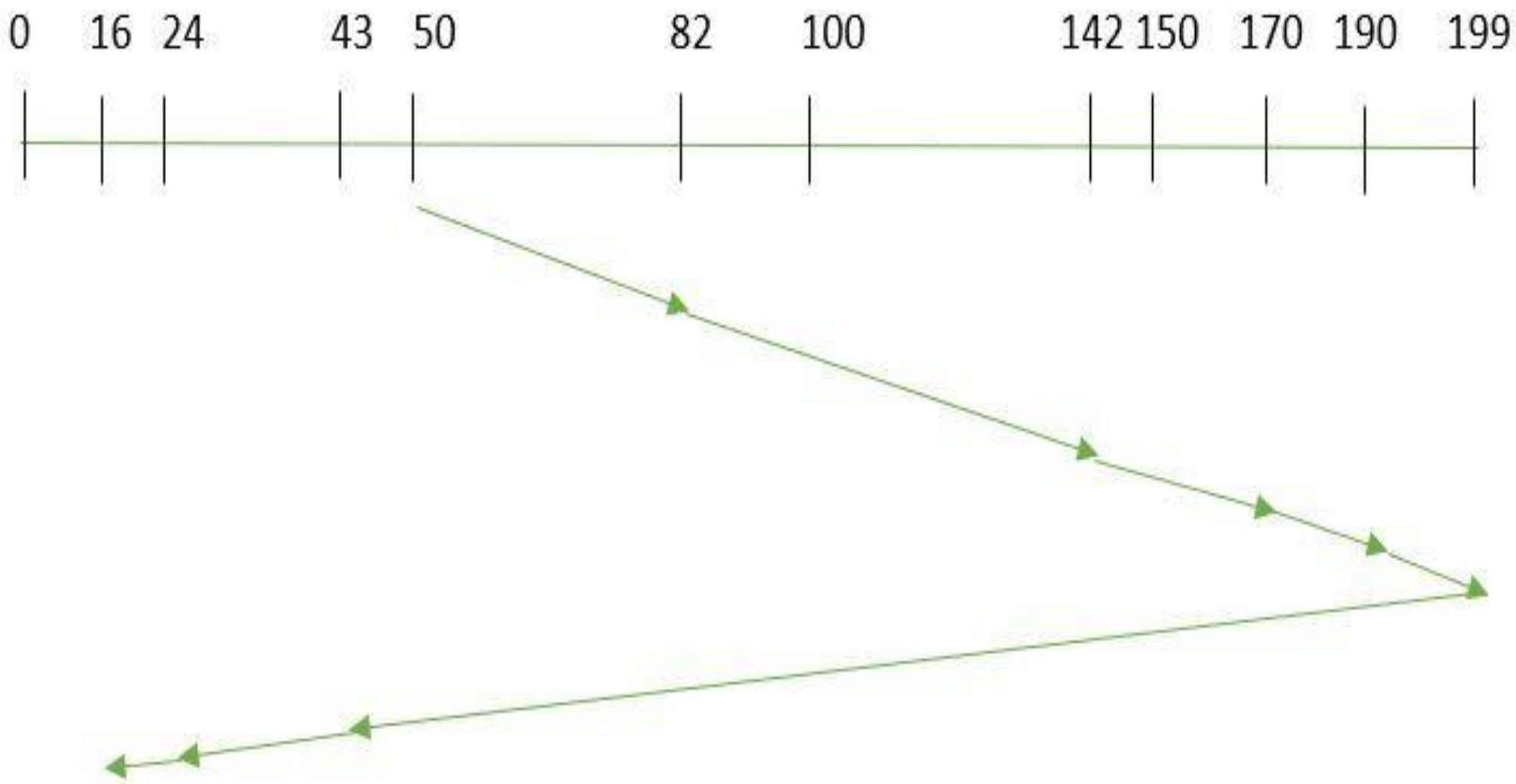
➢ SCAN

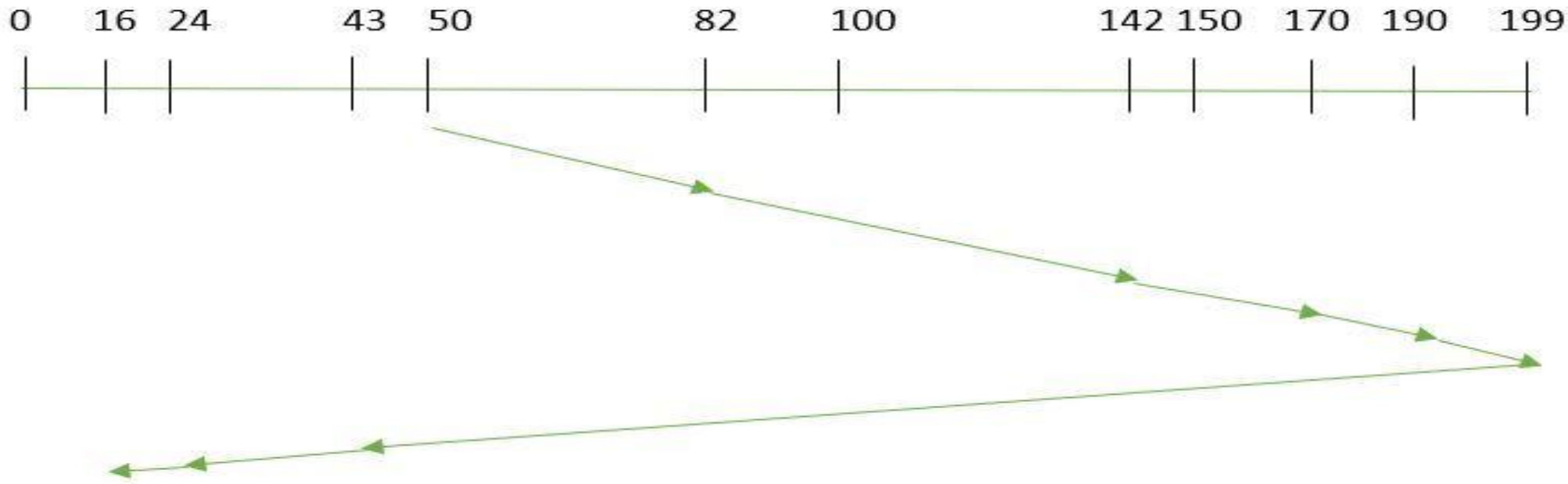
- In the **SCAN** algorithm the disk arm moves in a particular direction and services the requests coming in its path and after reaching the end of the disk, it reverses its direction and again services the request arriving in its path.

- In the **SCAN** algorithm the disk arm moves in a particular direction and services the requests coming in its path and after reaching the end of the disk, it reverses its direction and again services the request arriving in its path.

- **Example:** A disk contains 200 tracks (0-199). Suppose the requests to be addressed are-82,170,43,140,24,16,190. And the Read/Write arm is at 50, and it is also given that the disk arm should move "**towards the larger value**". Calculate the total overhead movement (total distance covered by the disk arm). If R/W head takes 1 nanosecond to move from one track to another then what will be the total time taken by R/W head?

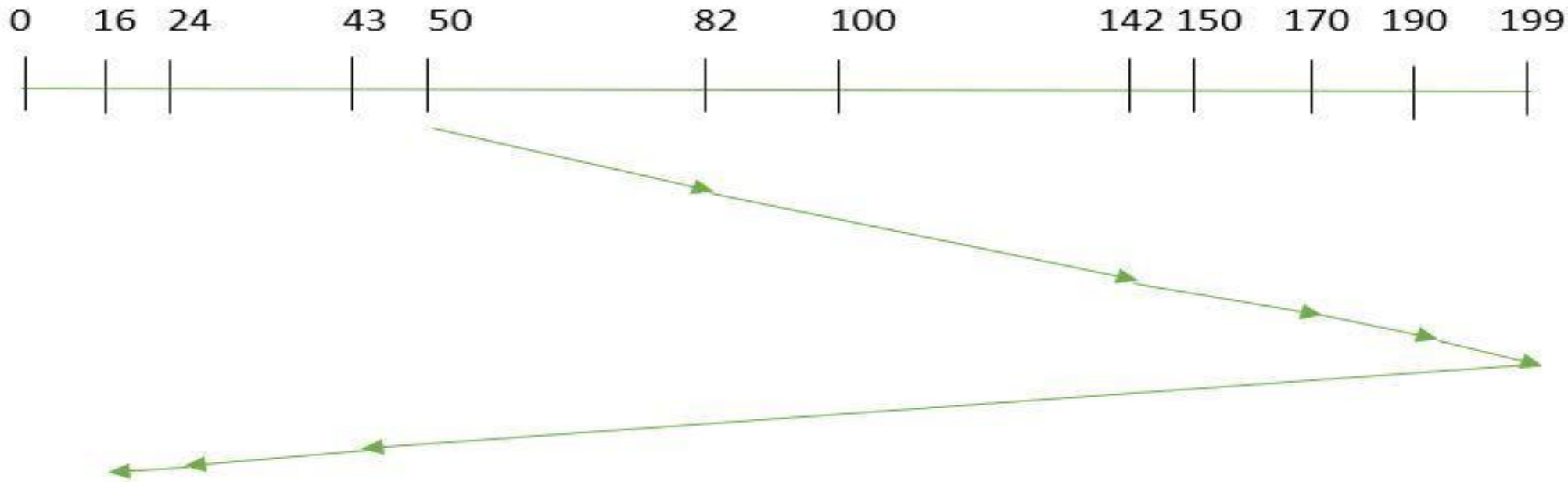
The order of request is- **82,170,43,140,24,16,190**





Therefore, the total overhead movement (total distance covered by the disk arm) is calculated as

```
= (199-50) + (199-16) = 332
```



Total time taken by R/W head = 332 x 1 **ns**

Advantages of SCAN

- ❑ High throughput
- ❑ Low variance of response time
- ❑ Average response time

Disadvantages of SCAN

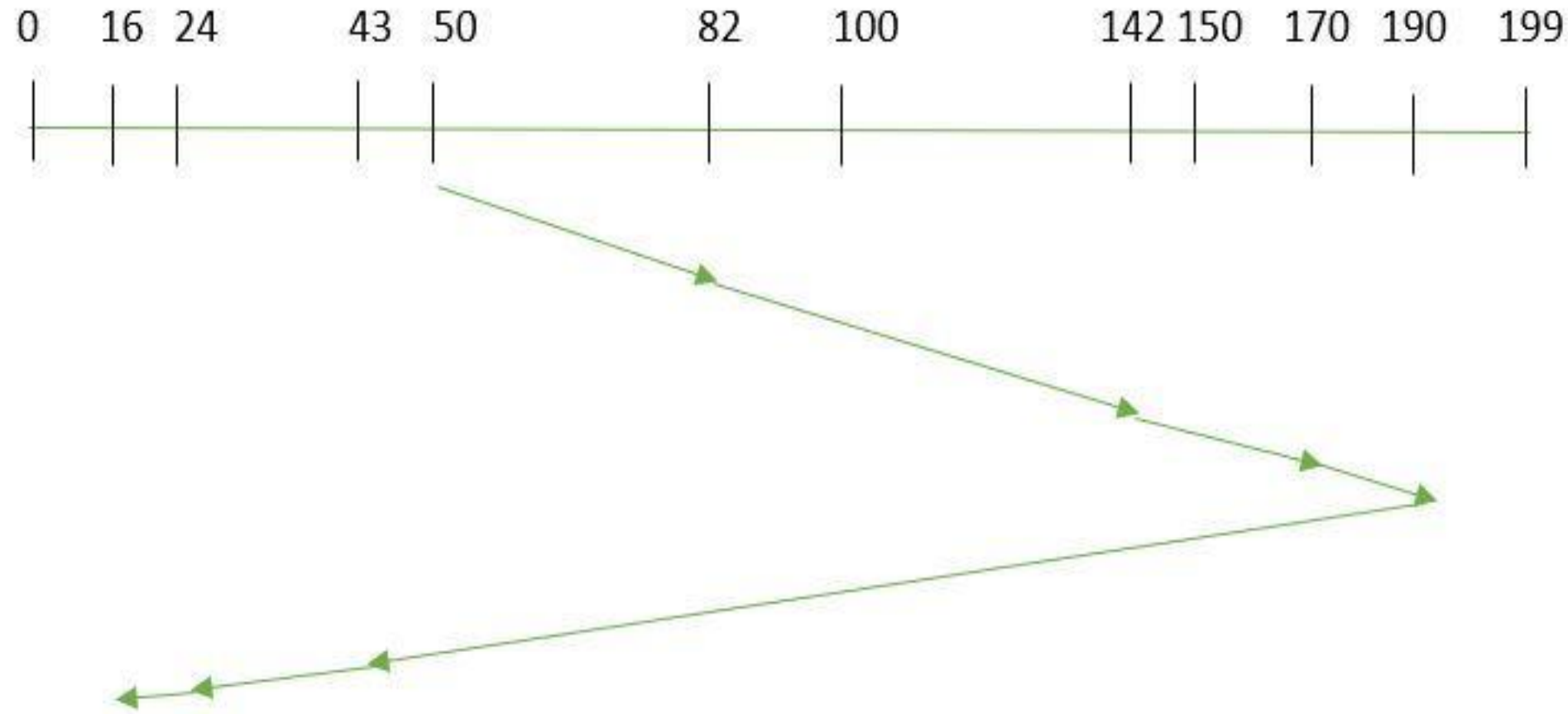
- ❑ Long waiting time for requests for locations just visited by disk.

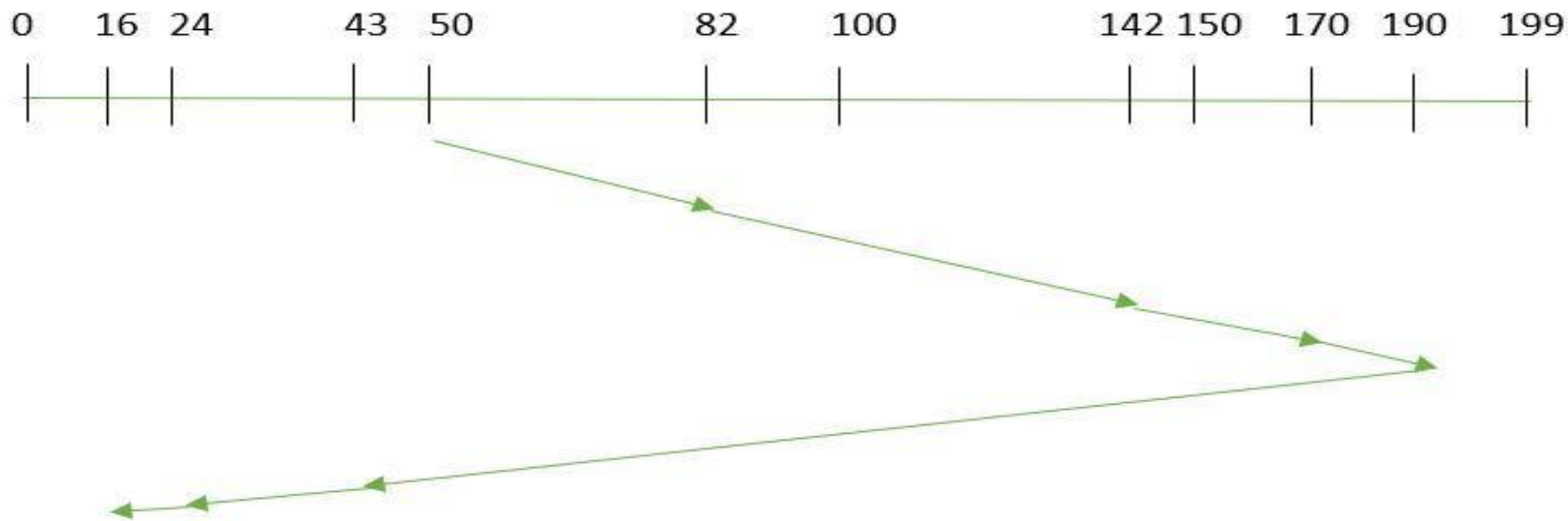
➢ LOOK

- LOOK Algorithm is similar to the SCAN disk scheduling algorithm except for the difference that the disk arm in spite of going to the end of the disk goes only to the last request to be serviced in front of the head and then reverses its direction from there only. Thus, **it prevents the extra delay** which occurred due to unnecessary traversal to the end of the disk.

- **Example:** A disk contains 200 tracks (0-199). Suppose the requests to be addressed are-82,170,43,140,24,16,190. And the Read/Write arm is at 50, and it is also given that the disk arm should move "**towards the larger value**". Calculate the total overhead movement (total distance covered by the disk arm)

The order of request is- **82,170,43,140,24,16,190**





So, the total overhead movement (total distance covered by the disk arm) is calculated as:

```
= (190-50) + (190-16) = 314
```

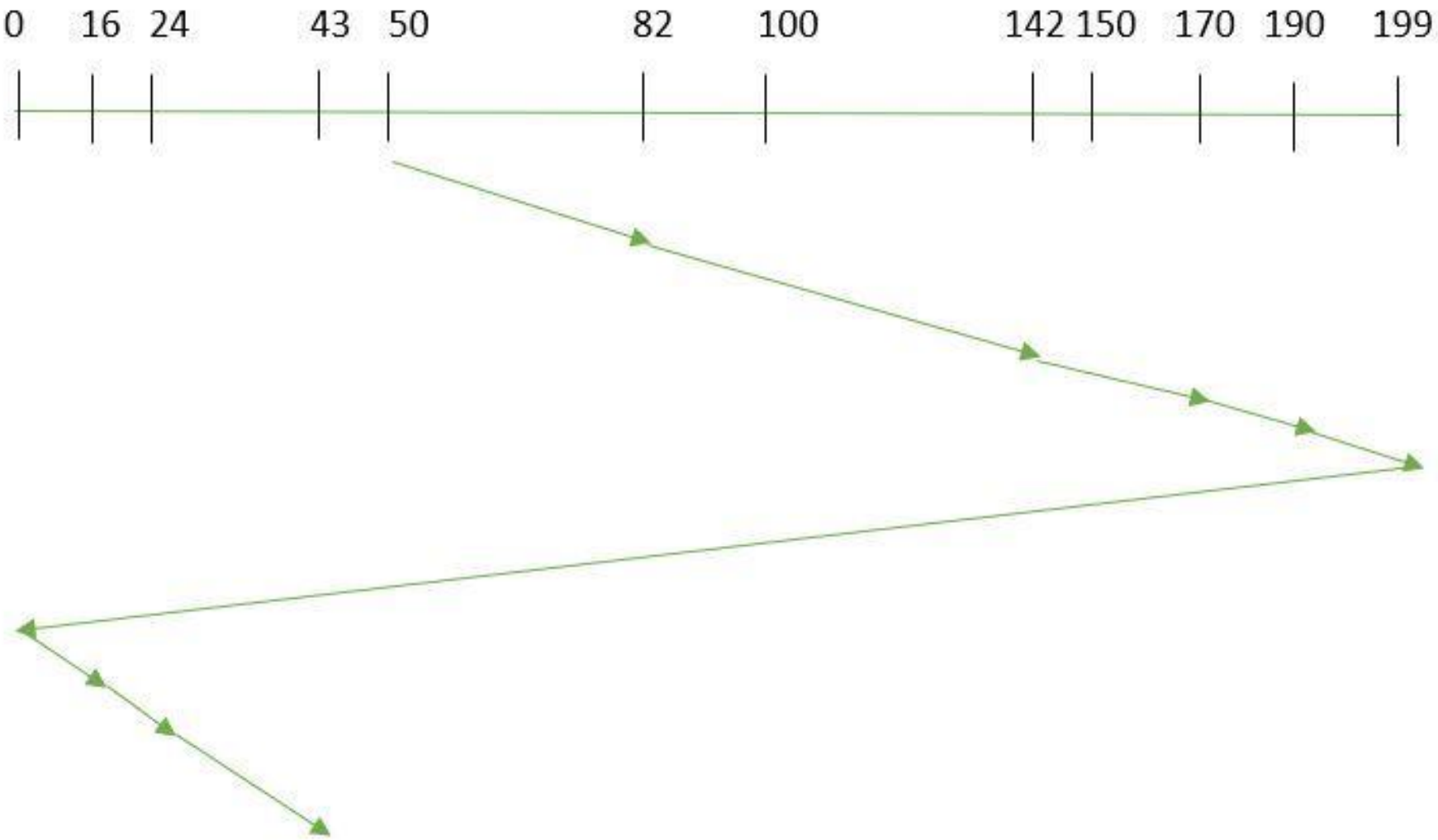
➢ C-SCAN

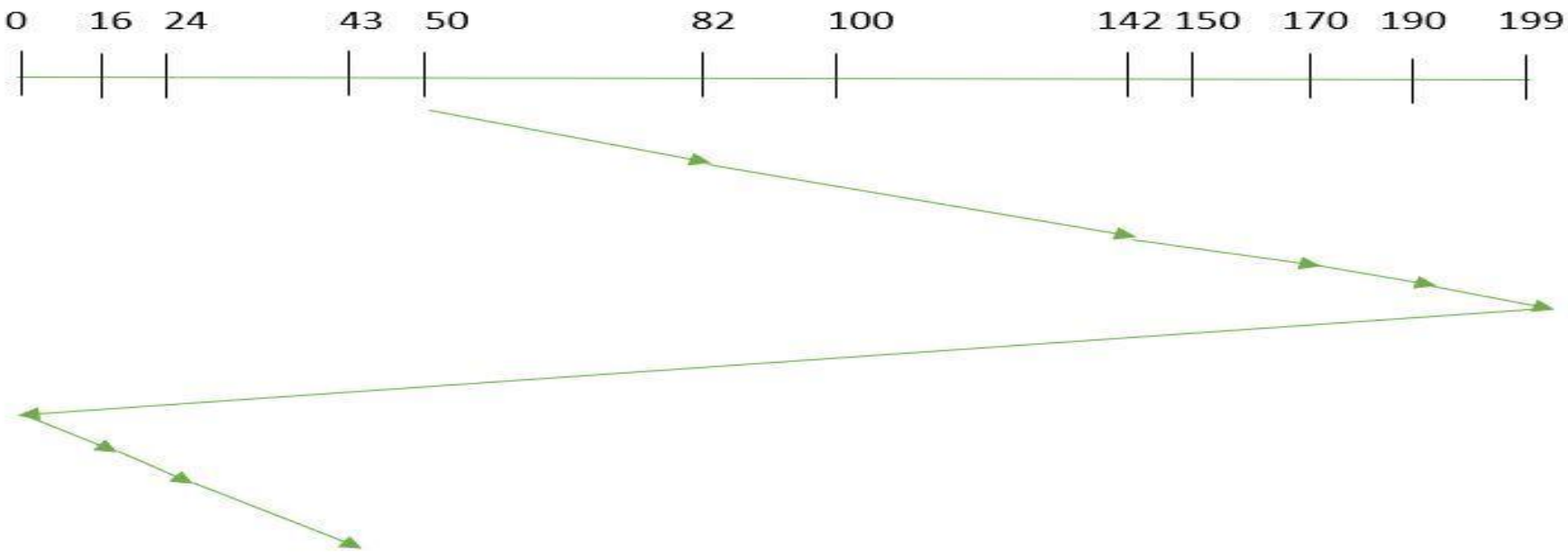
- In the SCAN algorithm, the disk arm again scans the path that has been scanned, after reversing its direction. So, it may be possible that too many requests are waiting at the other end or there may be zero or few requests pending at the scanned area.

- These situations are avoided in the CSCAN algorithm in which the disk arm instead of reversing its direction goes to the other end of the disk and starts servicing the requests from there. So, the disk arm moves in a circular fashion and this algorithm is also similar to the SCAN algorithm hence it is known as C-SCAN (**Circular SCAN**).

- **Example:** A disk contains 200 tracks (0-199). Suppose the requests to be addressed are-82,170,43,140,24,16,190. And the Read/Write arm is at 50, and it is also given that the disk arm should move "**towards the larger value**". Calculate the total overhead movement (total distance covered by the disk arm)

The order of request is- **82,170,43,140,24,16,190**





So, the total overhead movement (total distance covered by the disk arm) is calculated as:

```
=(199-50) + (199-0) + (43-0) = 391
```

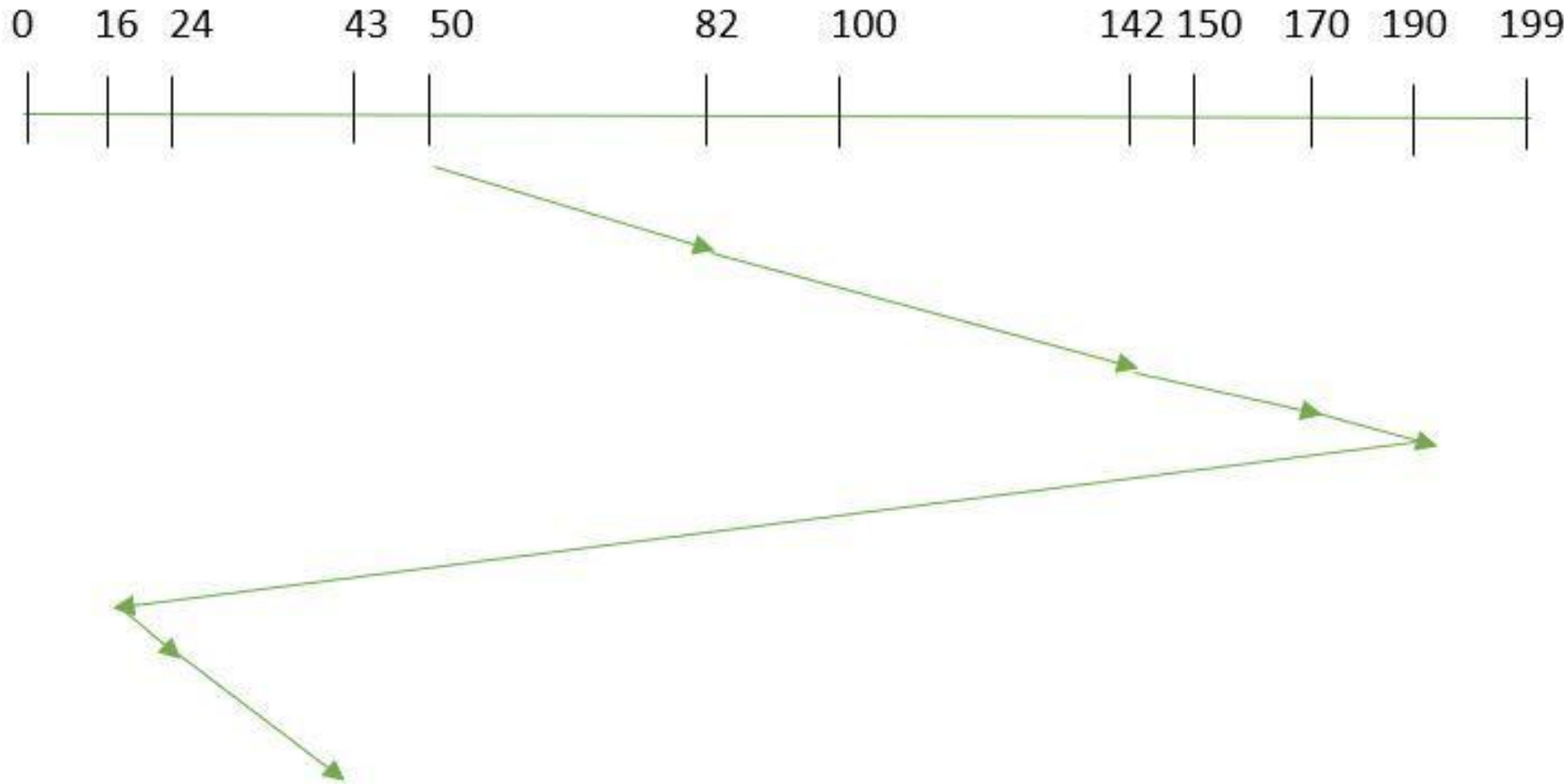
➢ C-LOOK

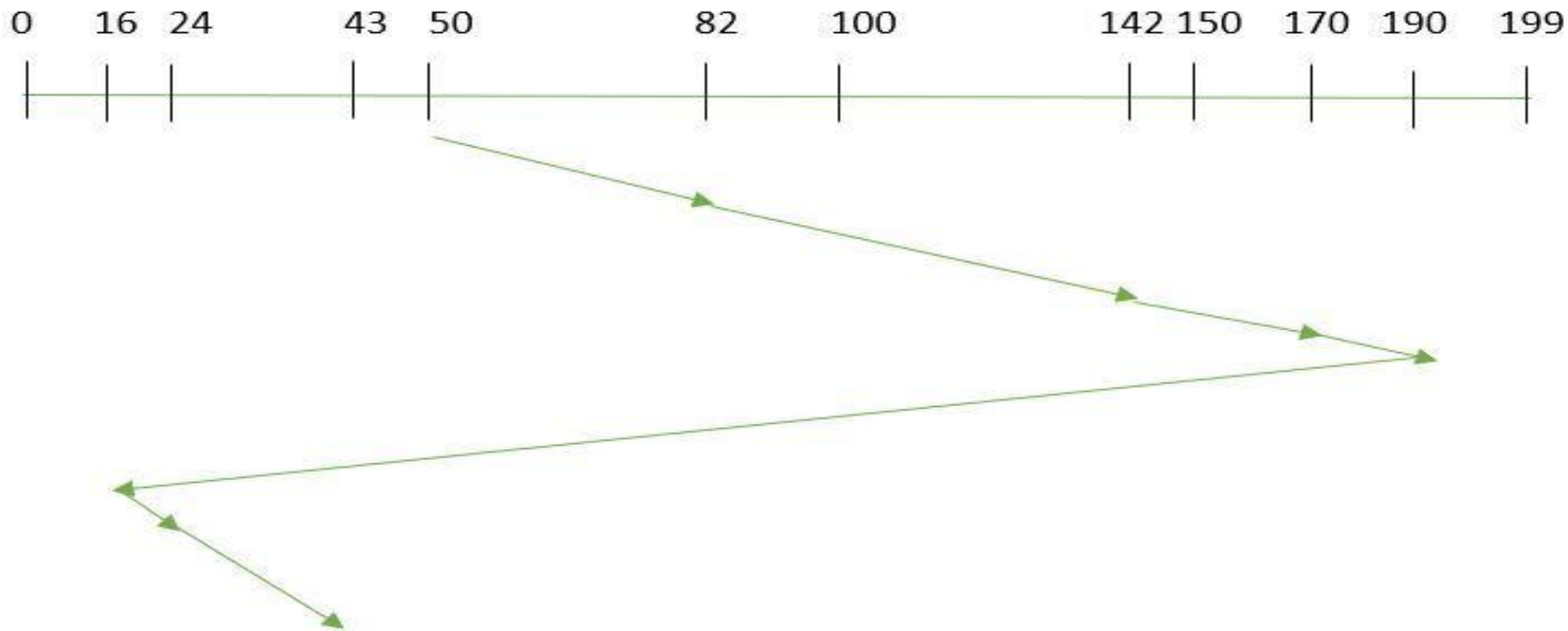
- As LOOK is similar to the SCAN algorithm, in a similar way, C-LOOK is similar to the CSCAN disk scheduling algorithm.

- In CLOOK, the disk arm in spite of going to the end goes only to the last request to be serviced in front of the head and then from there goes to the other end's last request. Thus, it also prevents the extra delay which occurred due to unnecessary traversal to the end of the disk.

- **Example:** A disk contains 200 tracks (0-199). Suppose the requests to be addressed are-82,170,43,140,24,16,190. And the Read/Write arm is at 50, and it is also given that the disk arm should move "**towards the larger value**". Calculate the total overhead movement (total distance covered by the disk arm)

The order of request is- **82,170,43,140,24,16,190**





So, the total overhead movement (total distance covered by the disk arm) is calculated as

```
= (190-50) + (190-16) + (43-16) = 341
```

References

1. Abraham Silberschatz, Peter Baer Galvin, Greg Gagne, “Operating System Concepts,” Eleventh Edition (Willey).

2. Andrew S. Tanenbaum, “Modern Operating Systems”, Fourth Edition (Pearson Publications), 2014.

3. https://www.geeksforgeeks.org/

4. https://www.javatpoint.com/

5. https://www.tutorialspoint.com/

6. https://www.nesoacademy.org/

7. https://www.tpointtech.com/