Disk Scheduling Algorithms

Disk scheduling is done by operating systems to schedule I/O requests arriving for the disk. Disk scheduling is also known as I/O scheduling. Disk scheduling is important because:

- Multiple I/O requests may arrive by different processes and only one I/O request can be served at a time by the disk controller. Thus other I/O requests need to wait in the waiting queue and need to be scheduled.
- Two or more requests may be far from each other so can result in greater disk arm movement.
- Hard drives are one of the slowest parts of the computer system and thus need to be accessed in an efficient manner.

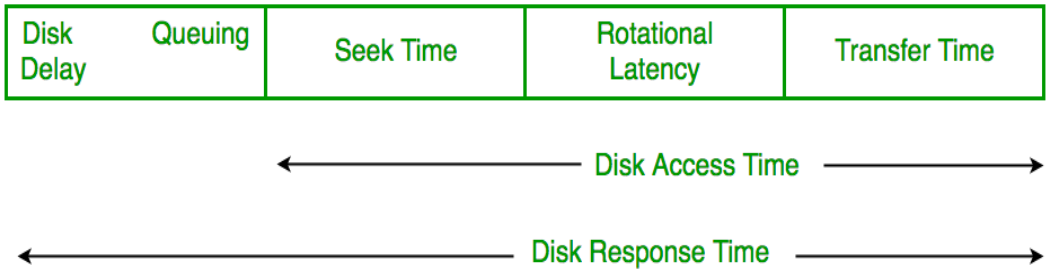
There are many Disk Scheduling Algorithms but before discussing them let's have a quick look at some of the important terms:

- **Seek Time:** Seek time is the time taken to locate the disk arm to a specified track where the data is to be read or write. So the disk scheduling algorithm that gives minimum average seek time is better.
- **Rotational Latency:** Rotational Latency is the time taken by the desired sector of disk to rotate into a position so that it can access the read/write heads. So the disk scheduling algorithm that gives minimum rotational latency is better.

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

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

Total Seek Time = Total head Movement * Seek Time



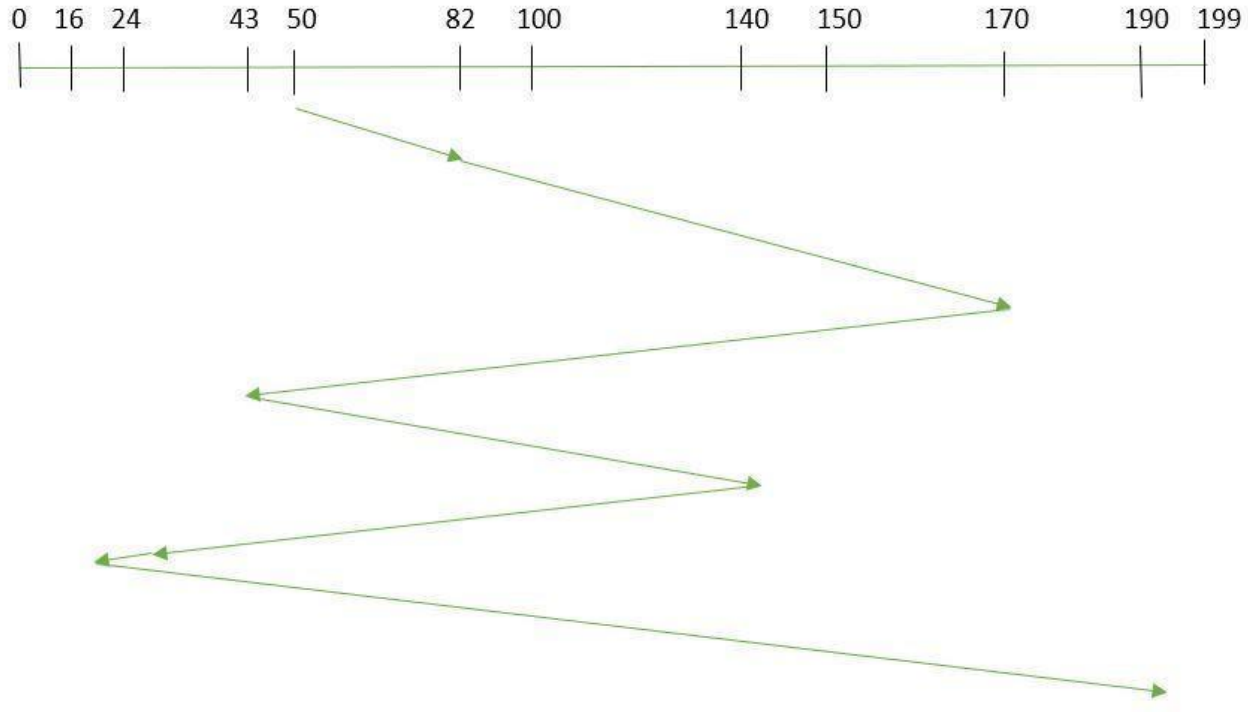
- **Disk Response Time:** Response Time is the average of time spent by a request waiting to perform its I/O operation. *Average Response time* is the response time of the all requests. *Variance Response Time* is measure of how individual request are serviced with respect to average response time. So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

Disk Scheduling Algorithms

1. **FCFS:** FCFS is the simplest of all the Disk Scheduling Algorithms. In FCFS, the requests are addressed in the order they arrive in the disk queue. Let us understand this with the help of an example.

Example:

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



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:

- Every request gets a fair chance
- No indefinite postponement

Disadvantages:

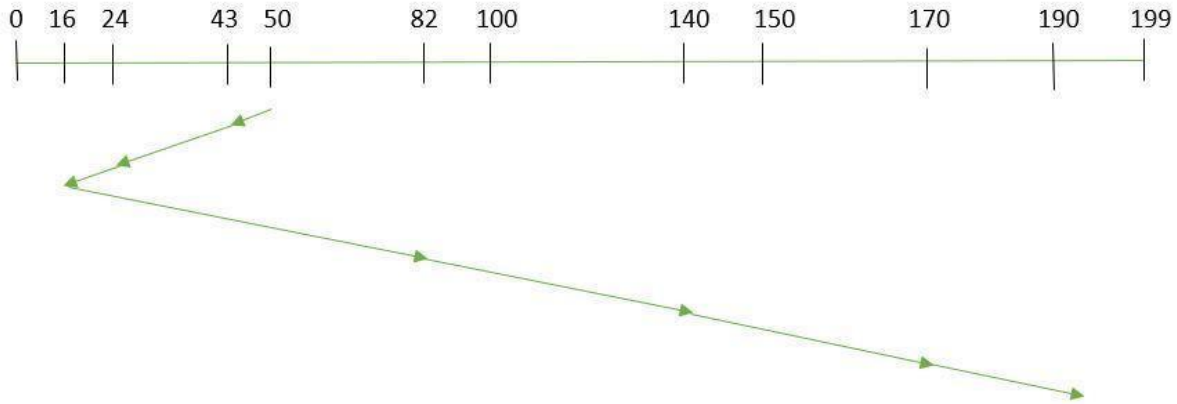
- Does not try to optimize seek time
- May not provide the best possible service

•

SSTF: In SSTF (Shortest Seek Time First), requests having 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 system.Let us understand this with the help of an example.

Example:

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



So,

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:

- Average Response Time decreases
- Throughput increases

Disadvantages:

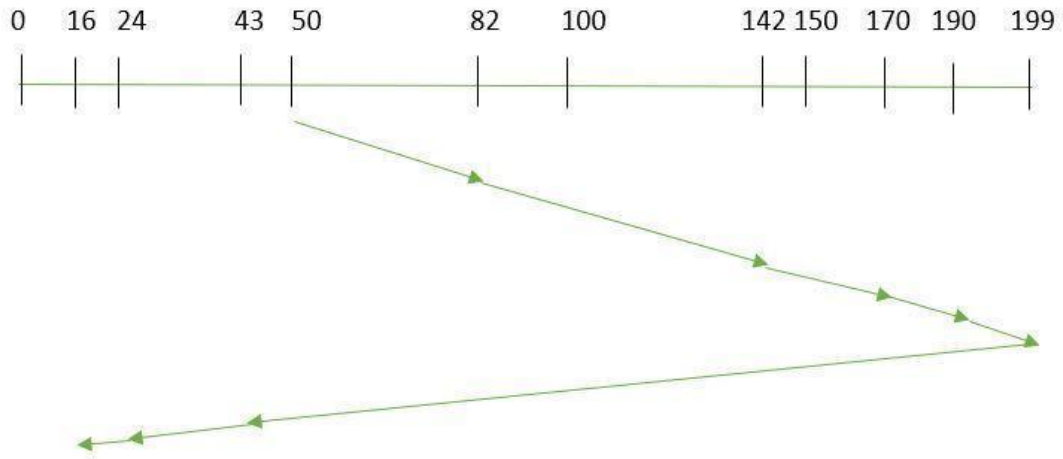
- Overhead to calculate seek time in advance
- Can cause Starvation for a request if it has a higher seek time as compared to incoming requests
- High variance of response time as SSTF favors only some requests

SCAN: In 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***. So, this algorithm works as an elevator and is hence also known as an **elevator algorithm.** As a result, the requests at the midrange are serviced more and those arriving behind the disk arm will have to wait.

Example:

1. 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".



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

1. =(199-50)+(199-16) =332

Advantages:

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

Disadvantages:

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

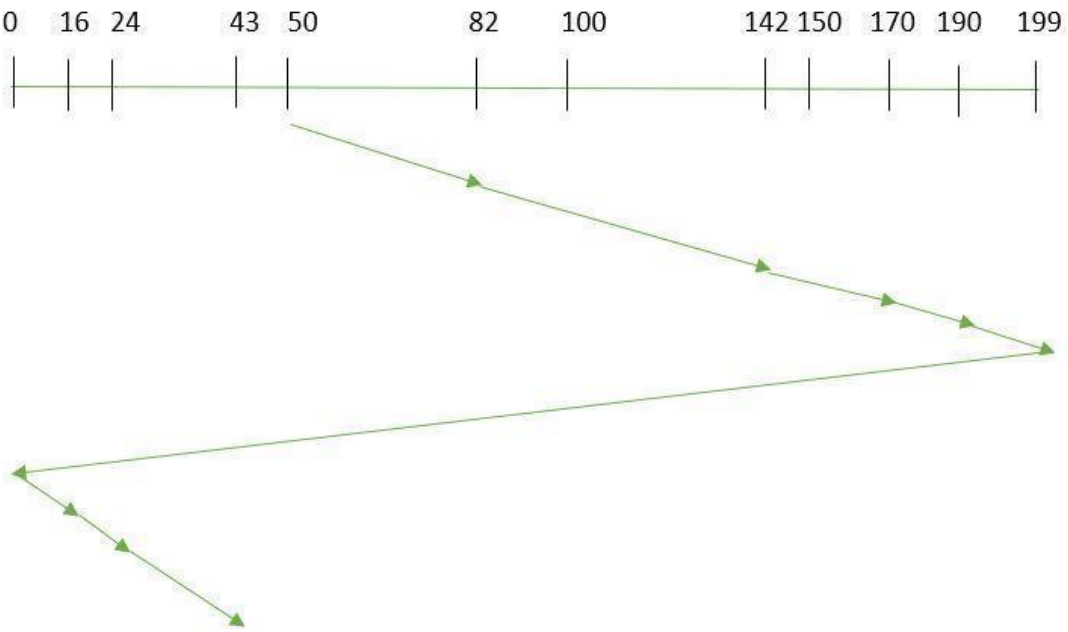
CSCAN: In SCAN algorithm, the disk arm again scans the path that has been scanned, after reversing its direction. So, it may be possible that too many requests are waiting at the

other end or there may be zero or few requests pending at the scanned area.

These situations are avoided in *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 SCAN algorithm and hence it is known as C-SCAN (Circular SCAN).

Example:

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



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

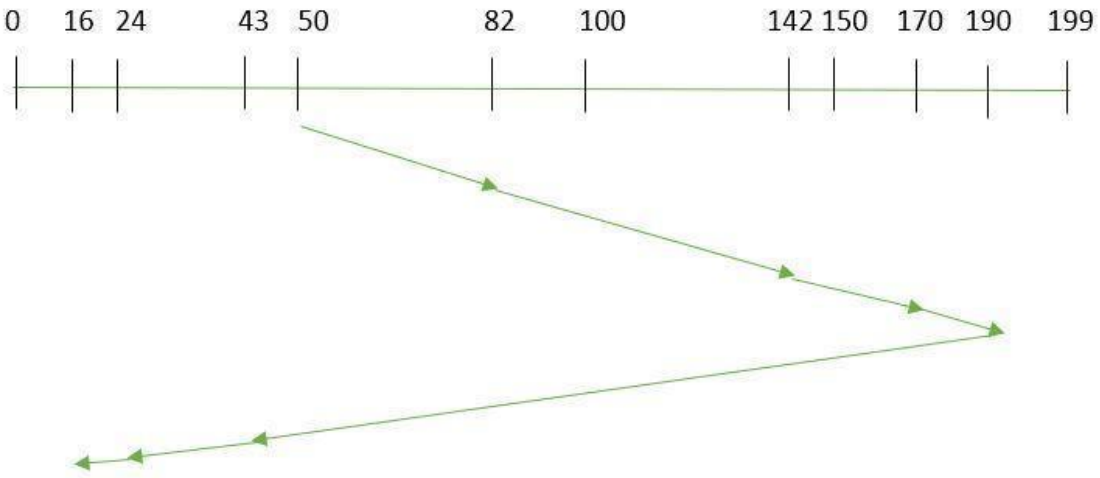
=(199-50)+(199-0)+(43-0) =391 Advantages:

- Provides more uniform wait time compared to SCAN

1. **LOOK:** It 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:

1. 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".**



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

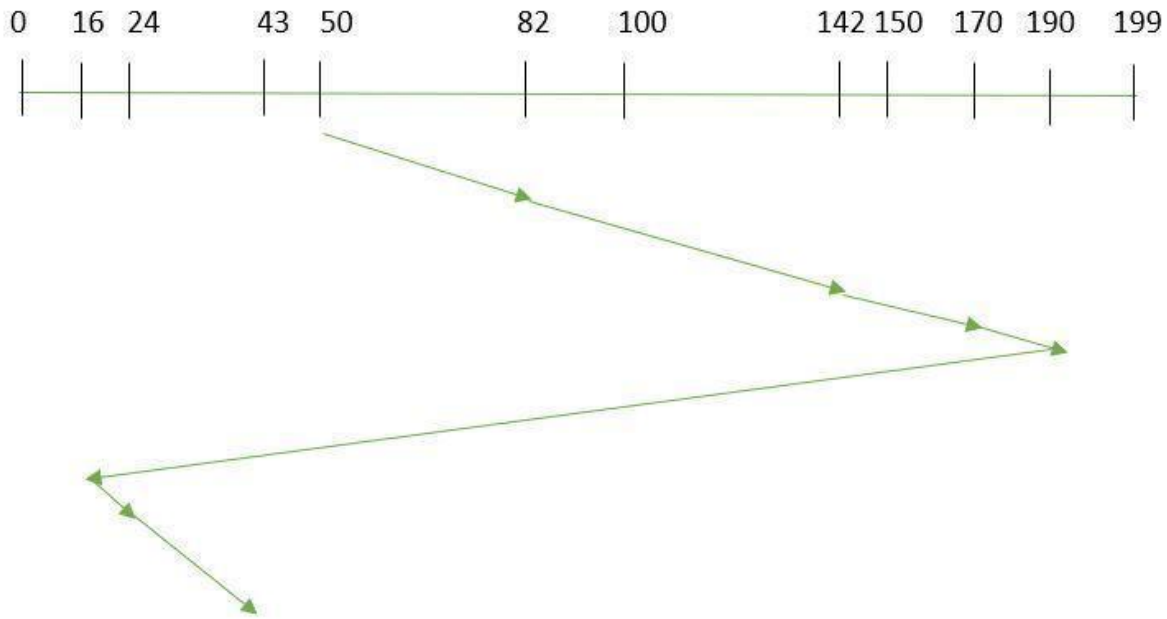
1. =(190-50)+(190-16) =314

2. **CLOOK:** As LOOK is similar to SCAN algorithm, in similar way, CLOOK is similar to CSCAN disk scheduling algorithm. In CLOOK, the disk arm in spite of going to the end goes only to the last request to be serviced in front of the head and then from there goes to the other end's last request. Thus, it also prevents the extra delay which occurred due to unnecessary traversal to the end of the disk.

Example:

1. 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”**



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

1. =(190-50)+(190-16)+(43-16) =341
2. **RSS**– It stands for random scheduling and just like its name it is nature. It is used in situations where scheduling involves random attributes such as random processing time, random due dates, random weights, and stochastic machine breakdowns this algorithm sits perfectly. Which is why it is usually used for analysis and simulation.
3. **LIFO**– In LIFO (Last In, First Out) algorithm, the newest jobs are serviced before the existing ones i.e. in order of requests that get serviced the job that is newest

or last entered is serviced first, and then the rest in the same order.

Advantages

- Maximizes locality and resource utilization
- Can seem a little unfair to other requests and if new requests keep coming in, it cause starvation to the old and existing ones.

1. **N-STEP SCAN** – It is also known as the N-STEP LOOK algorithm. In this, a buffer is created for N requests. All requests belonging to a buffer will be serviced in one go. Also once the buffer is full no new requests are kept in this buffer and are sent to another one. Now, when these N requests are serviced, the time comes for another top N request and this way all get requests to get a guaranteed service

Advantages

- It eliminates the starvation of requests completely

1. **FSCAN**– This algorithm uses two sub-queues. During the scan all requests in the first queue are serviced and the new incoming requests are added to the second queue. All new requests are kept on halt until the existing requests in the first queue are serviced.

Advantages

- FSCAN along with N-Step-SCAN prevents “arm stickiness” (phenomena in I/O scheduling where the scheduling algorithm continues to service requests at or near the current sector and thus prevents any seeking)

Each algorithm is unique in its own way. Overall Performance depends on the number and type of requests.

Note: *Average Rotational latency is generally taken as 1/2(Rotational latency).*

Exercise

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

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

Answer: (C)

Explanation: In Shortest-Seek-First algorithm, request closest to the current position of the disk arm and head is handled first.

In this question, the arm is currently at cylinder number 100. Now the requests come in the queue order for cylinder numbers 30, 85, 90, 100, 105, 110, 135 and 145.

The disk will service that request first whose cylinder number is closest to its arm. Hence 1st serviced request is for cylinder no 100 (as the arm is itself pointing to it), then 105, then 110, and then the arm comes to

service request for cylinder 90. Hence before servicing request for cylinder 90, the disk would had serviced 3 requests.
Hence option C.

2) Consider an operating system capable of loading and executing a single sequential user process at a time. The disk head scheduling algorithm used is First Come First Served (FCFS). If FCFS is replaced by Shortest Seek Time First (SSTF), claimed by the vendor to give 50% better benchmark results, what is the expected improvement in the I/O performance of user programs?
(A) 50% (B) 40% (C) 25% (D) 0% See this for a solution.
Answer: (D)

Explanation: Since Operating System can execute a single sequential user process at a time, the disk is accessed in FCFS manner always. The OS never has a choice to pick an IO from multiple IOs as there is always one IO at a time.

3) Suppose the following disk request sequence (track numbers) for a disk with 100 tracks is given: 45, 20, 90, 10, 50, 60, 80, 25, 70. Assume that the initial

position of the R/W head is on track 50. The additional distance that will be traversed by the R/W head when the Shortest Seek Time First (SSTF) algorithm is used compared to the SCAN (Elevator) algorithm (assuming that SCAN algorithm moves towards 100 when it starts execution) is _____ tracks.

(A) 8 (B) 9 (C) 10 (D) 11.

Answer: (C)

Explanation: In Shortest seek first (SSTF), closest request to the current position of the head, and then services that request next.
In SCAN (or Elevator) algorithm, requests are serviced only in the current direction of arm movement until the arm reaches the edge of the disk. When this happens, the direction of the arm reverses, and the requests that were remaining in the opposite direction are serviced, and so on.
Given a disk with 100 tracks

And Sequence 45, 20, 90, 10, 50, 60, 80, 25, 70.

Initial position of the R/W head is on track 50.

In SSTF, requests are served as following

Next Served	Distance Traveled
50	0
45	5
60	15
70	10
80	10
90	10
25	65
20	5
10	10

Total Dist = 130

If Simple SCAN is used, requests are served as following

Next Served	Distance Traveled
50	0
60	10
70	10
80	10
90	10
45	65 [disk arm goes to 99, then to 45]
25	20
20	5
10	10

Total Dist = 140

Less Distance traveled in SSTF = 130 - 140 = 10
Therefore, it is **not additional** but it is **less distance** traversed by SSTF than SCAN.

4) Consider a typical disk that rotates at 15000 rotations per minute (RPM) and has a transfer rate of 50 × 10^6 bytes/sec. If the average seek time of the disk is twice the average rotational delay and the controller's transfer time is 10 times the disk transfer time, the average time (in milliseconds) to read or write a 512-byte sector of the disk is _____ .

Answer: (A)

Explanation:

Disk latency = Seek Time + Rotation Time + Transfer Time + Controller Overhead

Seek Time? Depends no. tracks the arm moves and seek speed of disk

Rotation Time? depends on rotational speed and how far the sector is from the head

Transfer Time? depends on data rate (bandwidth) of disk (bit density) and the size of request

Disk latency = Seek Time + Rotation Time +

Transfer Time + Controller Overhead

Average Rotational Time = (0.5)/(15000 / 60) = 2 milliseconds

[On average half rotation is made]

It is given that the average seek time is twice the average rotational delay

So Avg. Seek Time = 2 * 2 = 4 milliseconds.

Transfer Time = 512 / (50 × 10^6 bytes/sec)

= 10.24 microseconds

Given that controller time is 10 times the average transfer time

Controller Overhead = 10 * 10.24 microseconds

= 0.1 milliseconds

Disk latency = Seek Time + Rotation Time +
Transfer Time + Controller Overhead
= 4 + 2 + $10.24 * 10^{-3}$ + 0.1 milliseconds
= 6.1 milliseconds