CHAPTER 07 File Systems

* File systems are a crucial part of any OS providing a structured way to store, organize and manage data on storage devices such as harddrives, SSDs and USB drives.

Types of File System

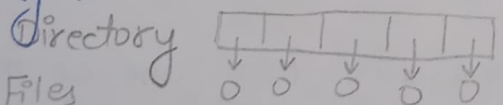
1. FAT (File Allocation Table): An older file system used by older versi -ons of windows and other operating systems.
2. NTFS (New Technology File System): A modern file system used by windows. It supports features such as file and folder permissions, compression and encryption.
3. ext (Extended File System): A file system commonly used on Linux and unix based operating systems.
4. HFS (Hierarchical File System): A file system used by macOS.
5. APFS (Apple File System): A new file system introduced by APPLE for their Macs and iOS devices.

* File Directories: The collection of files is a file directory. The directory is itself a file, accessible by various file manage -ment routines. The directory contains information about the files, including attributes, location and ownership.
* File Attributes: Name, type, address, Current length, Maximum length, Date last access, Date last updated, Owner id, Protecti -on Information.
* File Directory Structure

1. Single-Level Directory: A single directory is mainted for all the users.

Naming Problem: Users cannot have the same name for two files.

Grouping Problem: Users cannot group files according to their needs.

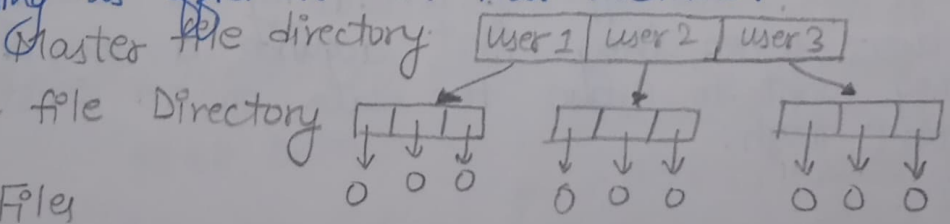


2. Two-Level Directory: Separate directories for each user is main -tained.

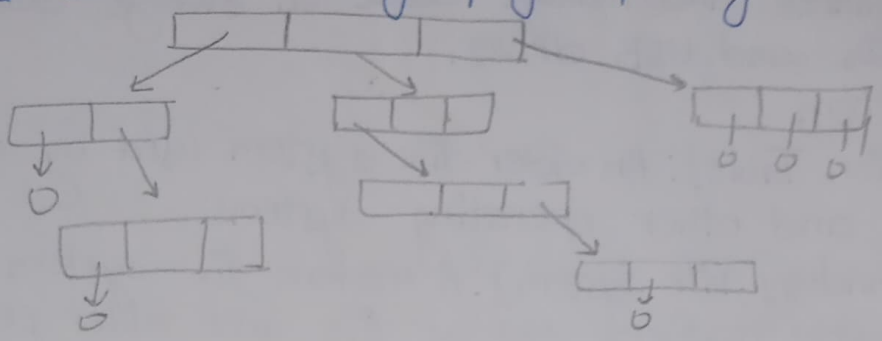
Path Name: Due to two levels there is a path name for every file to locate that file.

Now, we can have the same file name for different users.

Searching is efficient in this method.



3. Tree Structured Directory
The directory is maintained in the form of a tree. Searching is effic-ent and also there is grouping capability.



* Disk Formatting
It is a process to configure the data-storage devices such as hard-drive, floppy disk and flash drive when we are going to use them for the very first time or initial usage.
It is required when new Operating System is going to be used by the user. It also done when there is space issue and we require additional space for the storage of more data in the drives.
It has the cabability to erase the bad applications & various sophisticated Viruses.

Disk Formatting

Low-Level Formatting | Partitioning | High-level formatting

1. Low-Level Formatting (Physical Formatting)
It is the process of creating tracks, sectors & cylinders on the blank hard-disk. Low-Level formatting is performed by the hard-disk manufactures themselves.

2. Partitioning
Partitioning means divisions.
It is the process of dividing the hard-disk into one or more regions. The regions are called as partitions. It is performed by the users and it will affect the disk performance.

3. High-level Formatting (Logical Formatting)
It is the process of creating a file system and directory structu-re on a disk, allowing it to be recognized and used by an operating System. This type of formatting is usually performed by users when they first acquire a new disk, or when they want to erase the data on an existing disk and start fresh.
→ creating drives C:\>, E:, F:.....
→ creating disk blocks
→ installing file system
→ After this OS is stored (installed) in one of the drives.

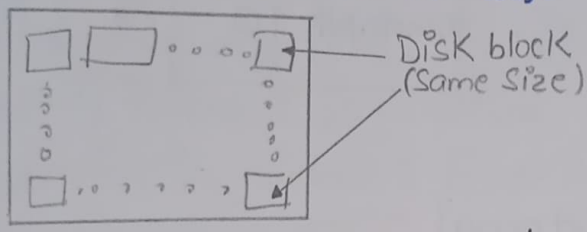
Logical Partitions

Primary Partition (one or 2)
OS + user files

Extended Partition (as many as you want)
user files

* Disk Blocks

Block is the smallest unit of data storage. It is used to read a file or write data to a file. Block is also a sequence of bits and bytes. Block is made up of sectors. A sector is a physical spot on a formatted disk that hold information. A block is made up of either one sector or even no. of sectors (2, 4, 6.....). A block is also called a physical record.



Assume: Disk = 64 disk blocks
$= 2^6$

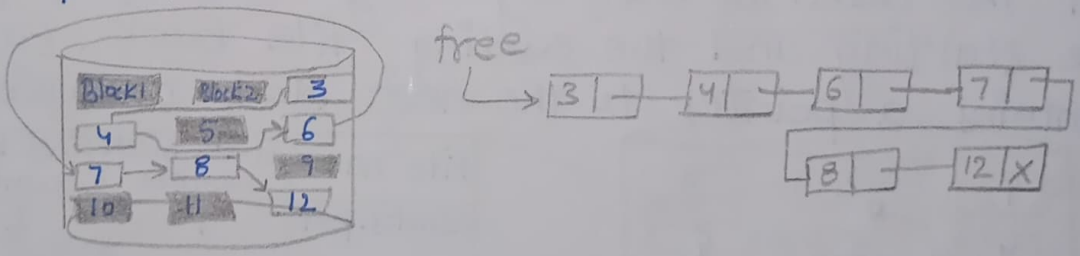
Disk block address = 6 bits

* Free Space Management

It is a critical aspect of OS as it involves managing the available storage space on the hard disk or other secondary storage devices. The OS uses various techniques to manage free space and optimize the use of storage devices.

1. Free List

The free disk blocks are linked together i.e., a free block contain a pointer to the next free block.



2. Bitmap Method

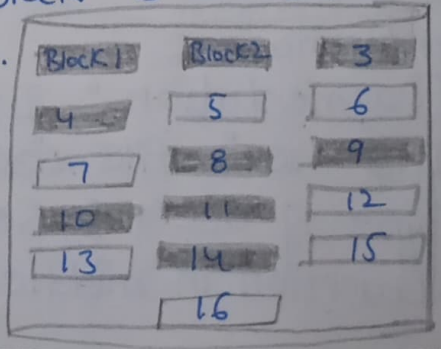
A Bitmap or Bit Vector is series or collection of bits where each bit corresponds to a disk block. The bit can take two values: 0 and 1. 0 indicates that the block is allocated and 1 indicates a free block.

Ex: Disk has 16 blocks = 2^4 blocks

0: free
1: occupied

16 bit bitmap

0000111100001101



1. No searching in free list, but in bitmap we search for first zero for free block.
2. Free list is faster in allocating a free block.
3. Free list size is variable, where as bitmap size is constant.

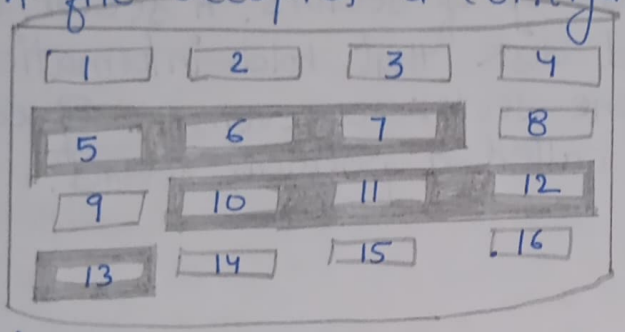
*File Allocation Methods

The allocation methods define how the files are stored in the disk blocks.

There are three main file allocation methods are:

1. Contiguous Allocation

Each file occupies a contiguous set of blocks on the disk.



File Name	File Start block no.	No. of blocks to store file
CoA.pptx	5	3
OSnote.pdf	10	4

Performance:

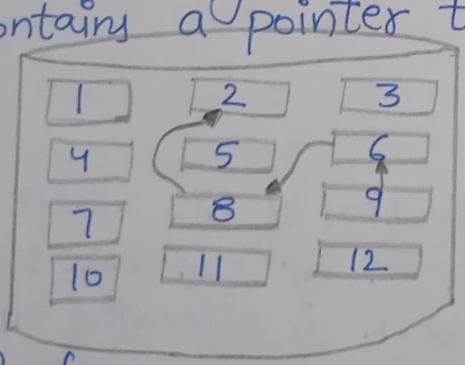
1. Fragmentation: Internal, External
2. Increase in File Size: Inflexible
3. Type of access: Sequential, Random/direct
4. Insertion in middle: Inflexible

disadvantage

advantage

2. Linked List Allocation

Each file is a linked list of disk blocks which need not be contiguous. The disk blocks can be scattered anywhere on the disk. The directory entry contains a pointer to the starting and the ending file block. Each block contains a pointer to the next block occupied by the file.



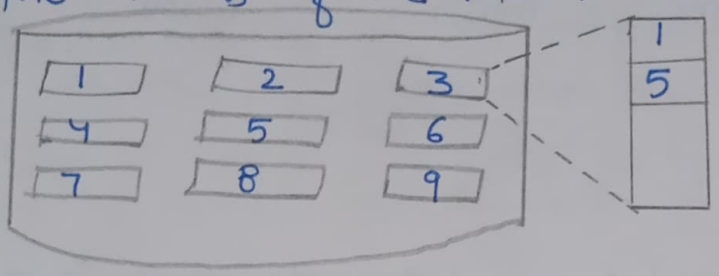
File name	Start block number	End block number
OSnotes.pdf	9	2

Performance:

1. Fragmentation: Internal (no any external fragmentation) → Advantage
2. Increase in file size: Flexible → Advantage
3. Type of access: Sequential → Disadvantage
4. Insertion in middle: Inflexible (because to reach to middle of the file then it will take time)

3. Indexed Allocation

A special block known as the index block contains the pointers to all the blocks occupied by a file. Each file has its own index block. The i^{th} entry in the index block contains the disk address of the i^{th} file block. The directory entry contains the address of the index block as shown in the image:



File Name	Index block
DSnotes.pdf	3

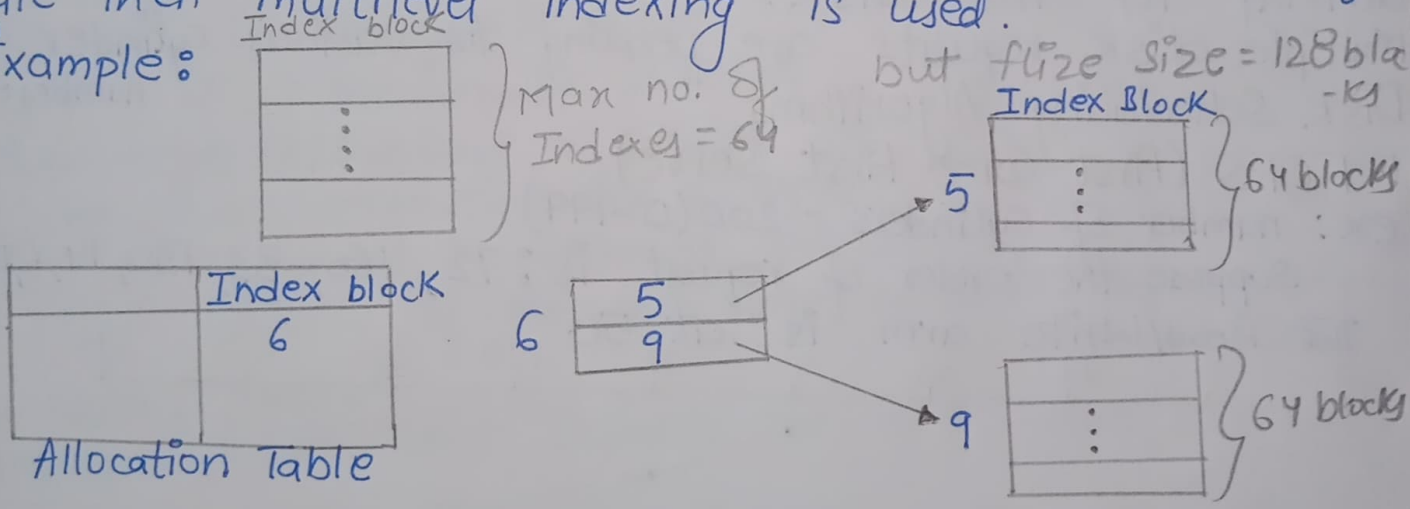
Performance

1. Fragmentation: Internal
2. Increase in File Size: Flexible
3. Type of access: Sequential, Random/direct
4. Insertion in middle: Flexible only disk block address are moved in index block.

Disadvantage: Space is occupied for storing index.

If one block size is not sufficient to store indexes of a file then multilevel indexing is used.

Example:



* Master Boot Record

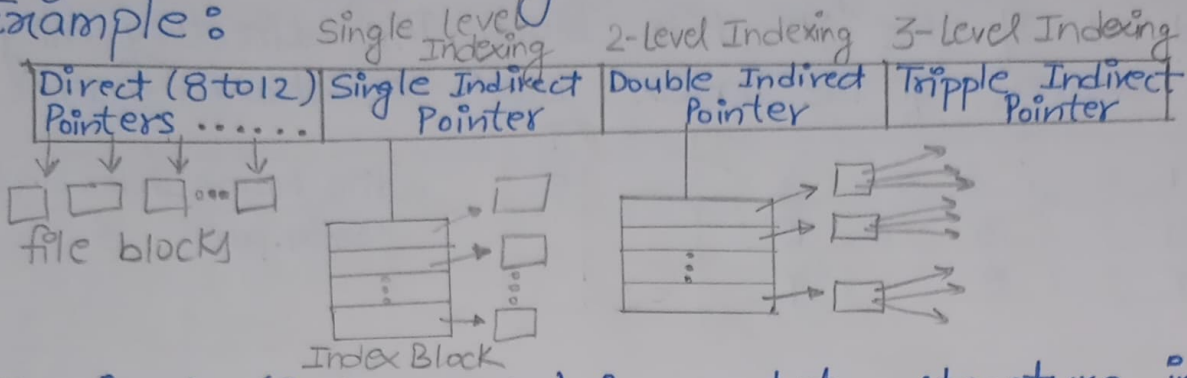
A master boot record (MBR) is a special type of boot sector at the very beginning of partitioned computer mass storage devices.

Contains the information regarding how and where the OS is located in the hard disk so that it can be booted in the RAM.

* Unix I-node Structure

The inode (index node) is a data structure in a unix-style file system that describes a file-system object such as a file or a directory.

Example:



The inode (index node) is a data structure in a Unix-style file system that describes a file System object such as a file or a directory.

Each inode stores the attributes and disk block locations of the object's data.

The number of Inode limits the total number of files/directories that can be stored in the file system.

* Disk Scheduling

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

Multiple disk requests are pending for various cylinder numbers.

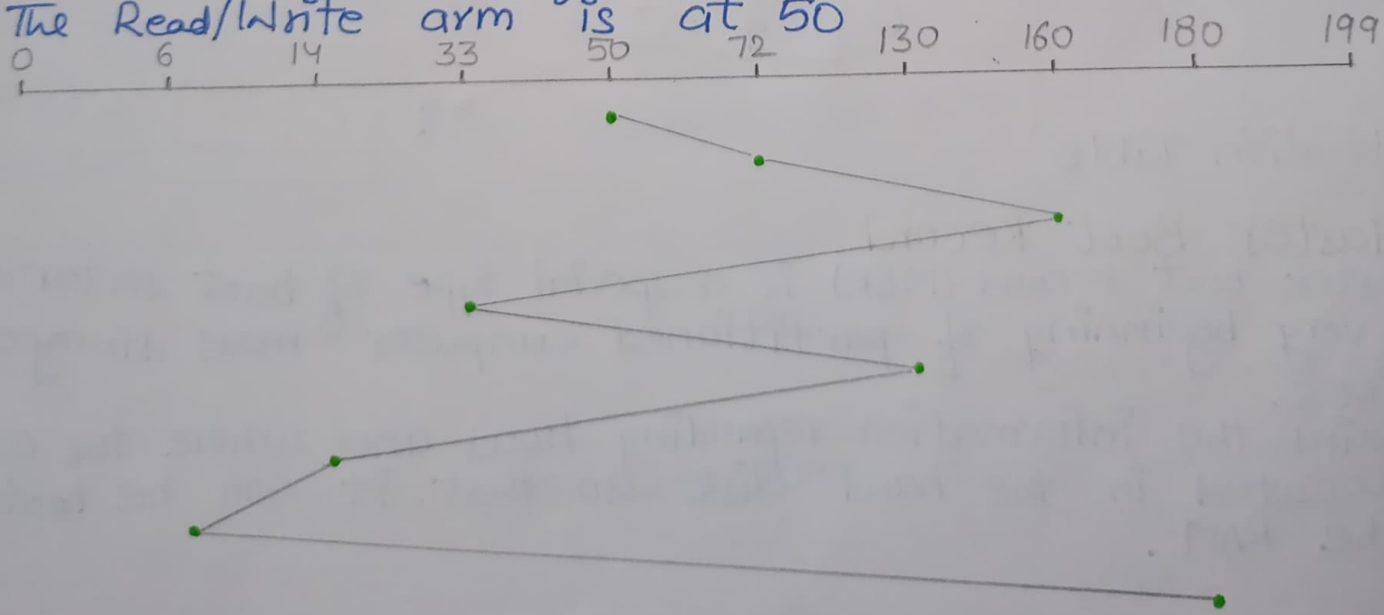
Disk Scheduling Algorithms

1. FCFS (First Come First Serve)

ex: number of cylinders = 200 (0-199)

Suppose the order of request is: 72, 160, 33, 130, 14, 6, 180

The Read/Write arm is at 50



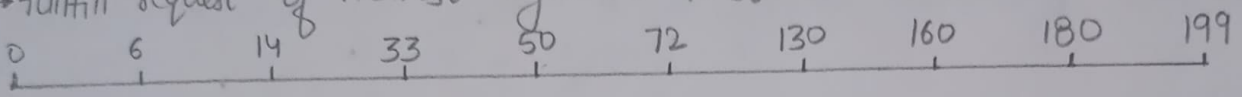
Total number of head (Cylinder) movements
$= (72-50) + (160-72) + (160-33) + (130-33) + (130-14) + (14-6) + (180-6)$

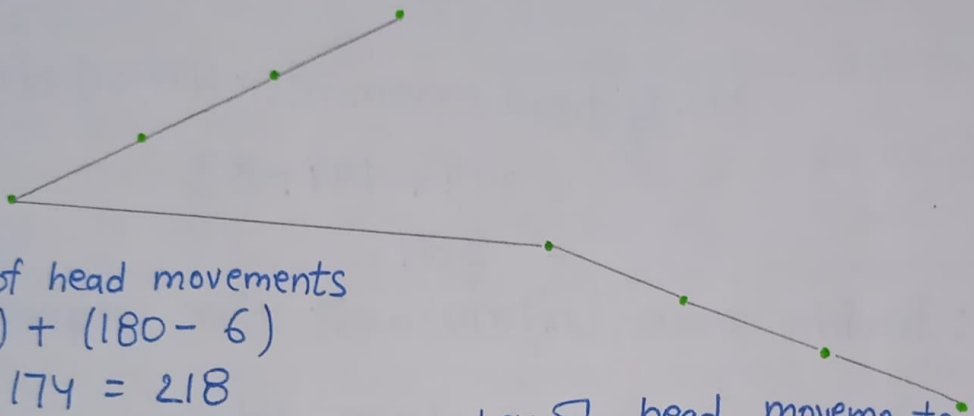
$= 632$

Advantages: (A) Every request gets a fair chance.
(B) No indefinite postponenement. (no starvation for disk requests)

Disadvantages: (A) Does not try to optimize seek time.
(B) May not provide the best possible service.

* SSTF (Shortest Seek Time First)
↳ fulfill request of nearest Cylinder first



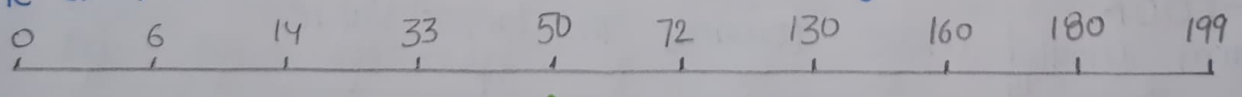


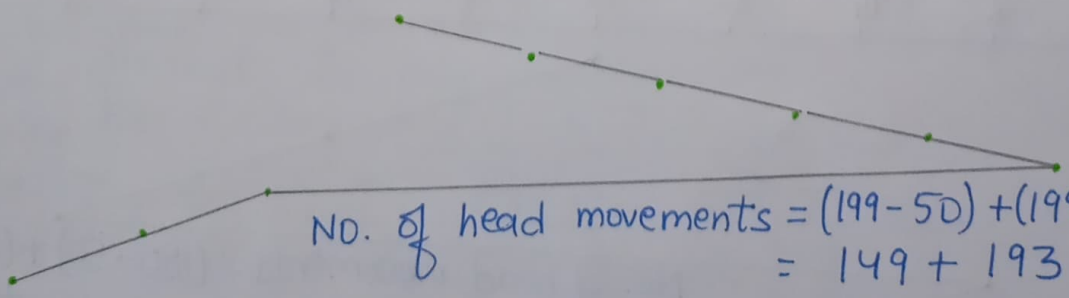
Number of head movements
$= (50-6) + (180-6)$
$= 44 + 174 = 218$

Advantages: (A) Minimum number of head movements.
(B) Average Response time decreases.
(C) Throughput increases.

Disadvantages: (A) Overhead to calculate seek time in advance.
(B) Can cause starvation for a request if it has higher seek time as compared to incoming requests.
(C) High variance of response time as SSTF favors only some requests.

* Scan (Elevator)
The arm should move "towards the larger value."





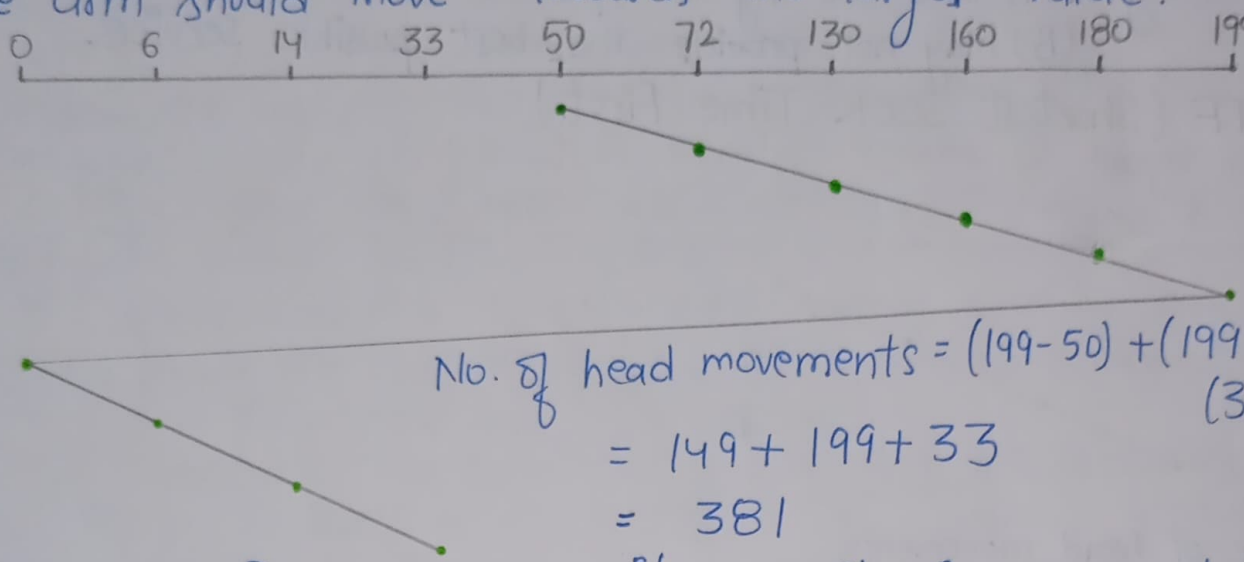
No. of head movements $= (199-50) + (199-6)$
$= 149 + 193$
$= 342.$

Advantages: (A) High throughput.
(B) Low variance of response time.
(C) Average response time.

Disadvantages: (A) Long waiting time for requests for locations just visited by disk arm.

* C-Scan

The arm should move "towards the larger value."

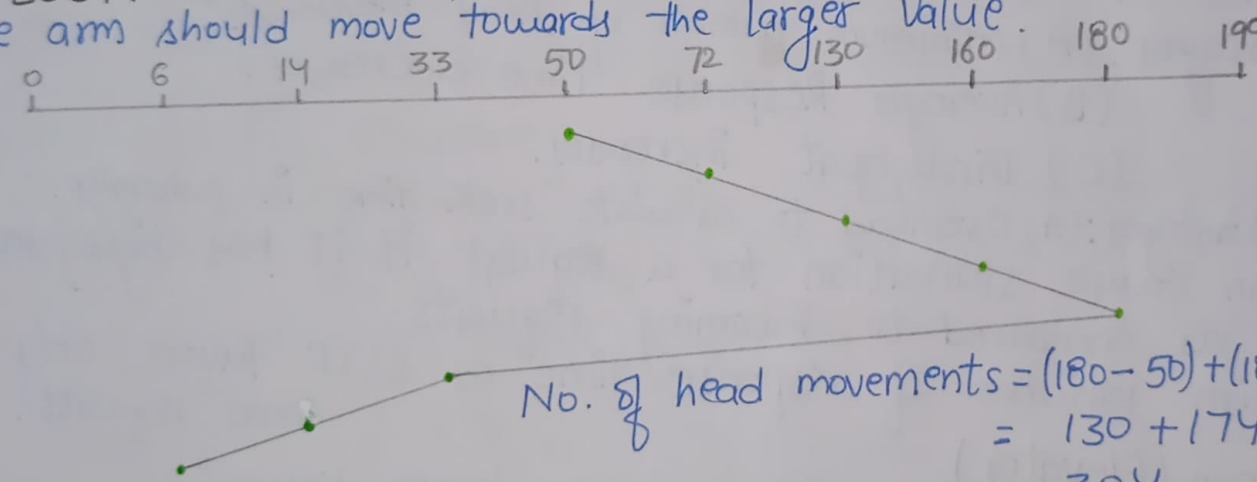


No. of head movements = (199-50) + (199-0) + (33-0)
= 149 + 199 + 33
= 381

Advantage: Provides more uniform wait time compared to SCAN.

* LOOK

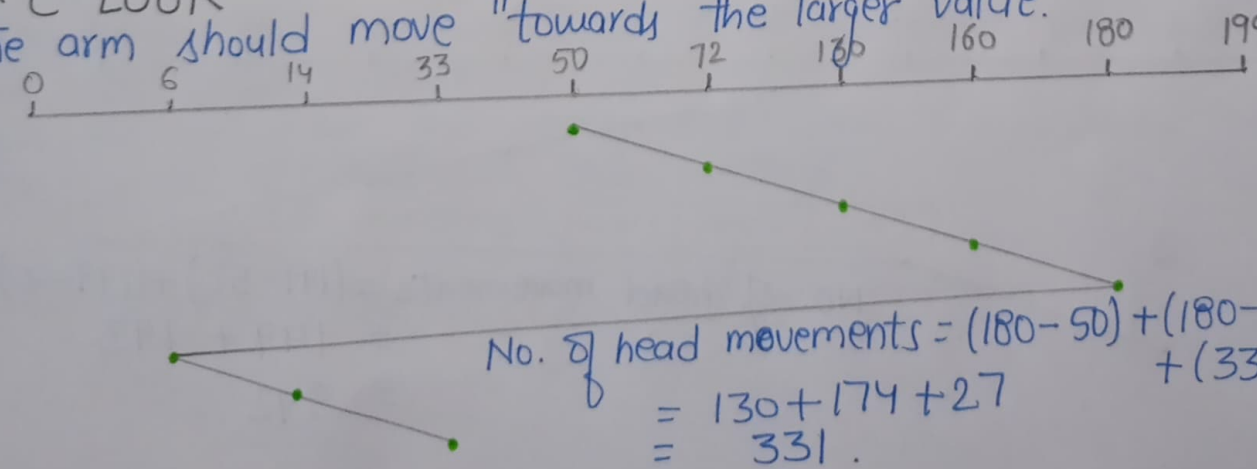
The arm should move "towards the larger value."



No. of head movements = (180-50) + (180-6)
= 130 + 174
= 304

* C-LOOK

The arm should move "towards the larger value."



No. of head movements = (180-50) + (180-6) + (33-6)
= 130 + 174 + 27
= 331.