Disk Management

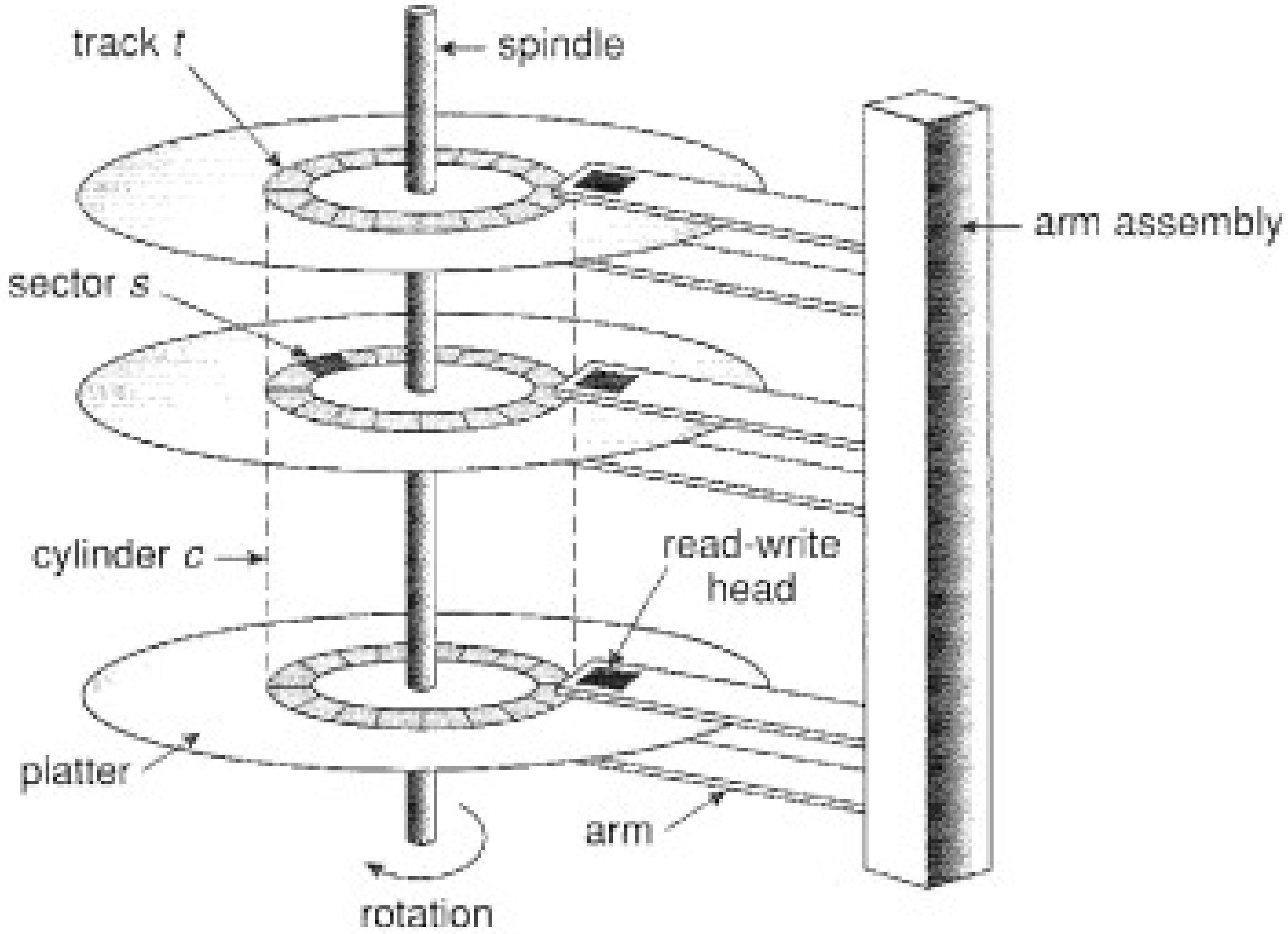
Disk Structure

- Disk drives are addressed as large 1-dimensional arrays of **logical blocks**, where the logical block is the smallest unit of transfer.
- The 1-dimensional array of logical blocks is mapped into the sectors of the disk sequentially.
  - Sector 0 is the first sector of the first track on the outermost cylinder.
  - Mapping proceeds in order through that track, then the rest of the tracks in that cylinder, and then through the rest of the cylinders from outermost to innermost.

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.
- Access time has two major components
    - **Seek time** is the time for the disk are to move the heads to the cylinder containing the desired sector.
    - **Rotational latency** is the additional time waiting for the disk to rotate the **desired sector** to the disk head.
- Minimize seek time
- 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.

Disk Scheduling



Disk Scheduling

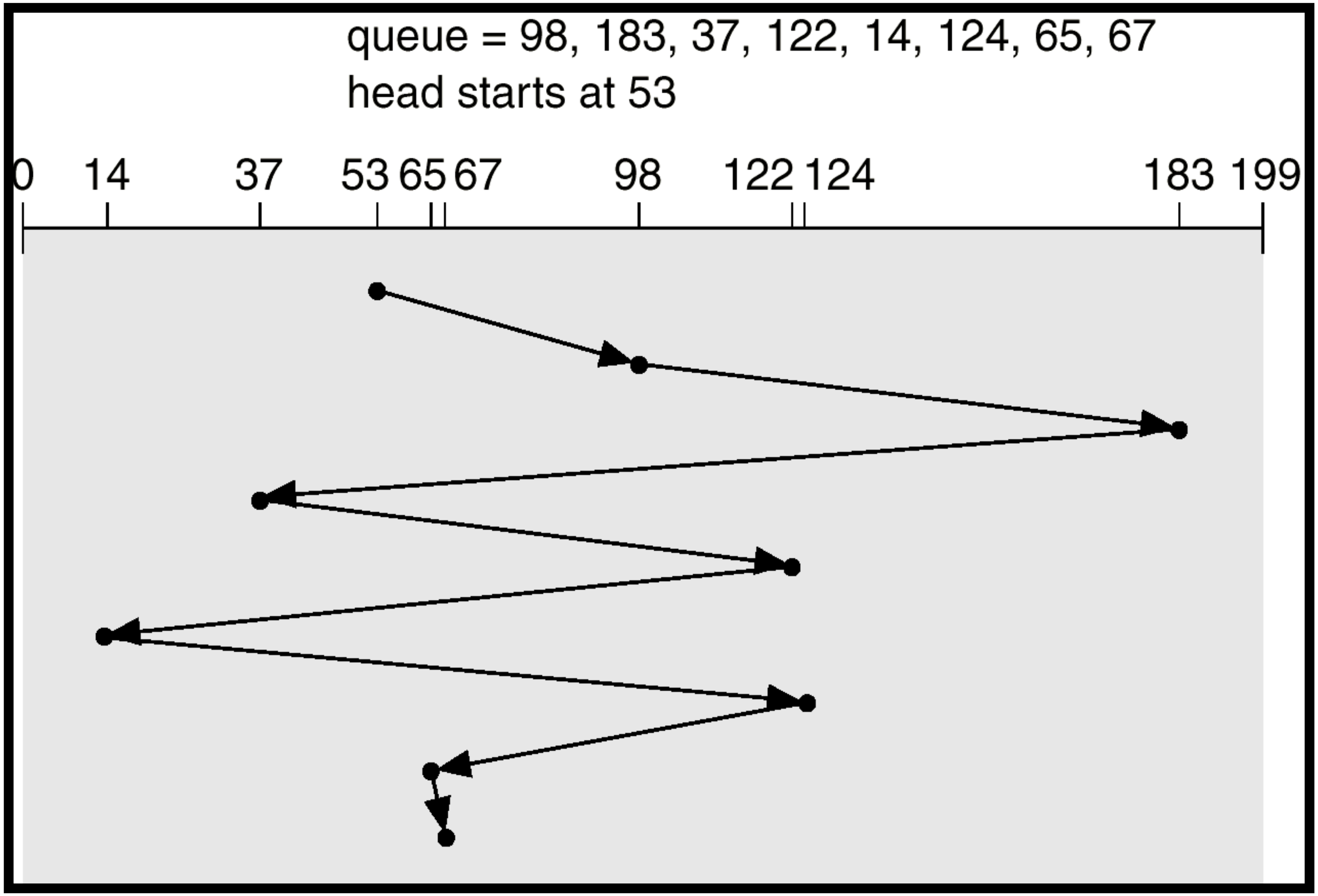
- Several algorithms exist to schedule the servicing of disk I/O requests.
- We illustrate them with a request queue (0-199).

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

Head pointer 53

FCFS

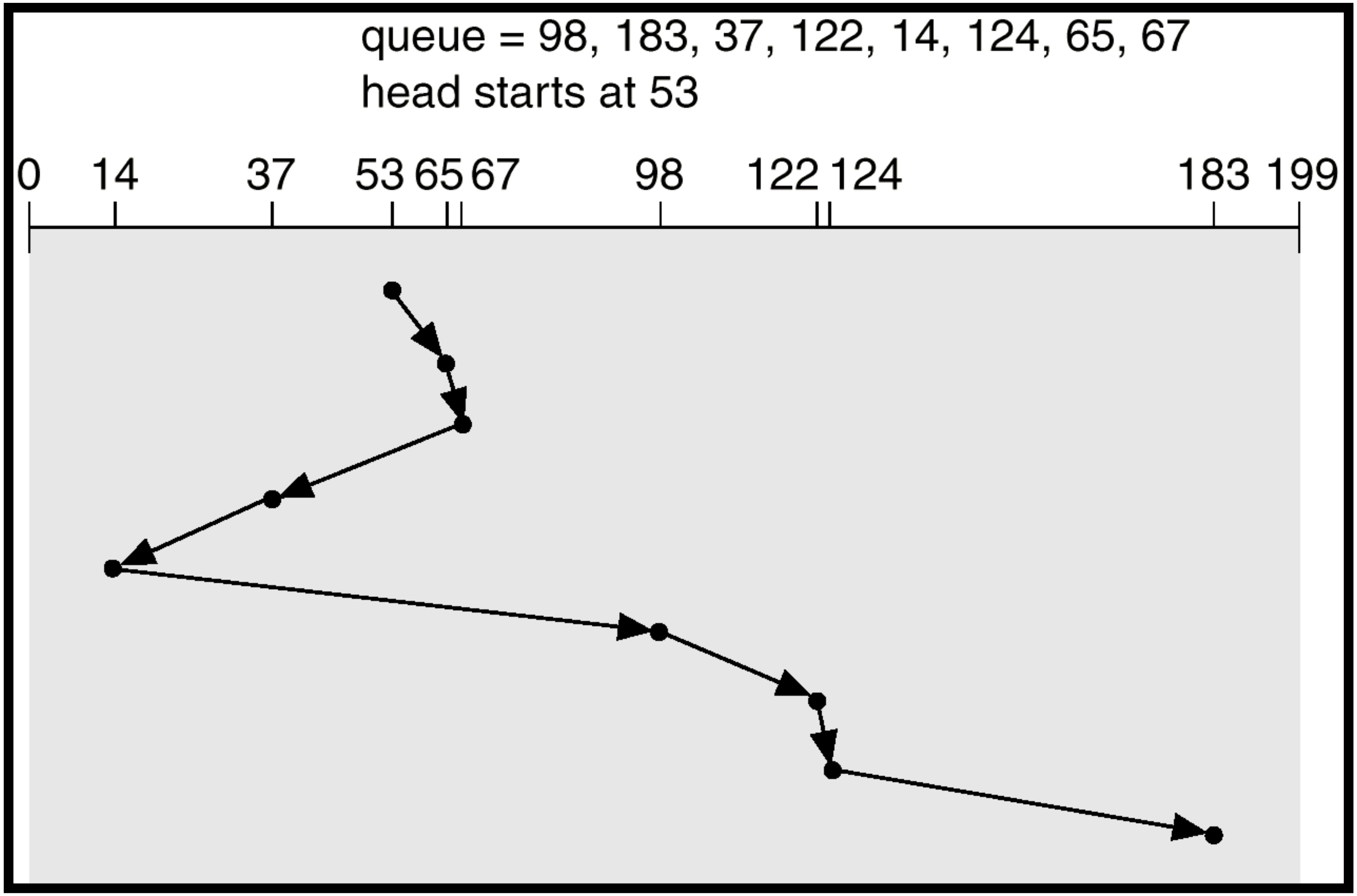
Illustration shows total head movement is 640.



SSTF

- 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.
- Illustration shows total head movement of 236 cylinders.

SSTF

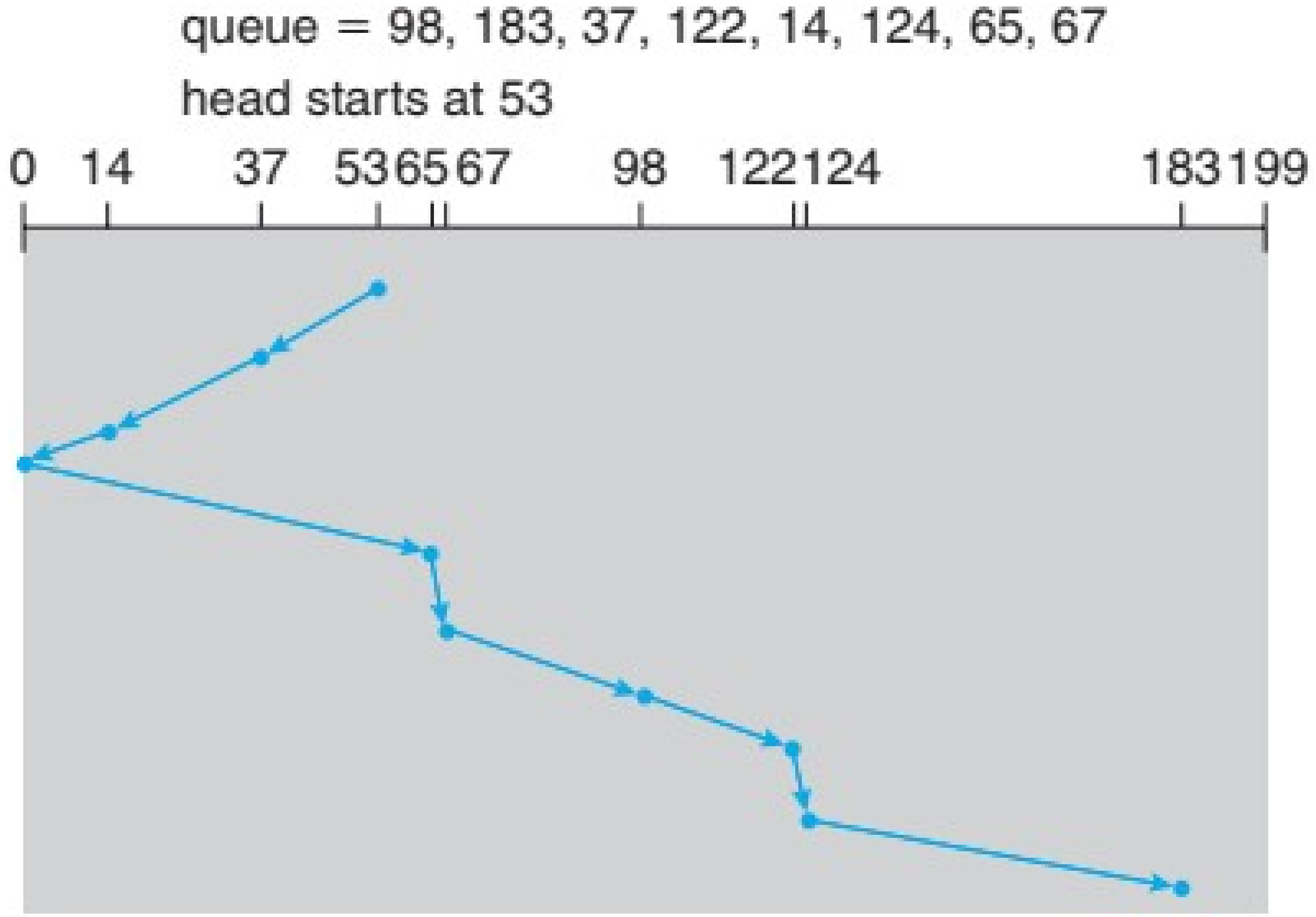


queue = 98, 183, 37, 122, 14, 124, 65, 67
head starts at 53
0
14
37
53
65
67
98
122
124
183
199

SCAN

- 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.
- Sometimes called the *elevator algorithm*.
- We need to know the direction of head movement.
- Illustration shows total head movement of 208 cylinders.

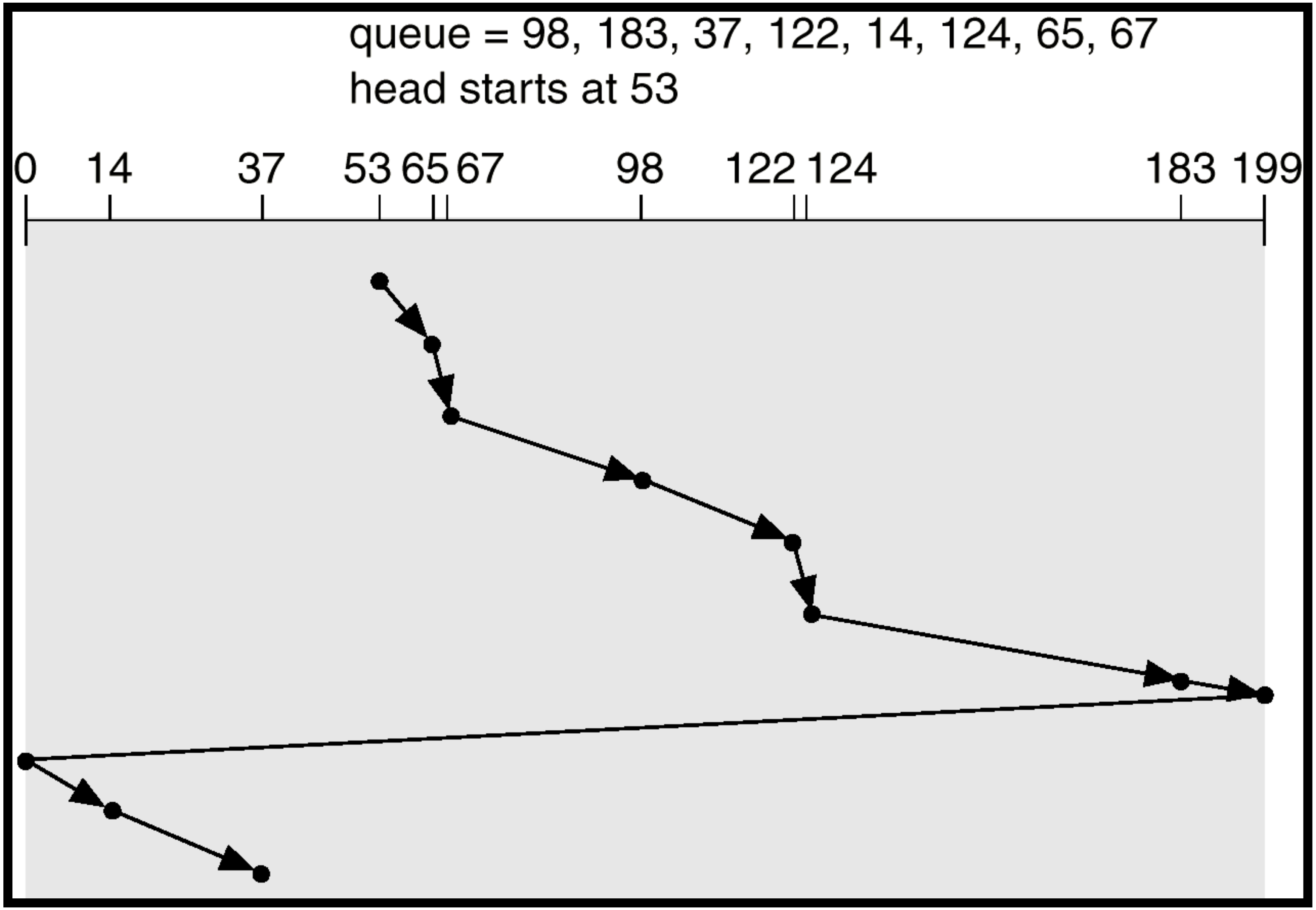
SCAN


queue = 98, 183, 37, 122, 14, 124, 65, 67
head starts at 53
0
14
37
53
65
67
98
122
124
183
199

C-SCAN

- 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.

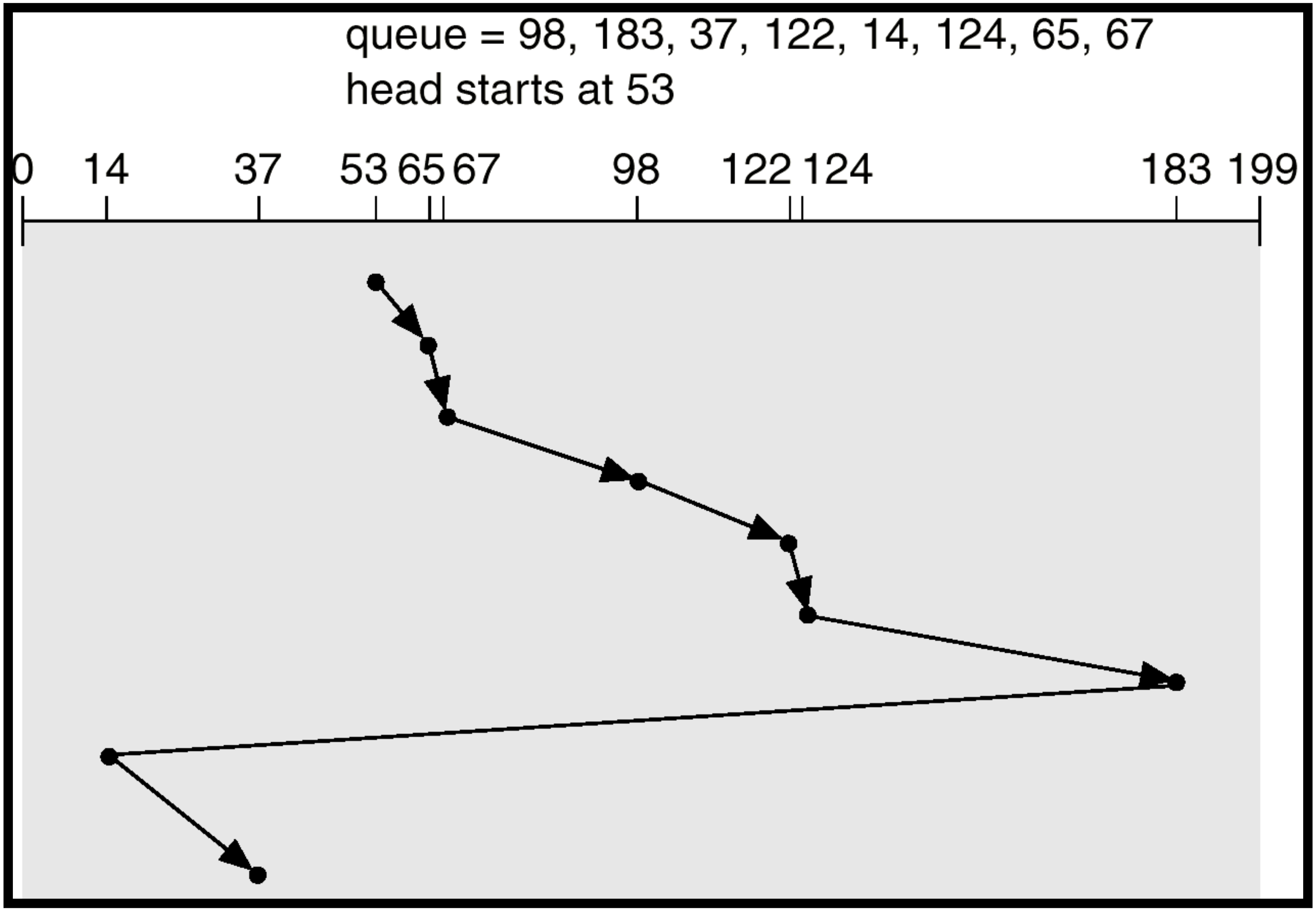
C-SCAN


queue = 98, 183, 37, 122, 14, 124, 65, 67
head starts at 53
0
14
37
53
65
67
98
122
124
183
199

C-LOOK

- 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.

C-LOOK


queue = 98, 183, 37, 122, 14, 124, 65, 67
head starts at 53
0
14
37
53
65
67
98
122
124
183
199

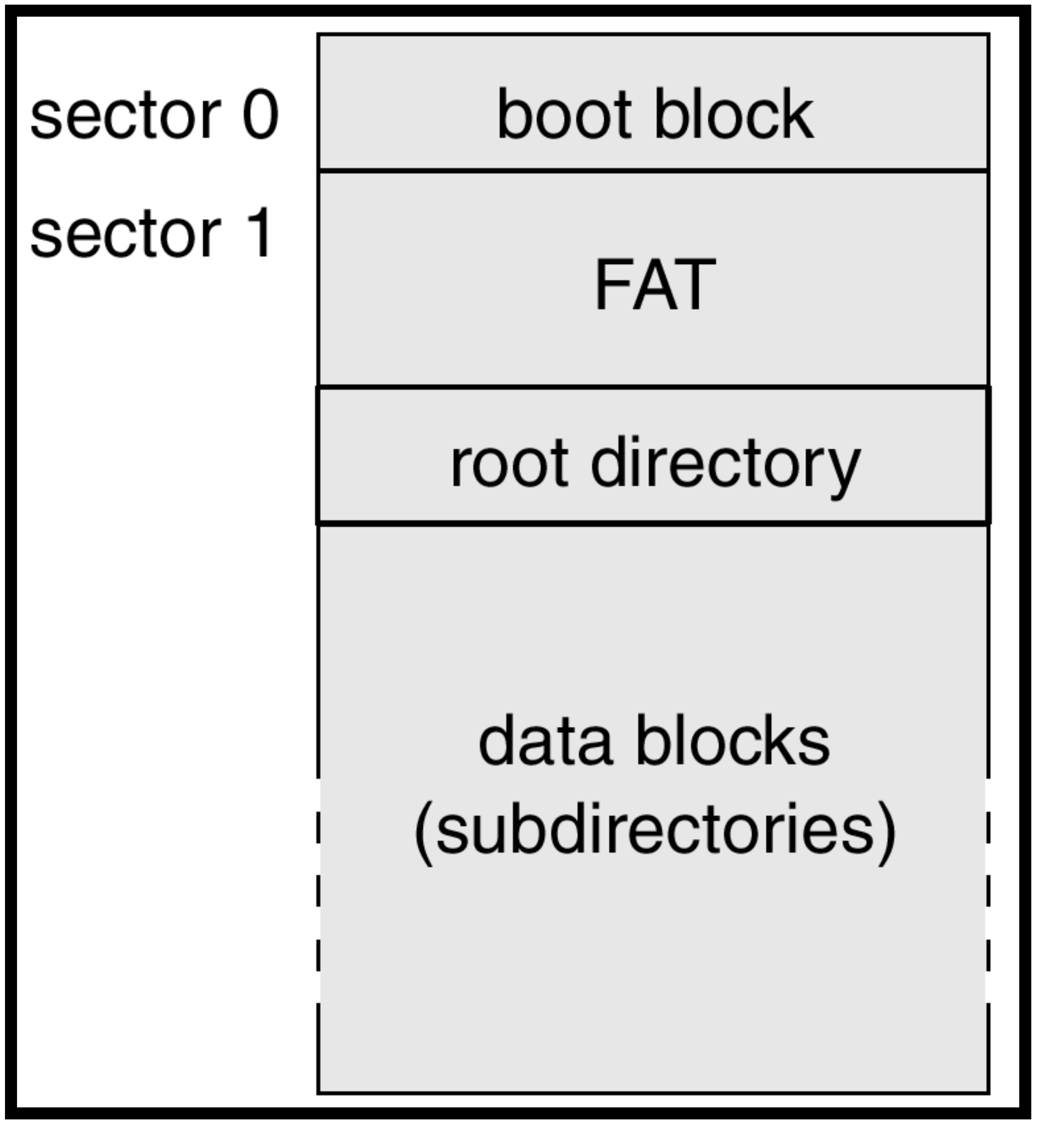
Selecting a Disk-Scheduling Algorithm

- SSTF is common and has a natural appeal
- SCAN and C-SCAN perform better for systems that place a heavy load on the disk.
- Performance depends on the number and types of requests.
- Requests for disk service can be influenced by the file-allocation method.
- 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.

Disk Management

- *Low-level formatting*, or *physical formatting* — Dividing a disk into sectors that the disk controller can read and write.
- To use a disk to hold files, the operating system still needs to record its own data structures on the disk.
    - *Partition* the disk into one or more groups of cylinders.
    - *Logical formatting* or "making a file system".
- Boot block initializes system.
    - The bootstrap is stored in ROM.
    - *Bootstrap loader* program.
- Methods such as *sector sparing* used to handle bad blocks.

MS-DOS Disk Layout



Swap-Space Management

- Swap-space — Virtual memory uses disk space as an extension of main memory.
- Swap-space can be carved out of the normal file system or more commonly, it can be in a separate disk partition.

Swap-Space Management

- Swap-space management is another low-level task of the operating system.
- Virtual memory uses disk space as an extension of main memory. Since disk access is much slower than memory access, using swap space significantly decreases system performance.
- The main goal for the design and implementation of swap space is to provide the best throughput for the virtual memory system.
- How swap space is **used**, where swap space is **located** on disk, and how swap space is **managed**.

Swap-Space Use

- Swap space is used in various ways by different operating systems, depending on the memory-management algorithms in use.
- For instance, systems that implement swapping may use swap space to hold an entire process image, including the code and data segments.
- Paging systems may simply store pages that have been pushed out of main memory.
- The amount of swap space needed on a system can therefore vary from a few megabytes of disk space to gigabytes, depending on the amount of physical memory, the amount of virtual memory it is backing, and the way in which the virtual memory is used.

Swap-Space Location

- A swap space can reside in one of two places:
- It can be carved out of the normal file system, or it can be in a separate disk partition.
- If the swap space is simply a large file within the file system, normal file-system routines can be used to create it, name it, and allocate its space.
- This approach, though easy to implement, is inefficient.
- External fragmentation can greatly increase swapping times by forcing multiple seeks during reading or writing of a process image.

Swap-Space Location

- We can improve performance by caching the block location information in physical memory and by using special tools to allocate physically contiguous blocks for the swap file, but the cost of traversing the file-system data structures remains.

Swap-Space Location

- Alternatively, swap space can be created in a separate raw partition.
- No file system or directory structure is placed in this space.
- Rather, a separate swap-space storage manager is used to allocate and deallocate the blocks from the raw partition.
- This manager uses algorithms optimized for speed rather than for storage efficiency, because swap space is accessed much more frequently than file systems (when it is used).

Swap-Space Location

- Internal fragmentation may increase, but this trade-off is acceptable because the life of data in the swap space generally is much shorter than that of files in the file system. Since swap space is reinitialized at boot time, any fragmentation is short-lived. The raw-partition approach creates a fixed amount of swap space during disk partitioning.

Swap-Space Management

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