Storage – Issues on where to store data.

Candidates: Disk, Tape, and rewritable CD/DVD

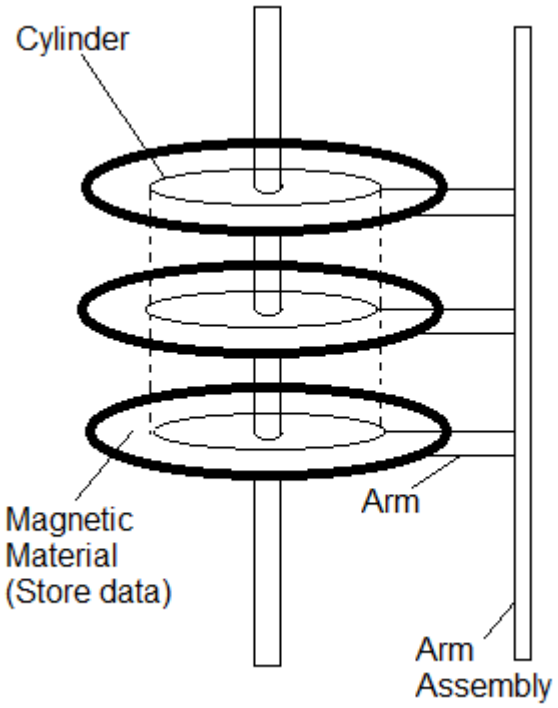
TAPE: traditional use of data storage.

Abandoned. Tapes are like those old Magnetic tapes. They store data sequentially. Thus random access to Tapes is too slow.

DISK:

Architecture

Made up of many platters (size varies in different disks) and an arm assembly.



Above is a typical disk architecture. Where arms are of same locations and double sided (on both sides of a platter). There is a header at the end of the arm which is used to fetch data from platters.

Cylinders are the tracks in the same location of different platters.

Tracks are a series of sectors in the same platter (they form a circle).

Sectors are the smallest units in the disk.

Access Time

Accessing random data in a disk costs =

Positioning Time + Transfer time

where, positioning time is the time cost to move the arm to the right cylinder and wait for the requested sector rotated to the header. I.e.,

Positioning time = Seek time + Rotational latency.

Transfer time is the time cost to transfer data from disk to memory (normally 20 – 40 Mbytes/s).

Reliability of Disk

MMTF (mean time to failure) = 100 K – 1 M Hours (several years)

Causes of a failure: Header crush.

There is little space between headers and the surfaces of platters. If the header touches the surface, the data in that segment would be ruined.

SSD (Solid State Disk)

NAND Flash Memory

+ much faster in random reads

+ no moving parts

SSD stores data in a random location to avoid a situation that access to certain spaces too often.

(To see more candidates of Storage, pls refer to the textbook)

DISK Scheduling

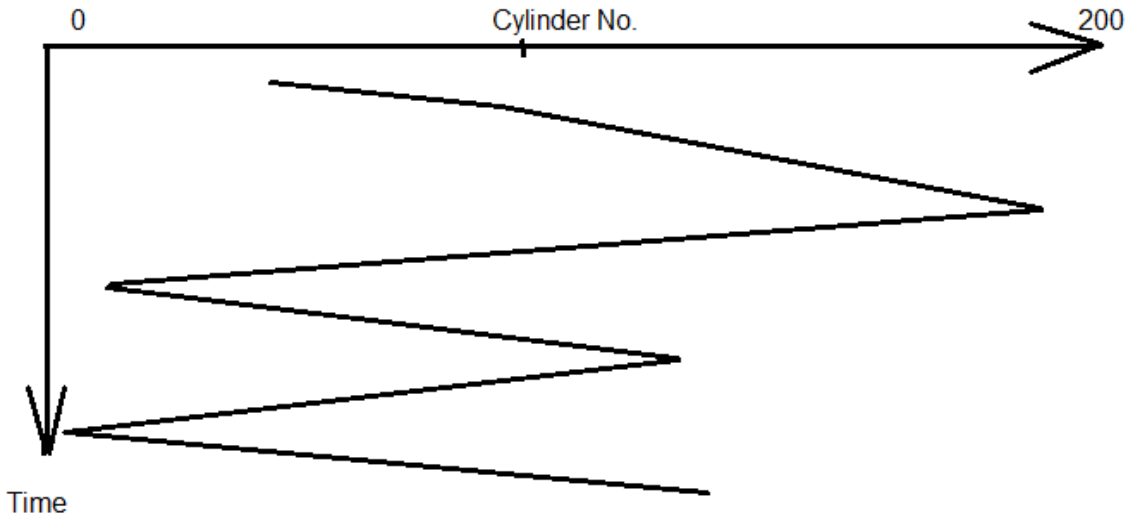
Consider a disk with 200 cylinders.

Access request: 98, 183, 37, 122, 14, 124, 65, 67 in a queue

Current arm position: 53

FIFO

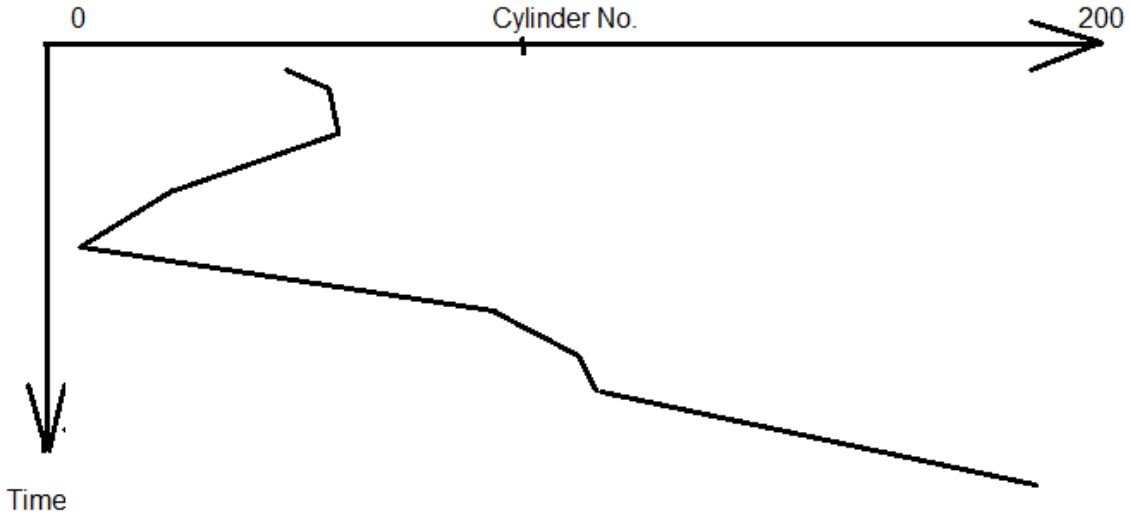
Move the arm to the position specified at the beginning of the wait queue.



This algorithm is just bad.

SSTF (Shortest Seek Time First)

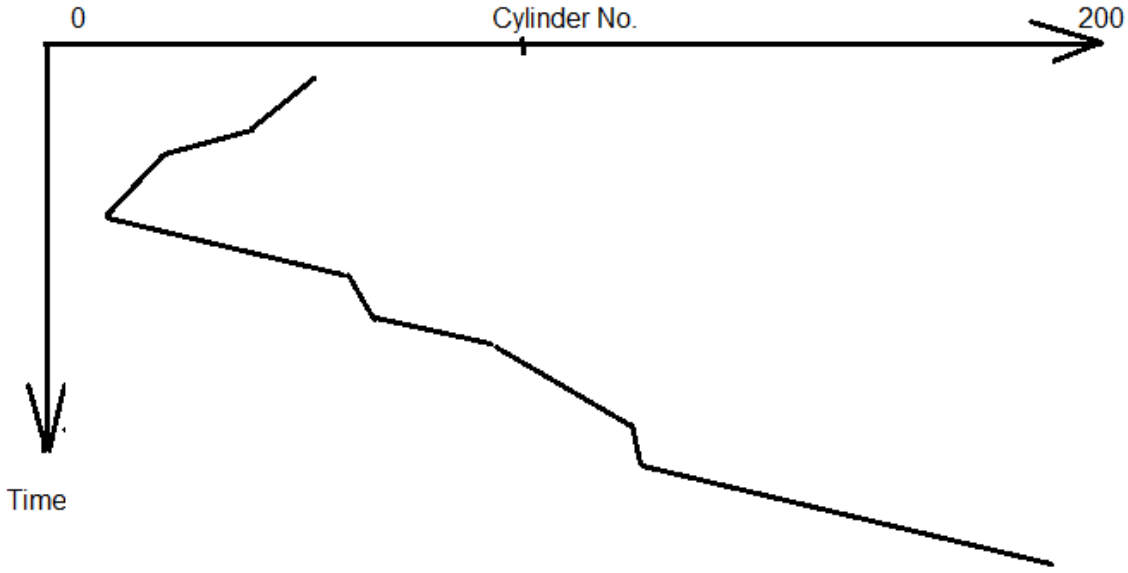
As name specified, always move the arm to the position in the queue which is closest.



May cause starvation – busy at some period of spaces, keep the rest periods waiting.

SCAN

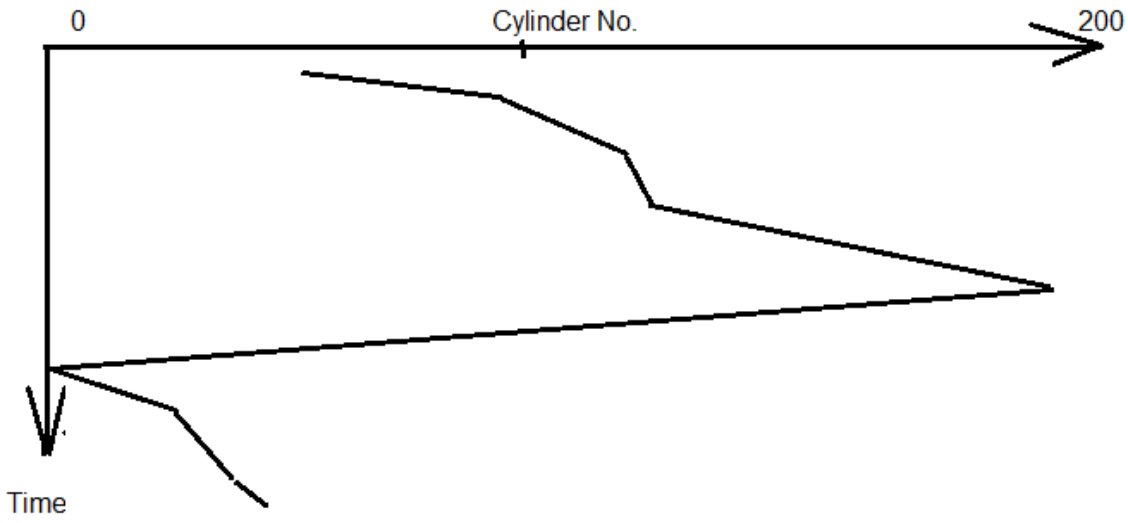
At each time point, move the arm to the position in the queue specified in certain direction. Turn back only when meets boundary.



This algorithm is not fair for jobs that come in right after the arm turn around.

C-SCAN

Like SCAN, but only go in one direction. If meet boundary, go to the very beginning.



LOOK and **C-LOOK**

Like Scan and C-Scan, but arms will not go to the boundary, instead, if no positions specified in the queue ahead, arms would turn around.

Formatting – Occurs in multiple levels.

Disk control handles formatting as follows.

1. Low-level formatting. Divide platters into sectors, add some error detection data.
2. Divide disk into partitions.
3. File system formatting

How to load OS into Memory

Bias runs some codes. They locate codes at the very top in disk, then load OS codes into Memory.

How to deal with bad blocks

Bad blocks are blocks in disks broken.

Our goal is to ensure never write data into bad blocks.
Operating system checks every sector during formatting. If it reads fail, remap to another sector automatically.
Problem: The remap function is done by OS, it make assumptions, which is time expensive.
Solution: Save some sectors as free sectors. If bad block detects, remap to one of the free sectors.

Swap Space Management

Differ from swapping in file systems: coz speed of swapping affects performances.
Two ways,

1. Swap space like swapping files
   Advantage, No need to store extra information.
   Disadvantage, Accessing file system is slow.

2. Take space as new partition
   Advantage, Faster
   Disadvantage, Waste space. If out of space, causes problems.