

Operating System: Hard Disk Drives

Sang Ho Choi (shchoi@kw.ac.kr)

School of Computer & Information Engineering
KwangWoon University

Hard Disk Drive

- Hard disk drives have been **the main form of persistent data storage** in computer systems for decades 주로 수십년동안 데이터 저장소로 지속성을 부여하는 주요한 형태였다.
 - The drive consists of a large number of sectors (512-byte blocks) 주 단위는 sector 단위
 - Address Space :
 - We can view the disk with n sectors as an array of sectors; 0 to $n-1$
- 총 1TB 데이터일때 512B로 나눈게 n 이 되고 이것은 주소공간으로 활용됨

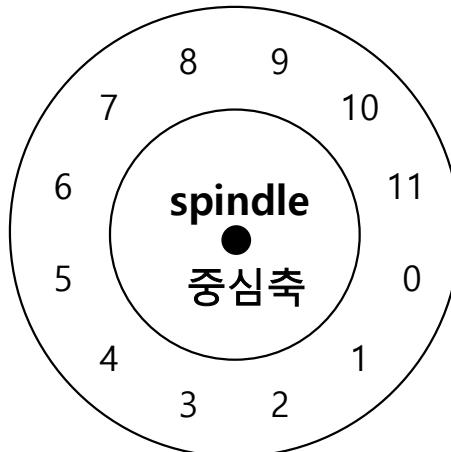
Interface

- The only guarantee is that a single 512-byte write is **atomic** 512 바이트는 원자적으로 수행됨 이 단위보다 적게 쓰여지진 않음 아예 안쓰이거나 다 쓰이거나 둘중 하나임
- Multi-sector operations are possible 일반적으로는 512바이트보다 더 크겠지?
 - Many file systems will read or write 4KB at a time 따라서 4KB로 읽거나 쓰는데 file system에서
 - Torn write:
 - If an untimely power loss occurs, only a portion of a larger write may complete 중간에 파워가 꺼지거나 하는 이슈가 발생하면 일정 부분만 쓰여질 수 있어 이걸 Torn write라고 함
- Accessing blocks in a contiguous chunk is the fastest access mode
 - A sequential read or write
 - Much faster than any more random access pattern

block에 접근할때 연속적인 chunk 단위로 접근하는게 랜덤으로 막 접근하는 것보다 훨씬 빠라

Basic Geometry

12개의 sector가 track을 이루고 있어



이 전체를 platter라고 하고 magnetic layer가 얇게 코팅된 aluminum임

A Disk with Just A Single Track (12 sectors)

- Platter (Aluminum coated with a thin magnetic layer)
 - A circular hard surface
 - Data is stored persistently by inducing magnetic changes to it
 - Each platter has 2 sides, each of which is called a surface
 - magnetic change를 유도하여 데이터가 저장되는 방식
 - 면이 두개 있을거고 앞뒷면을 각각 surface라고 한다.

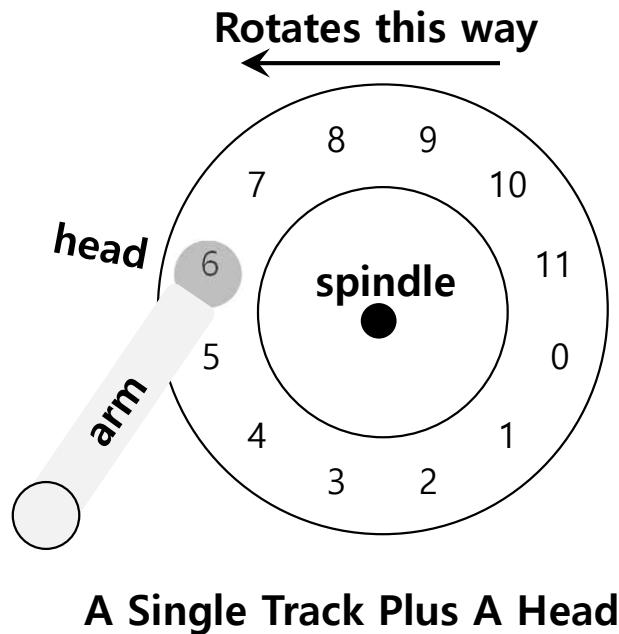
Basic Geometry (Cont.)

- **Spindle**
 - Spindle is connected to a motor that spins the platters around
 - The rate of rotations is measured in **RPM** (Rotations Per Minute)
 - Typical modern values : 7,200 RPM to 15,000 RPM
 - E.g., 10000 RPM : A single rotation takes about 6 ms

읽어보기

- **Track**
 - Concentric circles of sectors
 - Data is encoded on each surface in a track
 - A single surface contains thousands of tracks

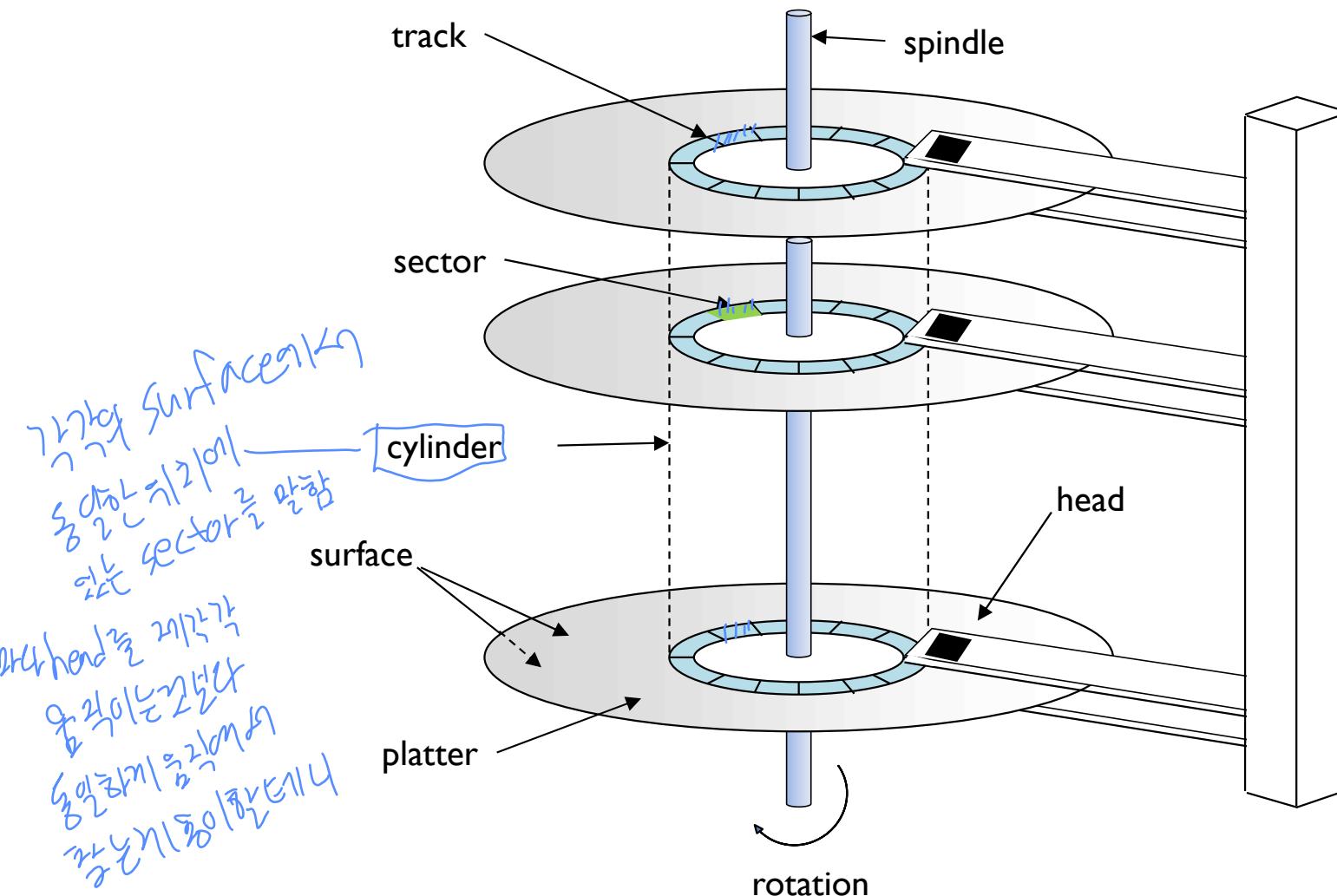
A Simple Disk Drive



- Disk head (One head per surface of the drive)
 - The process of *reading* and *writing* is accomplished by the **disk head**
 - Attached to a single disk arm, which moves across the surface

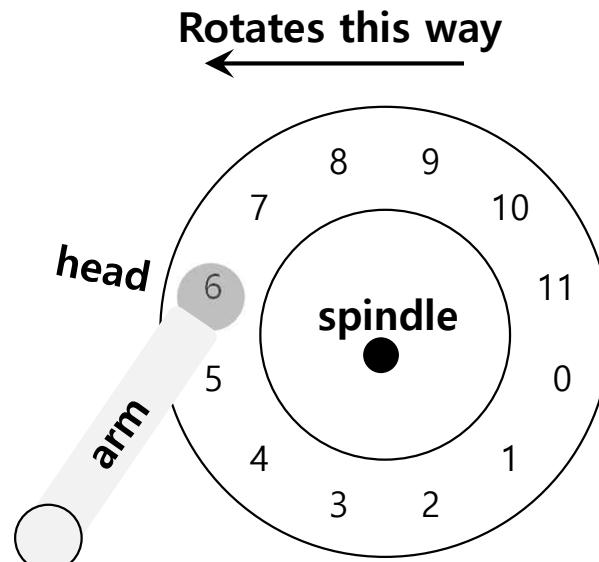
Hard Disk Internals

- Moving head disk mechanism



surfaces | 여러개의
tracks들이 있는데 이를
각각을 읽기 위해 head가 움직임

Rotational Delay



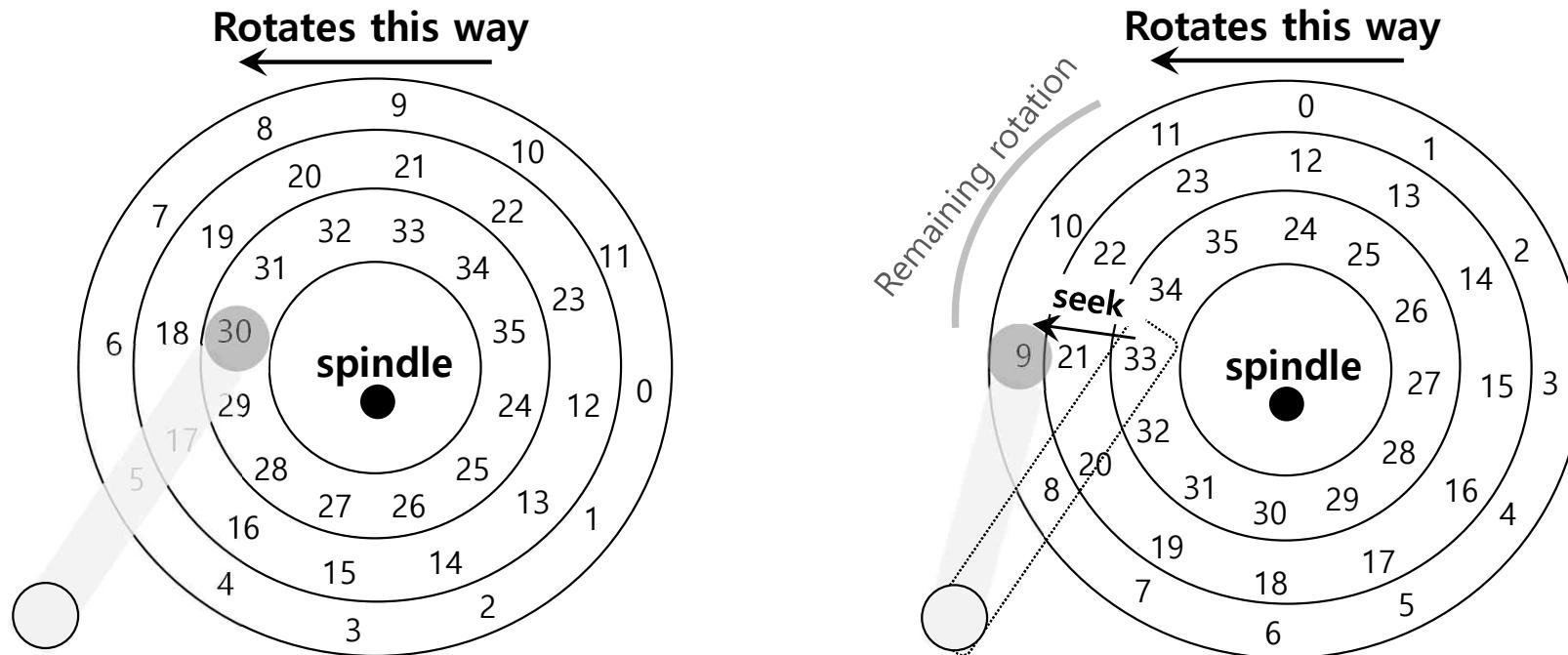
A Single Track Plus A Head

탐색하기 위해 회전할때 걸리는 시간

- **Rotational delay:** Time for the desired sector to rotate
 - Ex) Full rotational delay is R and we start at sector 6
 - Read sector 0: Rotational delay = $\frac{R}{2}$
 - Read sector 5: Rotational delay = $R-1$ (worst case)
회전 방향 반대부분에 한칸 떨어져있으면 worst case겠지?

Seek Time

track0| 다수일 때



Three Tracks Plus A Head (Right: With Seek)
(e.g., read to sector 11)

- **Seek:** Move the disk arm to the correct track
 - **Seek time:** Time to move head to the track contain the desired sector
 - One of the most costly disk operations

disk operation에서 가장 오래걸리는 시간임

원하는 데이터가 있는 track 쪽으로 이동하는데 걸리는 시간

Seek Time (Cont.)

- Acceleration → Coasting → Deceleration → Settling
 - Acceleration: The disk arm gets moving 가속해서 빨리 그쪽으로 가
 - Coasting: The arm is moving at full speed 최대 속도
 - Deceleration: The arm slows down 최대 속도에서 바로 속도가 0이 되진 않을거 아녀 속도를 늦춰
 - Settling: The head is *carefully positioned* over the correct track 올바른 부분에 조심스럽게 위치해
➤ The settling time is often quite significant, e.g., 0.5 to 2ms
0.5~2ms의 시간

Transfer time

- The final phase of I/O I/O 작업의 마지막 단계
 - Data is either *read from* or *written* to the surface
- Seek time or rotational delay >> data transfer time seek time이나 rotational delay보다 훨씬 적은 시간이 걸려

I/O service time

- I/O time ($T_{I/O}$) = $T_{seek} + T_{rotation} + T_{transfer}$
 - Seek time (T_{seek})
 - Time to move the disk head to the desired track
 - Seek time \approx seek distance
 - Rotational delay ($T_{rotation}$)
 - Time for the desired sector to rotate to the disk head
 - Best case = 0, Worst case = time for one rotation
 - Data transfer time ($T_{transfer}$)
 - Time for transferring data from disk media to the disk buffer, and vice versa
- The rate of I/O ($R_{I/O}$) = $\frac{\text{Size Transfer}}{T_{I/O}}$

I/O service time (Cont.)

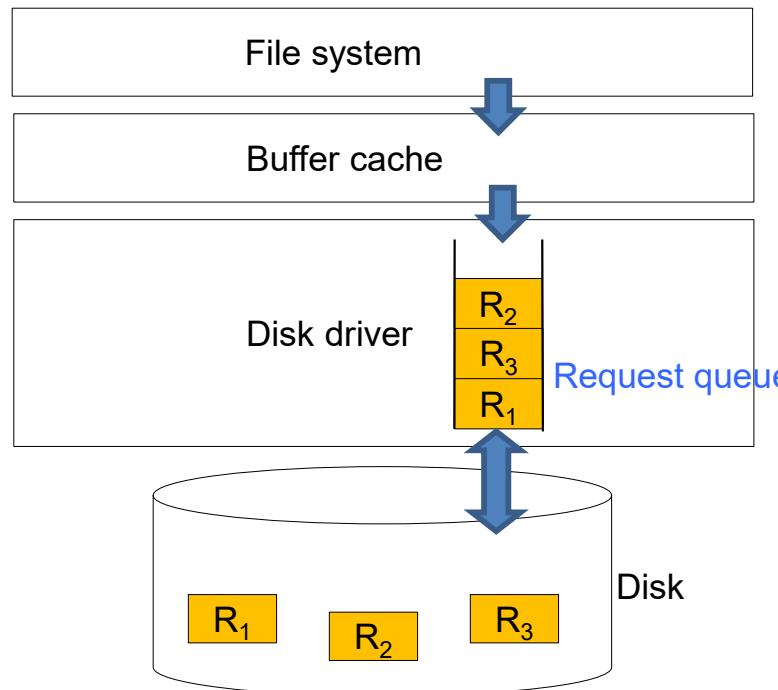
- Random workload: Issue 4KB read to random locations
- Sequential workload: Read 100MB consecutively

	Cheetah 15K.5	Barracuda
Capacity	300 GB	1 TB
RPM	15,000	7,200
Average Seek	4 ms	9 ms
Max Transfer	125 MB/s	105 MB/s
Platters	4	4
Cache	16 MB	16/32 MB
Interface	SCSI	SATA
Random	$T_{\text{seek}} = 4\text{ms}$ $T_{\text{rotation}} = 60 / 15000 / 2 = 2\text{ms}$ $T_{\text{transfer}} = 4\text{KB} / 125\text{MB} = 32\mu\text{s}$ $R_{\text{I/O}} \approx 4\text{KB} / 6\text{ms} = 0.66 \text{ MB/s}$	$T_{\text{seek}} = 9\text{ms}$ $T_{\text{rotation}} = 60 / 7200 / 2 = 4.2\text{ms}$ $T_{\text{transfer}} = 4\text{KB} / 105\text{MB} = 37\mu\text{s}$ $R_{\text{I/O}} \approx 4\text{KB} / 13.2\text{ms} = 0.31 \text{ MB/s}$
Sequential	$T_{\text{transfer}} = 100\text{MB} / 125\text{MB} = 0.8\text{s}$ $R_{\text{I/O}} \approx 100\text{MB} / 0.8\text{s} = 125 \text{ MB/s}$	$T_{\text{transfer}} = 100\text{MB} / 105\text{MB} = 0.95\text{s}$ $R_{\text{I/O}} \approx 100\text{MB} / 0.95\text{s} = 105 \text{ MB/s}$

random workload가 확실히 데이터 전송 속도가 떨어짐 시간으로 나눌때 상대적으로 작은 값은
강 무시하고 나눈거임

Disk Scheduling Algorithms

- Disk I/O scheduler
 - Generally, new requests for service will be placed in the request queue
 - Request - a scheduling unit in the disk I/O scheduler
 - ✓ type(R/W), disk address, the number of sectors



이전에 했던 스케줄링과 비슷하게 disk I/O scheduler가 scheduling의 기본 단위인 request로 큐에 넣어서 순차적으로 처리하는 거 평균적인 service time이 스케줄링 알고리즘에 의존한다 즉 더 좋은 알고리즘을 쓰면 service time에서 이점이 생기는거지

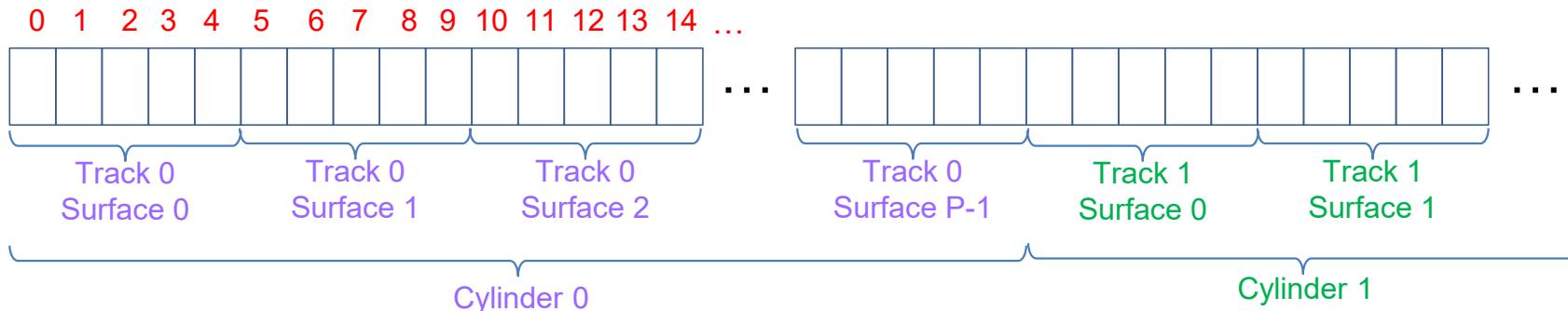
- When a request is completed, disk I/O scheduler chooses the next one
- Average disk I/O service time depends on disk I/O scheduling algorithm

Disk Scheduling Algorithms

- Objective of disk scheduling algorithm
 - Minimizes the seek time and rotational delay
 - Cylinder-based mapping in disk controller

disk scheduling의 목적은 결과적으로 I/O time에서 가장 시간을 많이 잡아먹는 seek time과 rotational delay를 최소화시키는 거임 그래서 이전에 말했던 cylinder라는 개념을 이용해서 surface마다 같은 track부분에 데이터를 저장해두는거지

근데 rotational delay는 disk마다 다르고 rotation speed도 다르고 track마다 sector의 수도 다르고 이런 하드웨어적인 차이 때문에 OS에서 제대로 측정하기 어려워



But, operating system cannot estimate the rotational delay

➤ The rotation speed and the number of sectors per track of disks are different

– Generally, disk scheduling focus on minimizing the seek time

따라서 seek time을 최소화 하는거에 초점을 맞추게 된다.

- Evaluation of disk scheduling algorithms

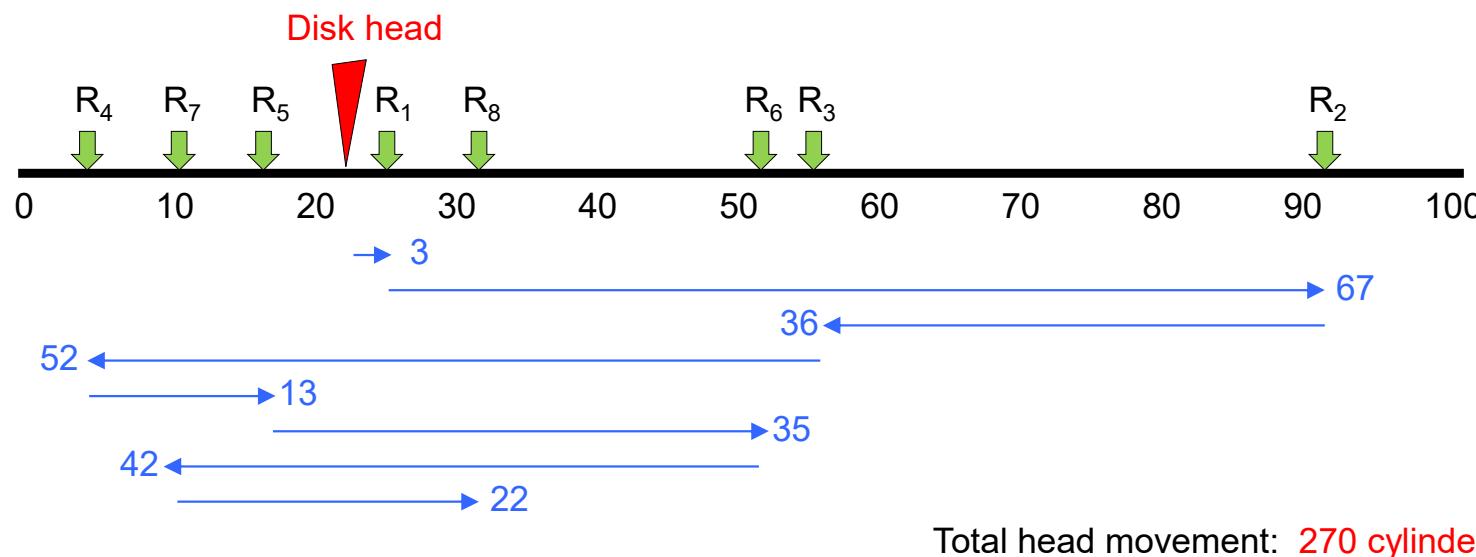
- The state of the request queue and the current position of disk head are given
 - For each disk scheduling algorithm, the total head movements is computed

head가 움직인 총 횟수가 결국 seek time에 비례할테니 이를 측정하기로 하자

FCFS (First Come First Served)

- Services the requests **in the arrival order**
- E.g.
 - Queue state → $R_1(25), R_2(92), R_3(56), R_4(4), R_5(17), R_6(52), R_7(10), R_8(32)$
 - Position of disk head → 22

도착한 순서대로 ㄱㄱ



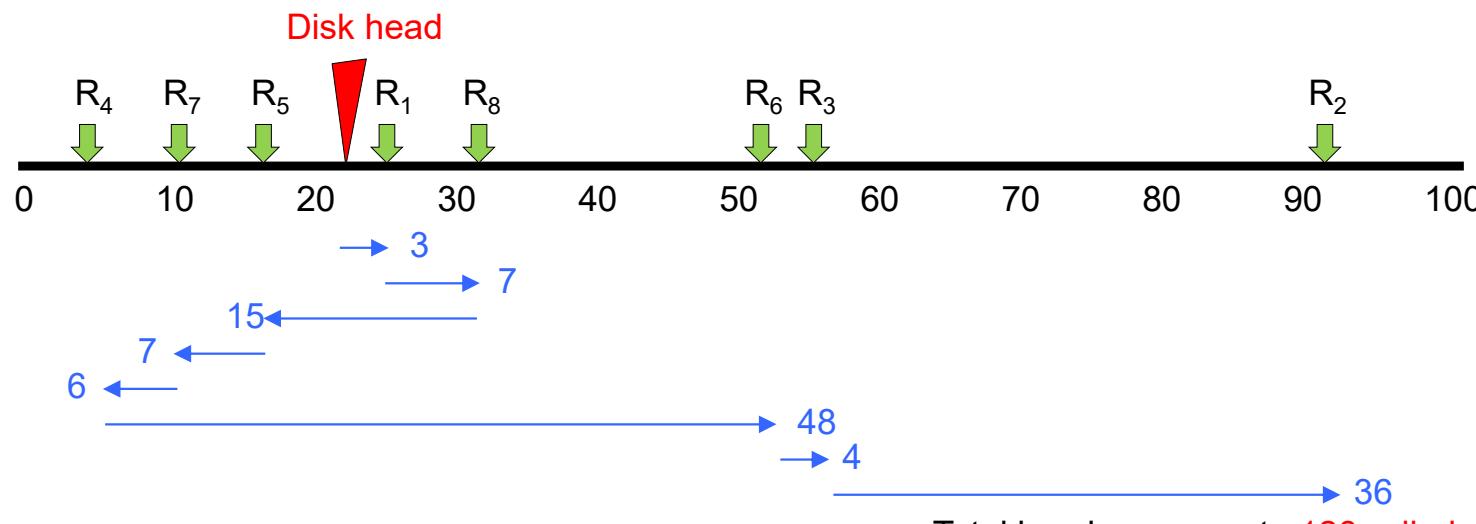
- It has **too long seek distance**

너무 마니 움직여!!!!!!

SSTF (Shortest Seek Time First)

- Services the request **with the minimum seek time** from the current head position
- E.g.
 - Queue state → $R_1(25), R_2(92), R_3(56), R_4(4), R_5(17), R_6(52), R_7(10), R_8(32)$
 - Position of disk head → 22

현 시점에서 가장 가까운 곳부터
가는 greedy algorithm임



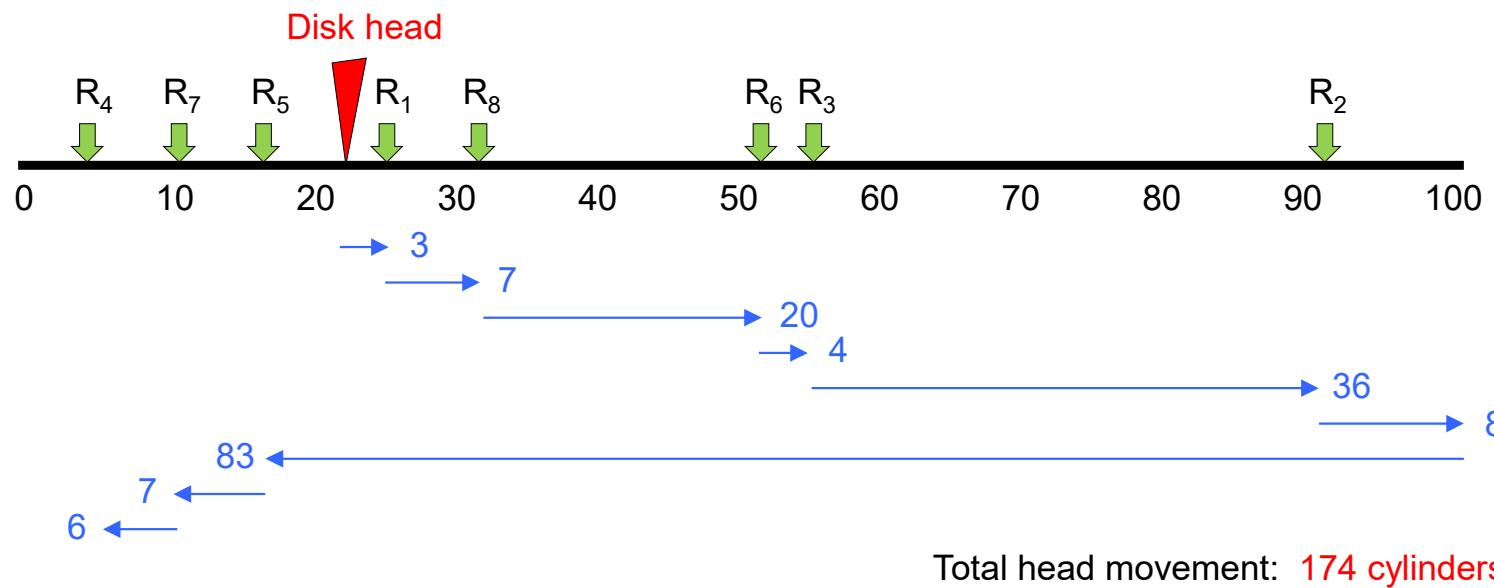
- It is a **greedy algorithm**
- It may cause **starvation** of some requests

이동하고 있는데 계속 새로운 request가 오는데 얘가 가까우면 계속 새로운 애만 실행할거아냐 그럼 미리 요청한 애는 처리를 안해주니까 starvation이 발생한다는 거임

SCAN

- The disk head starts at one end of the disk, and moves toward the other end, servicing requests until it gets to the end, where head movement is reversed and servicing continues
- It is also called **elevator algorithm**
- E.g.
 - Queue state → $R_1(25), R_2(92), R_3(56), R_4(4), R_5(17), R_6(52), R_7(10), R_8(32)$
 - Position of disk head → 22

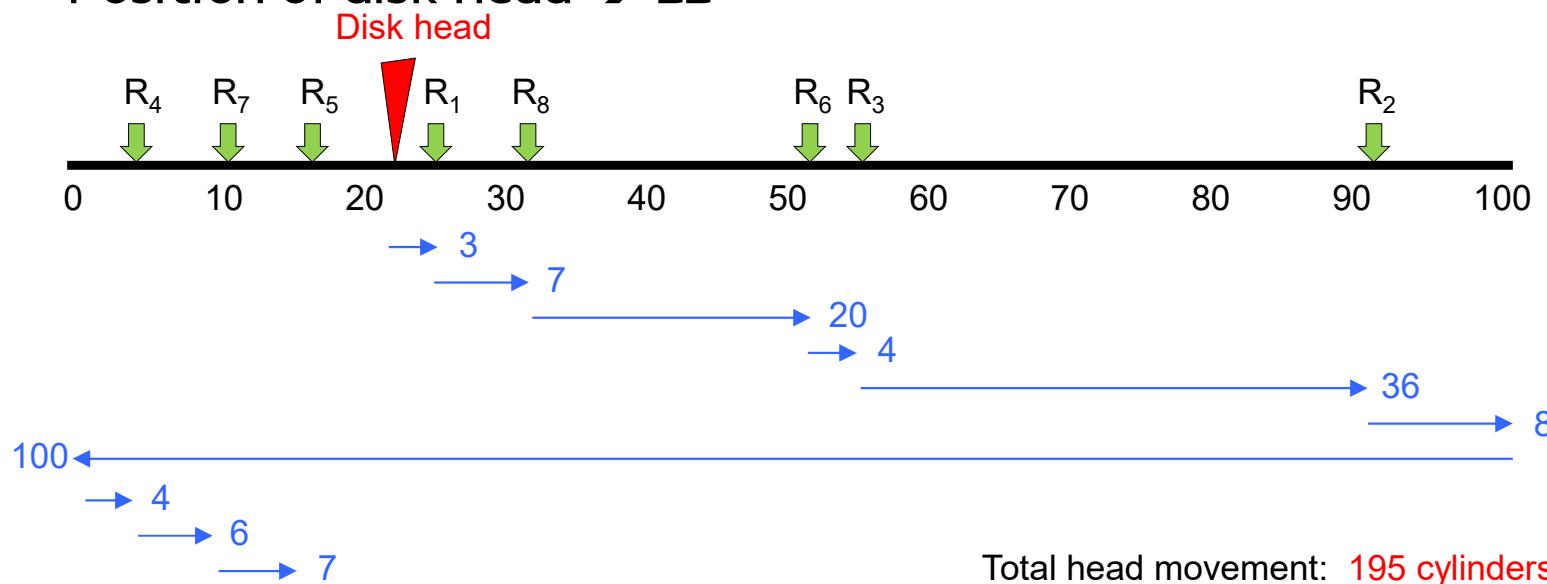
탐색을 한쪽 끝쪽으로 쭉 가고 끝까지 가면 반대쪽으로 가는거



C-SCAN

- It is a variant of SCAN
- It provides a **more uniform wait time** than SCAN
- When it reaches the other end, it immediately returns to the beginning, **without servicing any requests on the return trip**
- E.g.

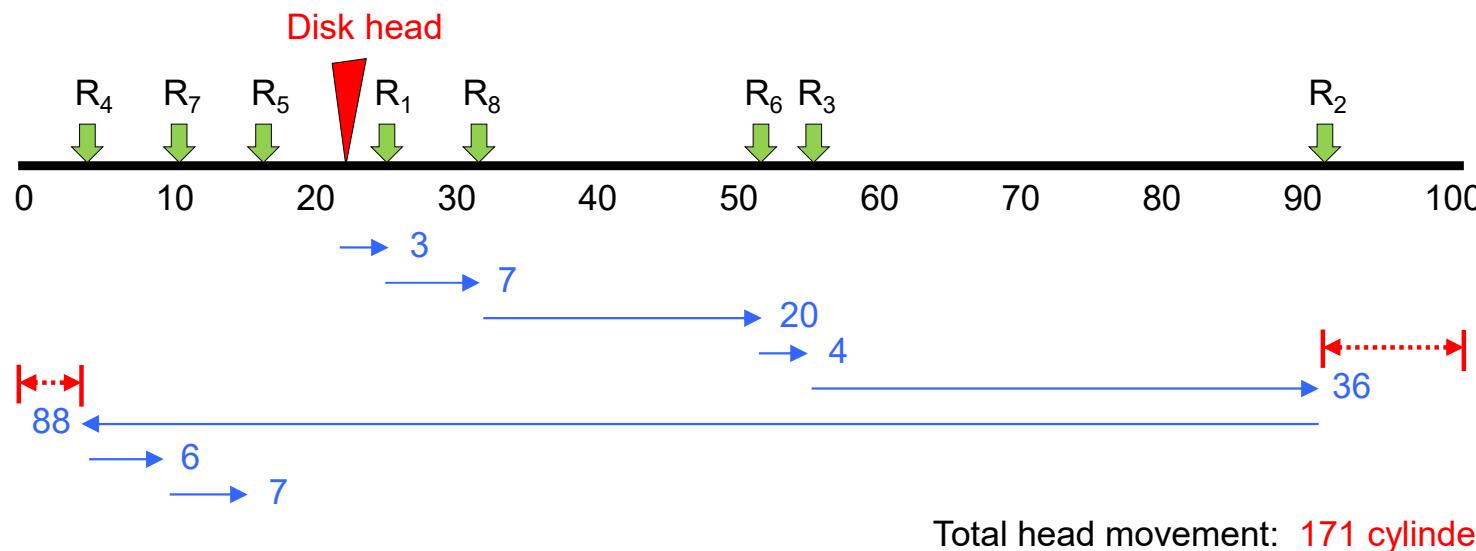
- Queue state → $R_1(25), R_2(92), R_3(56), R_4(4), R_5(17), R_6(52), R_7(10), R_8(32)$
- Position of disk head → 22



C-LOOK

- It is a practical implementation of C-SCAN
- The disk head goes only as far as the final request in each direction. Then, it reverses direction immediately **without going all the way to the end of disk**
- E.g.
 - Queue state → $R_1(25), R_2(92), R_3(56), R_4(4), R_5(17), R_6(52), R_7(10), R_8(32)$
 - Position of disk head → 22

C-SCAN에서 조금 보완해서 굳이 끝까지 가지 않고 마지막 요청까지만 가고 다시 돌아오는거



Disk cache

- Hold data read from or written to the disk
 - Allow the drive to quickly respond to requests
 - Small amount of memory (usually around 8 or 16 MB)
- Write on cache
 - Writeback (Immediate reporting)
 - Acknowledge a write has completed when it has **put the data in its memory**
캐시에만 저장하고 썻다고 하는거
 - faster but dangerous
전원꺼지거나 하면 위험할 빠르긴한데
 - Write through
 - Acknowledge a write has completed after the write has **actually been written to disk**
캐시에 쓰고 디스크에 서도 써지면 알림

Disk Formatting

- Physical formatting (low-level formatting)
 - Before using a disk, it must be divided into sectors that the disk controller can read and write
 - Most hard disks are physically formatted in manufacturing process
 - During this process, a special data structure for each sector is added

디스크가 사용되기 전
에 포맷되어야 하겠지
sector 단위로

이런걸 대개 제조 공
정에서 한다

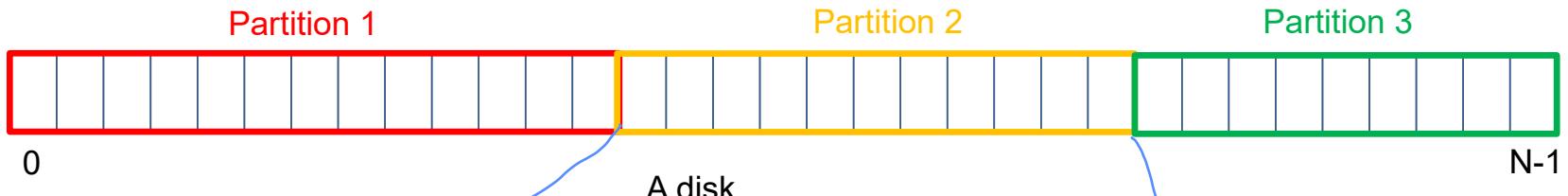


- The header and trailer contain information used by the disk controller
- header나 trailer라는 공간이 추가적으로 존재하는데 이곳에 sector에 대한 정보가 저장된다.
- E.g. sector number, error-correcting code (ECC)
- ← *아래*

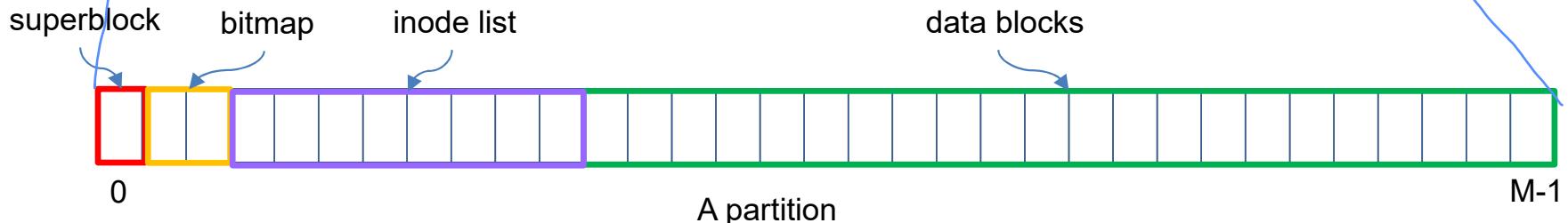
오늘날

Disk Formatting

- Partition
 - A disk can be divided into **multiple logical disks**



- Operating system can treat each partition **as though it were a separate disk**
하나의 디스크를 목적으로 맞게 여러 개의 logical disk로 나누는 행위
OS는 각각을 개별적인 disk로 판단해
- Logical formatting
 - Operating system store the **initial file system data structures**.
 - Superblock, free space bitmap, etc.



파일 시스템을 관리하기 위해서 필요한 전체 레이아웃을 작성하는 과정