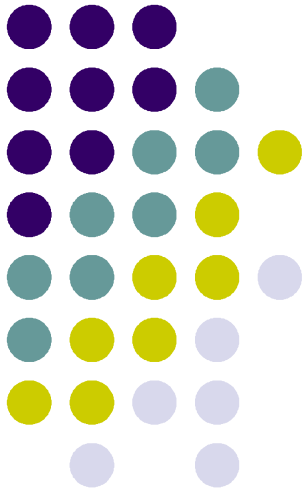
Disks and Disk Scheduling

Brian Railing

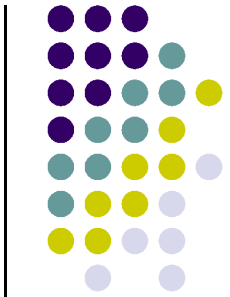
Friday, October 31st 2003

15-410 Fall 2003

Original lecture given by Steve Muckle on Monday, March 31st 2003
Additional Slides Taken from Eno Thereska's July Systems Talk

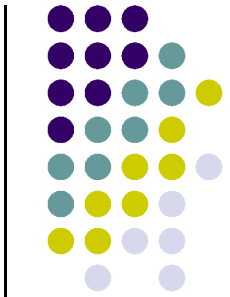


Overview



- Anatomy of a Hard Drive
- Common Disk Scheduling Algorithms

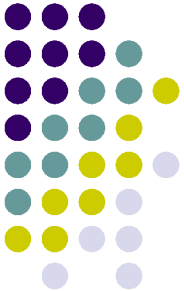
Project Discussion (3)



- Project 3 is over!
- War stories?
- Sage advice?

- Sign ups for interviews will begin soon
- Watch bboard

Project Discussion (4)



- File System project out today
- Lots of code
- Planning will save you pain and suffering
- Read it tonight (this afternoon even!)

Anatomy of a Hard Drive

- On the outside, a hard drive looks like this

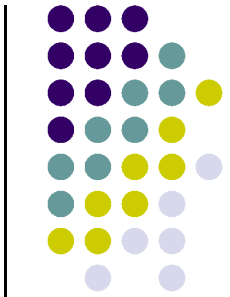
Taken from “How Hard Disks Work”
http://computer.howstuffworks.com/hard-disk2.htm

Anatomy of a Hard Drive

- If we take the cover off, we see that there actually is a “hard disk” inside

Taken from “How Hard Disks Work”
http://computer.howstuffworks.com/hard-disk2.htm

Anatomy of a Hard Drive

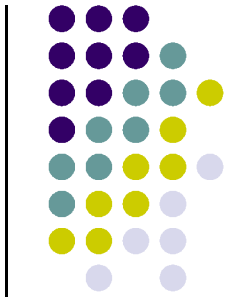


- A hard drive usually contains multiple disks, called *platters*

- These spin at thousands of RPM (5400, 7200, etc)

Taken from “How Hard Disks Work”
http://computer.howstuffworks.com/hard-disk2.htm

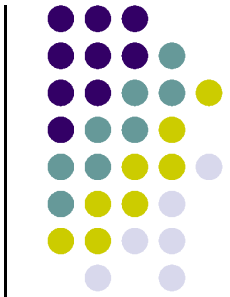
Anatomy of a Hard Drive



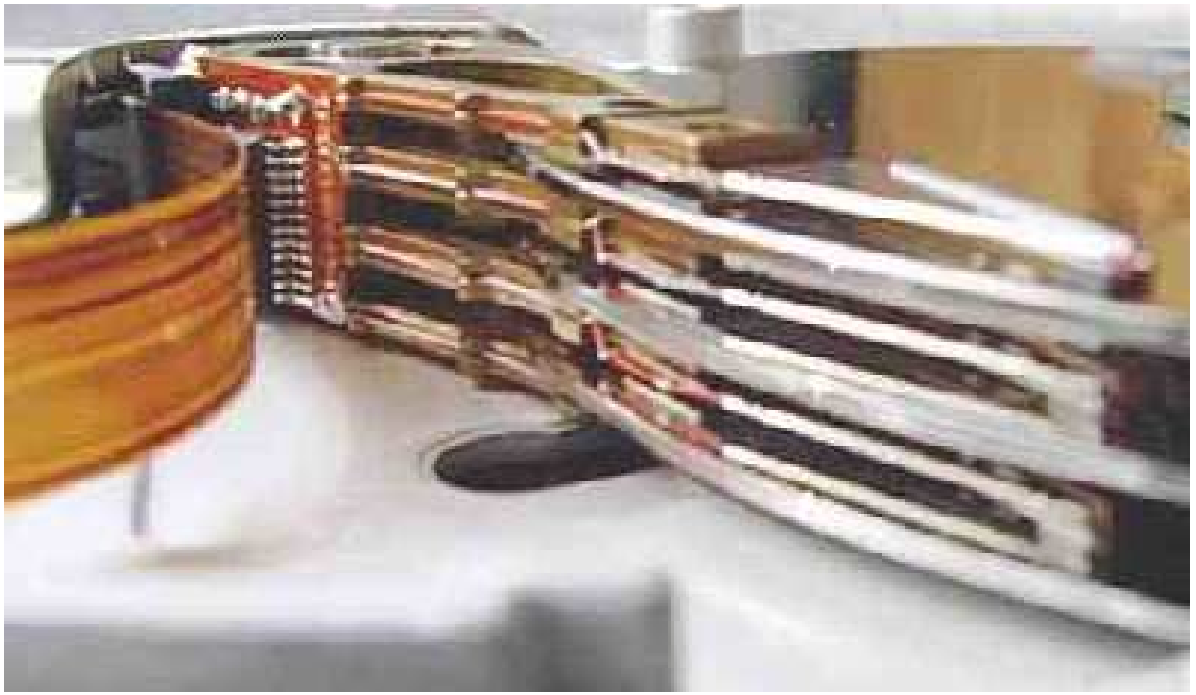
- Information is written to and read from the platters by the *read/write heads* on the *disk arm*

Taken from “How Hard Disks Work”
http://computer.howstuffworks.com/hard-disk2.htm

Anatomy of a Hard Drive

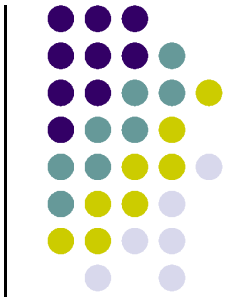


- Both sides of each platter store information
- Each side of a platter is called a *surface*
- Each surface has its own read/write head

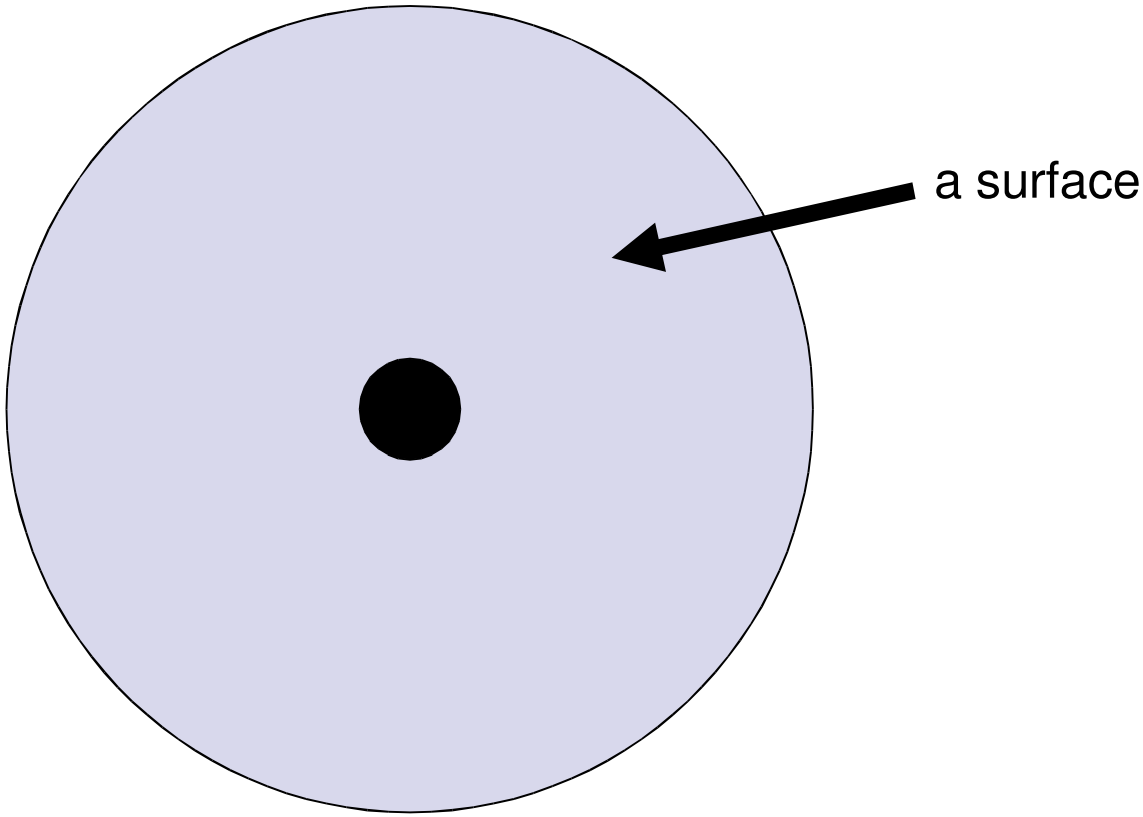


Taken from “How Hard Disks Work”
http://computer.howstuffworks.com/hard-disk2.htm

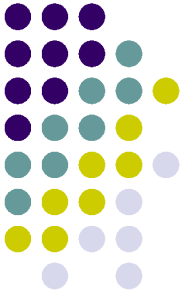
Anatomy of a Hard Drive



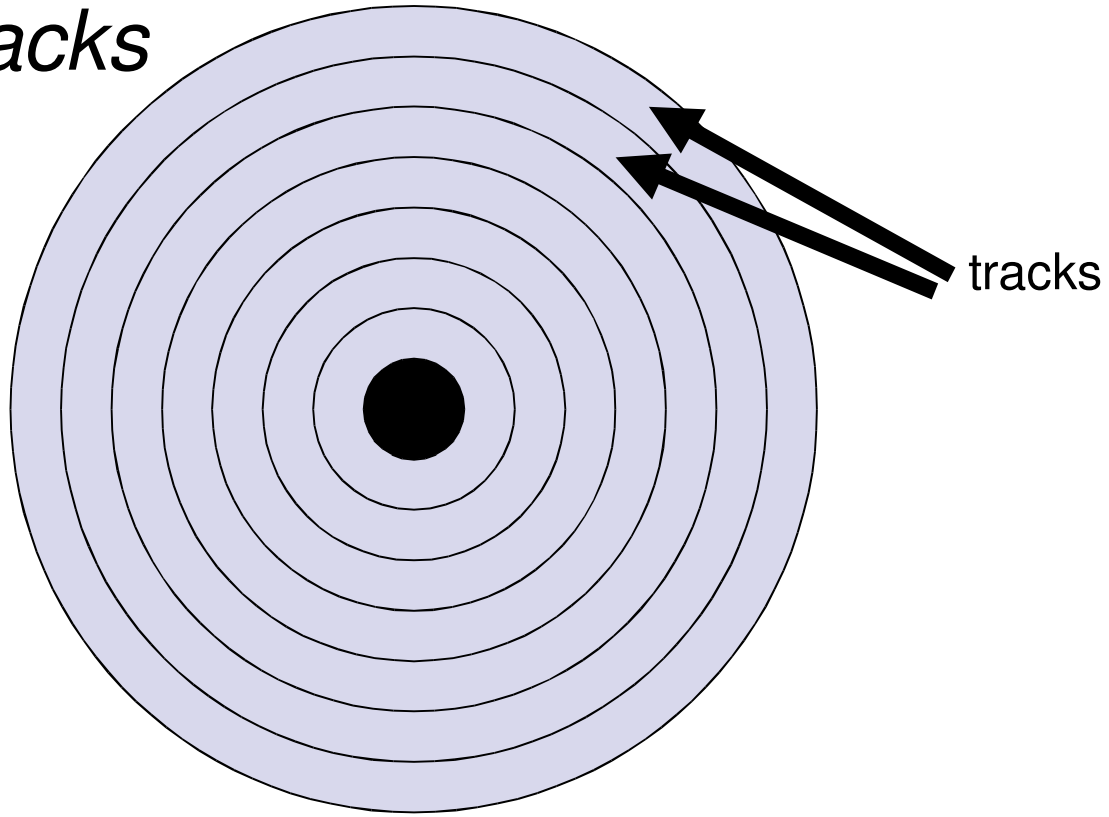
- How are the surfaces organized?



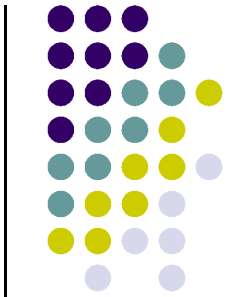
Anatomy of a Hard Drive



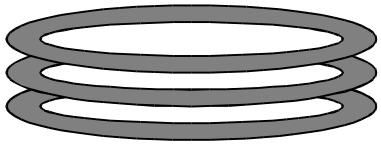
- Each surface is divided by concentric circles, creating *tracks*



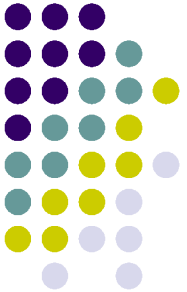
Anatomy of a Hard Drive



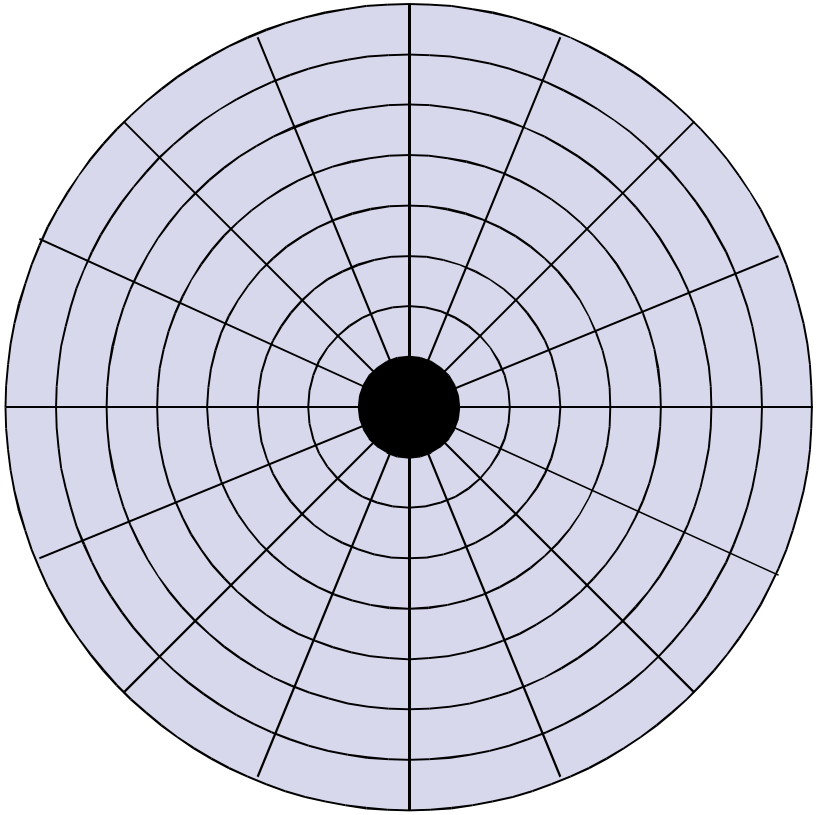
- The matching tracks on all surfaces are collectively called a *cylinder*



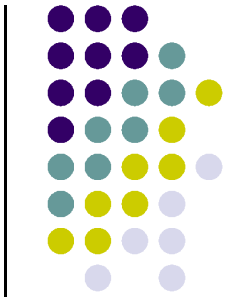
Anatomy of a Hard Drive



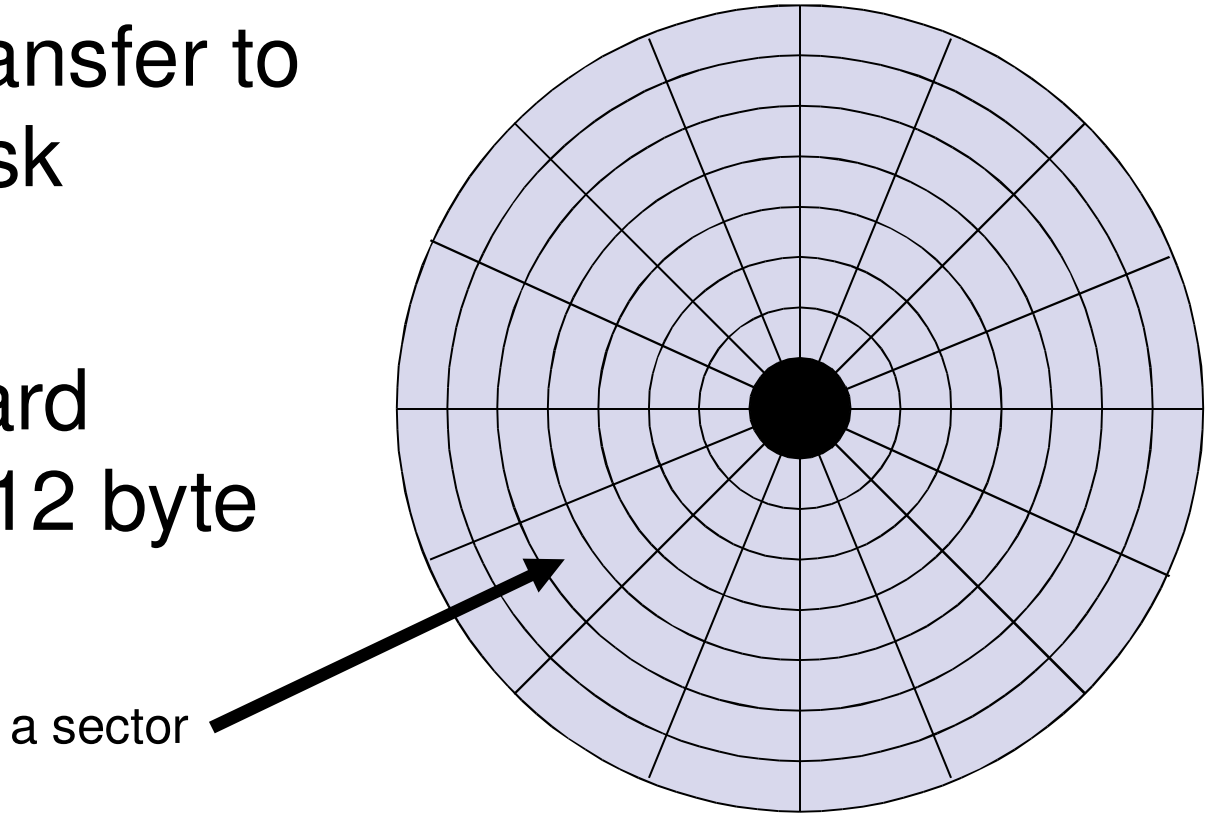
- These tracks are further divided into *sectors*



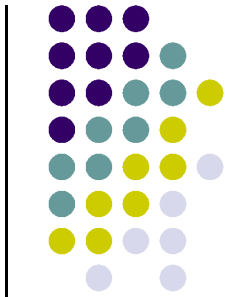
Anatomy of a Hard Drive



- A sector is the smallest unit of data transfer to or from the disk
- Most modern hard drives have 512 byte sectors

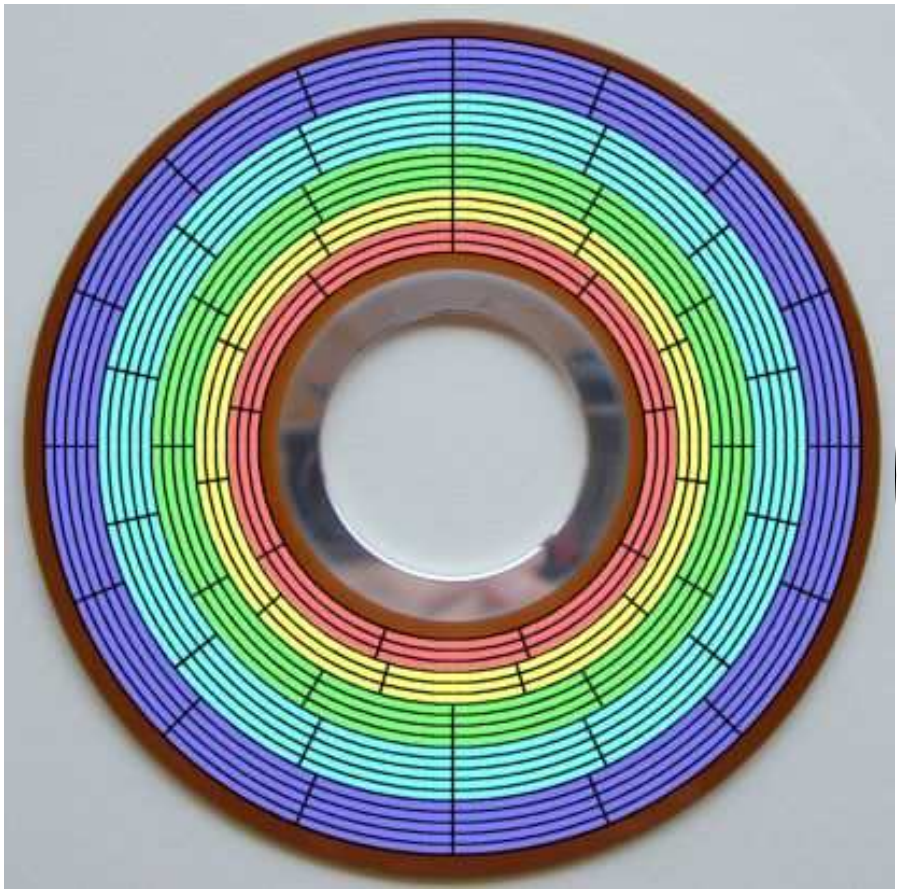


Anatomy of a Hard Drive



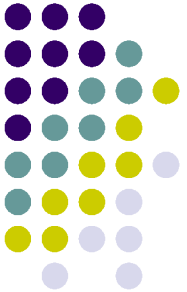
- Does this mean that
  sectors
  on the outside of a
  surface
  are larger than those on
  the inside?

- Modern hard drives
  fix this with
  zoned bit recording

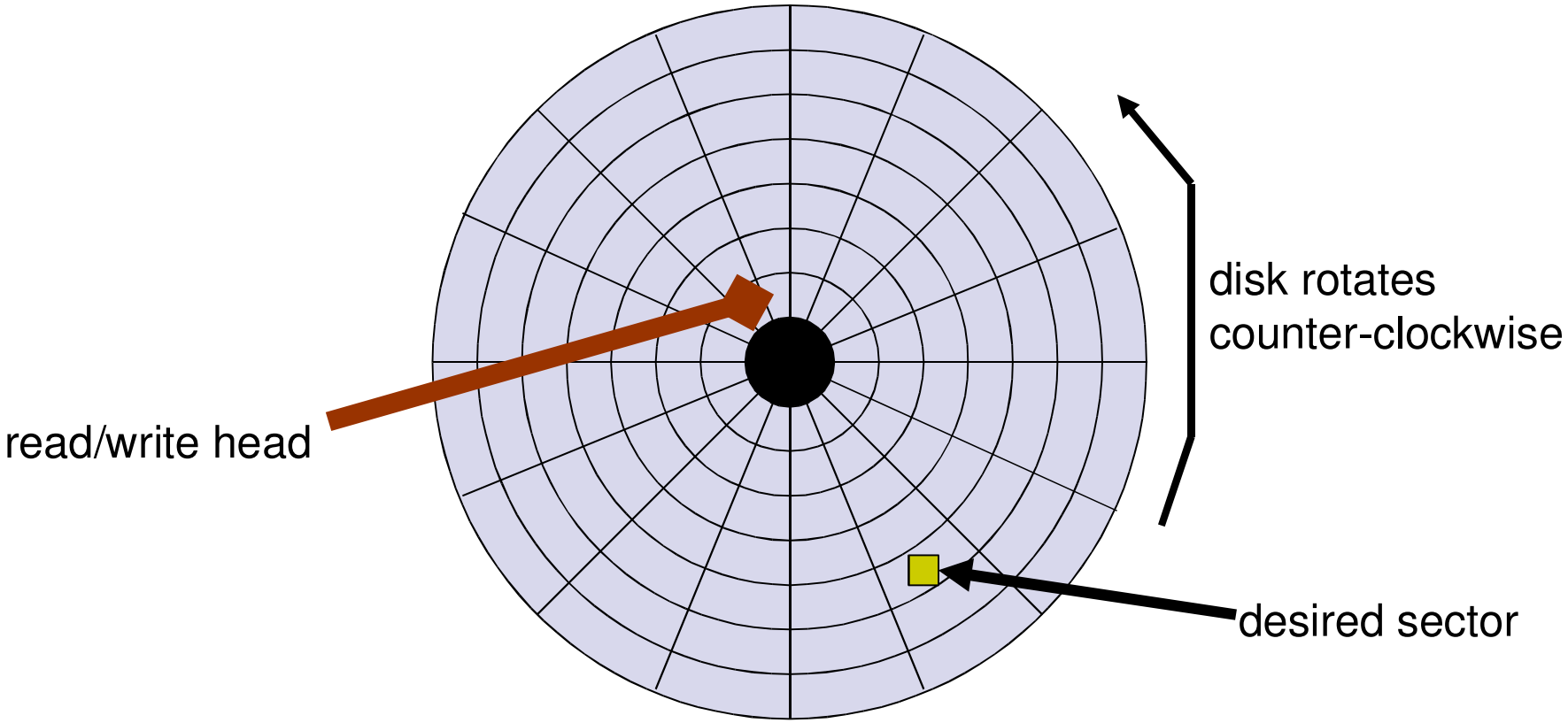


Taken from “Reference Guide – Hard Disk Drives”
http://www.storagereview.com/map/lm.cgi/zone

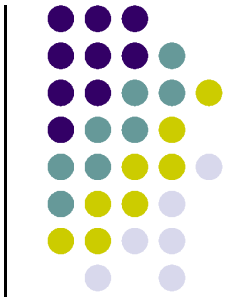
Anatomy of a Hard Drive



- Why don't we read in a sector from the disk



Anatomy of a Hard Drive

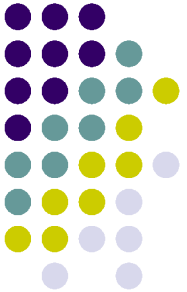


- We need to do two things to transfer a sector

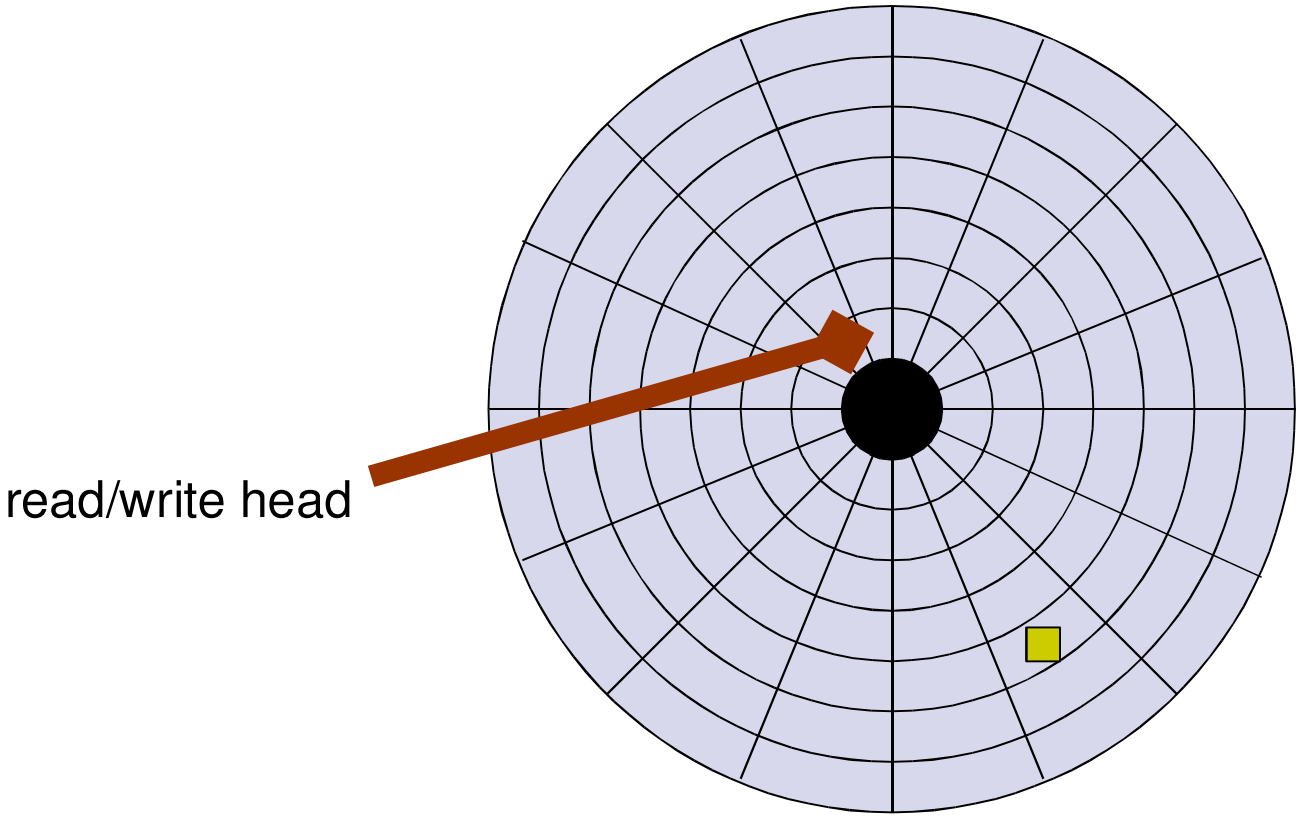
  1. Move the read/write head to the appropriate track (seek)

  2. Wait until the desired sector spins around

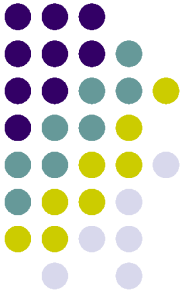
Anatomy of a Hard Drive



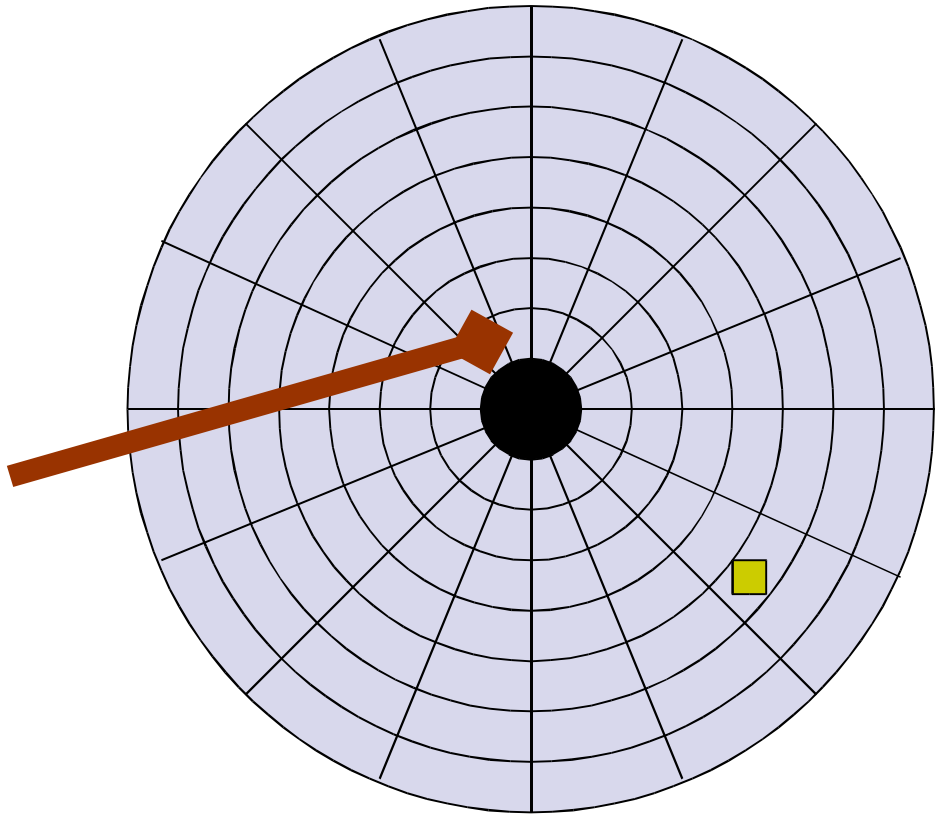
- Why don't we read in a sector from the disk



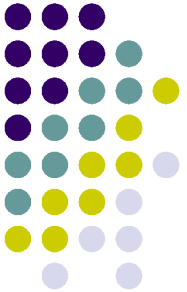
Anatomy of a Hard Drive



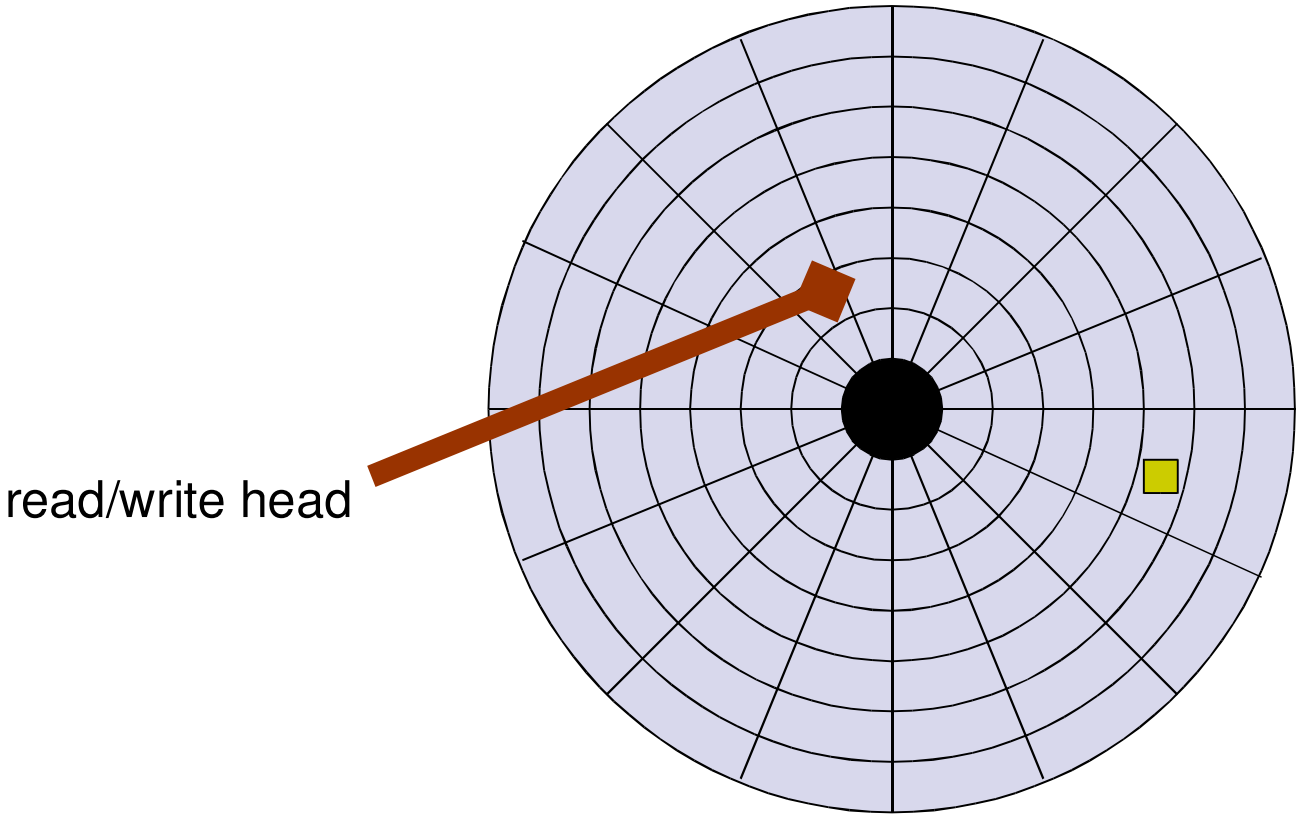
- Why don't we read in a sector from the disk



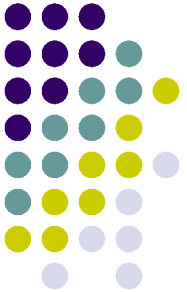
Anatomy of a Hard Drive



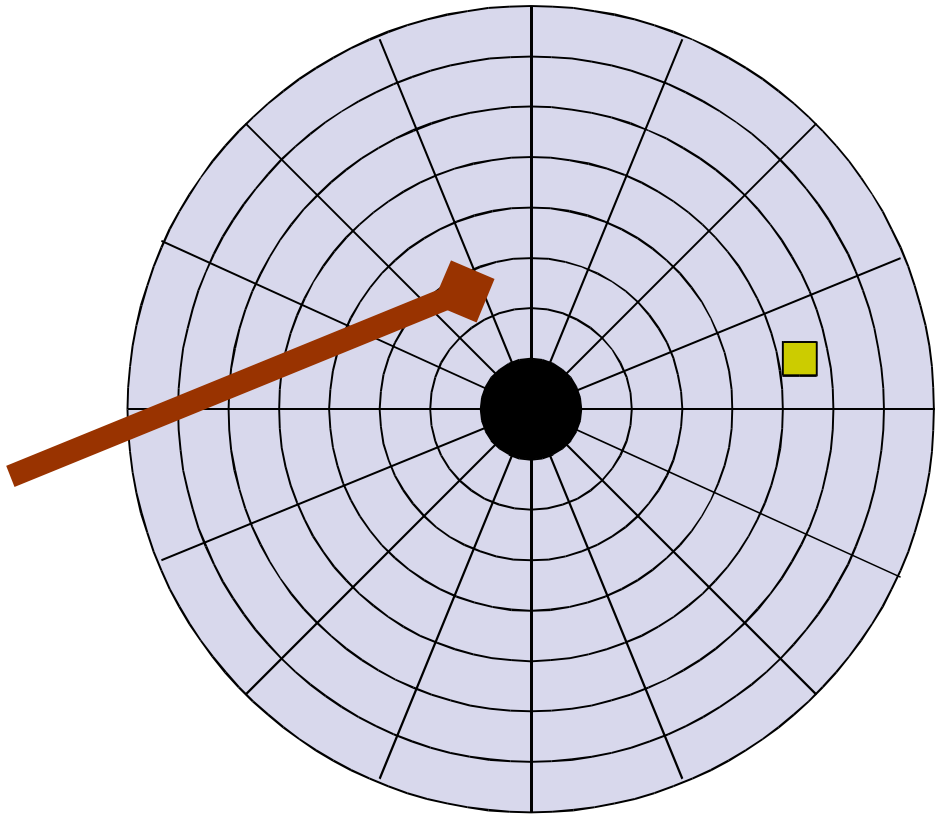
- Why don't we read in a sector from the disk



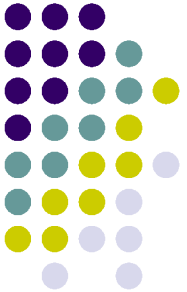
Anatomy of a Hard Drive



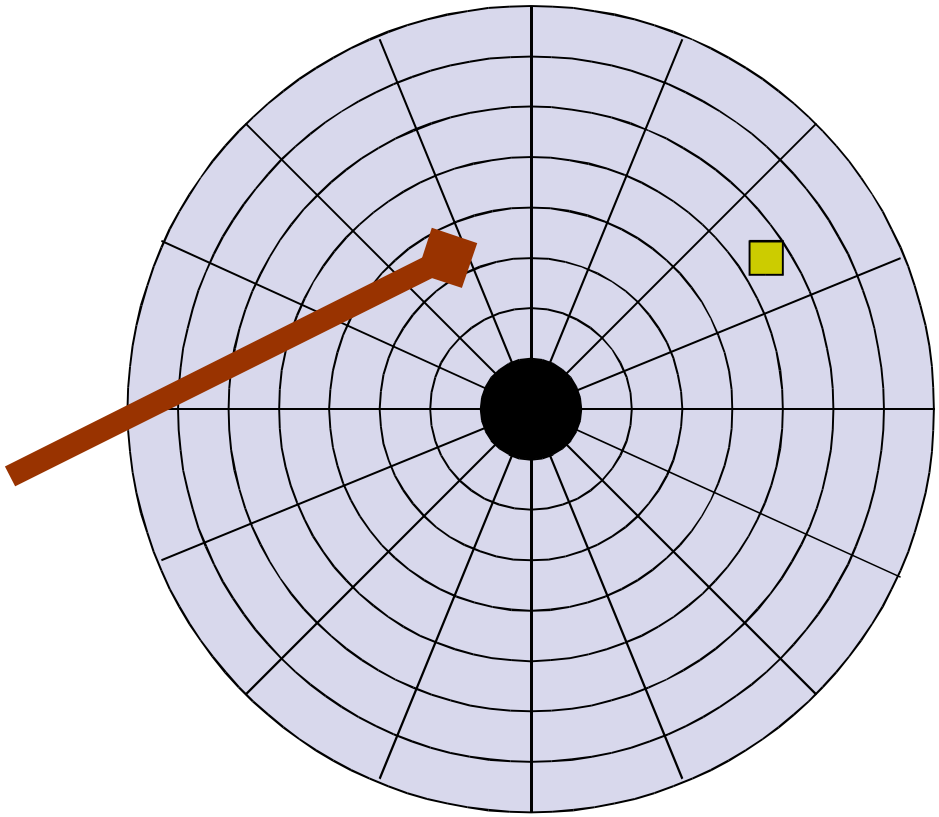
- Why don't we read in a sector from the disk



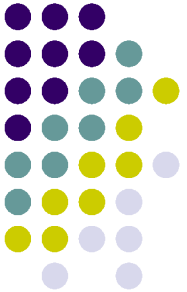
Anatomy of a Hard Drive



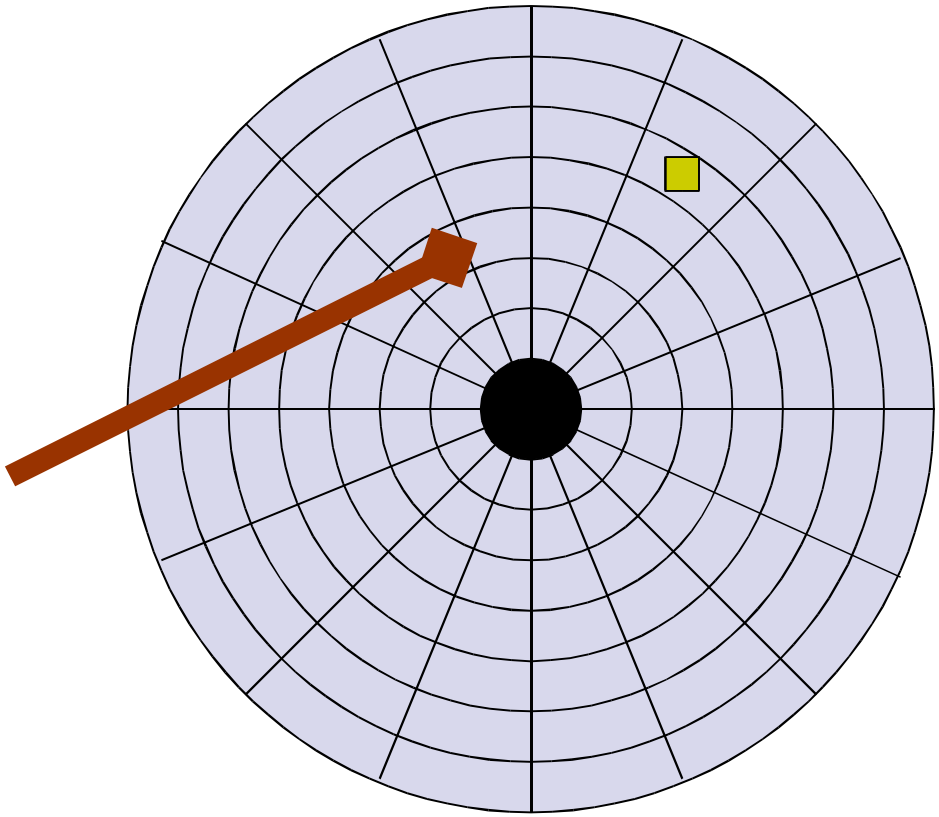
- Why don't we read in a sector from the disk



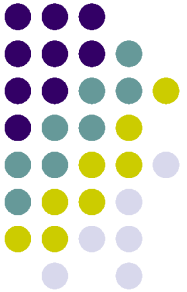
Anatomy of a Hard Drive



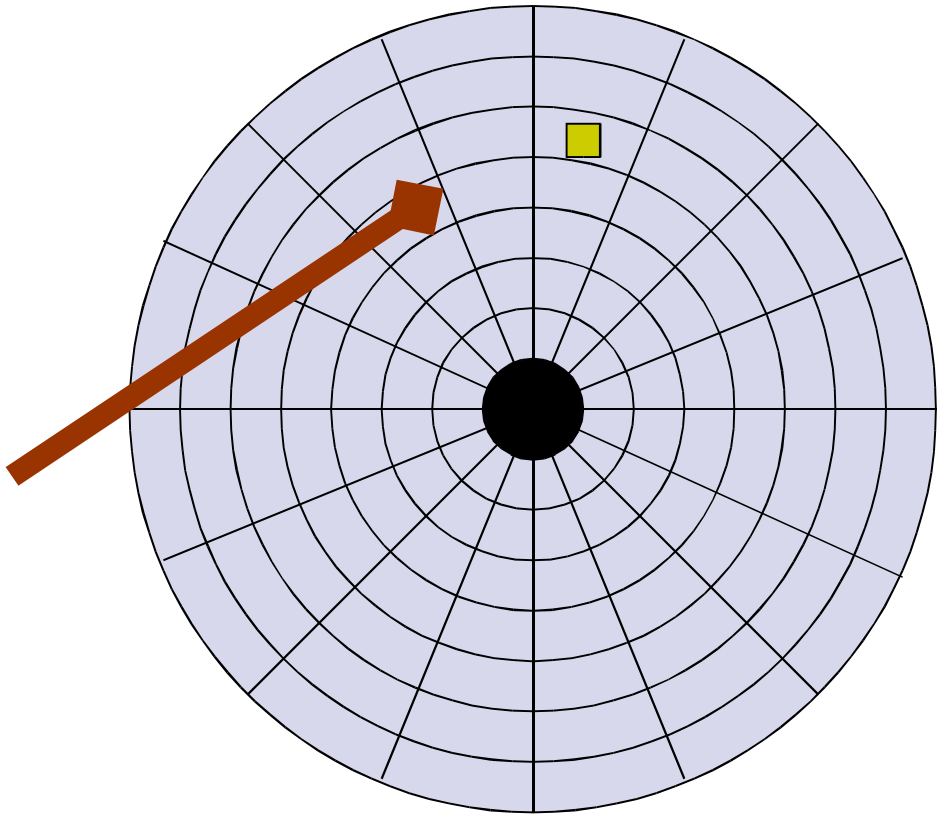
- Why don't we read in a sector from the disk



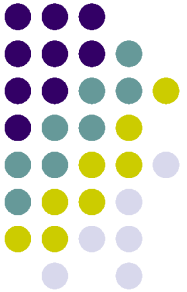
Anatomy of a Hard Drive



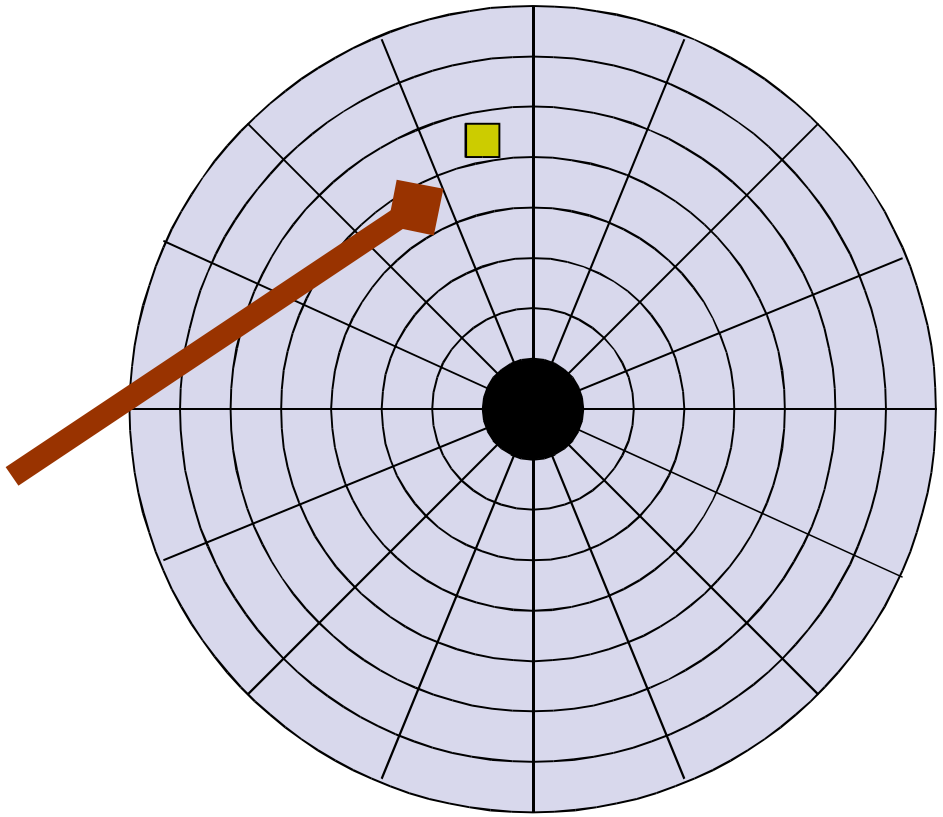
- Why don't we read in a sector from the disk



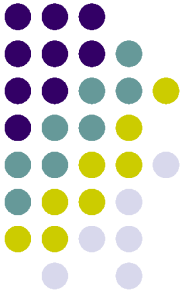
Anatomy of a Hard Drive



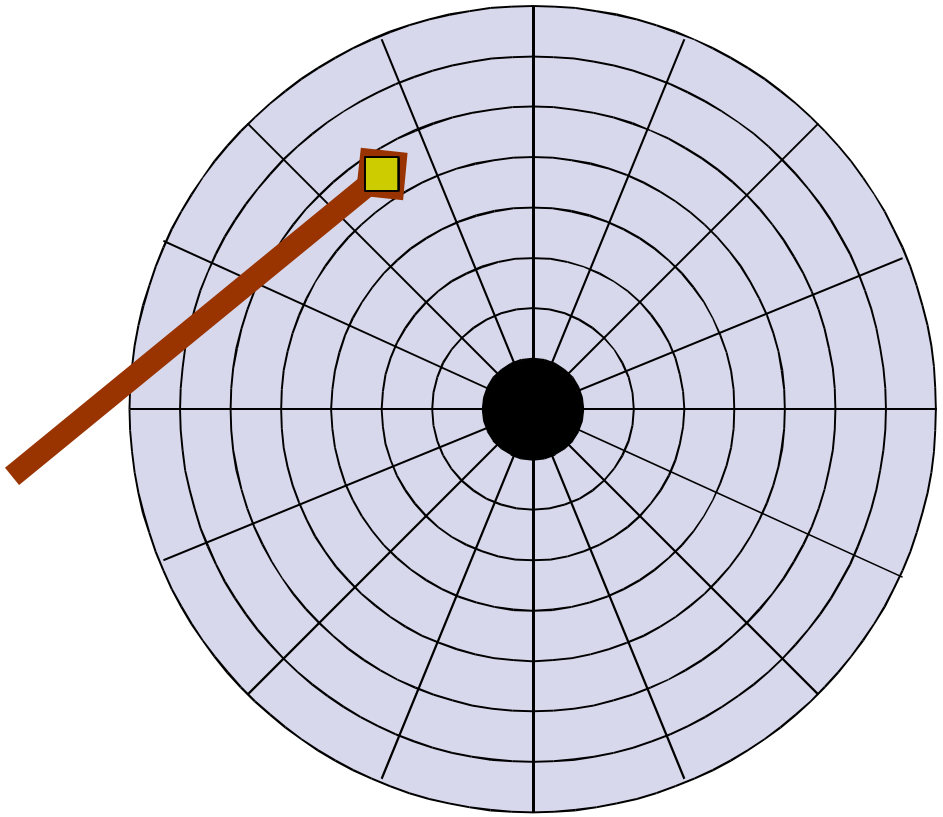
- Why don't we read in a sector from the disk



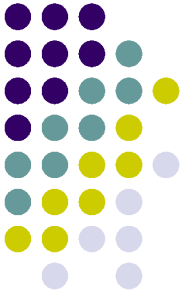
Anatomy of a Hard Drive



- Why don't we read in a sector from the disk

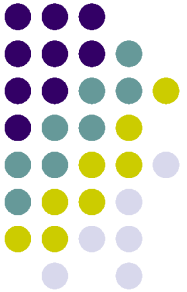


Anatomy of a Hard Drive



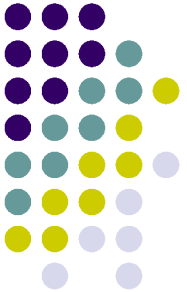
- On average, we will have to move the read/write head over half the tracks
- The time to do this is the average seek time, and is ~10ms
- We will also have to wait half a rotation
- The time to do this is rotational latency, and on a 5400 rpm drive is ~5.5ms

Anatomy of a Hard Drive



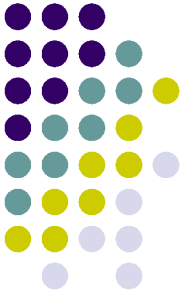
- There are many other things that determine overall disk access time including
  - settle time, the time to stabilize the read/write head after a seek
  - command overhead, the time for the disk to process a command and start doing something
- The things are fairly minor compared to seek time and rotational latency

Anatomy of a Hard Drive



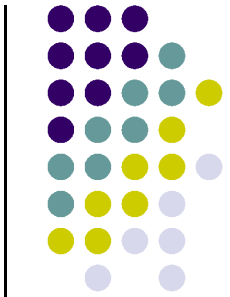
- Total drive random access time is on the order of 15 to 20 milliseconds
  - But wait! Disk transfer rates are tens of Mbytes
- Oh man, disks are slow
- What can we, as operating system programmers, do about this?

Disk Scheduling Algorithms



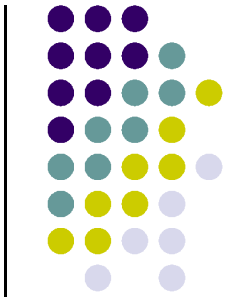
- The goal of a disk scheduling algorithm is to be nice to the disk
- We can help the disk by giving it requests that are located close to each other on the disk
- This minimizes seek time, and possibly rotational latency
- There exist a variety of ways to do this

Disk Scheduling Algorithms



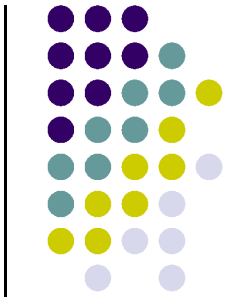
- What the OS knows about the disk?
  - Logical Block Numbers
  - Interface: IDE or SCSI
- What happened to sectors, tracks, etc?
  - They are hidden behind the logical block numbers
- How are they used?
  - File systems assign logical blocks to files

First Come First Served (FCFS)



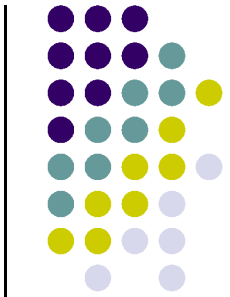
- Requests are sent to the disk as they are generated by the OS
- Trivial to implement
- Fair – no request will be starved because of its location on the disk
- Provides an unacceptably high mean response time

SCAN



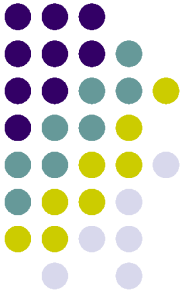
- Send requests in ascending cylinders
- When last cylinder is reached, reverse the scan
- Mean response time is worse than SSTF, but better than FCFS
- Better response time variance than SSTF
- Unfair – why?

LOOK



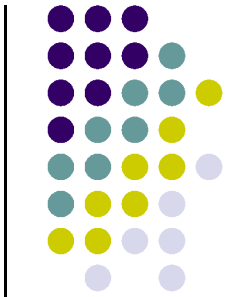
- Just like SCAN – sweep back and forth through cylinders
- If there are no more requests in our current direction we reverse course
- Improves mean response time, variance
- Still unfair though

CSCAN



- Send requests in ascending (or descending) cylinders
- When the last cylinder is reached, seek all the way back to the beginning
- Long seek is amortized across all accesses
- Variance is improved
- Fair
- Still missing something though…

C-LOOK

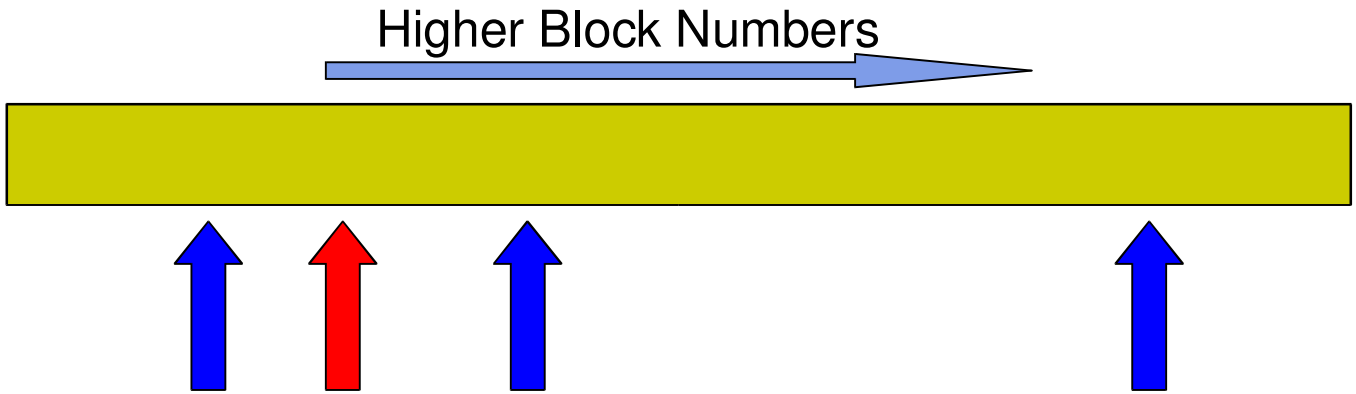


- CSCAN + LOOK
- Only scan in one direction, as in CSCAN
- If there are no more requests in current direction go back to furthest request
- Very popular

C-LOOK

Blue are requests

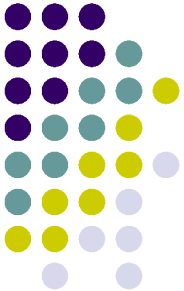
Yellow is disk

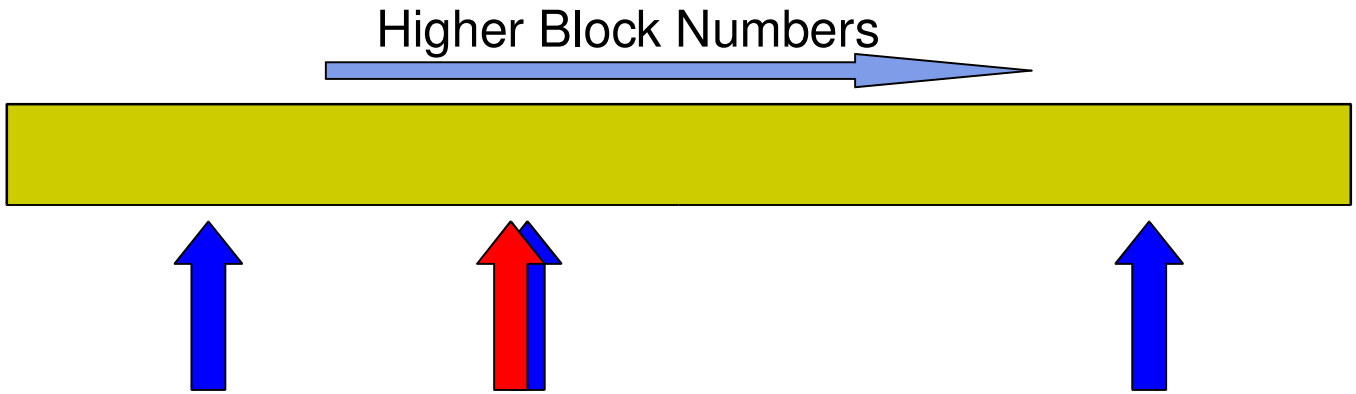


Red is disk head

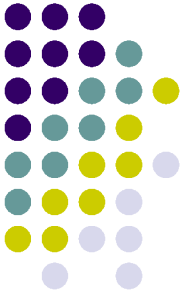
Green is completed requests

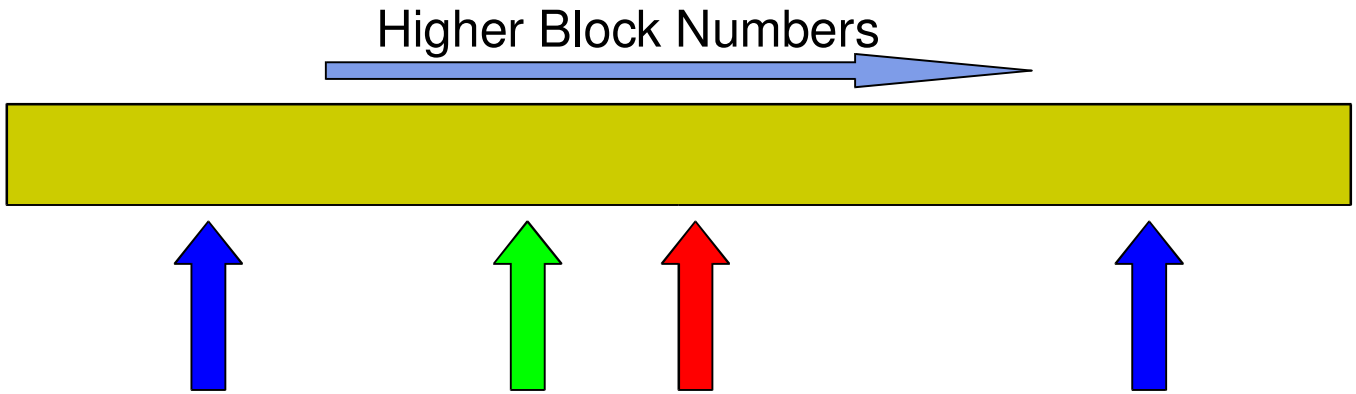
C-LOOK



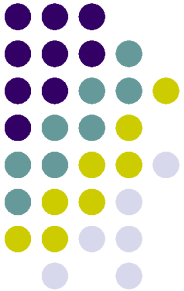


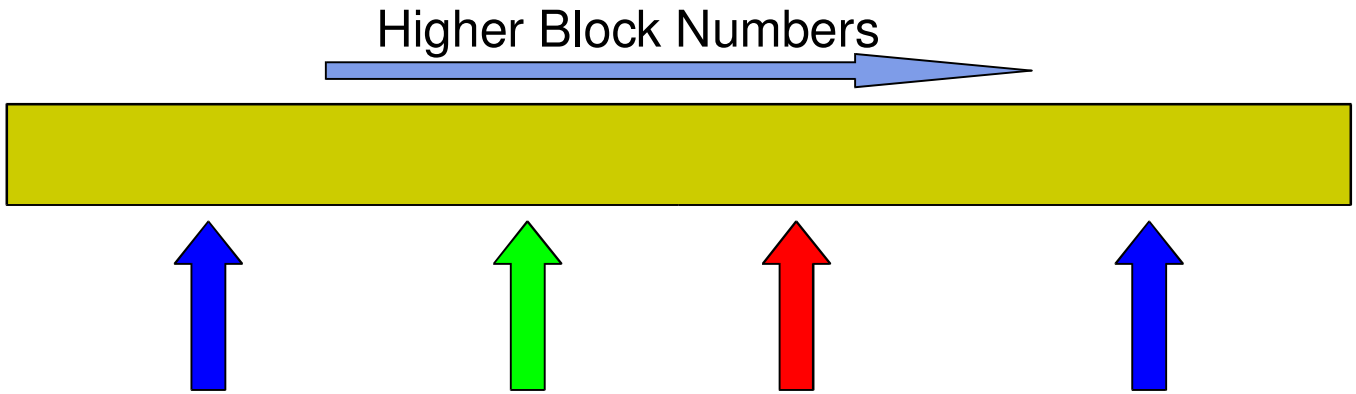
C-LOOK



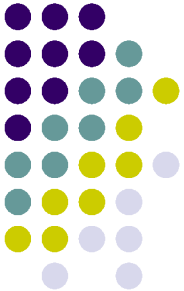


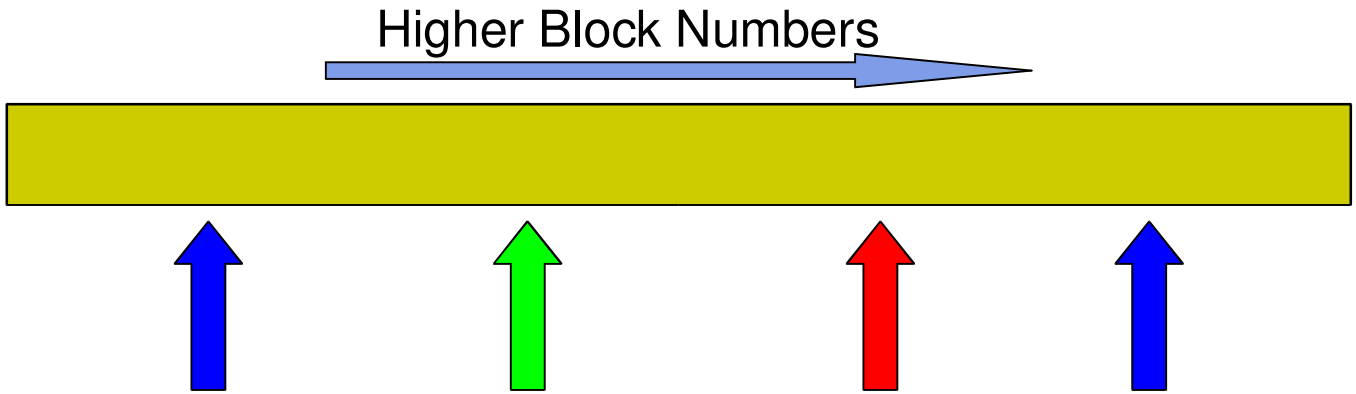
C-LOOK



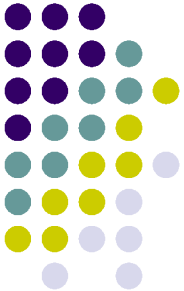


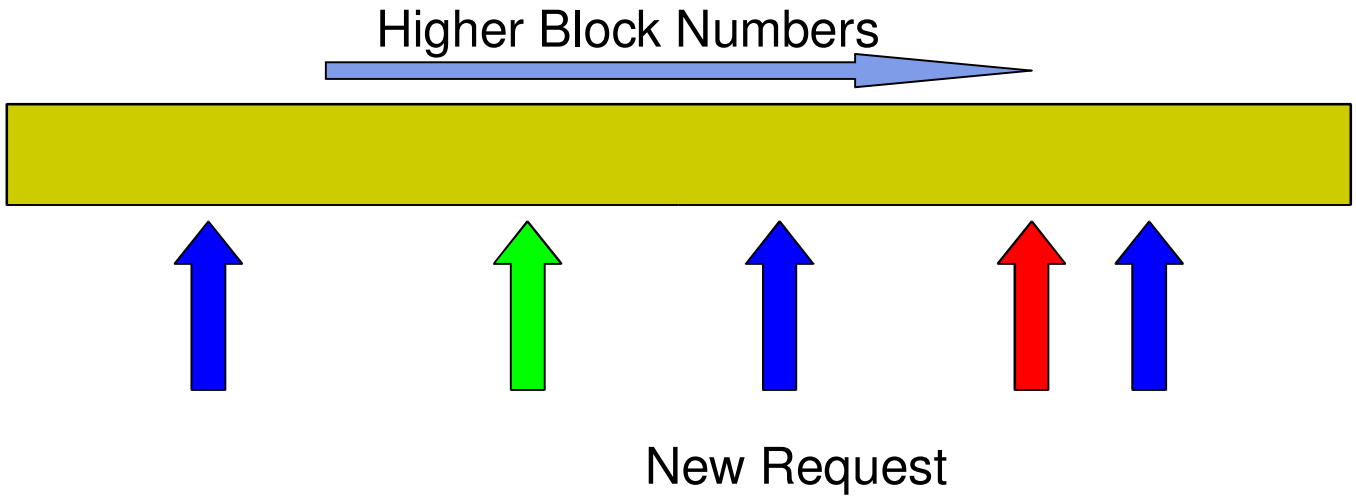
C-LOOK



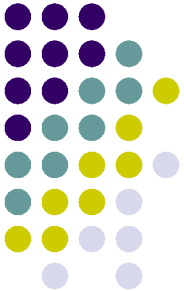


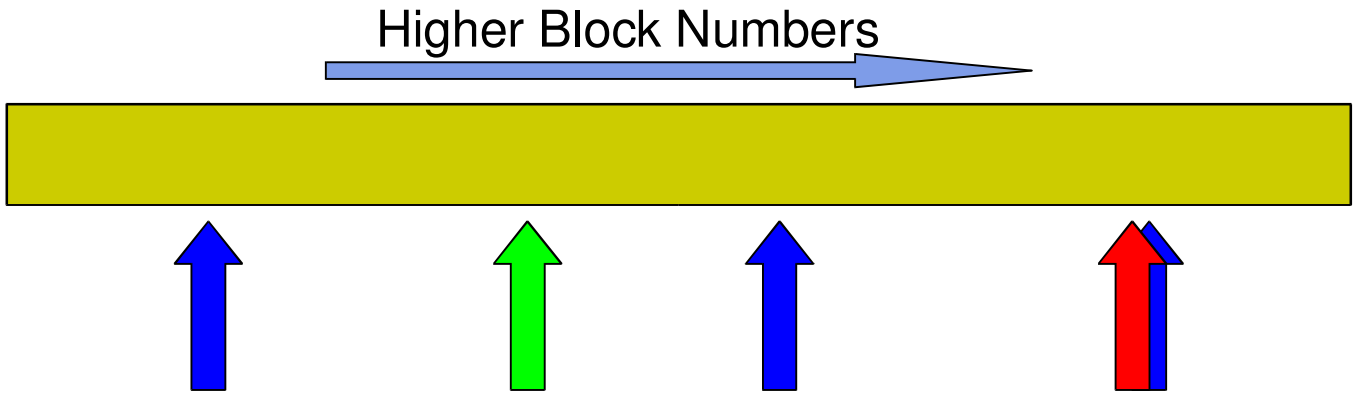
C-LOOK



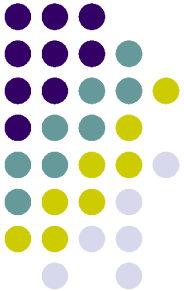


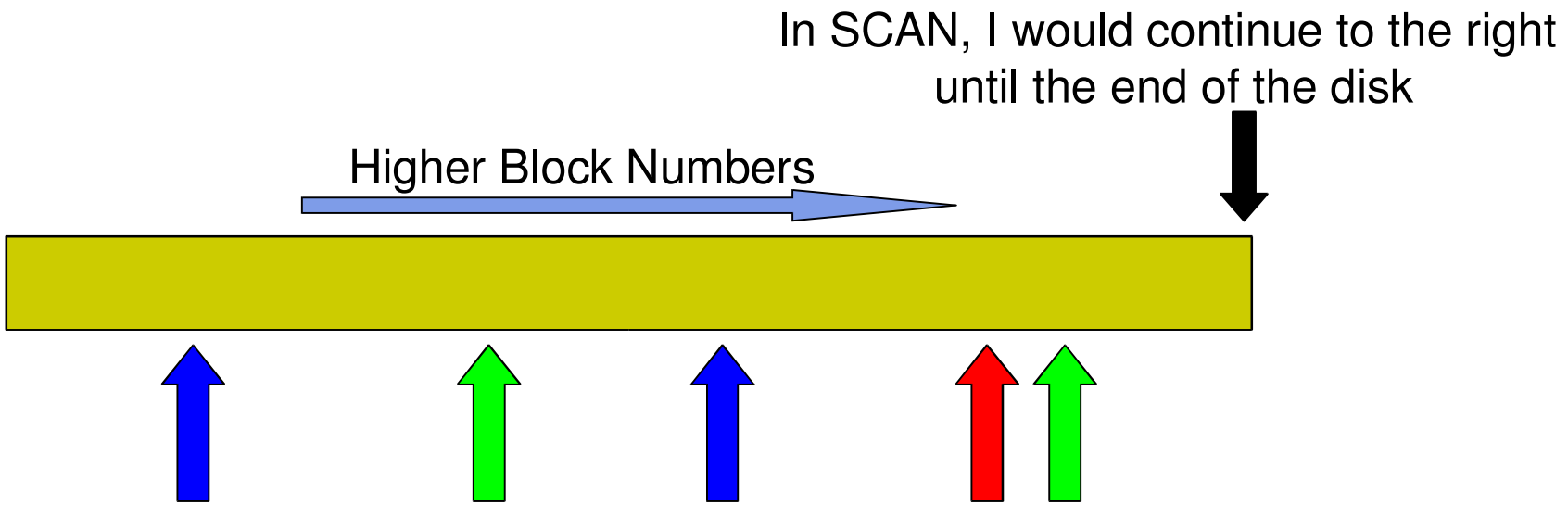
C-LOOK



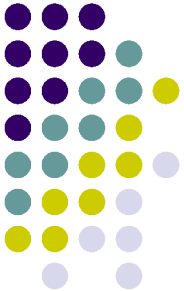


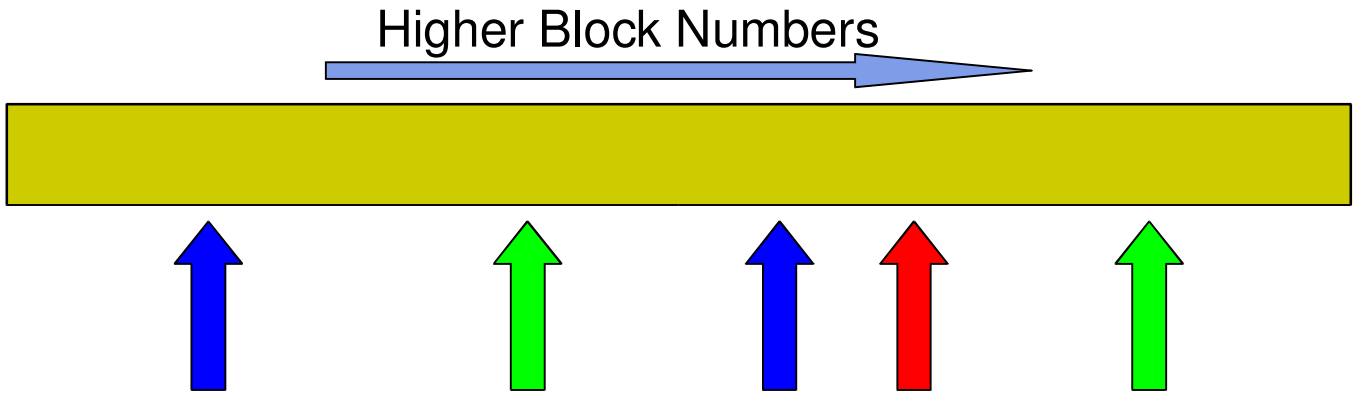
C-LOOK



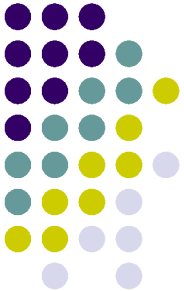

In SCAN, I would continue to the right
until the end of the disk
Higher Block Numbers

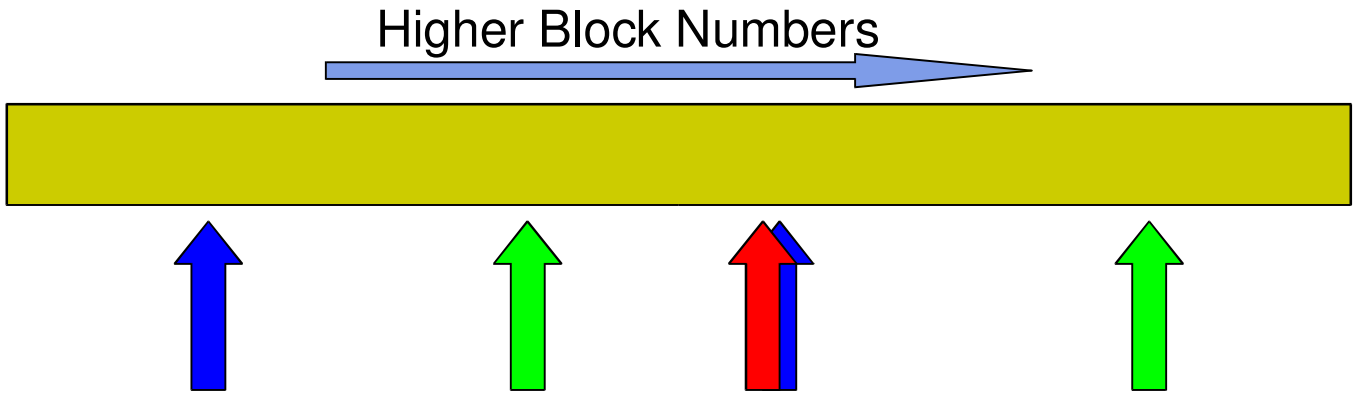
C-LOOK





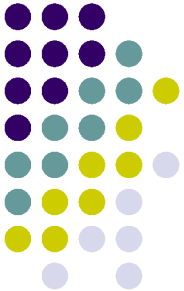
C-LOOK

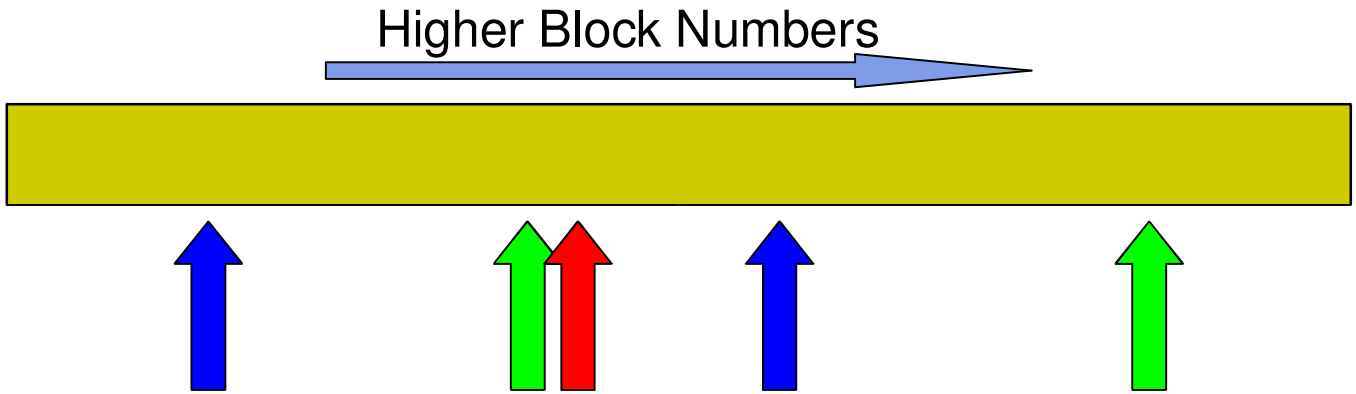




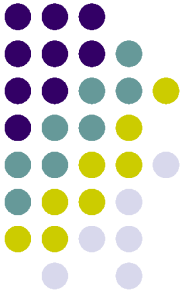
In LOOK, we would have read this request

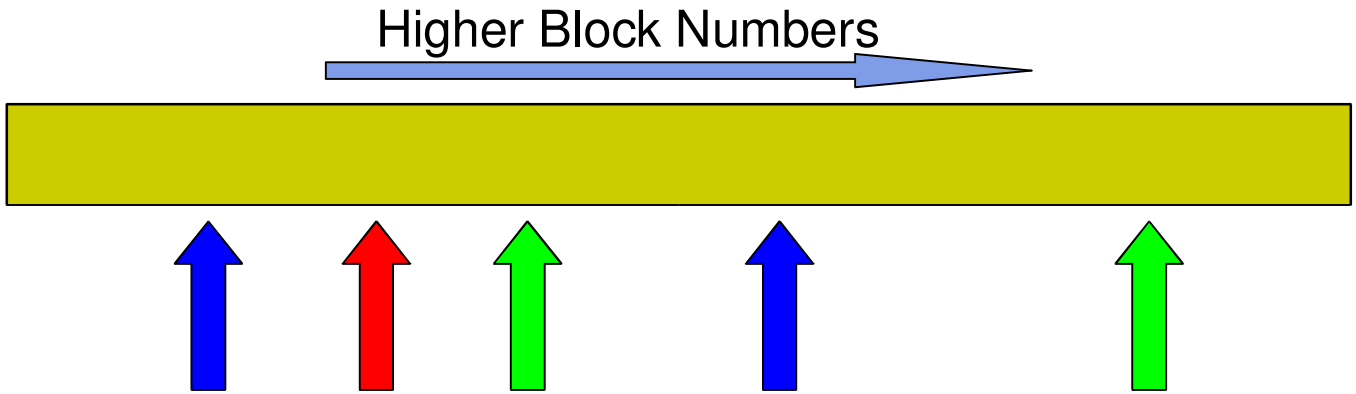
C-LOOK



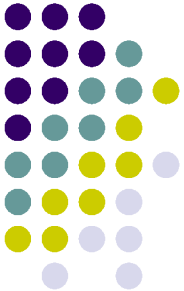


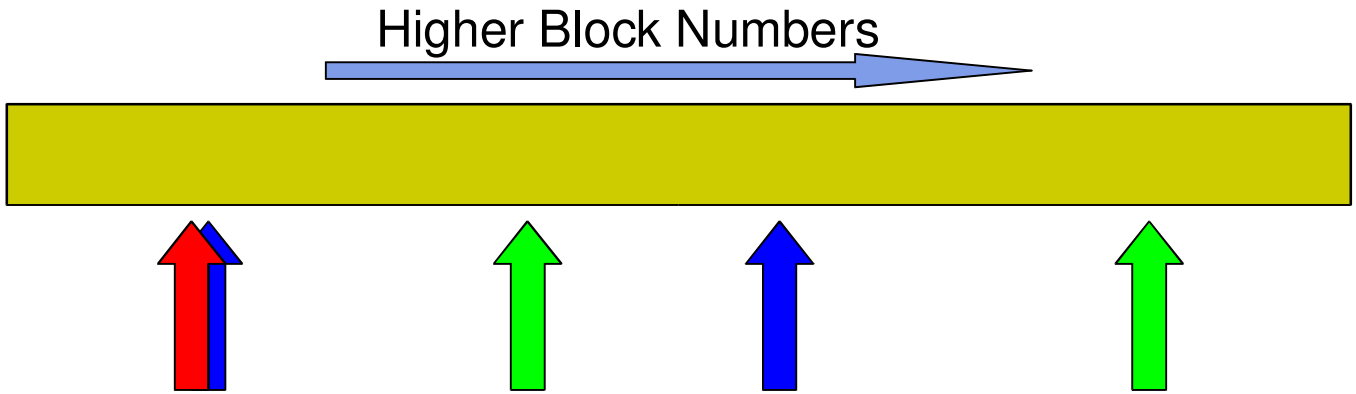
C-LOOK



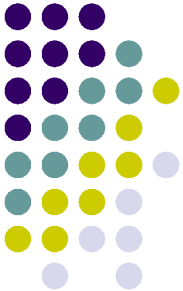


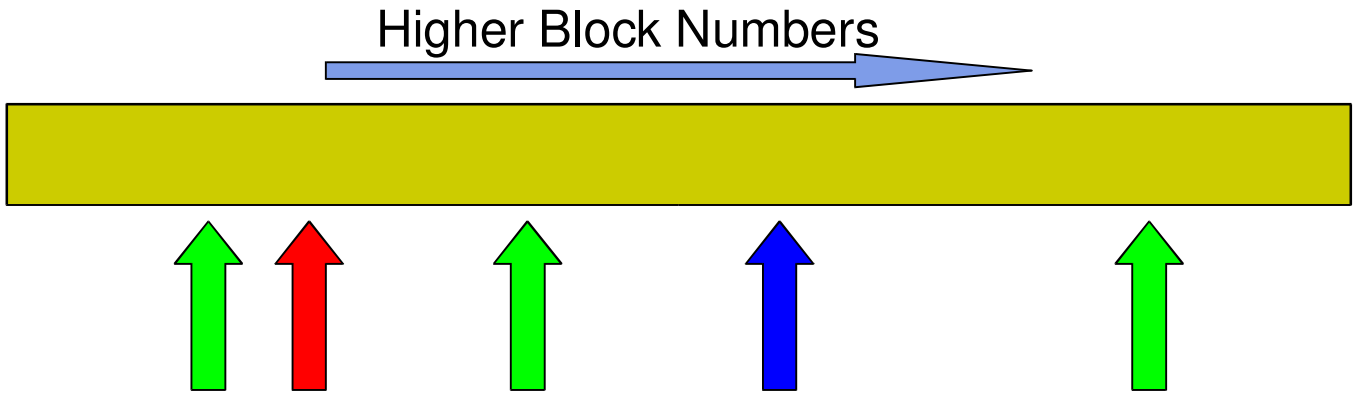
C-LOOK



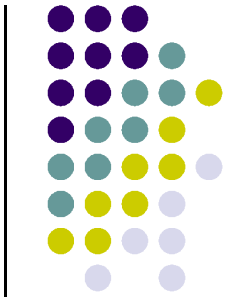


C-LOOK



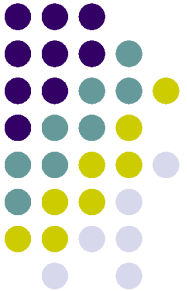


Shortest Seek Time First (SSTF)



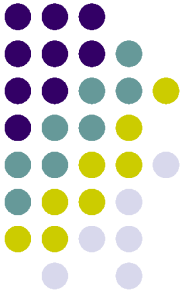
- Always send the request with the shortest seek time from current head position
- Generates very fast response time
- Intolerable response time variance, however
- Why?

Shortest Positioning Time First (SPTF)



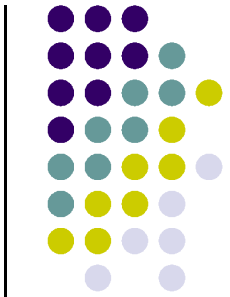
- Similar to Shortest Seek Time First
- Always select request with shortest total positioning time (rotational latency + seek time)
- More accurate greedy algorithm than SSTF
- Same starvation problems

Weighted Shortest Positioning Time First (WSPTF)



- SPTF, but we age requests to prevent starvation
- Aging policy is very flexible
- Excellent performance
- Why don't we use this?

Conclusions



- Disks are complicated
- Disks are very slow