Week 8: Storage Management

L8.1: *Algorithms & Complexity Analysis*

Algorithms and Programs

Algorithm

- A sequence of unambiguous instructions for solving a problem, i.e. a sequence of steps that can be followed to produce a result
- An algorithm is a *recipe* for solving a problem.
- Example: *If you want to make a cake, you need a recipe. The recipe is the algorithm. The cake is the result.*
- An algorithm must terminate.

Program

- A computer program is an algorithm that has been coded into some programming language. It is a sequence of steps that can be followed to produce a result.
- A program may terminate or run forever (e.g. a web server)

Analysis of Algorithms

Why ?

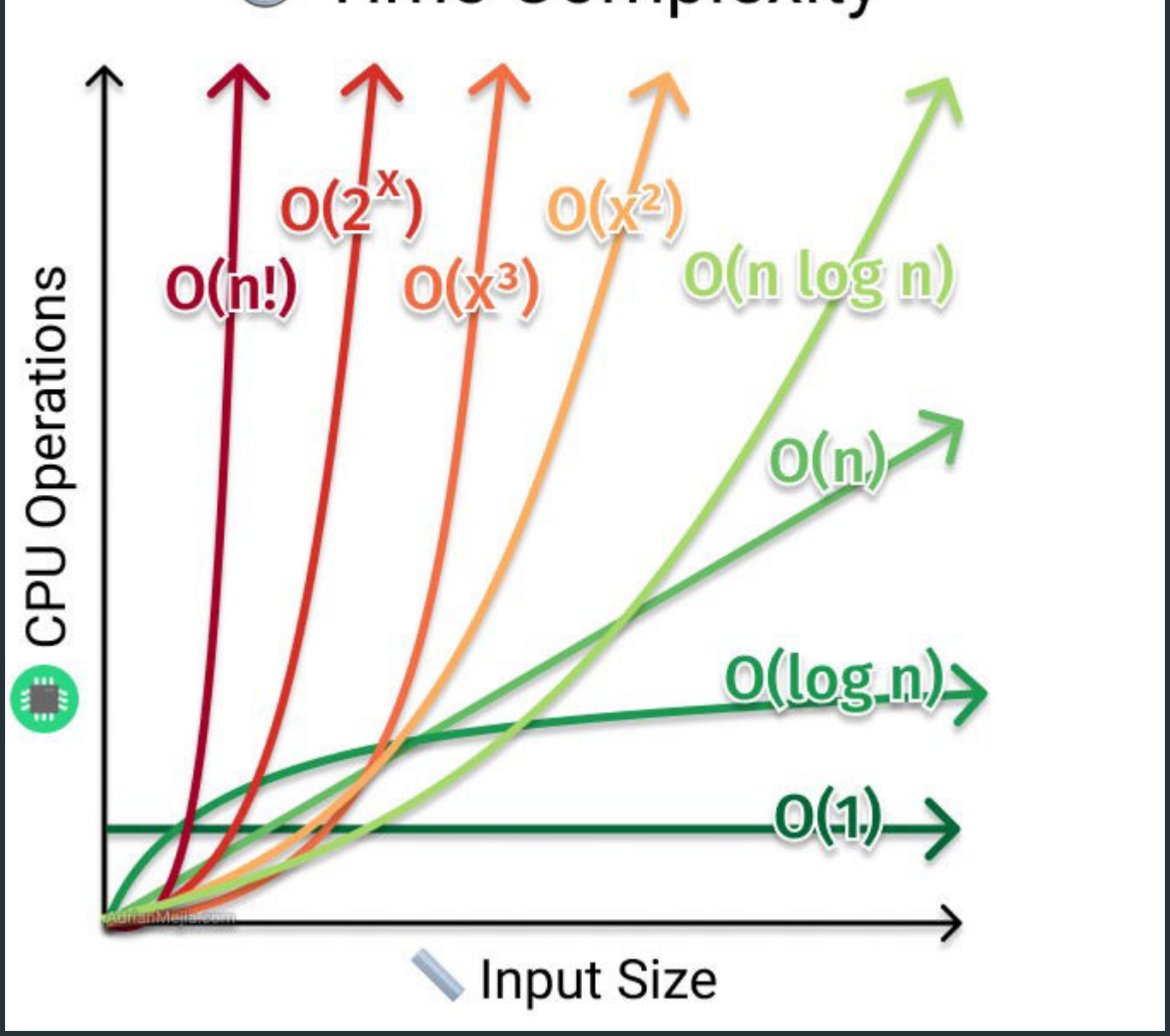
- **Predict performance** of an algorithm
- **Compare** two algorithms for the same problem
- **Understand theoretical basis** for solving problems

What to analyze ?

- **Time** - How long does it take for an algorithm to complete?
- **Space** - How much extra memory does it need?

How to analyze ?

- ***Counting models***
  - `Total running time = Sum of cost x frequency for all operations`
- ***Asymptotic analysis***
  - Cannot compare actual times, hence compare *Growth* or how time increases with input size
  - **Big-O notation** - upper bound on the growth of an algorithm
  - **Big-Omega notation** - lower bound on the growth of an algorithm
  - **Big-Theta notation** - both upper and lower bound on the growth of an algorithm



- ***Generating functions***
- ***Master Theorem***

Where to Analyze ?

- **Worst case** - Maximum number of steps taken on any instance of size n
- **Average case** - Average number of steps taken on any instance of size n
- **Best case** - Minimum number of steps taken on any instance of size n

---

L8.2 & 8.3: *Data Structures*

Data Structure

- A data structure specifies the way of organizing and storing in memory data that enables efficient access and modification of the data.
  - **Linear Data structures**
  - **Non-Linear Data structures**
- For applications related to data management, the key operations are:

- **Create**
- **Insert**
- **Delete**
- **Find / Search**
- **Close**

Linear Data Structures

- A linear data structure is a data structure where data elements are arranged sequentially or linearly, i.e. each element (data) is connected to it's previous and next element.
- Examples:
  - **Arrays**
    - The data elements are stored at contiguous memory locations.
  - **Linked Lists**
    - The data elements are linked using pointers.
  - **Stacks**
    - It follows LIFO (Last In First Out) principle, i.e. the element inserted at the last is the first element to come out.
  - **Queues**
    - It follows FIFO (First In First Out) principle, i.e. the element inserted at the first is the first element to come out.

> For more understanding of linear data structures, check this
> You can learn about searching techniques in linear data structures here

	Array		**Linked List**	
	Unordered	Ordered	Unordered	Ordered
Access	O(1)	O(1)	O(n)	O(n)
Insert	O(n)	O(n)	O(1)	O(1)
Delete	O(n)	O(n)	O(1)	O(1)
Search	O(n)	O(logn)	O(n)	O(n)

- Stacks and Queues are abstract data structures. They can be implemented using arrays or linked lists.
- So, the space complexity of linear data structures is O(n).
- All of them having complexities that are identical for Worst case as well as Average case.

Non-linear data structures can be used to trade-off between extremes and achieve a balance good performance for all.

Non-Linear Data Structures

- A non-linear data structure is a data structure in which data items are not arranged in a sequence and each

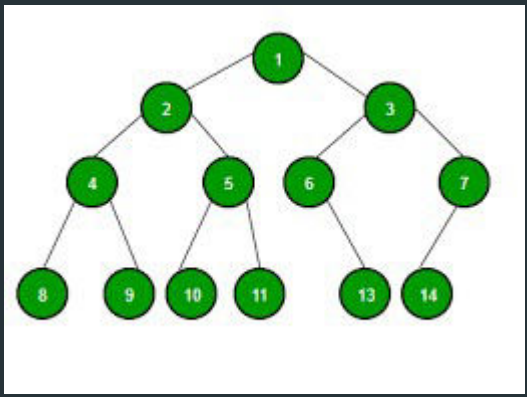
element may have multiple paths to connect to other elements.

- Unlike linear data structures, where an element is at most connected to 2 elements, non-linear data structures can be connected to any number of elements.
- Traversing is not possible in single run as elements are not arranged in sequential manner.
- Example:
  - **Graph**
    - A graph is a non-linear data structure consisting of nodes and edges.
    - The edges are lines or arcs that connect any two nodes in the graph.
    - They can be directed or undirected, and can be weighted or unweighted.
  - **Tree**
    - A tree is a non-linear data structure in which data elements are organized in hierarchical structure. The topmost node is called root of the tree.
    - The elements that are directly under an element are called its children.
    - The element directly above something is called its parent. Finally, elements with no children are called leaves.
    - They can be rooted or unrooted, binary or n-arry, balanced or unbalanced, etc.
  - **Hash Table**
    - In a hash table we store data in linked list which are accessed by a key which refers to the index of the array where the linked list is stored.
    - Each element in an array is a linked list.
  - and there are many more...

> For more understanding of Graph, check this

Tree

- Tree is a connected acyclic graph representing hierarchical structure.



- **Root node**:
  - The topmost node in a tree is called the root node.
  - There is only one root node in a tree.
- **Parent node**:
  - A node that has sub-nodes connected to it is called a parent node.
  - A node can be a parent node to multiple nodes.
  - A node can be a parent node as well as a child node.
- **Child node**:

    - A node that is connected to a parent node is called a child node.
    - A node can be a child node to multiple nodes.
    - A node can be a parent node as well as a child node.
- **Leaf node**:
    - A node that does not have any child node is called a leaf node.
    - A leaf node is also called an external node.
- **Internal node**:
    - A node that has at least one child node is called an internal node.
- **Subtree**:
    - A subtree is a tree that is part of a larger tree.
- **Path**:
    - A path is a sequence of nodes connected by edges of a tree.
- **Sibling**:
    - Nodes that have the same parent node are called siblings.
- **Arity**:
    - Number of children of a node is called its arity.
- **Levels**:
    - Root node has level 0, its children have level 1, and so on.
- **Height**:
    - Maximum level of any node in a tree is called its height.
- **Binary tree**:
    - A tree which arity 2, i.e. each node has at most 2 children.

A tree maybe:

- rooted or unrooted
- Binary or n-ary
- Balanced or unbalanced
- Disconnected (forest) or connected

Examples:

- Composite attributes
- Family Genealogy
- Search trees

> **Facts:**
>
> 1. A tree with n nodes has $n - 1$ edges.
> 2. The maximum number of nodes at level l of a binary tree is 2^l.
> 3. If h is the height of a binary tree with n nodes, then:
> $h + 1 \leq n \leq 2^{h+1} - 1$
> 4. For a k-ary tree with n nodes:
> $log_k(n) \leq h \leq n$

Hash Table

- A hash table is a data structure that maps keys to values for highly efficient lookup.
- It elements an associative array abstract data type, a structure that can map keys to values by using a hash function to compute an index into an array of buckets or slots, from which the desired value can be found.

A hash table may be using:

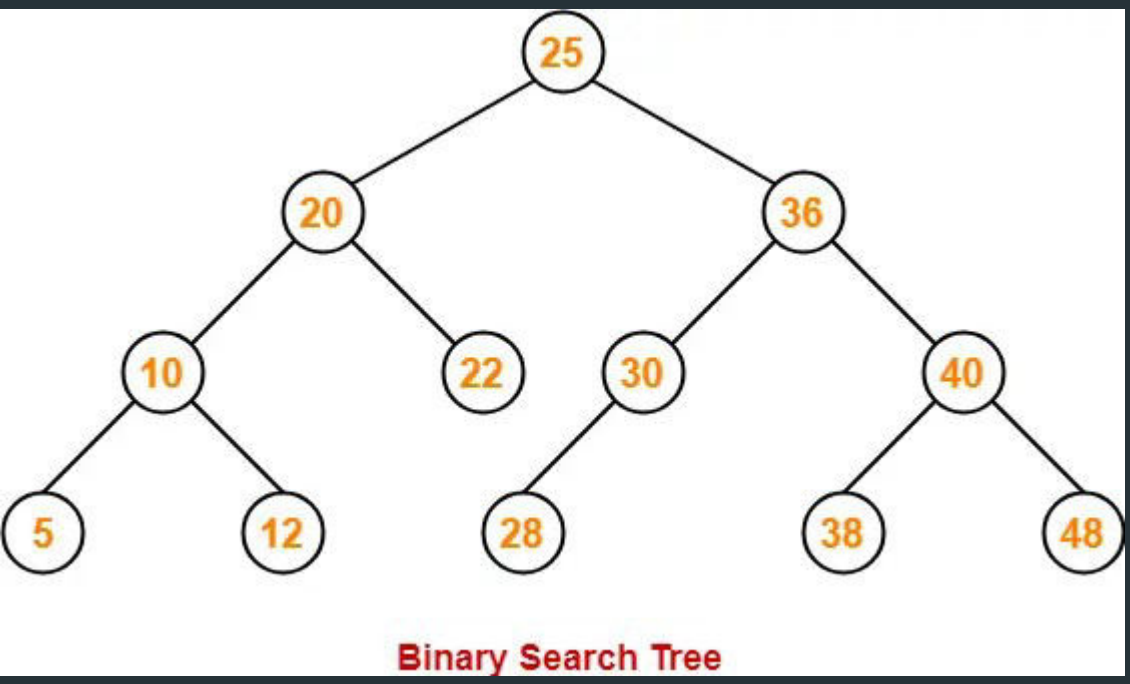
- Static or dynamic schemes
- open addressing
- 2-choice hashing

Examples:

- Associative arrays
- Database indexing
- Caches

Binary Search and Binary Search Tree

- Binary search is efficient in search of a key in a sorted array, i.e. O(log n).
- As the binary search splits the array, we can conceptually consider the Middle element to be the root of a tree and the left and right sub-arrays to be the left and right sub-trees and progress recursively, we get a Binary Search Tree.
- So, A Binary Search Tree is a binary tree in which the root node is greater than all the nodes in the left sub-tree and less than all the nodes in the right sub-tree.



Binary Search Tree

Searching in a Binary Search Tree

- Searching in a Binary Search Tree takes O(h), where h is the height of the tree.
- **Worst Case**: If the BST is skewed, then time complexity is O(n), where n is the number of nodes.
- **Best Case**: If the BST is balanced, then time complexity is O(log n), height (h) becomes log n.

Complexities of Various Data Structures

	Data Structure	Time Complexity								Space Complexity
		Average				Worst				Worst
		Access	Search	Insertion	Deletion	Access	Search	Insertion	Deletion	
Linear Data Structures	Array	Θ(1)	Θ(n)	Θ(n)	Θ(n)	O(1)	O(n)	O(n)	O(n)	O(n)
	Stack	Θ(n)	Θ(n)	Θ(1)	Θ(1)	O(n)	O(n)	O(1)	O(1)	O(n)
	Queue	Θ(n)	Θ(n)	Θ(1)	Θ(1)	O(n)	O(n)	O(1)	O(1)	O(n)
	Singly-Linked List	Θ(n)	Θ(n)	Θ(1)	Θ(1)	O(n)	O(n)	O(1)	O(1)	O(n)
	Doubly-Linked List	Θ(n)	Θ(n)	Θ(1)	Θ(1)	O(n)	O(n)	O(1)	O(1)	O(n)
Non-Linear Data Structures	Skip List	Θ(log(n))	Θ(log(n))	Θ(log(n))	Θ(log(n))	O(n)	O(n)	O(n)	O(n)	O(n log(n))
	Hash Table	N/A	Θ(1)	Θ(1)	Θ(1)	N/A	O(n)	O(n)	O(n)	O(n)
	Binary Search Tree	Θ(log(n))	Θ(log(n))	Θ(log(n))	Θ(log(n))	O(n)	O(n)	O(n)	O(n)	O(n)
	Cartesian Tree	N/A	Θ(log(n))	Θ(log(n))	Θ(log(n))	N/A	O(n)	O(n)	O(n)	O(n)
	B-Tree	Θ(log(n))	Θ(log(n))	Θ(log(n))	Θ(log(n))	O(log(n))	O(log(n))	O(log(n))	O(log(n))	O(n)
	Red-Black Tree	Θ(log(n))	Θ(log(n))	Θ(log(n))	Θ(log(n))	O(log(n))	O(log(n))	O(log(n))	O(log(n))	O(n)
	Splay Tree	N/A	Θ(log(n))	Θ(log(n))	Θ(log(n))	N/A	O(log(n))	O(log(n))	O(log(n))	O(n)
	AVL Tree	Θ(log(n))	Θ(log(n))	Θ(log(n))	Θ(log(n))	O(log(n))	O(log(n))	O(log(n))	O(log(n))	O(n)
	KD Tree	Θ(log(n))	Θ(log(n))	Θ(log(n))	Θ(log(n))	O(n)	O(n)	O(n)	O(n)	O(n)

L8.4: *Physical Storage*

Physical Storage

Classification of Physical storage media

- **Speed** - How fast can data be accessed?
- **Cost** - How much does it cost?
- **Reliability** - Data loss on power failure or system crash, physical failure.
- **Volatile storage** - Data is lost when power is turned off.
- **Non-volatile storage** - Data is retained when power is turned off.
- **Cache** - Small, very fast, volatile and most costly memory managed by hardware.
- **Main memory** - Larger, fast, volatile memory managed by hardware.

Flash Memory

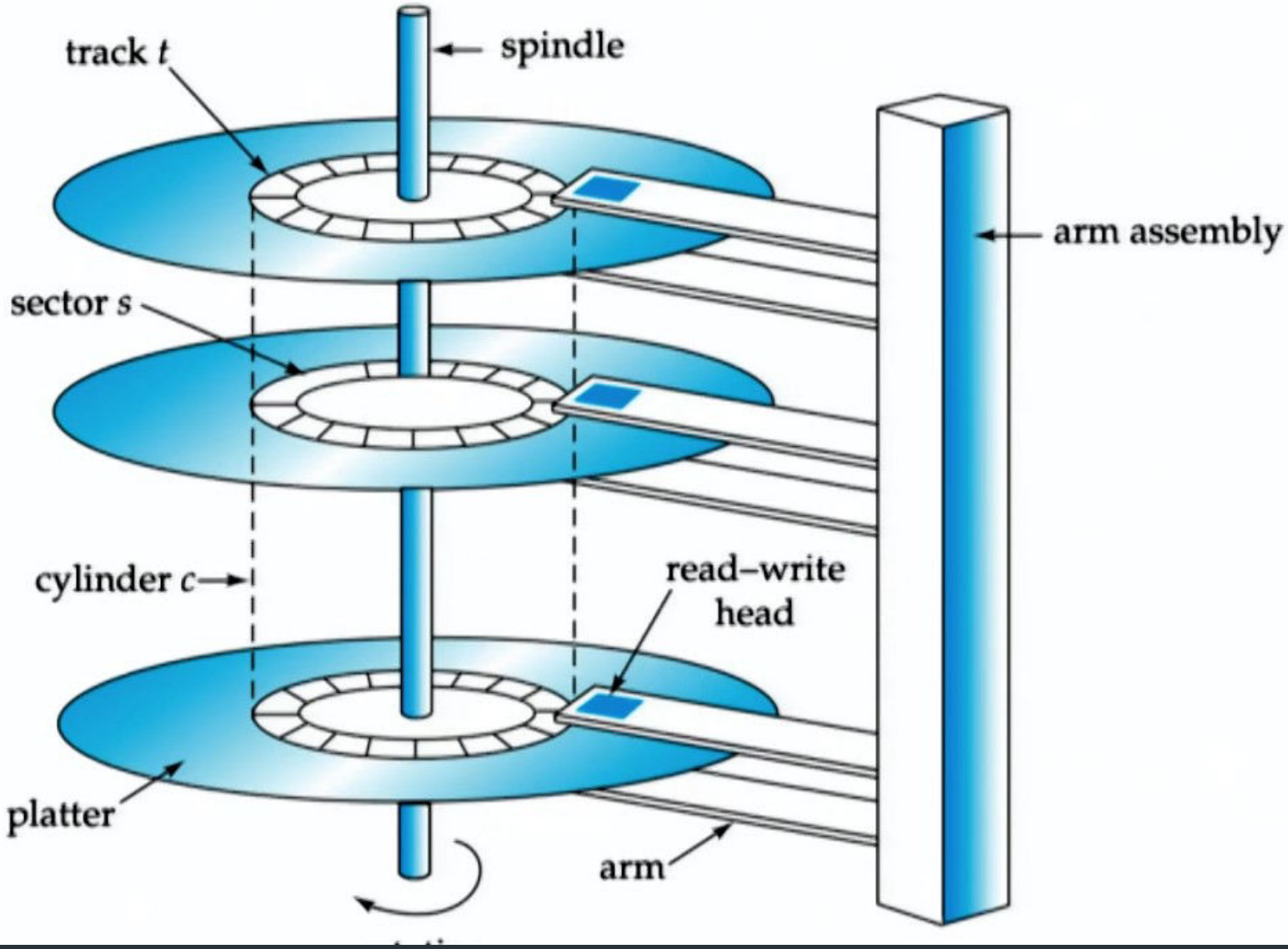
- Flash memory is a non-volatile storage technology that retains data even when power is removed.
- It's commonly used in portable devices, memory cards, and USB drives.
- *Advantages*: Fast read speeds, low power consumption, compact size, durability, and resistance to shocks.
- *Disadvantages*: Limited write endurance (limited number of write cycles per cell).
- Moderately priced, highly reliable for read operations, read speeds are fast, while write speeds are slower.

Magnetic Disk

- Data is stored on spinning disk and read/written magnetically.

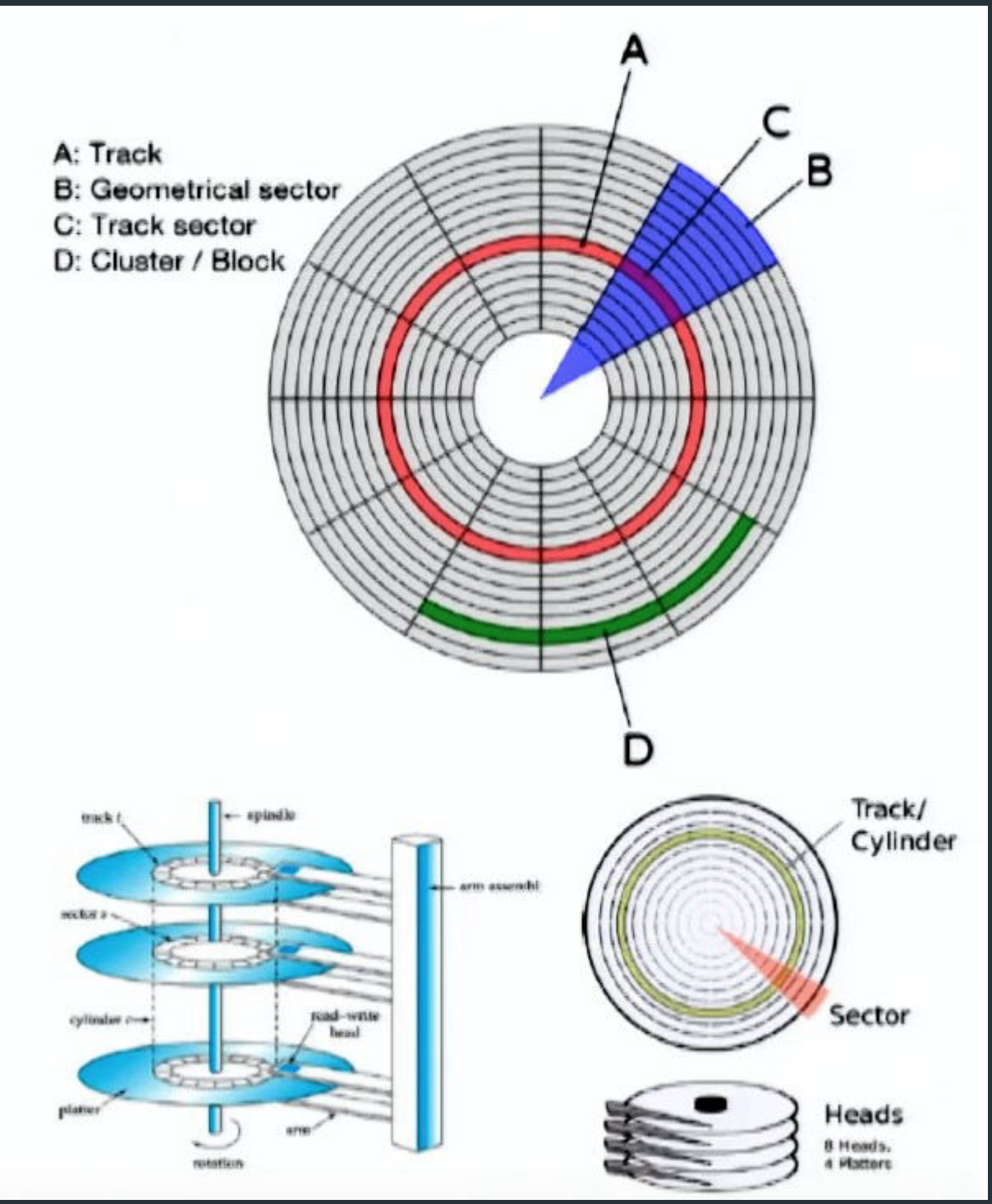
- Primary medium for long-term storage.
- Data must be moved from disk to main memory for access and written ○ack for storage, much slower than main memory.
- **Direct access**: Possible to read data on disk in any order.

Mechanism



- Read-write head
  - Positioned very close to the platter surface
  - Reads or write magnetically encoded information.
- Surface of platter divided into circular **tracks**
  - Over 50k-100K tracks per platter on typical hard disks
- Each track is divided into **sectors**
  - A sector is the smallest unit of data that can be read or written
  - Sector size is typically 512 bytes
- To read/write a sector
  - disk arm swings to position head on right track
  - platter spins: read/write as sector passed under head
- Head-disk assemblies

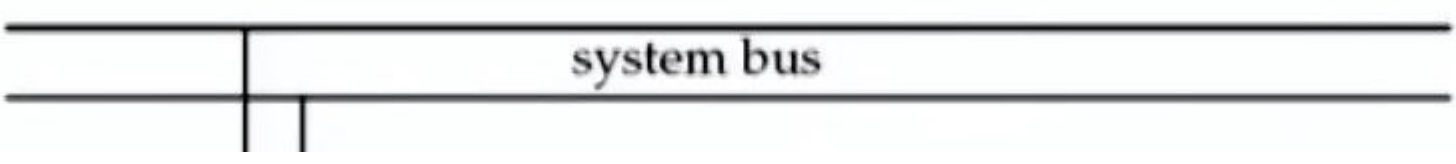
- multiple disk platters on a single spindle (1 to 5 usually)
- one head per platter, mounted on a common arm.

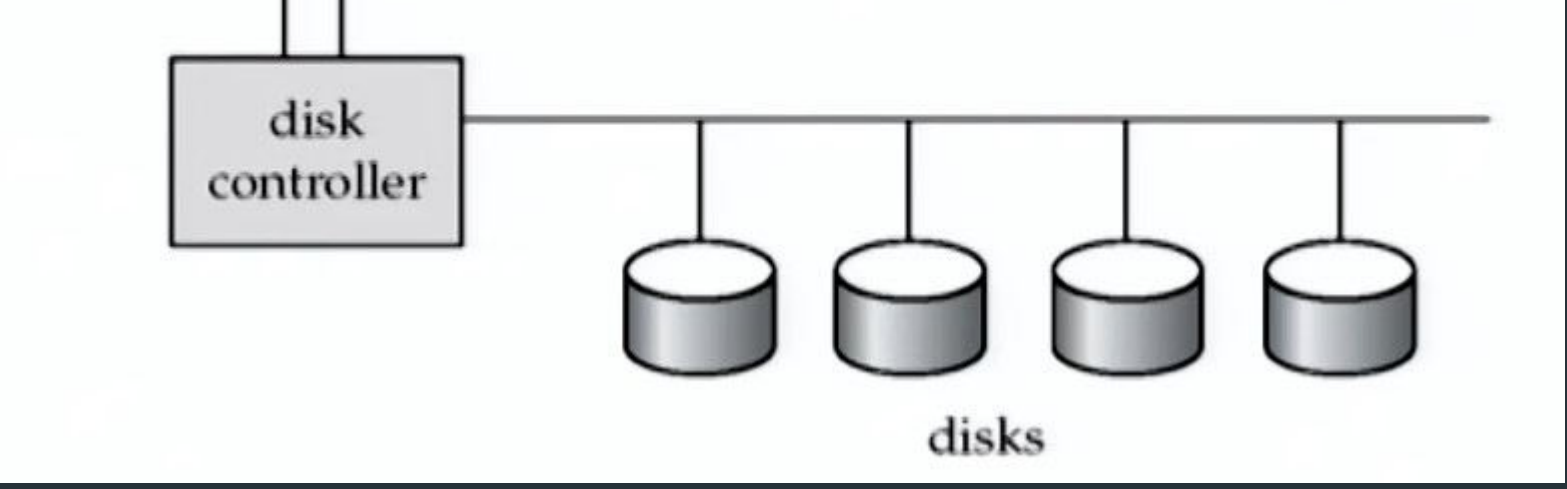


Dis controller

- Interface between the computer system and the disk drive hardware
- Accepts high-level commands to read or write a sector
- Computes and attaches checksums to each sector to verify that correct read back
- Perform remapping of bad sectors

Disk Subsystem





- **Disk Interface Standards Families**: ATA, SATA, SCSI, SAS
- **Storage Area Networks (SAN)** connects disks by a high-speed network to number of servers.
- **Network Attached Storage (NAS)** provides a file system interface using networked file system protocol

Performance Measures

- **Access Time**: Time from a read or write request issue to start of data transfer.
- **Seek Time**: Time to move the disk arm to the desired track.
- **Rotational Latency**: Time for the desired sector to rotate under the disk head.
- **Data transfer Rate**: Rate at which data is transferred to or from the disk.
- **Mean Time to Failure (MTTF)**: Average time the disk is expected to run continuously without any failure.

Optical Storage

- Non-volatile, data is read optically from a spinning disk using a laser.
- CD-ROM (640 MB) and DVD (4.7 to 17 GB) are most popular forms.
- Write once, read many (WORM) optical disks are used for archival storage.
- Reads and write slower than magnetic disks, but cheaper and more durable.

Tape Storage

- Non-volatile, used primarily for backup and for achival data.
- Sequential access, very slow, but very cheap and durable.

Flash Drives

- Flash drives, or USB drives, are portable storage devices using flash memory. They connect via USB ports and come in various sizes.
- *Advantages*: Compact, lightweight, no moving parts, high data transfer rates, and plug-and-play convenience.
- *Disadvantages*: Limited write cycles, smaller capacities compared to traditional HDDs.
- Affordable, reliable for read operations, high read speeds, moderate write speeds.

Flash Storage

- Flash storage refers to non-volatile memory technology used to store data electronically without the need

for power.

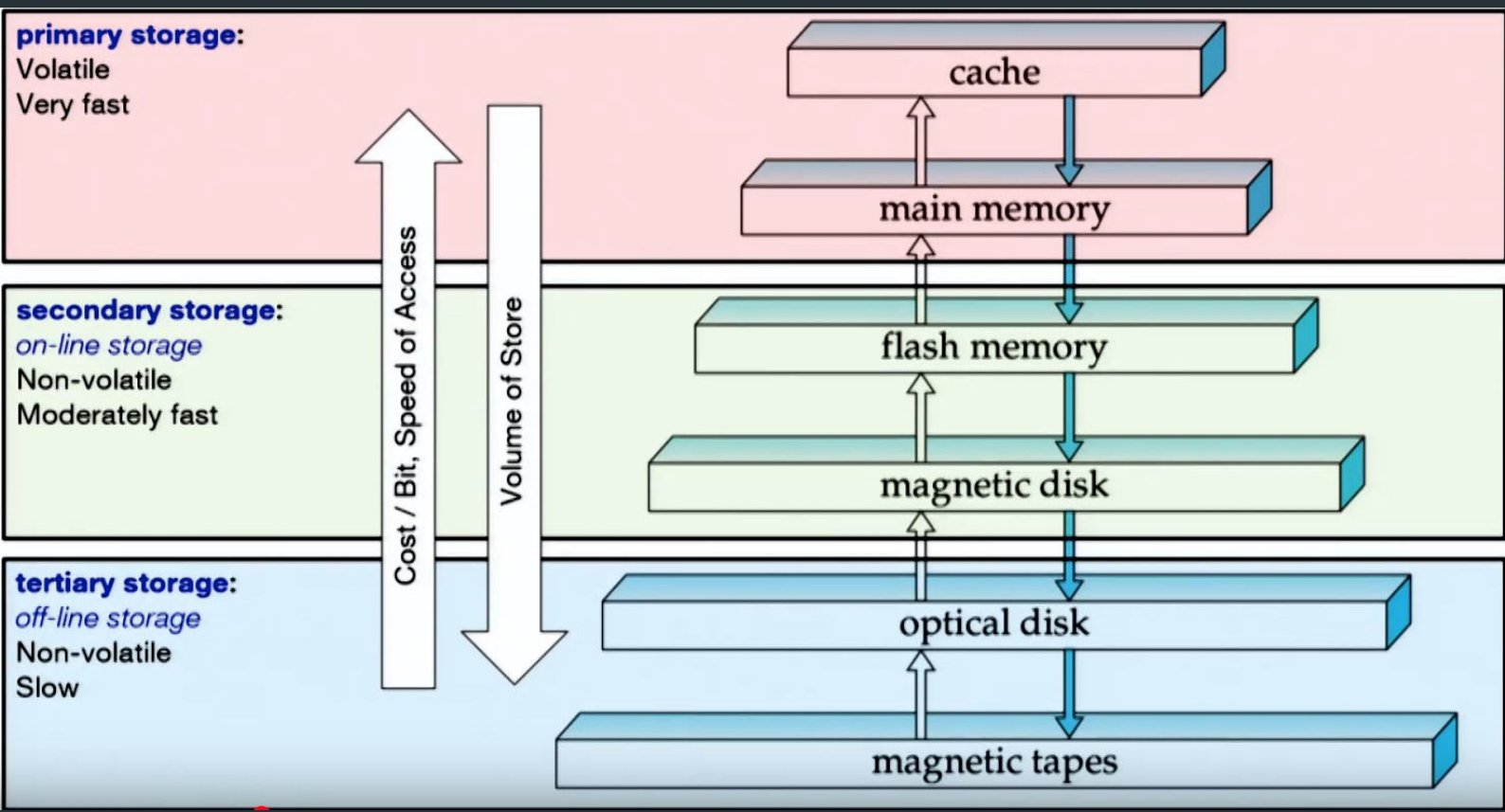
- *Advantages*: Offers high-speed data access, low power consumption, and resistance to physical shocks due to no moving parts.
- *Disadvantages*: Limited write endurance, higher cost compared to traditional HDDs for larger capacities.
- Can be more expensive than HDDs but cost-effective for performance gains, reliable for read operations, and provides faster data access than HDDs.

SSD

- SSDs are storage devices that use flash memory to provide high-speed, non-volatile data storage.
- *Advantages*: Lightning-fast data access, low power usage, silent operation, and longer lifespan than HDDs.
- *Disadvantages*: Generally more expensive, write endurance concerns (though improving), and cost per gigabyte is higher.
- Higher initial cost but becoming more affordable, reliable for both read and write operations, significantly faster read/write speeds than HDDs.

HDD

- HDDs use spinning disks to read/write data magnetically, offering long-term storage solutions.
- *Advantages*: Cost-effective for large storage capacities, improving technology for faster performance, and stability for long-term data storage.
- *Disadvantages*: Slower access times compared to SSDs, mechanical parts can be prone to damage, and consumes more power.
- Generally cheaper per gigabyte, reliable for read operations, slower read/write speeds compared to SSDs.



Cloud Storage

- Cloud storage is a service model in which data is maintained, managed, backed up remotely and made available to users over a network (typically the Internet).
- Applications access cloud storage through traditional storage protocols or directly via an API.
- Example of cloud storage providers: Amazon S3, Google Cloud Storage, Microsoft Azure Storage, Dropbox, etc.

L8.5: *File Structure* 🔗

- You can refer to this video lecture 🔗