OS TOPICES

Q1. What is an Operating System?

Answer: An **Operating System** is an interface between computer hardware and computer users. It performs tasks like memory management, file management, process management, controlling peripheral devices such as disk drives and printers, and handling input and output. Example: **Linux Operating System**, **Windows Operating System**, etc.

Q2. List some of the standard services by OS?

Answer: Some of the standard services by OS are :

- **User interface**
- **Program execution**
- **I/O operations**
- **File-system manipulation**
- **Communications**
- **Error detection**
- **Resource allocation**
- **Accounting**
- **Protection and security**

Q3. What is the operating system's primary purpose?

Answer: There are two primary purposes of an operating system(OS):

- It is designed to make sure that a computer system performs well by managing its computational activities.
- It provides an environment for the development and execution of programs.

Q5. What are the different operating systems?

- **Batched operating systems**
- **Distributed operating systems**
- **Time-Sharing operating systems**
- **Multiprogrammed operating systems.**
- **Real-time operating systems**

Q6. What is a socket in OS?

Answer: A **socket** is a software construct that allows **two applications to communicate** over a network. It is **one endpoint** of a two-way communication link. The **other endpoint** is another socket. The two sockets are identified by their **IP addresses and port numbers**. When two sockets connect, they create a **socket connection.** This connection allows the two applications **to exchange data**.

Q7. Explain various activities/ functions of OS?

ANS:

Memory Management : It refers to the management of Main Memory or Primary Memory, an extensive array of words, each having its address. Main memory provides fast storage that can be accessed directly by the CPU.

Some of its activities are:

- It keeps track of the main memory, i.e., what part is in use by whom and what part is not in use.
- When a process requests, it allocates the memory to it.
- When a process no longer needs the memory, it **de-allocates** the memory.

Processor Management : It helps **the OS to decide** which process gets the processor when and for how much time.

Some of its activities are:

- It keeps track of the processor and status of the process.
- When a process requests, it allocates the processor (CPU) to it.
- When a process no longer needs the processor, it de-allocates it.

Device Management : An OS manages device communication via their respective drivers.

Some of its activities are:

- It keeps track of all devices.
- Efficiently allocates the device.
- De-allocates devices.

File Management : A file system is usually organized into directories for easy navigation and usage. These directories may contain files and other directories.

Some of its activities are:

- It keeps **track of location, information, uses, status**, etc. The collective facilities are often known as the file system.
- Decides who **gets the** resources.
- Allocates the resources.
- De-allocates the resources.

Security – It prevents unauthorized access to programs and data.

Job accounting – It keeps track of resources and time used by various users and jobs.

Error detecting aids – It helps in detecting the production of dumps, traces, error messages, and other debugging and error detecting aids.

Coordination between other software and users – It coordinates and assigns compilers, assemblers, interpreters, and other software to the multiple users of the computer systems.

Q8.What is the Batch Operating System?
Answer:

- ✓ It doesn't interact with the computer directly. There is an operator which takes similar jobs having the exact requirement and groups them into batches.
- ✓ It is the responsibility of the operator to sort the jobs with similar needs.

Q9. What are the benefits and disadvantages of a Batch Operating System?
Answer: Advantages of Batch Operating System:

- ➢ Multiple users can share the **batch systems**
- ➢ Batch system's idle time is significantly less
- ➢ It is easy to manage extensive work repeatedly in batch systems

Disadvantages of Batch Operating System:

- Batch systems are hard to **debug**
- If any job fails, the other jobs will have to wait for an unknown time.

- **Q10. What are Time-Sharing Operating Systems?**
  Answer:
- **In a time-sharing Operating System**, each user gets the CPU time as they use a single system, and each task is given some time to execute so that all the tasks work smoothly. Hence, this system is also known as Multitasking Systems.
- The time that each task gets to execute is called quantum. After this time interval is over, the OS switches over to the next task.

Q11. Explain the difference between multitasking and multiprogramming.

Answer: Multitasking refers to the ability of an OS to execute multiple tasks simultaneously by rapidly switching between them. Multiprogramming involves keeping multiple programs in memory at the same time to improve CPU utilization.

Q12. What is a process in the context of an OS?

Answer: A process is an independent program in execution. It consists of both program code and the resources (like memory and CPU registers) needed for its execution.

Q13. What is virtual memory, and why is it used?

Answer: Virtual memory is a memory management technique that uses a combination of RAM and disk space to provide the illusion of a larger memory than physically available. It allows running larger programs and enhances multitasking.

Q14. Explain the concept of a file system.

Answer: A file system is a method for storing, organizing, and retrieving files on a storage device. It provides a hierarchical structure for **files and directories** and offers mechanisms for file manipulation and access control.

Q15. What is a shell in an operating system?

Answer: A **shell** is a **command-line interface** (CLI) or **graphical user interface** (GUI) that allows users to interact with the operating system by entering commands or using a graphical interface to perform tasks.

Q16. What is a device driver, and why is it important in an OS?

Answer: A **device driver** is software that facilitates communication between the operating system and hardware devices like printers, keyboards, and graphics cards. It is essential for proper hardware functionality and system stability.

Q17. What is the difference between preemptive and non-preemptive scheduling in process management?

Answer: Preemptive scheduling allows the OS to **interrupt a running process** and allocate the CPU to another process with higher priority. Non-preemptive scheduling only switches processes when the running process voluntarily gives up the CPU.

Q18. What is the role of the BIOS (Basic Input/Output System) in computer booting?

Answer: The BIOS is responsible for initializing hardware components during the boot process, performing a POST (Power-On Self-Test), and locating and loading the bootloader, which then loads the operating system.

Q19. What is a deadlock in operating systems? How can it be prevented or resolved?

Answer: A **deadlock** is a situation in which two or more processes are unable to proceed because they are each waiting for the other to release a resource. Deadlocks can be prevented using techniques like resource allocation graphs or resolved using methods like deadlock detection and recovery.

Q20. Explain the concept of a page fault.

Answer: A **page fault** occurs when a program tries to access a **portion of memory** that is **currently not in physical RAM** but instead **stored on disk**. The OS then retrieves the **required page** from disk into RAM, allowing the program to continue execution.

Q21. What is a system call, and why is it important for applications?

Answer: A system call is a programmatic way for applications to request services from the operating system, such as file operations, process control, or hardware access. It provides a secure and controlled interface to the OS kernel.

Q22. What is a real-time operating system (RTOS), and where is it commonly used?

Answer: A **real-time operating system** is designed to provide predictable and guaranteed response times for critical tasks. It is commonly used in **embedded systems, industrial automation, robotics, and other applications** where timing is crucial.

Q23. What is a GUI (Graphical User Interface), and how does it differ from a CLI (Command-Line Interface)?

Answer: A GUI is a user interface that uses graphical elements like windows, icons, and menus to interact with the operating system, while a CLI relies on text-based commands and requires users to type commands.

Q24. What is a process, and what information does the process control block (PCB) contain?

Formal Definition: A process is a program that's **running on a computer**. The process control block (PCB) is like a data paper that keeps track of important information about the process, such as where it's running, what it's doing, and other details to manage it.

Q25. Explain the differences between a process and a thread.

Formal Definition: A **process** is an **independent program** that runs on a computer. Threads are smaller parts of a process that can perform tasks independently, like running different parts of the program.

Q26. What is a context switch, and why is it necessary?

- **Simple Definition:** A context switch is like pausing one game to play another. It's needed so the computer can do different jobs quickly.
- **Formal Definition:** A context switch is the action of the computer switching from one task to another. It's necessary to efficiently **manage different tasks** and make sure they all get a turn to run.

Q27. Describe process synchronization and its importance in operating systems.

- **Simple Definition:** Process synchronization is like taking turns with toys. It's important so computer programs don't fight over stuff.
- **Formal Definition:** Process synchronization is a way for computer programs or parts of programs to work together without causing conflicts. It's crucial in operating systems to ensure that tasks run smoothly and don't interfere with each other.

Q28. What is a deadlock, and how can it be prevented or resolved?

- **Simple Definition:** A deadlock is when two friends hold onto the same toy and won't let go. To fix it, you need someone to help or ask nicely.
- **Formal Definition:** A deadlock is a situation where multiple processes or threads are stuck because they're waiting for each other to release something. It can be prevented with careful design and resolved by outside intervention or by making one of the processes let go.

Q29. Explain the concept of inter-process communication (IPC).

- **Simple Definition:** IPC is like talking to your friend through toy walkie-talkies. You share secrets and help each other out.
- **Formal Definition:** Inter-process communication (IPC) is a method for different computer programs or processes to communicate with each other. It allows them to share information, work together, and coordinate their actions, much like using walkie-talkies to talk and collaborate.

Summary for the Interviewer In short :

Processes are like recipes for tasks, and the PCB is the list of instructions. Threads are like smaller tasks within a big task. Context switching is like changing toys to work on different tasks. Process synchronization is like sharing toys or taking turns to play nicely. Deadlock is when tasks won't let go of something; it can be prevented and resolved with help. IPC is like talking on the phone between computer programs.

Memory Management:

Q30. What is virtual memory, and how does it work?

- **Simple Definition:** Virtual memory is like a magical backpack for a computer. It helps the computer use lots of space, even if it's not very big.
- **Formal Definition:** Virtual memory is a technique that lets a computer use more memory than it actually has. It does this by using a combination of RAM (real memory) and space on the computer's hard drive to store data that might not fit entirely in RAM.

Q31. Differentiate between paging and segmentation.

- **Simple Definition: Paging** is like putting different pieces of a puzzle into separate boxes. Segmentation is like dividing a big coloring book into sections.
- **Formal Definition:** Paging and segmentation are methods used in memory management. Paging divides memory into fixed-size blocks, while segmentation divides it into variable-sized sections for more flexible storage.

Q32. What is thrashing, and how can it be avoided?

- **Simple Definition:** Thrashing is when the computer is so busy moving things around it can't get any real work done. To avoid it, the computer needs to be given fewer tasks at once.
- **Formal Definition:** Thrashing happens when a computer spends more time swapping data between RAM and disk than actually processing tasks. It can be avoided by adjusting the number of tasks the computer is working on or by adding more memory.

Q33. How does the operating system manage memory allocation and deallocation?

- **Simple Definition:** The operating system is like a tidy room organizer. It decides where to put toys and when to put them away.
- **Formal Definition:** The operating system is responsible for allocating memory (finding space for new data) and deallocating memory (cleaning up space that's no longer needed) to make sure programs run smoothly.

Q34. Describe the role of page tables in virtual memory systems.

- **Simple Definition: Page tables** are like a map that helps you find your toys in your magical backpack. You look at the map to find what you need.
- **Formal Definition:** Page tables are data structures used in virtual memory systems. They keep track of which parts of virtual memory correspond to actual physical memory (RAM). When the computer needs to find something in virtual memory, it consults the page table to locate it.

Summary for the Interviewer:

Virtual memory is like a magical backpack for computers. It can use more memory than it has. Paging and segmentation are ways to organize memory. Thrashing is when the computer is too busy moving data. The operating system organizes and cleans up memory. Page tables are like maps for finding things in virtual memory.

File Systems:

Q35. What are the key components of a file system?

- **Simple Definition:** A **file system** is like a big library for a computer. It has **books (files**), **shelves (folders),** and a **librarian (the operating system)** who keeps everything organized.

- **Formal Definition:** A **file system** is a way for a computer to **organize and store** files and directories. Key components include files (the data), directories (like folders to keep files in), and the file management software provided by the **operating system**.

Q36. Explain the differences between FAT, NTFS, and ext4 file systems.

- **Simple Definition:** These are like different types of books (file systems) in the library. **FAT** is simple, **NTFS** is smart, and **ext4 is** super organized.

- **Formal Definition: FAT (File Allocation Table), NTFS (New Technology File System**), and **ext4** are different methods for **organizing and storing** data on a computer's storage devices. They have varying features and capabilities, with NTFS and ext4 being more advanced and capable than FAT.

Q37. How does the operating system handle file permissions and security?

- **Simple Definition:** The **operating system** is like a security guard. It decides who can enter the library and who can read certain books. It uses keys (permissions) to keep things safe.

- **Formal Definition:** The operating system controls access to files and folders by assigning permissions to users and groups. It determines who can read, write, or execute files, ensuring data security and privacy.

Q38. What is a file descriptor, and how does it relate to file I/O?

- **Simple Definition:** A **file descriptor** is like a library card for a book. You need it to borrow a book (read or write data).

- **Formal Definition:** A file descriptor is a unique number or identifier that the operating system assigns to an open file. It's used in file input and output operations (file I/O) to keep track of files and manipulate data.

Q39. Discuss the importance of file system consistency and recovery.

- **Simple Definition:** Consistency is like making sure **all the books are in the right place**, and recovery is like fixing things when they get messy.

- **Formal Definition:** File system consistency ensures that data is correctly stored and organized. Recovery mechanisms help restore the file system to a stable state if errors or crashes occur, preventing data loss.

Summary for the Interviewer:

A file system is like a library with books and folders, managed by the operating system. FAT, NTFS, and ext4 are different ways to organize data. The OS controls who can access files. File descriptors are like library cards for data access. Consistency keeps things organized, and recovery fixes problems.

Scheduling Algorithms:

Q40. Describe various CPU scheduling algorithms (e.g., FCFS, SJF, RR, Priority Scheduling).

- **Simple Definition:** These are like different ways to decide who gets to play with a toy next. Some wait in line (FCFS), some play a quick game (SJF), **some take turns** (RR), and some are like special guests (Priority).

- **Formal Definition:** CPU scheduling algorithms are methods used by the operating system to decide which process gets to use the computer's central processing unit (CPU) next. FCFS (First-Come, First-Served) serves processes in the order they arrive. SJF (Shortest Job First) prioritizes shorter tasks. RR (Round Robin) lets processes take turns equally. Priority Scheduling assigns importance to processes.

Q41. What are the advantages and disadvantages of each scheduling algorithm?

- **Simple Definition:** Some ways to pick who goes next are fair but slow (FCFS), some are fast but not always fair (SJF), some are balanced but might take long (RR), and some are like a VIP party (Priority).

- **Formal Definition:** Each scheduling algorithm has its own strengths and weaknesses. FCFS is fair but can be slow. SJF is fast for short tasks but not always fair. RR is balanced but may take longer. Priority Scheduling is flexible but can favor high-priority tasks over others.

Q42. How does the operating system decide which process to execute next?

- **Simple Definition:** The operating system is like a teacher. It looks at the students (processes) and lets the one with the best reason (priority or order) go next.

- **Formal Definition:** The operating system uses the scheduling algorithm to choose which process gets access to the CPU. It considers factors like priority, execution time, and fairness to decide who runs next.

Summary for the Interviewer:

Scheduling algorithms are like ways to decide who gets to use the computer next. FCFS waits in line, SJF picks the shortest task, RR takes turns, and Priority treats some tasks as special. Each has pros and cons. The operating system chooses using these rules. Priority inversion can slow things down if the wrong task gets the resource.

Security and Protection:

Q43. Define authentication, authorization, and auditing in the context of security.

- **Simple Definition:**
  - Authentication is like showing your secret handshake to prove you are who you say you are.
  - Authorization is like asking a grown-up if you're allowed to do something.
  - Auditing is like keeping a list of who did what to make sure everyone plays nicely.
- **Formal Definition:**
  - Authentication is the process of confirming the identity of a user or system.
  - Authorization is the process of granting or denying access rights or permissions to resources.
  - Auditing involves keeping records of activities to monitor and review them for security and compliance purposes.

Q44. Describe the differences between symmetric and asymmetric encryption.

- **Simple Definition:**
  - **Symmetric encryption** is like using the same secret key to lock and unlock a treasure chest.
  - Asymmetric encryption is like having two special keys, one for locking and one for unlocking.
- **Formal Definition:**
  - Symmetric encryption uses the same key for both encryption and decryption.
  - Asymmetric encryption uses a pair of keys, one for encryption and one for decryption, which are mathematically related but distinct.

Q46. What is a firewall, and how does it enhance system security?

- **Simple Definition: A firewall** is like a security guard at the door of your house. It checks who's coming in and keeps the bad guys out.
- **Formal Definition:** A firewall is a security device or software that monitors and controls incoming and outgoing network traffic. It acts as a barrier between a trusted network and untrusted networks (like the internet) to block or allow data based on a set of security rules.

Q47. Discuss the role of user accounts and password policies in security.

- **Simple Definition:** User accounts are like having your name on a toy box. Passwords are like secret codes to open the box. You need a strong code to keep your toys safe.
- **Formal Definition:** User accounts are individual profiles that allow users to access a computer system. Password policies establish rules for creating strong, secure passwords to protect user accounts from unauthorized access.

Summary for the Interviewer:

Authentication proves who you are, authorization says what you can do, and auditing watches what everyone does. Least privilege means only giving the necessary access. Symmetric and asymmetric encryption are like keys for boxes. Firewalls are like security guards for computer networks. User accounts and passwords keep things safe.

Networking:

Q48. What is a socket, and how is it used in network programming?

- **Simple Definition:**
  - A socket is like a phone you use to talk to your friend. It helps computers talk to each other on the internet.
- **Formal Definition:**
  - A socket is an endpoint for sending or receiving data across a computer network. It is used in network programming to establish connections and exchange data between computers.

Q49. Describe the OSI model and its layers.

- **Simple Definition:**
  - **The OSI model is like a sandwich with many layers. Each layer helps make sure your message reaches your friend.**
- **Formal Definition:**
  - The OSI (Open Systems Interconnection) model is a conceptual framework that standardizes how computer systems communicate. It divides the communication process into **seven** layers, each with a specific role in facilitating network communication.

Q50. Explain the purpose of ARP (Address Resolution Protocol) and ICMP (Internet Control Message Protocol).

- **Simple Definition:**
  - ARP is like asking, "Who has my friend's address?" ICMP is like sending secret messages to check if your friend is okay.
- **Formal Definition:**
  - ARP (Address Resolution Protocol) is used to map an IP address to a physical MAC (Media Access Control) address on a local network. ICMP (Internet Control Message Protocol) is used for network diagnostics, including error reporting and checking if a remote host is reachable.

Q51. How does NAT (Network Address Translation) work in routing packets?

- **Simple Definition:**
  - NAT is like a postman who changes the address on your letter so it can be delivered to your friend's house.
- **Formal Definition:**
  - NAT (Network Address Translation) is a technique used in routers to modify the source or destination IP addresses of packets as they pass through the router. It allows multiple devices on a local network to share a single public IP address when communicating with external networks like the internet.

Summary for the Interviewer:

Sockets are like phones for computers to talk over the internet. The OSI model has seven layers to help messages travel. ARP finds addresses, and ICMP sends messages. NAT changes addresses to share the internet.

Linux Specific:

Q52. Describe the Linux boot process.

- **Simple Definition:**
  - The Linux boot process is like waking up in the morning. The computer starts, loads its clothes, and gets ready to work.
- **Formal Definition:**
  - **The Linux boot process** is the **series of steps a computer goes through when it's turned on or restarted**. It involves **hardware checks, loading the operating system, and preparing the computer to run**.

Q53. How does the Linux file system hierarchy (e.g., /bin, /etc, /home) work?

- **Simple Definition:**
  - The **Linux file system** is like a big tree with special folders. /bin is for tools, /etc is for settings, and /home is where people keep their stuff.
- **Formal Definition:**
  - The Linux file system hierarchy is a structured way of organizing files and directories on a Linux system. It uses specific directory names like /bin for essential system binaries, /etc for system configuration files, and /home for user-specific files and directories.

Q54. What are the basic Linux shell commands, and how can they be used for system administration?

- **Simple Definition:**
  - Linux shell commands are like magic words to tell the computer what to do. You can use them to clean up, organize, and do all sorts of chores on the computer.
- **Formal Definition:**
  - Linux shell commands are text-based instructions that you type into the command line interface (CLI) to perform various tasks on a Linux system. They can be used for system administration to manage files, users, settings, and more.

Summary for the Interviewer:

The Linux boot process is like a computer waking up. The file system has special folders for different things. Shell commands are like magic words to tell the computer what to do, and they help with computer chores and tasks.

| CHEATSHEET | OPERATING SYSTEMS |

1. **Basics**

- **Operating System (OS):** Software that manages computer hardware and software resources, providing common services for computer programs.
- **Kernel:** Core part of the OS, managing system resources and communication between hardware and software.

2. **Types of Operating Systems**

- **Batch OS:** Processes batches of jobs without user interaction.
- **Time-Sharing OS:** Multiple users can access the system simultaneously.
- **Distributed OS:** Manages a group of distinct computers and makes them appear as a single computer.
- **Embedded OS:** Designed for embedded systems.
- **Real-Time OS (RTOS):** Processes data as it comes in, typically used in systems that require immediate processing.
- **Network OS:** Provides services to computers connected in a network.
- **Mobile OS:** Designed specifically for mobile devices.

3. **Processes and Threads**

- **Process:** A program in execution, with its own address space.
- **Thread:** The smallest unit of processing, sharing the process's resources.
- **Process States: New, Ready, Running, Waiting, Terminated.**
- **Multithreading: Running multiple threads within a single process.**

4. **Process Scheduling**

- **Scheduling Algorithms:**
    - **FCFS (First-Come, First-Served):** Processes are attended in the order they arrive.
    - **SJF (Shortest Job First):** Processes with the shortest burst time are scheduled first.
    - **Priority Scheduling:** Processes are scheduled based on priority.
    - **Round Robin (RR):** Each process gets a small unit of CPU time in a cyclic order.
    - **Multilevel Queue:** Processes are divided into queues based on priority.

5. **Memory Management**

- **RAM (Random Access Memory):** Volatile memory used by processes.
- **Virtual Memory:** Extends physical memory onto the disk.
- **Paging:** Divides memory into fixed-size pages.
- **Segmentation:** Divides memory into segments of varying sizes.
- **Page Replacement Algorithms:**
    - **FIFO (First-In, First-Out):** Replaces the oldest page.
    - **LRU (Least Recently Used):** Replaces the least recently used page.
    - **Optimal Page Replacement:** Replaces the page that will not be used for the longest time.

6. **File Systems**

- **File:** A collection of data stored in a storage device.
- **Directory:** A container that holds files and other directories.
- **File System Types:** NTFS, FAT32, ext3, ext4, HFS+, APFS.
- **File Operations:** Create, Open, Read, Write, Delete, Close.
- **File Attributes:** Metadata about the file, such as name, size, type, permissions.

7. **I/O Management**

- **I/O Devices:** Hardware used for input and output operations.
- **Device Drivers:** Software that controls a hardware device.
- **I/O Scheduling:** Manages the order in which I/O requests are processed.
- **Buffering:** Storing data temporarily while it is being moved from one place to another.

8. **Concurrency**

- **Concurrency:** Multiple processes or threads executing simultaneously.
- **Race Condition:** When multiple processes or threads access shared data concurrently, leading to unpredictable results.
- **Mutex (Mutual Exclusion):** Prevents multiple processes from accessing a critical section simultaneously.
- **Semaphore:** A signaling mechanism to control access to shared resources.
- **Deadlock:** **A situation where two or more processes are unable to proceed because each is waiting for the other to release resources.**
    - **Deadlock Prevention:** Ensuring at least one of the necessary conditions for deadlock cannot occur.
    - **Deadlock Avoidance:** Using algorithms like Banker's algorithm to avoid unsafe states.
    - **Deadlock Detection and Recovery:** Allowing deadlock to occur, then detecting and recovering from it.

9. **Security and Protection**

- **Authentication:** Verifying the identity of a user or process.
- **Authorization:** Granting or denying access to resources based on permissions.
- **Encryption:** Protecting data by transforming it into an unreadable format.
- **Firewall:** Controls incoming and outgoing network traffic.
- **Antivirus:** Software that detects and removes malware.
- **User and Group Management:** Managing permissions and access rights for users and groups.

10. **System Calls**

- **Definition:** Interface between **user programs and the OS.**
- **Types:**
    - **Process Control:** fork(), exit(), wait()
    - **File Management:** open(), read(), write(), close()
    - **Device Management:** ioctl(), read(), write()
    - **Information Maintenance:** getpid(), alarm(), sleep()
    - **Communication:** pipe(), shmget(), mmap()

11. **Common Commands (Unix/Linux)**

- **File Operations:**
    - ls: List directory contents.
    - cd: Change directory.
    - cp: Copy files.
    - mv: Move or rename files.
    - rm: Remove files.
    - cat: Concatenate and display files.
- **Process Management:**
    - ps: Report a snapshot of current processes.
    - top: Display and update sorted information about processes.
    - kill: Terminate processes.
    - bg: Resume suspended jobs in the background.
    - fg: Resume suspended jobs in the foreground.
- **System Information:**
    - uname: Print system information.
    - df: Report filesystem disk space usage.
    - du: Estimate file space usage.
    - uptime: Tell how long the system has been running.
    - free: Display amount of free and used memory.

12. Virtualization

- **Virtual Machine (VM):** Emulates a physical computer system.
- **Hypervisor:** Software that creates and manages VMs.
    - **Type 1:** Runs directly on hardware (e.g., VMware ESXi).
    - **Type 2:** Runs on a host operating system (e.g., VirtualBox, VMware Workstation).
- **Containerization:** Lightweight virtualization using containers (e.g., Docker).

13. **Common OS Examples**

- **Windows:** Microsoft's OS with a graphical user interface.
- **Linux:** Open-source Unix-like OS, commonly used for servers.
- **macOS:** Apple's Unix-based OS for Mac computers.
- **Unix:** Multiuser, multitasking OS, known for stability and security.

im_krishna_8 ★ 88 July 12, 2024 12:30 PM 679 VIEWS

Concurrency

Concurrency means working or processing on things time by time like some time for this process(work) some time for this process(work) in simple words Concurrency means dealing with multiple task time by time peridically.

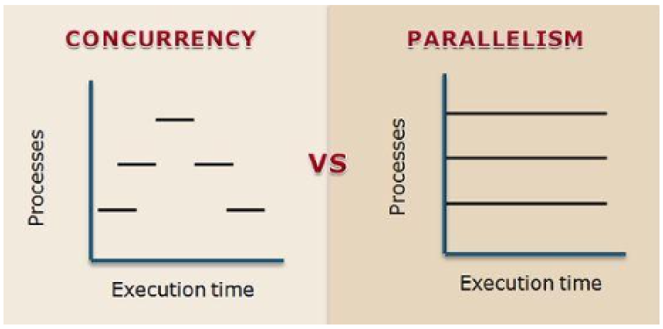
Example: Imagine you are cooking and also watching TV. You cook for a few minutes, then watch TV for a few minutes, and keep switching between the two tasks. Both tasks are making progress, but not necessarily at the same time.

Parallelism

Parallelism means working on multiple things at same time. both process is work on different cpu.

Example: Imagine two people in a kitchen. One person is cooking while the other is watching TV. Both activities are happening simultaneously, without any switching between the two.

may be this image clear your remaining doubts.

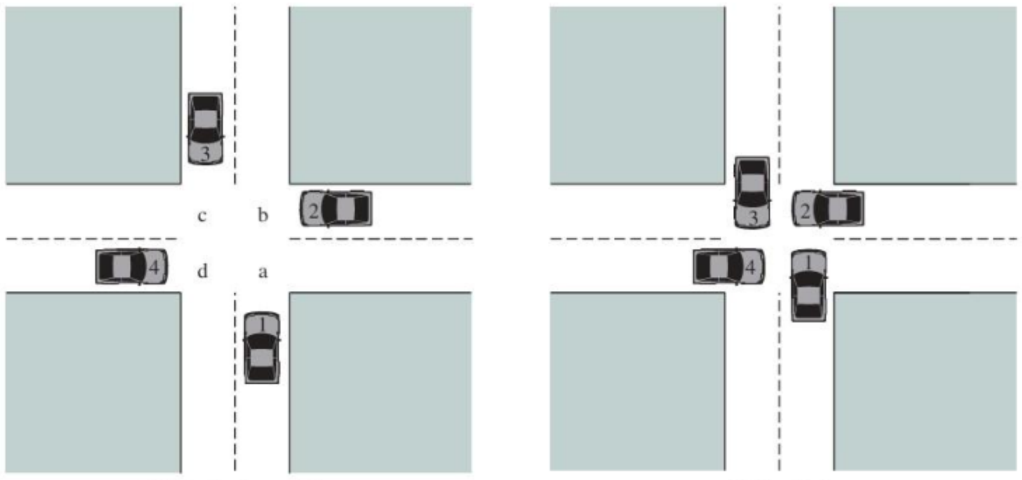


Thank you for reading....

DeadLock

Deadlock can be defined as the **permanent blocking of a set of processes that either compete for system resources or communicate with each other**. A set of processes is deadlocked when each process in the set is blocked awaiting an event (typically the freeing up of some requested resource) that can only be triggered by another blocked process in the set. Deadlock is permanent because none of the events is ever triggered. Unlike other problems in concurrent process management, there is no efficient solution in the general case.

This image is taken from Operating_System(william Stallings)



(a) Deadlock possible (b) Deadlock

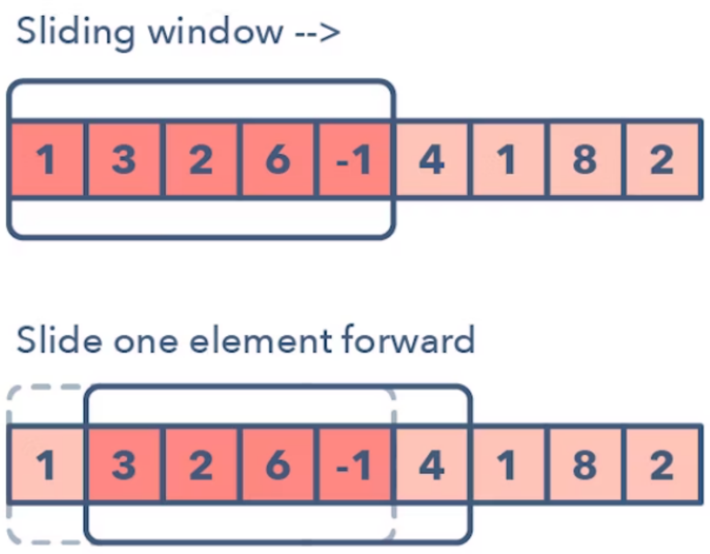
All deadlocks involve conflicting needs for resources by two or more processes. A common example is the traffic deadlock. Figure 6.1a shows a situation in which four cars have arrived at a four-way stop intersection at approximately the same time. The four quadrants of the intersection are the resources over which control is needed. In particular, if all four cars wish to go straight through the intersection, the resource requirements are as follows:

Car 1, traveling north, needs quadrants a and b.
Car 2 needs quadrants b and c.
Car 3 needs quadrants c and d.
Car 4 needs quadrants d and a.

Problem Solving Popular Technique

1. Sliding Window

The Sliding Window pattern is used to perform a required operation on a specific window size of a given array or linked list, such as finding the longest subarray containing all 1s. Sliding Windows start from the 1st element and keep shifting right by one element and adjust the length of the window according to the problem that you are solving. In some cases, the window size remains constant and in other cases the sizes grows or shrinks.



Following are some ways you can identify that the given problem might require a sliding window:

- The problem input is a linear data structure such as a **linked list, array, or string**
- You're asked to find the longest/shortest substring, subarray, or a desired value

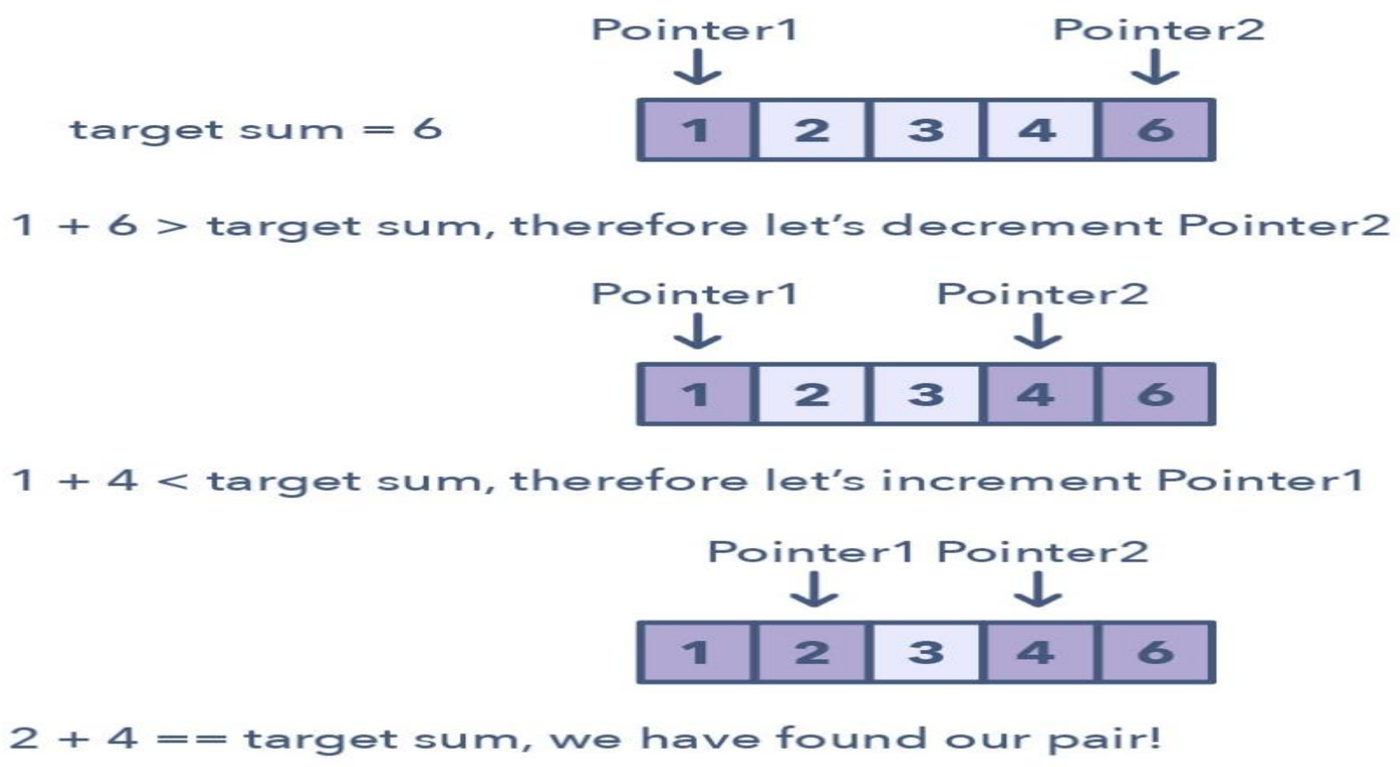
Common problems you use the sliding window pattern with:

- Maximum sum subarray of size 'K' (easy)
- Longest substring with 'K' distinct characters (medium)
- String anagrams (hard)

2. Two Pointers or Iterators

Two Pointers is a pattern where two pointers iterate through the data structure in tandem until one or both of the pointers hit a certain condition.Two Pointers is often useful when searching pairs in a sorted array or linked list; for example, when you have to compare each element of an array to its other elements.

Two pointers are needed because with just pointer, you would have to continually loop back through the array to find the answer. This back and forth with a single iterator is inefficient for time and space complexity — a concept referred to as asymptotic analysis. While the brute force or naive solution with 1 pointer would work, it will produce something along the lines of $O(n^2)$. In many cases, two pointers can help you find a solution with better space or runtime complexity.


Pointer1
Pointer2
target sum = 6
1 2 3 4 6
1 + 6 > target sum, therefore let's decrement Pointer2
Pointer1
Pointer2
1 2 3 4 6
1 + 4 < target sum, therefore let's increment Pointer1
Pointer1
Pointer2
1 2 3 4 6
2 + 4 == target sum, we have found our pair!

Ways to identify when to use the Two Pointer method:

- It will feature problems where you deal with sorted arrays (or Linked Lists) and need to **find a set of elements that fulfill certain constraints**
- The set of elements in the **array is a pair, a triplet, or even a subarray**

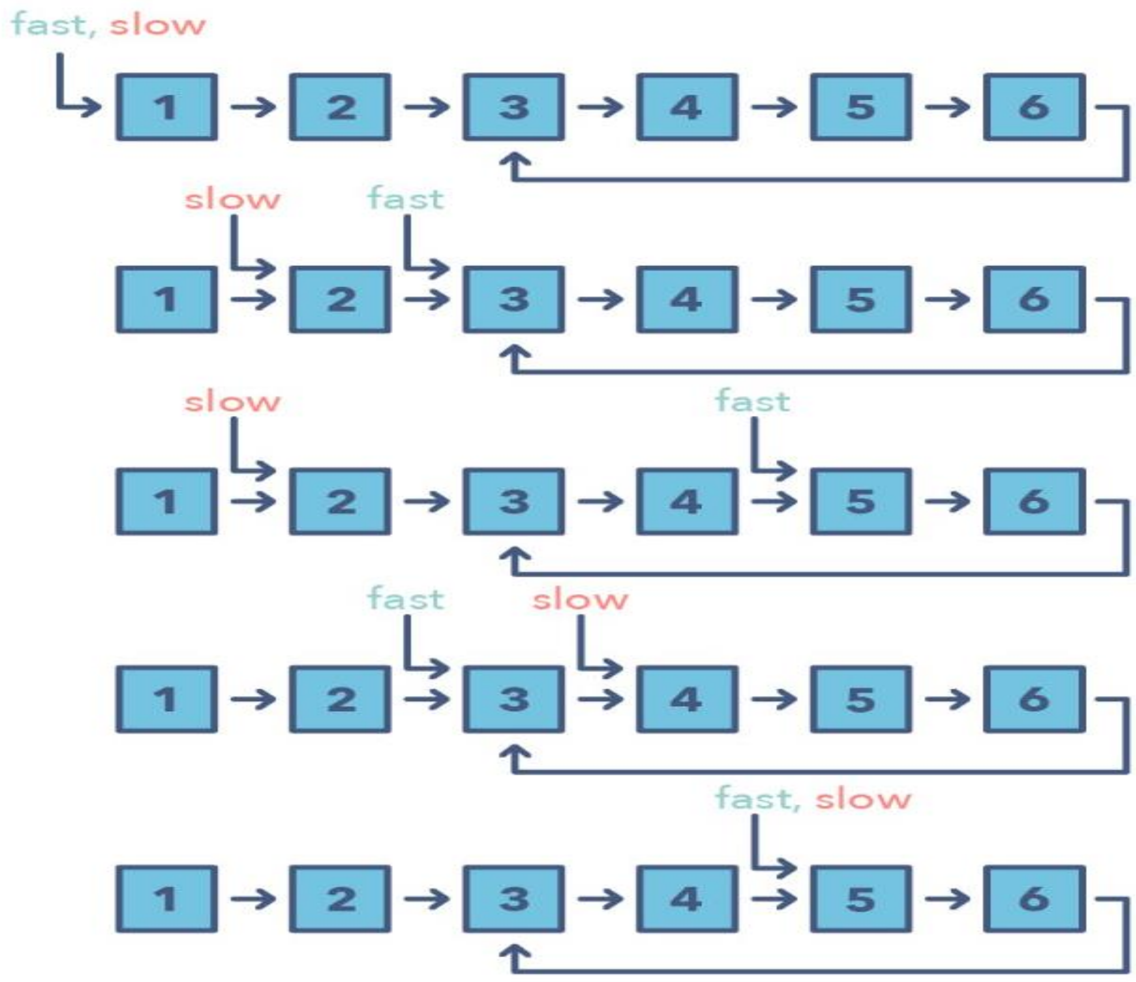
Here are some problems that feature the Two Pointer pattern:

- Squaring a sorted array (easy)
- Triplets that sum to zero (medium)
- Comparing strings that contain backspaces (medium)

3. Fast and Slow pointers

The **Fast and Slow pointer** approach, also known as the Hare & Tortoise algorithm, is a pointer algorithm that uses two pointers which move through the array (or sequence/linked list) at different speeds. This approach is quite useful when dealing with cyclic linked lists or arrays.

By moving at different speeds (say, in a cyclic linked list), the algorithm proves that the two pointers are bound to meet. The fast pointer should catch the slow pointer once both the pointers are in a cyclic loop.



How do you identify when to use the Fast and Slow pattern?

- The problem will deal **with a loop in a linked list or array**
- When you need to know the **position of a certain element** or the **overall length of the linked list**.

When should I use it over the Two Pointer method mentioned above?

- There are some cases where you shouldn't use the Two Pointer **approach such as in a singly linked list where you can't** move in a backwards direction. **An example of when to use the Fast and Slow pattern is when you're trying to determine if a linked list is a palindrome.**

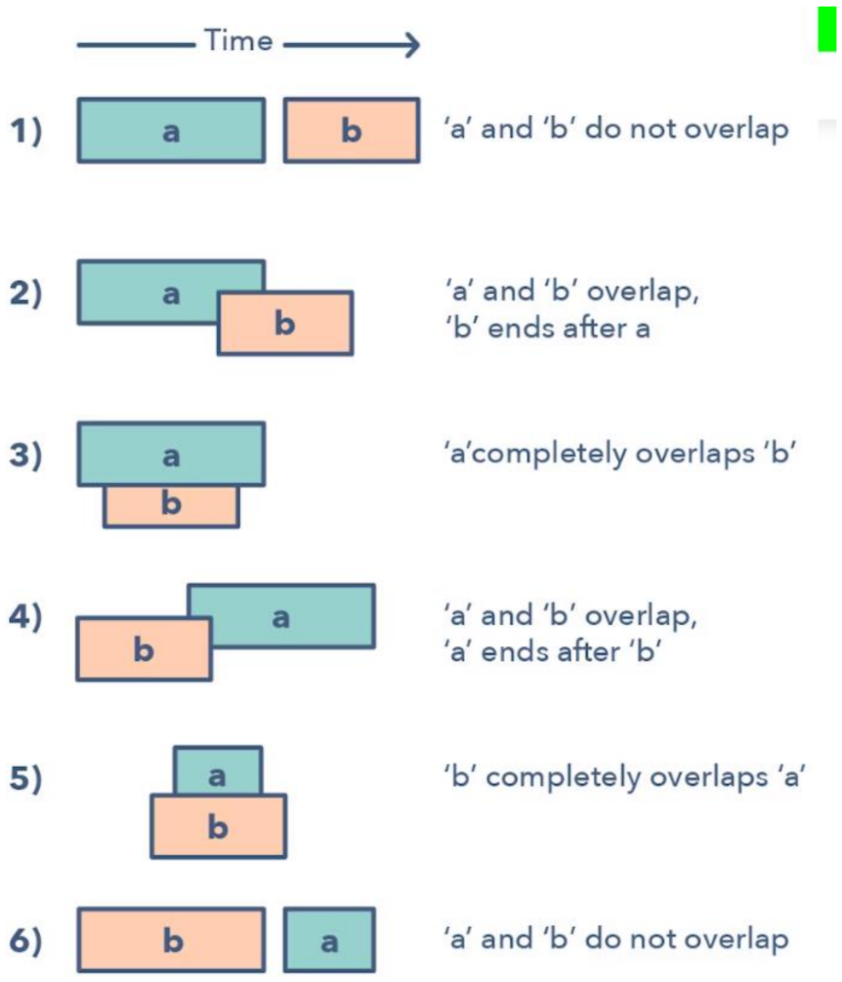
Problems featuring the fast and slow pointers pattern:

- Linked List Cycle (easy)
- Palindrome Linked List (medium)
- Cycle in a Circular Array (hard)

4. Merge Intervals

The **Merge Intervals pattern** is an efficient technique to deal with **overlapping intervals**. In a lot of problems involving intervals, you either need to find overlapping intervals or merge intervals if they overlap. The pattern works like this:

Given two intervals (‘a’ and ‘b’), there will be six different ways the two intervals can relate to each other:


Time
1)
a
b
'a' and 'b' do not overlap
2)
a
b
'a' and 'b' overlap,
'b' ends after a
3)
a
b
'a'completely overlaps 'b'
4)
b
a
'a' and 'b' overlap,
'a' ends after 'b'
5)
a
b
'b' completely overlaps 'a'
6)
b
a
'a' and 'b' do not overlap

Understanding and recognizing these six cases will help you help you solve a wide range of problems from inserting intervals to optimizing interval merges.

How do you identify when to use the Merge Intervals pattern?

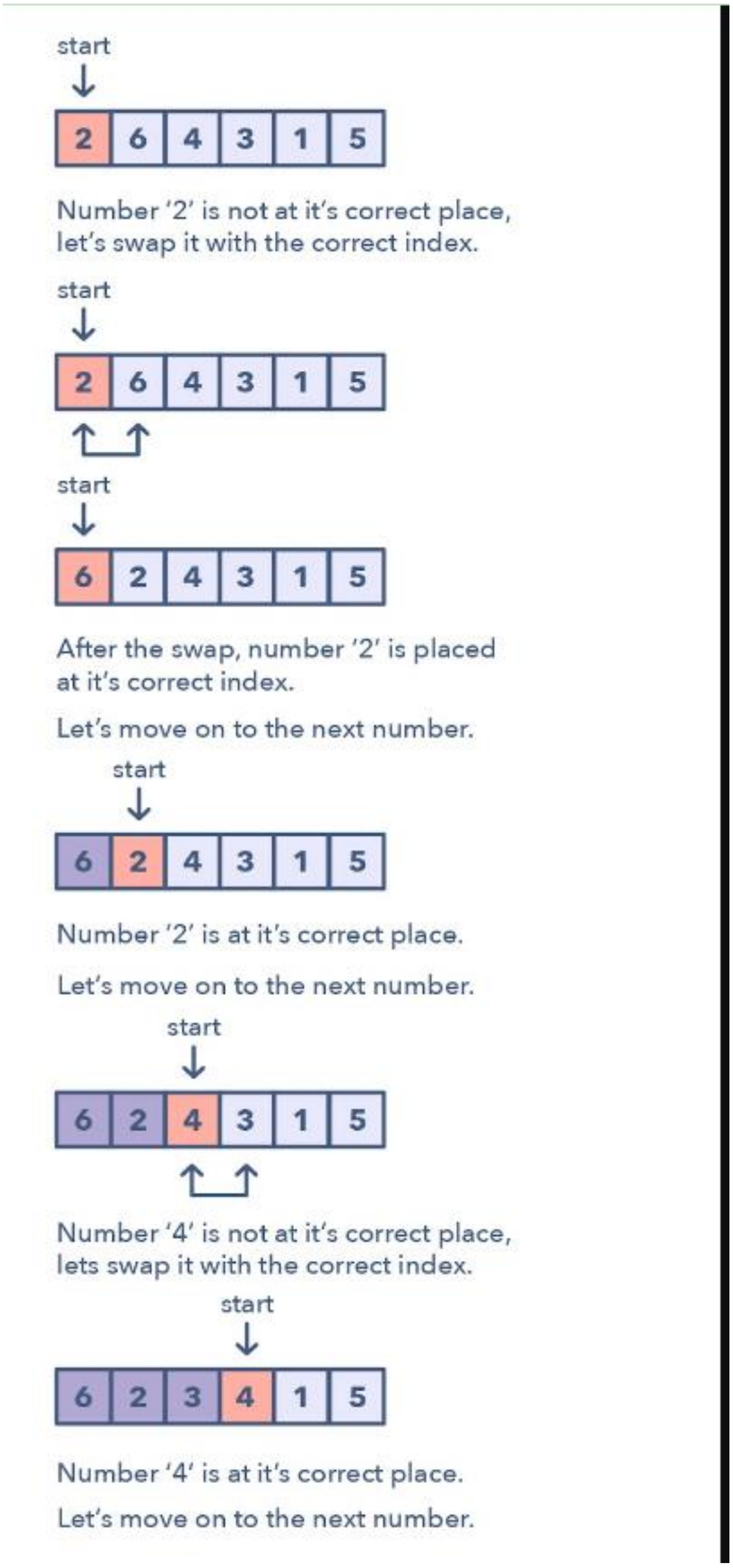
- If you’re asked to produce a list with only mutually exclusive intervals
- If you hear the term “overlapping intervals”.

Merge interval problem patterns:

- Intervals Intersection (medium)
- Maximum CPU Load (hard)

5. Cyclic sort

This pattern describes an interesting approach to deal with problems involving arrays containing numbers in a given range. The **Cyclic Sort pattern** iterates over the array one number at a time, and if the current number you are iterating is not at the correct index, you swap it with the number at its correct index. You could try placing the number in its correct index, but this will produce a complexity of O(n^2) which is not optimal, hence the Cyclic Sort pattern.


start
2 6 4 3 1 5
Number '2' is not at it's correct place, let's swap it with the correct index.
start
2 6 4 3 1 5
start
6 2 4 3 1 5
After the swap, number '2' is placed at it's correct index.
Let's move on to the next number.
start
6 2 4 3 1 5
Number '2' is at it's correct place.
Let's move on to the next number.
start
6 2 4 3 1 5
Number '4' is not at it's correct place, lets swap it with the correct index.
start
6 2 3 4 1 5
Number '4' is at it's correct place.
Let's move on to the next number.

How do I identify this pattern?

- They will be problems involving a **sorted array with numbers in a given range**
- If the problem asks you **to find the missing/duplicate/smallest number in an sorted/rotated array**

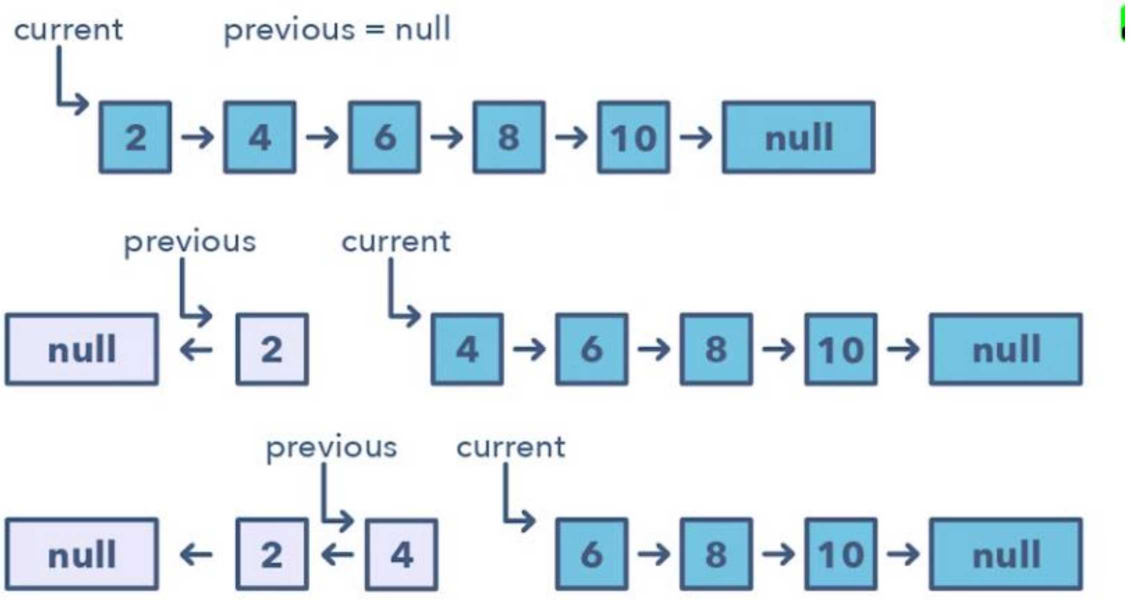
Problems featuring cyclic sort pattern:

- Find the Missing Number (easy)
- Find the Smallest Missing Positive Number (medium)

6. In-place reversal of linked list

In a lot of problems, you may be asked to reverse the links between a set of nodes of a linked list. Often, the constraint is that you need to do this in-place, i.e., using the existing node objects and without using extra memory. This is where the above mentioned pattern is useful.

This pattern reverses one node at a time starting with one variable (current) pointing to the head of the linked list, and one variable (previous) will point to the previous node that you have processed. In a lock-step manner, you will reverse the current node by pointing it to the previous before moving on to the next node. Also, you will update the variable "previous" to always point to the previous node that you have processed.



How do I identify when to use this pattern:

- **If you're asked to reverse a linked list without using extra memory**

Problems featuring in-place reversal of linked list pattern:

- **Reverse a Sub-list (medium)**
- **Reverse every K-element Sub-list (medium)**

7. Tree BFS

This pattern is based on the Breadth First Search (BFS) technique to traverse a tree and uses a queue to keep track of all the nodes of a level before jumping onto the next level. Any problem involving the traversal of a tree in a level-by-level order can be efficiently solved using this approach.

The Tree BFS pattern works by pushing the root node to the queue and then continually iterating until the queue is empty. For each iteration, we remove the node at the head of the queue and “visit” that node. After removing each node from the queue, we also insert all of its children into the queue.

How to identify the Tree BFS pattern:

- **If you’re asked to traverse a tree in a level-by-level fashion (or level order traversal)**

Problems featuring Tree BFS pattern:

- **Binary Tree Level Order Traversal (easy)**
- **Zigzag Traversal (medium)**

8. Tree DFS

Tree DFS is based on the Depth First Search (DFS) technique to traverse a tree.

You can use recursion (or a stack for the iterative approach) to keep track of all the previous (parent) nodes while traversing.

The Tree DFS pattern works by starting at the root of the tree, if the node is not a leaf you need to do three things:

1. **Decide whether to process the current node now (pre-order), or between processing two children (in-order) or after processing both children (post-order).**
2. **Make two recursive calls for both the children of the current node to process them.**

How to identify the Tree DFS pattern:

- **If you’re asked to traverse a tree with in-order, preorder, or postorder DFS**
- **If the problem requires searching for something where the node is closer to a leaf**

Problems featuring Tree DFS pattern:

- **Sum of Path Numbers (medium)**
- **All Paths for a Sum (medium)**

9. Two heaps

In many problems, we are given a **set of elements** such that we can divide them into two parts. To solve the problem, we are interested in knowing **the smallest element in one part a**nd **the biggest element in the other part**. This pattern is an efficient approach to solve such problems.

This pattern uses two heaps; **A Min Heap to find the smallest element** and a Max Heap to find the biggest element. The pattern works by storing the first half of numbers in a Max Heap, this is because you want to find the largest number in the first half. You then store the second half of numbers in a Min Heap, as you want to find the smallest number in the second half. At any time, the median of the current list of numbers can be calculated from the top element of the two heaps.

Ways to identify the Two Heaps pattern:

- Useful in situations like Priority Queue, Scheduling
- If the problem states that you need to find the smallest/largest/median elements of a set
- Sometimes, useful in **problems featuring a binary tree data structure**

Problems featuring

- Find the Median of a Number Stream (medium)

10. Subsets

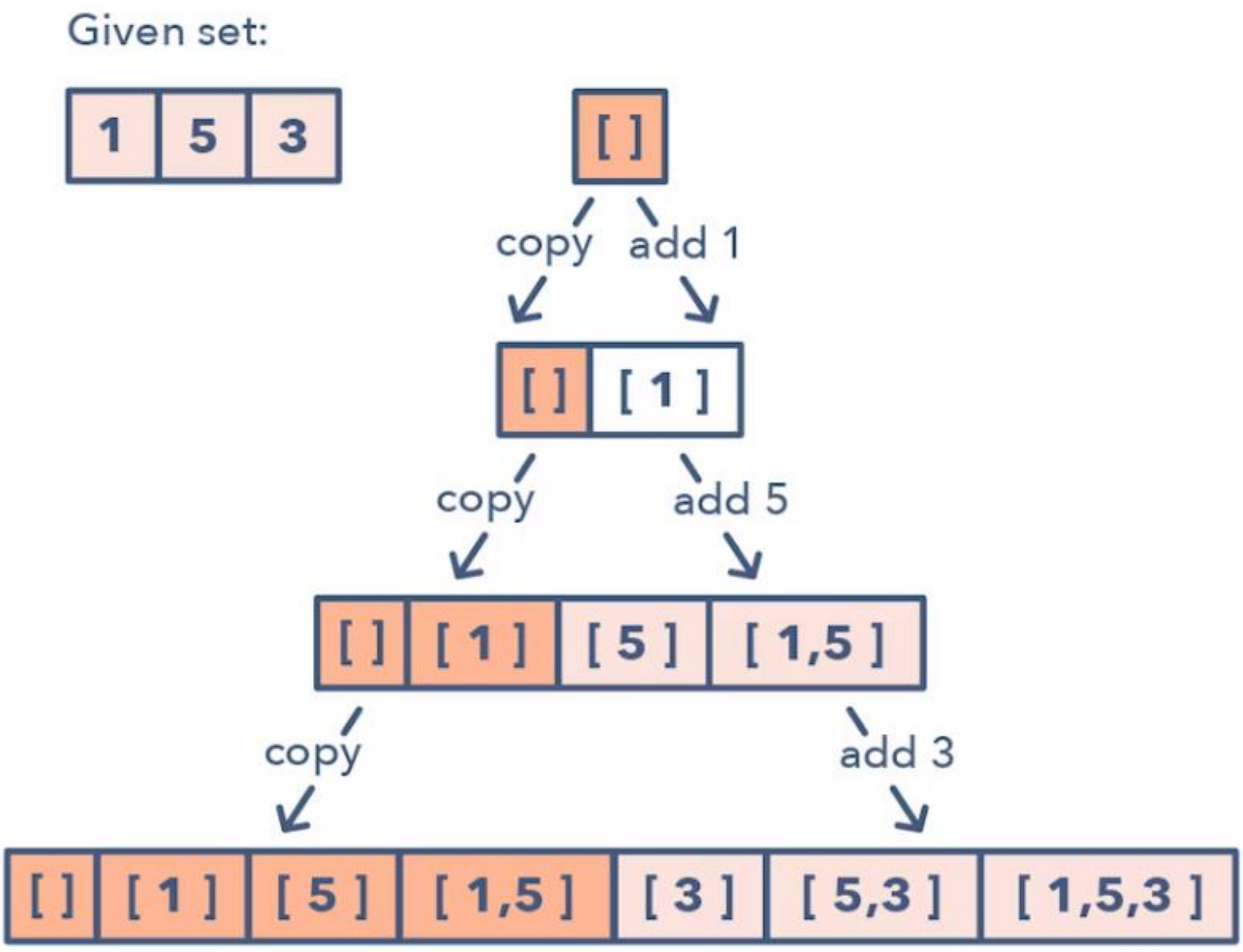
A huge number of coding interview problems involve dealing with Permutations and Combinations of a given set of elements. The pattern Subsets describes an **efficient Breadth First Search (BFS) approach to handle all these problems.**

The pattern looks like this:

Given a set of [1, 5, 3]

1. Start with an empty set: [[]]
2. Add the first number (1) to all the existing subsets to create new subsets: [[], [1]];
3. Add the second number (5) to all the existing subsets: [[], [1], [5], [1,5]];
4. Add the third number (3) to all the existing subsets: [[], [1], [5], [1,5], [3], [1,3], [5,3], [1,5,3]].

Here is a visual representation of the Subsets pattern:



How to identify the Subsets pattern:

- **Problems where you need to find the combinations or permutations of a given set**

Problems featuring Subsets pattern:

- **Subsets With Duplicates (easy)**
- **String Permutations by changing case (medium)**

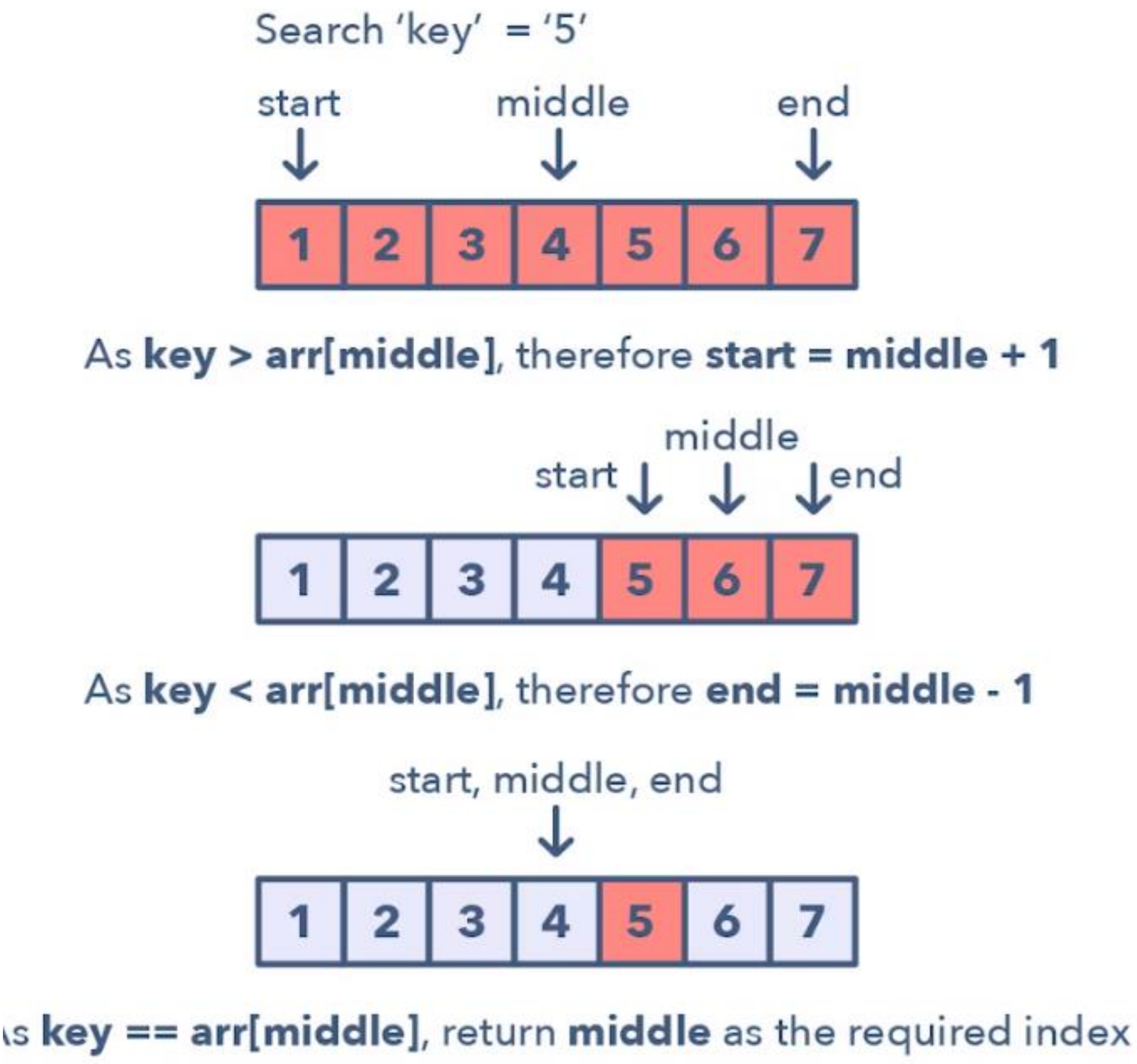
11. Modified binary search

Whenever you are given **a sorted array, linked list, or matrix, and are asked to find a certain element**, the best algorithm you can use is the Binary Search. This pattern describes an efficient way to handle all problems involving Binary Search.

The patterns looks like this for an ascending order set:

1. First, find the middle of start and end. An easy way to find the middle would be: middle = (start + end) / 2. But this has a good chance of producing an integer overflow so it's recommended that you represent the middle as: middle = start + (end — start) / 2
2. If the key is equal to the number at index middle then return middle
3. If 'key' isn't equal to the index middle:
4. Check if key < arr[middle]. If it is reduce your search to end = middle — 1
5. Check if key > arr[middle]. If it is reduce your search to end = middle + 1

Here is a visual representation of the Modified Binary Search pattern:

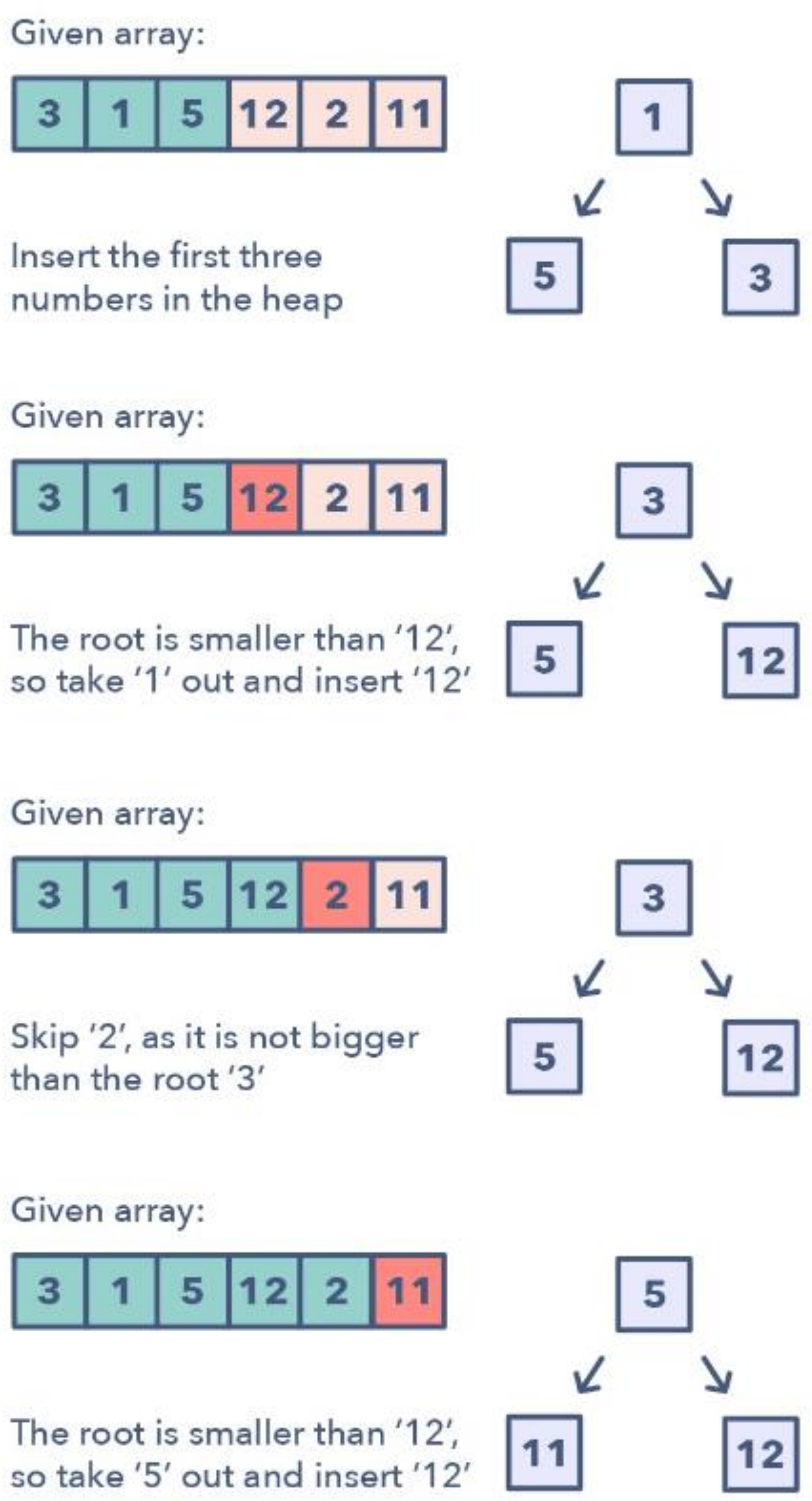

Search 'key' = '5'
start
middle
end
1 2 3 4 5 6 7
As **key > arr[middle]**, therefore **start = middle + 1**
middle
start
end
1 2 3 4 5 6 7
As **key < arr[middle]**, therefore **end = middle - 1**
start, middle, end
1 2 3 4 5 6 7
ıs **key == arr[middle]**, return **middle** as the required index

12. Top K elements

Any problem that asks us to find the top/smallest/frequent 'K' elements among a given set falls under this pattern.

The best data structure to keep track of 'K' elements is Heap. This pattern will make use of the Heap to solve multiple problems dealing with 'K' elements at a time from a set of given elements. The pattern looks like this:

1. **Insert 'K' elements into the min-heap or max-heap based on the problem.**
2. **Iterate through the remaining numbers and if you find one that is larger than what you have in the heap, then remove that number and insert the larger one.**


Given array:
3 1 5 12 2 11
1
5 3
Insert the first three numbers in the heap
Given array:
3 1 5 12 2 11
3
5 12
The root is smaller than '12', so take '1' out and insert '12'
Given array:
3 1 5 12 2 11
3
5 12
Skip '2', as it is not bigger than the root '3'
Given array:
3 1 5 12 2 11
5
11 12
The root is smaller than '12', so take '5' out and insert '12'

There is no need for a sorting algorithm because the heap will keep track of the elements for you.

How to identify the Top 'K' Elements pattern:

- **If you're asked to find the top/smallest/frequent 'K' elements of a given set**
- **If you're asked to sort an array to find an exact element**

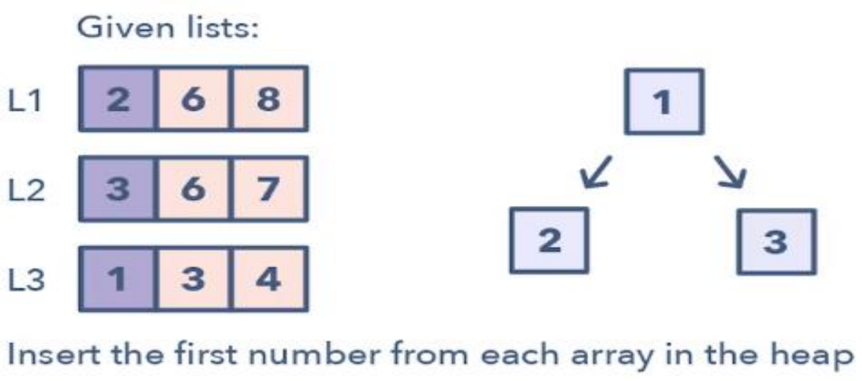
Problems featuring Top 'K' Elements pattern:

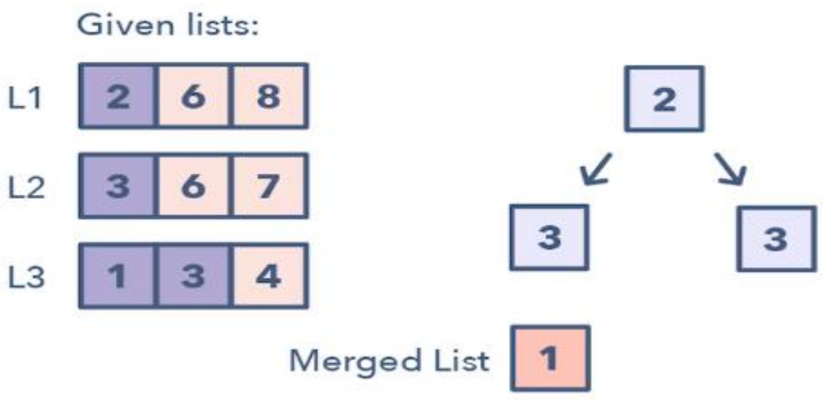
- **Top 'K' Numbers (easy)**
- **Top 'K' Frequent Numbers (medium)**

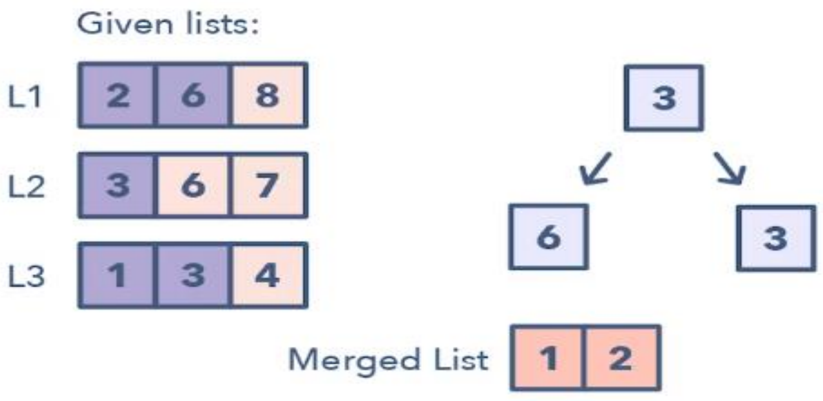
13. K-way Merge

K-way Merge helps you solve problems that **involve a set of sorted arrays.**

Whenever you're given 'K' sorted arrays, you can use a Heap to efficiently perform a sorted traversal of all the elements of all arrays. You can push the smallest element of each array in a Min Heap to get the overall minimum. After getting the overall minimum, push the next element from the same array to the heap. Then, repeat this process to make a sorted traversal of all elements.







The pattern looks like this:

1. Insert the first element of each array in a Min Heap.
2. After this, take out the smallest (top) element from the heap and add it to the merged list.
3. After removing the smallest element from the heap, insert the next element of the same list into the heap.
4. Repeat steps 2 and 3 to populate the merged list in sorted order.

How to identify the K-way Merge pattern:

- The problem will **feature sorted arrays, lists, or a matrix**
- If the problem asks you to **merge sorted lists, find the smallest element in a sorted list.**

Problems featuring the K-way Merge pattern:

- Merge K Sorted Lists (medium)
- K Pairs with Largest Sums (Hard)

14. Topological sort

Topological Sort is used to find a linear ordering of elements that have dependencies on each other. For example, if event ‘B’ is dependent on event ‘A’, ‘A’ comes before ‘B’ in topological ordering.

How to identify the Topological Sort pattern:

- The problem will deal with graphs that have no directed cycles
- If you’re asked to update all objects in a sorted order
- If you have a class of objects that follow a particular order

Problems featuring the Topological Sort pattern:

- Task scheduling (medium)
- Minimum height of a tree (hard)