**Day 18 - 18th July 2025**

**Task - 4:**

**What are the applications of heap sort?**

### 1. Priority Queue

* A priority queue is like a to-do list where the most important task comes first.
* Heap Sort is used to manage and sort tasks by importance.
* For example: In an operating system, important programs run before others.

2. Sorting Large Data

* Heap Sort is great when you're sorting big amounts of data and you don't have a lot of extra memory.
* It doesn’t need extra space like some other sorting methods.

3. Finding the Kth Largest or Smallest Number

* Want to find the 3rd biggest number in a list?
* Heap Sort helps do that quickly without sorting everything.

4. Real-Time Systems

* Heap Sort always takes about the same amount of time, even in the worst case.
* That makes it useful in systems where speed and timing must be predictable (like traffic lights or robots).

5. External Sorting

* When data is too big to fit in memory (like huge files on disk), heap sort helps sort pieces of it efficiently.

6. Used in Algorithms

* Some famous algorithms like:  
  + Dijkstra’s shortest path (used in maps)
  + A\* search (used in games)
* These use heaps, so Heap Sort is a natural fit.

**Task - 6**

**How does heap sort work ? explain the technique in 5 .. algorithm**

### **Algorithm Steps:**

### Step 1: Build a Max Heap

* Convert the input array into a Max Heap (a binary tree where each parent is greater than its children).
* This ensures the largest element is at the root.

Step 2: Swap Root with Last Element

* Swap the first element (maximum) with the last element in the heap.
* This puts the largest value at the end (its correct sorted position).

Step 3: Reduce Heap Size

* Ignore the last element (it's sorted now).
* Reduce the heap size by 1.

### Step 4: Heapify the Root

* Call the heapify function on the root to restore the heap property.
* This moves the new root to its correct place in the heap.

Step 5: Repeat

* Repeat steps 2 to 4 until all elements are sorted.

Task -

## **Algorithm Steps for Radix Sort:**

1. Find the maximum number to determine the number of digits (D).
2. Start from the least significant digit (units place).
3. Use Counting Sort (or any stable sort) to sort elements based on the current digit.
4. Repeat the sorting process for every digit (tens, hundreds, etc.).
5. After all digits are processed, the array will be sorted.

**Pseudocode for Radix Sort:**

RADIX\_SORT(arr)

max = find the maximum number in arr

exp = 1

WHILE (max / exp > 0)

DO counting sort on arr based on digit at 'exp' place

exp = exp \* 10

**Counting Sort by Digit**

COUNTING\_SORT\_BY\_DIGIT(arr, exp)

create output array of same size as arr

create count array of size 10, initialized to 0

// Count occurrences of digits (0 to 9) at 'exp' place

FOR i = 0 to arr.length - 1

digit = (arr[i] / exp) % 10

count[digit]++

// Change count[i] so it contains actual position in output

FOR i = 1 to 9

count[i] = count[i] + count[i - 1]

// Build output array (stable sort)

FOR i = arr.length - 1 down to 0

digit = (arr[i] / exp) % 10

output[count[digit] - 1] = arr[i]

count[digit]--

// Copy output back to arr

FOR i = 0 to arr.length - 1

arr[i] = output[i]