arr = [8, 4, 5, 3, 2, 7, 6, 1]

FULL TRACE: MERGE SORT mergeSort(arr, 0, 7)

Step	Function Call	What Happens
1	mergeSort(arr, 0, 7)	mid = 3 → calls mergeSort(arr, 0, 3)
2	mergeSort(arr, 0, 3)	mid = 1 → calls mergeSort(arr, 0, 1)
3	mergeSort(arr, 0, 1)	mid = 0 → calls mergeSort(arr, 0, 0)
4	mergeSort(arr, 0, 0)	base case → returns
5	returns to mergeSort(arr, 0, 1)	now calls mergeSort(arr, 1, 1)
6	mergeSort(arr, 1, 1)	base case → returns
7	back to mergeSort(arr, 0, 1)	now calls merge(arr, 0, 0, 1)
8	merge(arr, 0, 0, 1)	merges [8] and [4] → result: [4, 8]
9	returns to mergeSort(arr, 0, 3)	now calls mergeSort(arr, 2, 3)
10	mergeSort(arr, 2, 3)	mid = 2 → calls mergeSort(arr, 2, 2)
11	mergeSort(arr, 2, 2)	base case → returns
12	back to mergeSort(arr, 2, 3)	calls mergeSort(arr, 3, 3)
13	mergeSort(arr, 3, 3)	base case → returns
14	back to mergeSort(arr, 2, 3)	calls merge(arr, 2, 2, 3)
15	merge(arr, 2, 2, 3)	merges [5] and [3] → result: [3, 5]
16	back to mergeSort(arr, 0, 3)	calls merge(arr, 0, 1, 3)
17	merge(arr, 0, 1, 3)	merges [4, 8] and [3, 5] → result: [3, 4, 5, 8]

Now second half: mergeSort(arr, 4, 7)

Step	Function Call	What Happens
18	mergeSort(arr, 4, 7)	mid = 5 → calls mergeSort(arr, 4, 5)
19	mergeSort(arr, 4, 5)	mid = 4 → calls mergeSort(arr, 4, 4)
20	mergeSort(arr, 4, 4)	base case → returns
21	back to mergeSort(arr, 4, 5)	calls mergeSort(arr, 5, 5)
22	mergeSort(arr, 5, 5)	base case → returns
23	back to mergeSort(arr, 4, 5)	calls merge(arr, 4, 4, 5)
24	merge(arr, 4, 4, 5)	merges [2] and [7] → result: [2, 7]
25	back to mergeSort(arr, 4, 7)	calls mergeSort(arr, 6, 7)
26	mergeSort(arr, 6, 7)	mid = 6 → calls mergeSort(arr, 6, 6)
27	mergeSort(arr, 6, 6)	base case → returns
28	back to mergeSort(arr, 6, 7)	calls mergeSort(arr, 7, 7)
29	mergeSort(arr, 7, 7)	base case → returns
30	back to mergeSort(arr, 6, 7)	calls merge(arr, 6, 6, 7)
31	merge(arr, 6, 6, 7)	merges [6] and [1] → result: [1, 6]
32	back to mergeSort(arr, 4, 7)	calls merge(arr, 4, 5, 7)
33	merge(arr, 4, 5, 7)	merges [2, 7] and [1, 6] → result: [1, 2, 6, 7]

Heap Sort

arr = [4, 10, 3, 5, 1]

Step	Action	Array State	Explanation
1	Build Max-Heap	[10, 5, 3, 4, 1]	Start from the last non-leaf node → heapify from bottom up
2	Swap root with last (i=4)	[1, 5, 3, 4, 10]	Move largest (10) to end, reduce heap size
3	Heapify root (i=0 to i=3)	[5, 4, 3, 1, 10]	Fix heap after removing 10
4	Swap root with index 3	[1, 4, 3, 5, 10]	Move largest (5) to its position
5	Heapify root (i=0 to i=2)	[4, 1, 3, 5, 10]	Fix heap after removing 5
6	Swap root with index 2	[3, 1, 4, 5, 10]	Move largest (4) to position
7	Heapify root (i=0 to i=1)	[3, 1, 4, 5, 10]	Root already max heap
8	Swap root with index 1	[1, 3, 4, 5, 10]	Move largest (3) to sorted position
9	Heapify (size = 1)	[1, 3, 4, 5, 10]	Only one element left, heap is sorted