Build Heap

1. Let index = length/2-1. This is the parent of the last node in the tree, i.e. list[index + 1] . . . list[length-1] are leaves

Original Array:
[5, 22, 9, 76, 63, 81, 48, 92, 54, 28]

Step-by-step explanation:

a. Start from the last parent node:
Formula: index = length // 2 - 1 = 10 // 2 - 1 = 4
So we begin heapifying from index 4 and go up to index 0.

b. Heapify from index 4 to 0:

i. Index 4 (value 63): Children are at indices 9 (28) and no right child. 63 > 28 → no swap.

ii. Index 3 (76): Children 7 (92) and 8 (54). 92 > 76 → swap → new subtree: 92 becomes parent.

iii. Now at index 7 (76): No children → done.

iv. Index 2 (9): Children 5 (81) and 6 (48). 81 > 9 → swap. Now 81 is parent.

1. Index 5 (9) has no children → done.

v. Index 1 (22): Children 3 (92) and 4 (63). 92 > 22 → swap.

1. Index 3 (22): Children 7 (76) and 8 (54) → 76 > 22 → swap.
2. Index 7 (22): No children → done.

vi. Index 0 (5): Children 1 (92) and 2 (81). 92 > 5 → swap.

1. Index 1 (5): Children 3 (76) and 4 (63) → 76 > 5 → swap.
2. Index 3 (5): Children 7 (22) and 8 (54) → 54 > 5 → swap.
3. Index 8 (5): No children → done.

Resulting Max-Heap: [92, 76, 81, 54, 63, 9, 48, 22, 5, 28]

2. Convert the subtree with root of list[index] into a heap.

a. Given list[a] is root of tree, list[b] is left child (root *2 +1), list[c] is right child (root*2+2), if exists

b. Compare list[b] with list[c] to determine larger child, list[largerIndex]

c. Compare list[a] with list[largerIndex]. If list[a] < list[largerIndex], then swap, else already a heap

d. If swap, repeat step 2 for the subtree of list[largerIndex]

Procedure:

1. Swap root (max value) with last item.
2. Decrease the heap size by 1 (ignore the last index now).
3. Heapify the root again.
4. Repeat until one item is left.

Sorted Steps (in descending extraction order, because it's a max-heap):

- Extract 92 → [**92** is placed at end]
- Heapify → Extract 81 → [**81** placed before 92]
- Continue until array becomes: [92, 81, 76, 63, 54, 48, 28, 22, 9, 5]

Final Sorted Array (ascending order):
[5, 9, 22, 28, 48, 54, 63, 76, 81, 92]

(Note: We got the descending order during extraction, but you can reverse it to get ascending.)

3. Convert the subtree with the root of list[index-1] into a heap, repeat until list[0]

From the slide layout, there are three sets of 10 boxes (indices [0] to [9]):

- **First set**: Initial array input.
    - Fill: 5, 22, 9, 76, 63, 81, 48, 92, 54, 28
- **Second set**: After max-heap construction.
    - Fill: 92, 76, 81, 54, 63, 9, 48, 22, 5, 28
- **Third set**: After full heap sort.
    - Fill *to get ascending sort* : *5, 9, 22, 28, 48, 54, 63, 76, 81, 92*

	Step 1	Step 2	Step 3	Step 4	Step 5	Step 6
[0]	5	5	5	5	92	92
[1]	22	22	22	92	76	76
[2]	9	9	81	81	81	81
[3]	76	92	92	22	5	54
[4]	63	63	63	63	63	63
[5]	81	81	9	9	9	9
[6]	48	48	48	48	48	48
[7]	92	76	76	76	76	22
[8]	54	54	54	54	54	5
[9]	28	28	28	28	28	28

	Sorted Array
[0]	92
[1]	76
[2]	81
[3]	54
[4]	63
[5]	9
[6]	48
[7]	22
[8]	5
[9]	28

	Step 1	Step 2	Step 3	Step 4	Step 5	Step 6	Step 7	Step 8	Step 9
[0]	81	76	63	54	48	28	22	9	5
[1]	76	63	54	28	28	22	5	5	9
[2]	48	48	48	48	9	9	9	22	22
[3]	54	54	22	22	22	5	28	28	28
[4]	63	5	5	5	5	48	48	48	48
[5]	9	9	9	9	54	54	54	54	54
[6]	28	28	28	63	63	63	63	63	63
[7]	22	22	76	76	76	76	76	76	76
[8]	5	81	81	81	81	81	81	81	81
[9]	92	92	92	92	92	92	92	92	92