arr = | -1 | 54 | 53 | 55 | 52 | 50 |
0 1 2 3 4 5

Note :→ i = n/2 to i > 0

Non-leaf nodes should be heapified, all the remaining leaf nodes (n/2 + 1) will be done automatically.

Max Heap :→ | 55 | 53 | 54 | 52 | 50 |
1 2 3 4 5

↓ HeapSort :→ arr[i] arr[size]

array → heap → heapify(arr, n, i) → heapify

O(n) → no. elements

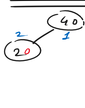
s(a[i], a[size]) size--

log n
n log n
Sorted

50	53	54	52	55
54	55	50	54	55
52	53	50	55	54
53	52	50	53	54
50	52	53	54	55
52	50			
50	52			

arr[] 60, 80, 40, 20, 90, 70

Max heap :→



| -1 | 90 | 80 | 70 | 20 | 60 | 40 |
0 1 2 3 4 5 6

→ O(n)

↳ Sort

| 90 | 80 | 70 | 20 | 60 | 40 |
1 2 3 4 5 6

↓ size

swap(a[1], a[size]) | 20 | 80 | 70 | 20 | 80 | 90 |
1 2 3

size--
heapify( )

n log(n)

0	1	2	3	4	5	6
0	1	1	0	1	0	1
left	0	1	2	3	4	5
right	6	5	4	3	2	1

left++ right--

if (left < right)
a[left] > a[right]
l++
r--

| 1 | 2 | 3 | 2 | 3 |
0 1 2 3 4

i → for(int i=1, i<n, i++)
j → if(arr[j] != arr[i])
j++
arr[j] = arr[i]
arr[j] = arr[i]

j = 0
i = 1

0, 1, 2
return
2+1 = 3

| 1, 2, 3, 5, 6, 7 |
0 1 2 3 4 5

O(n)

(element = index + 1) ×

(i+1)th = a[i] + 1

S(n) = 8 × 9 / 2

S(a) = 36 − 32 = 4

| 1, 2, 3, 5, 6, 7, 8 |
0 1 2 3 4 5 6
s e

(6-10 USA)

Missing Element from Sorted Array

Constraint → Time Complexity

ele = index + 1

O(log N) → sorted

mid = s + (e-s)/2 = s+e/2 ↳ Binary Search

m = 3

if arr[mid] = mid + 1 = 3+1 ×

e = m-1

come to the left ; 1,2,3,5,6,7,8

e = i+1 e = m-1 ; 0 1 2 (mid = 3)

⇒ if (mid == 0 || arr[mid-1] == mid

① ② arr[2] == mid

return m+1 3 == 3

↳ s = m+1 1,2,3,4,5 n = 6