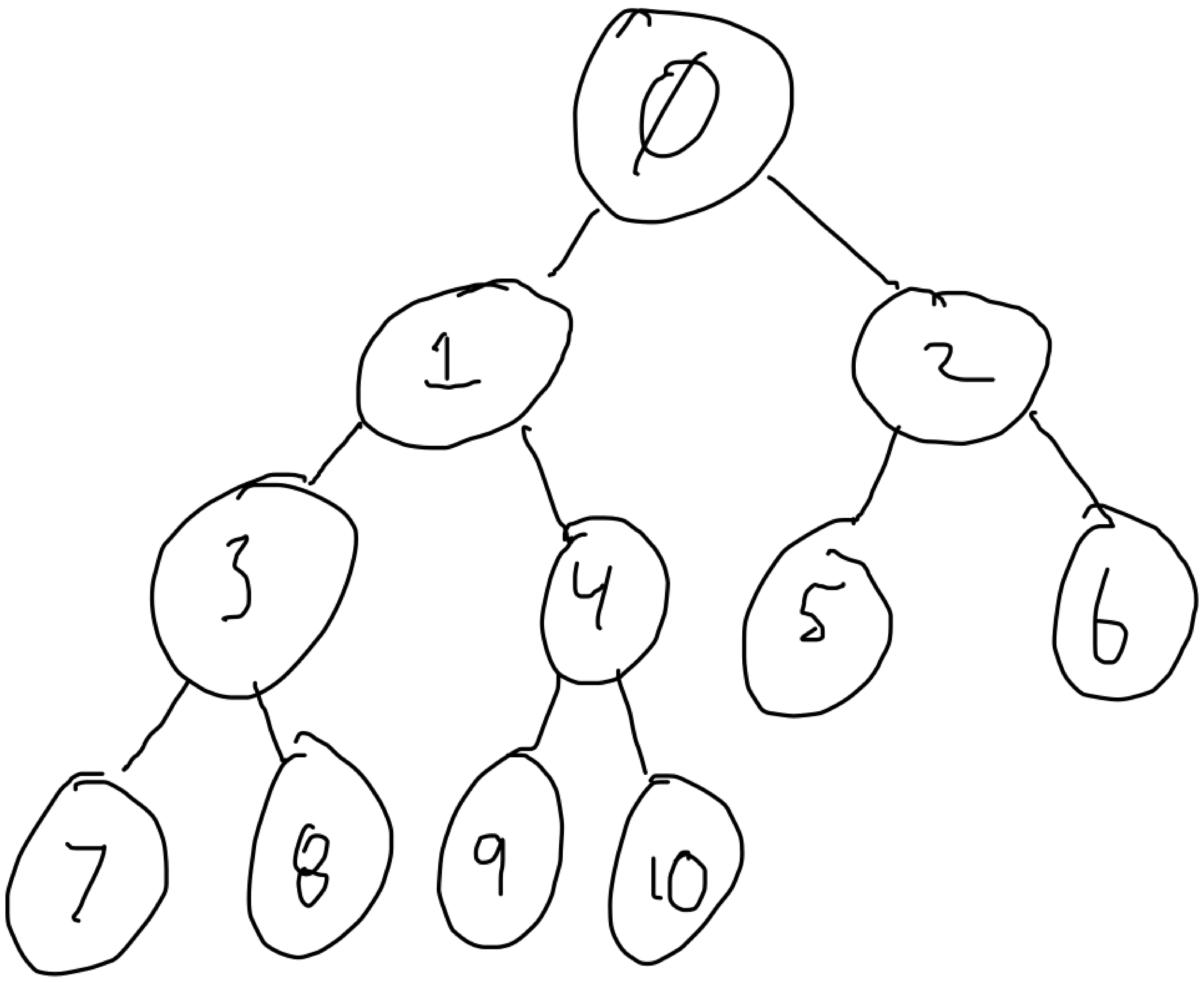
Trees in Arrays

0	1	2	3	4	5	6	7	8	9	10



node	parent	LC	RC
0		1	2
1	0	3	4
2	0	5	6
3	1	7	8
4	1	9	10
5	2	11	12
6	2	13	14

if a node is at index i

parent is at $\left\lfloor \frac{i-1}{2} \right\rfloor$

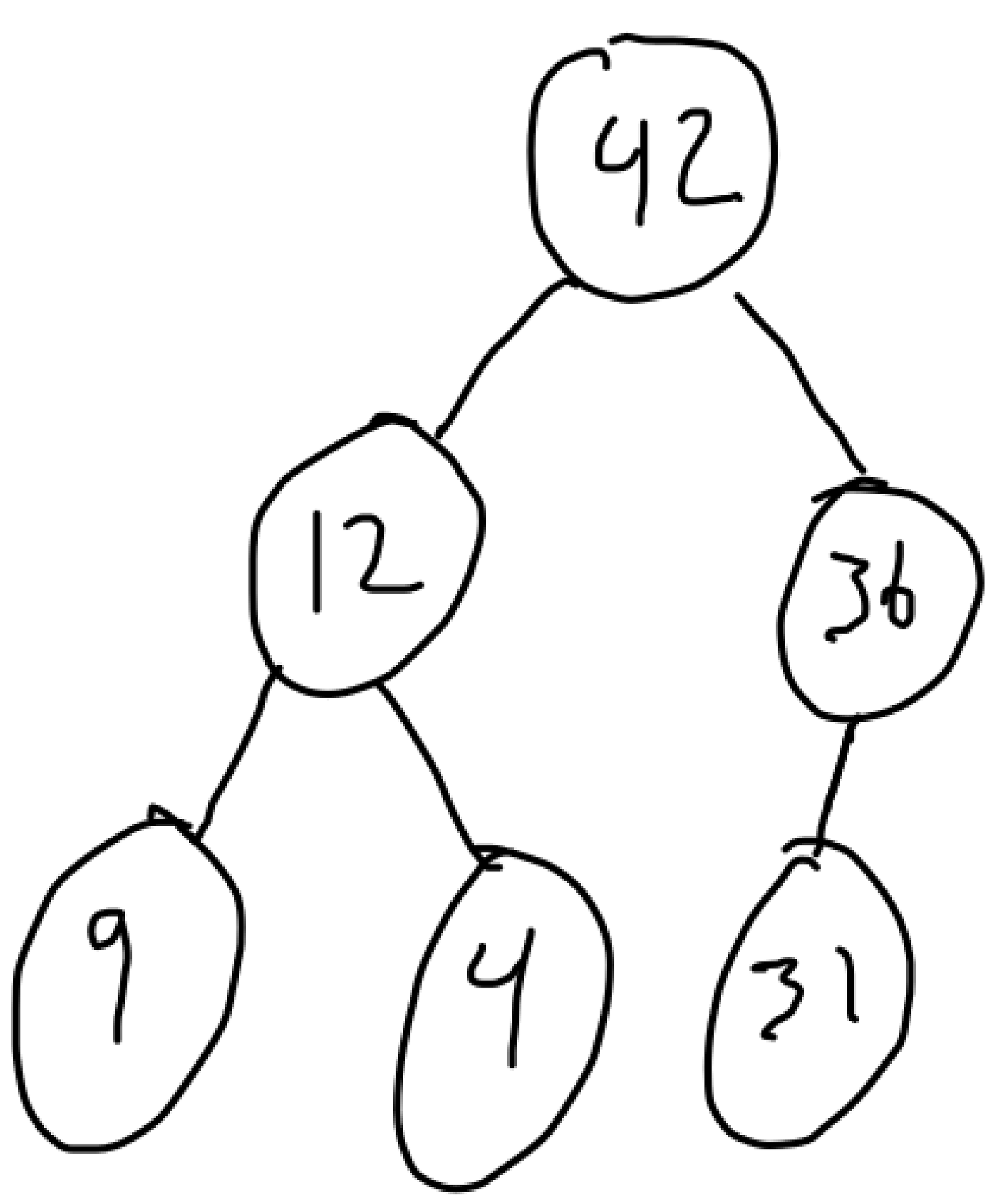
LC is at $i \times 2 + 1$

RC is at $i \times 2 + 2$

Heaps

- a binary tree
- that is left-complete
- can be in an array
- for a max heap (typical):
  - parents $\geq$ children
    - recursive
- for a min heap
  - parents $\leq$ children
    - recursive

Examples:



Heaps are often used for priority queues

Heap Operations

- add a value
  - add at the next available leaf - $O(1)$
  - percolate up
    - while the new value is greater than its parent, swap with parent - $O(h) = O(\lg n)$
- remove max value (root)
  - remove root value - $O(1)$
  - replace with last leaf
  - percolate down
    - while one child of the leaf value is greater, swap with the greater child - $O(h) = O(\lg n)$

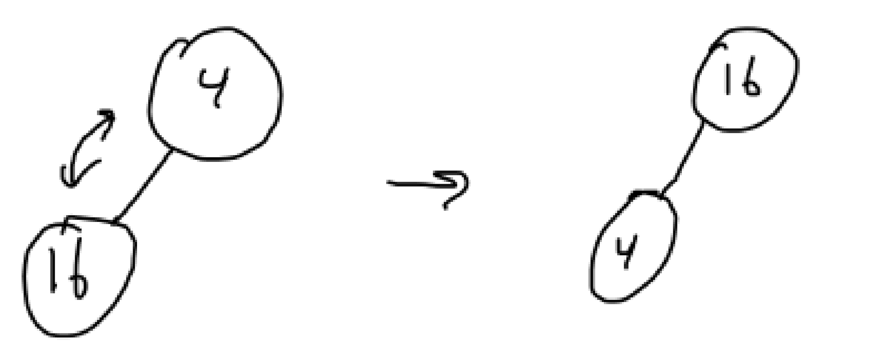
Both heap operations are $O(\lg n)$

Heap Examples

add 4

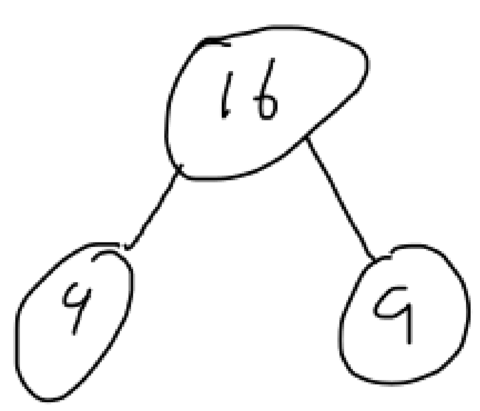
[4]

add 16



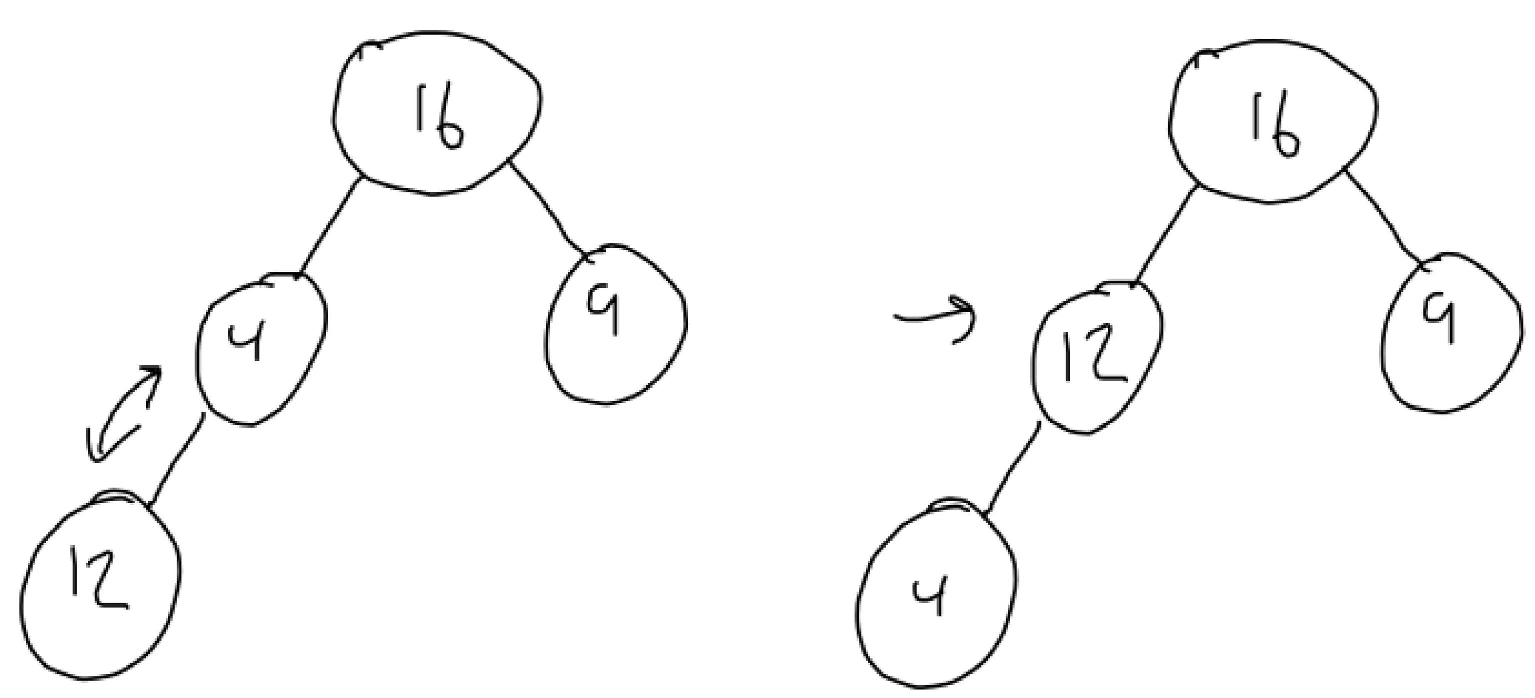
[4, 16] → [16, 4]

add 9



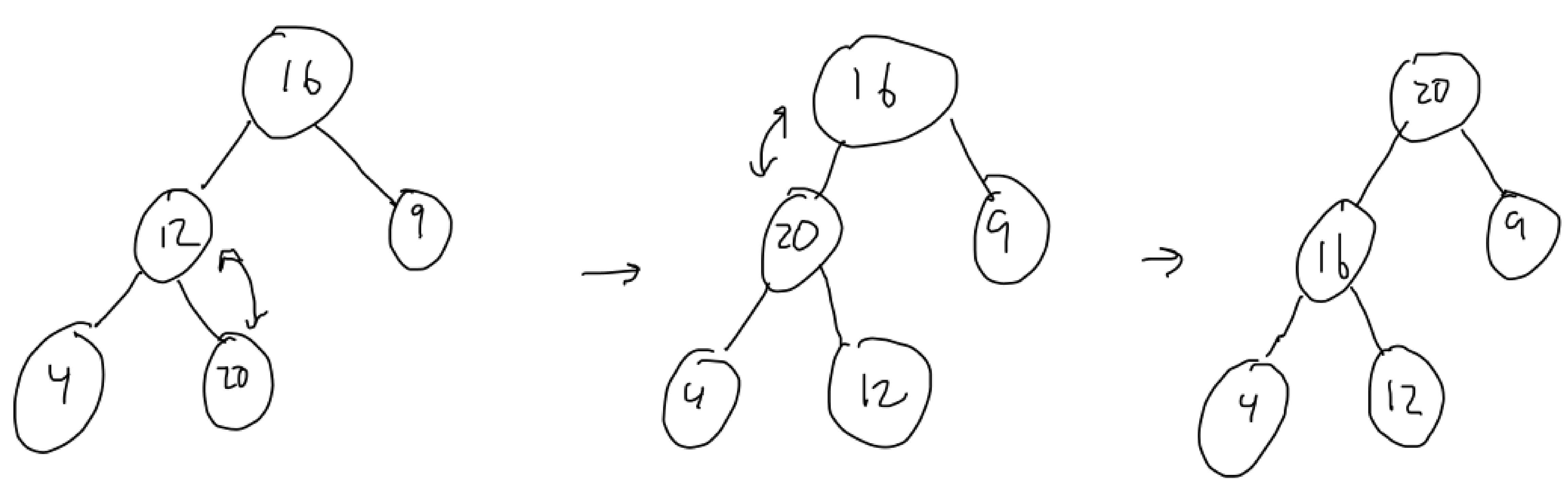
[16, 4, 9]

add 12



[16, 4, 9, 12] → [16, 12, 9, 4]

add 20

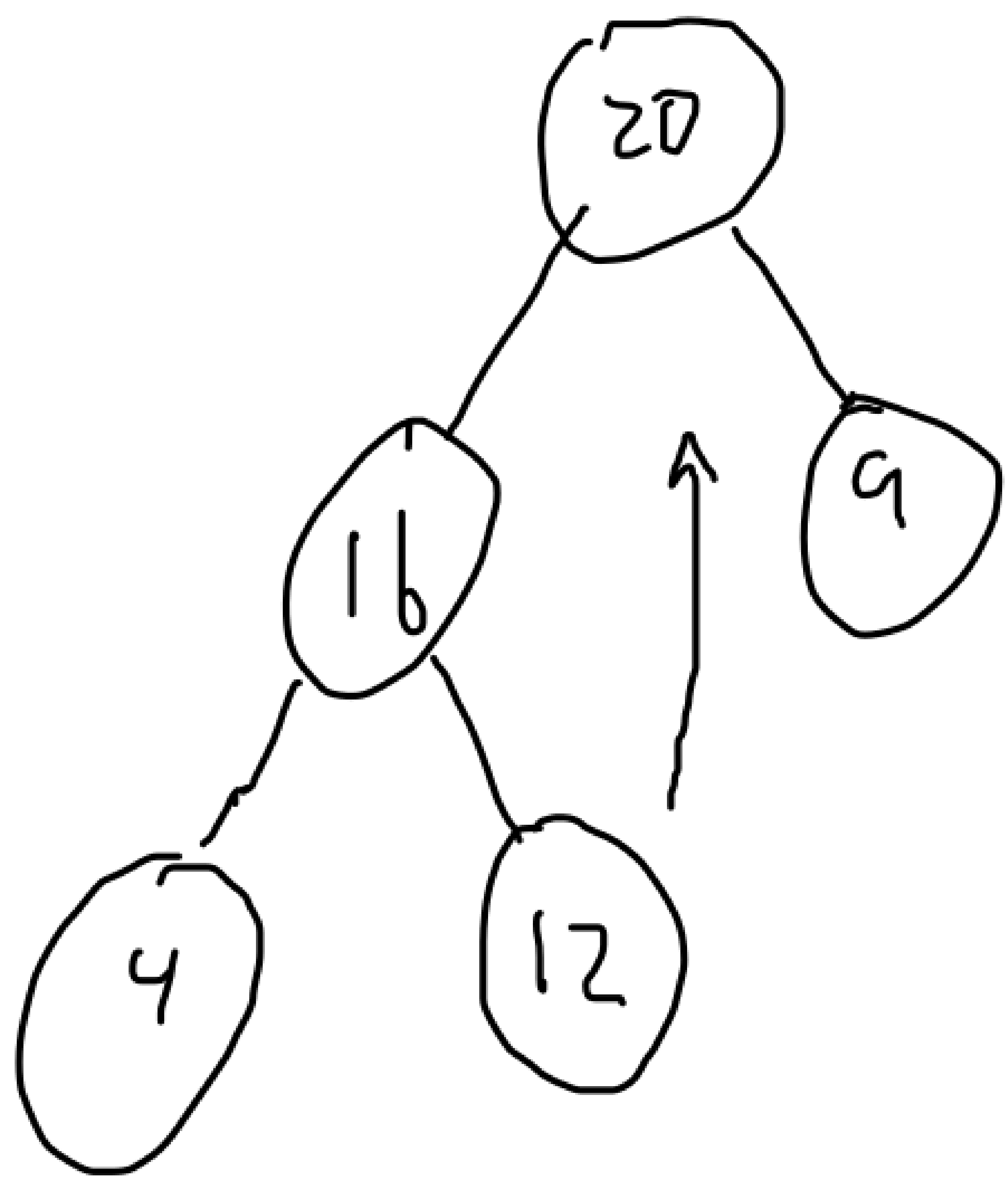


[16, 12, 9, 4, 20] → [16, 20, 9, 4, 12]

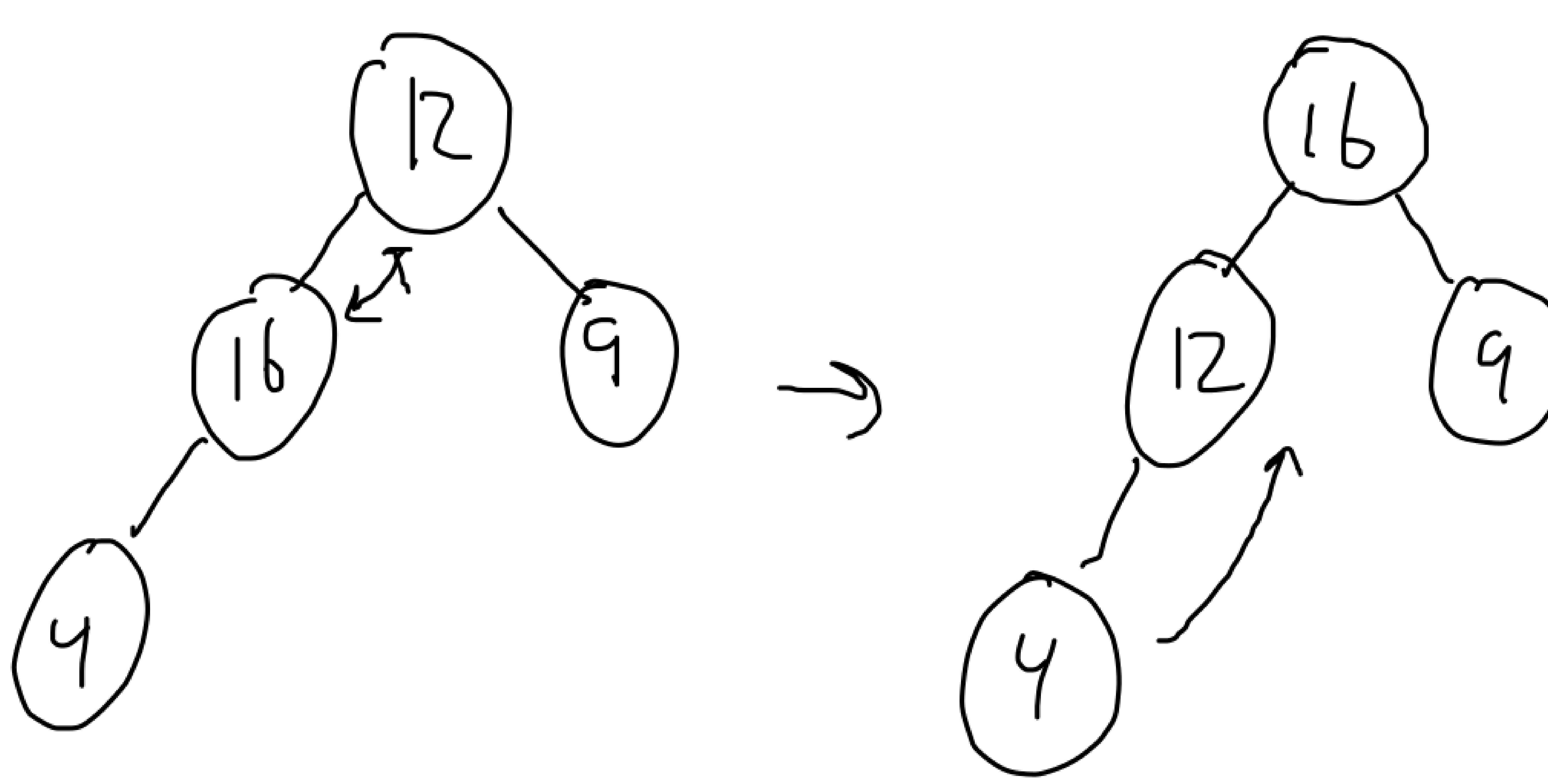
↓

[20, 16, 9, 4, 12]

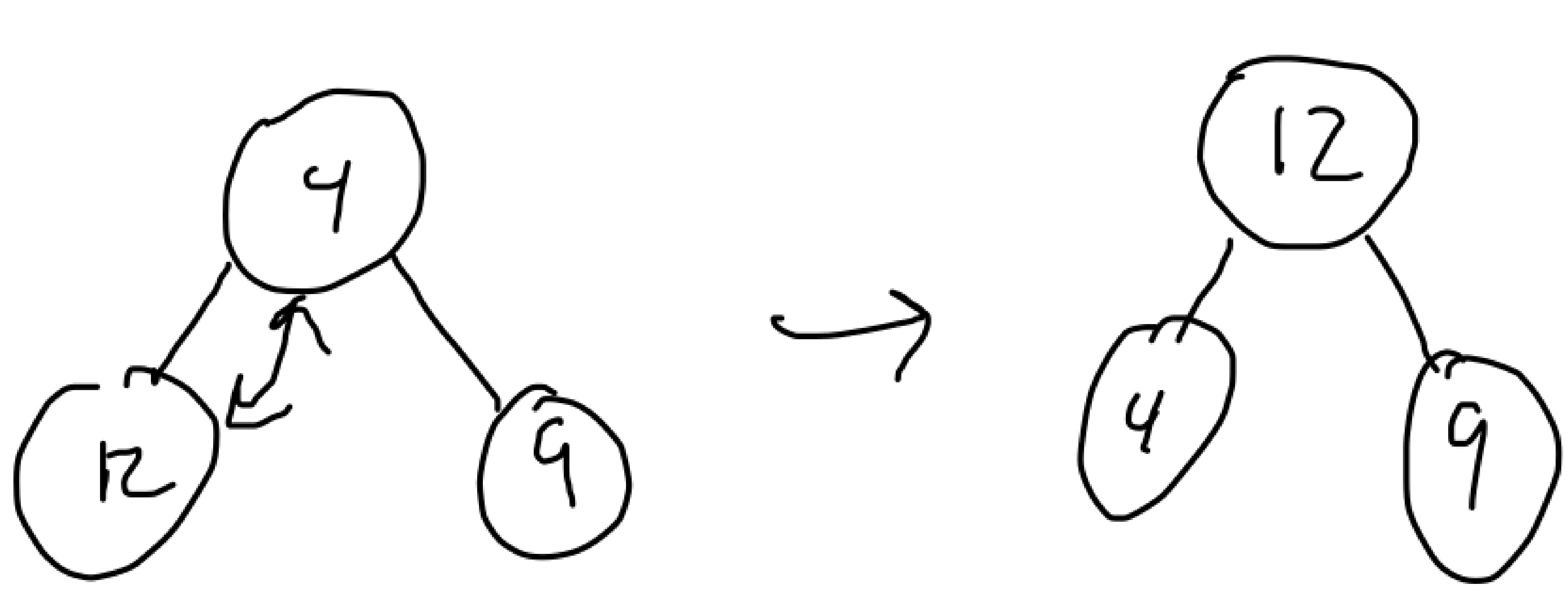
Heap operations



remove (get 20)



remove (get 16)

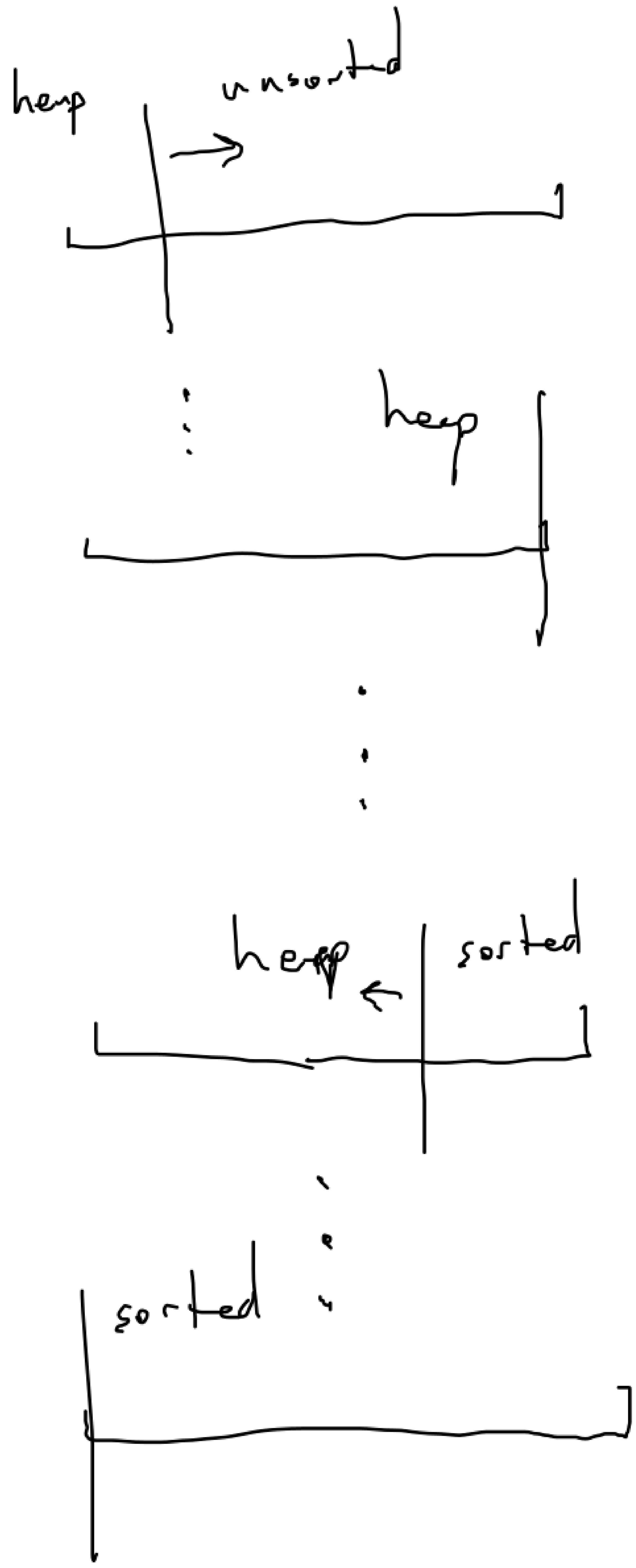


Heapsort

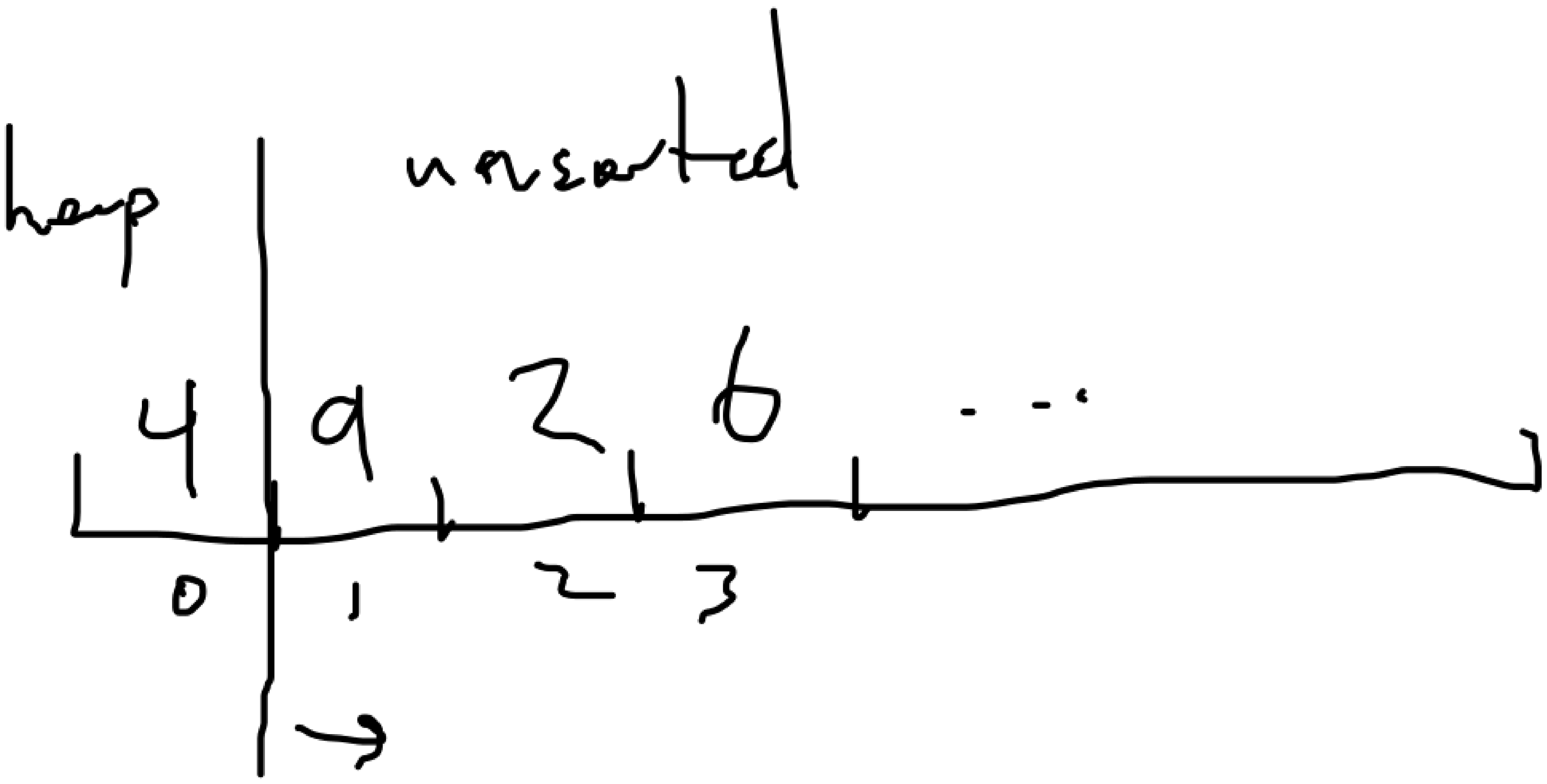
- add all items to a heap — $n \times O(\lg n)$
- remove all items, putting them right to left — $n \times O(\lg n)$

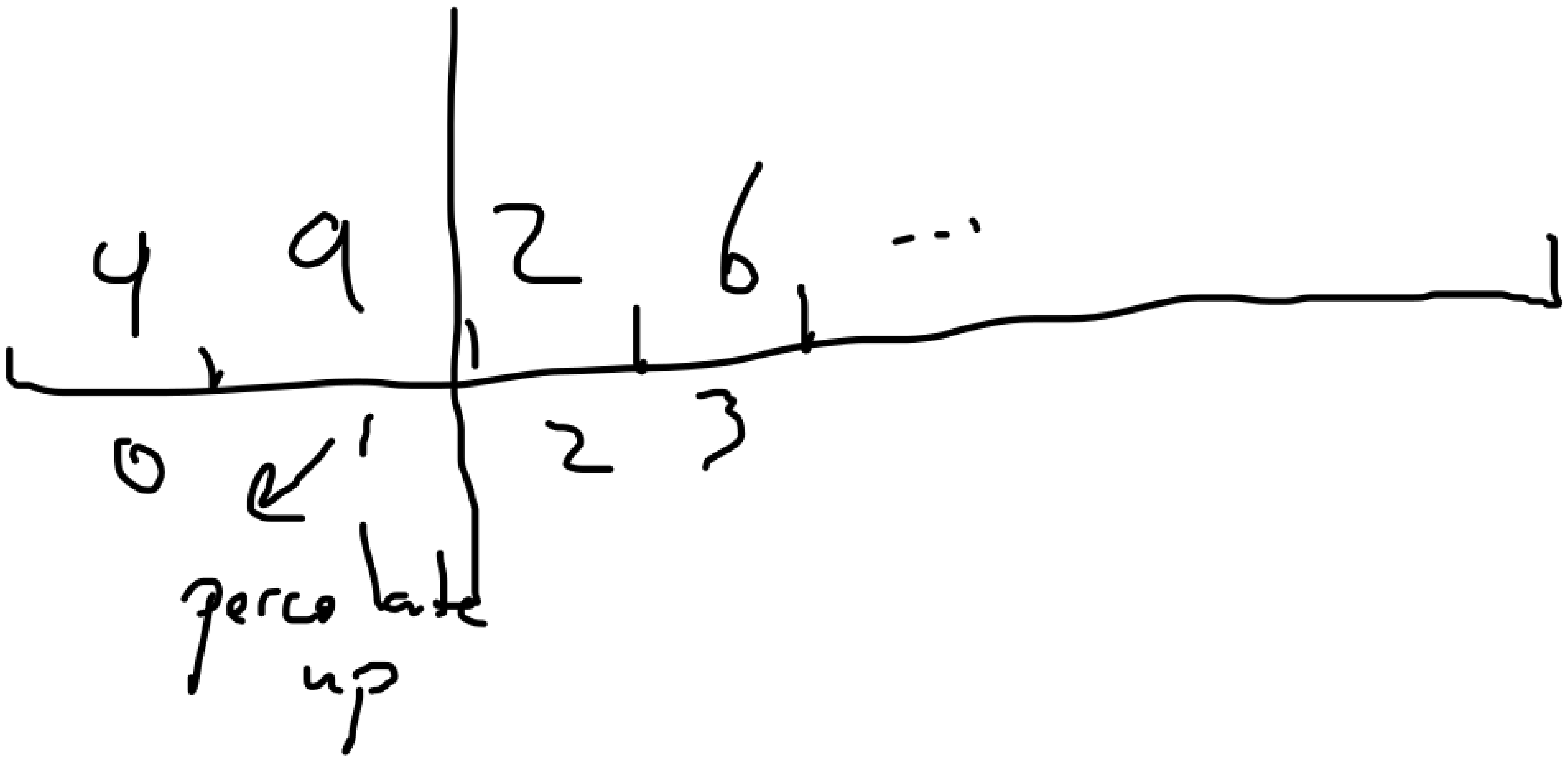
$$\frac{\begin{aligned} &n \times O(\lg n) \\ &n \times O(\lg n) \end{aligned}}{O(n \lg n)}$$

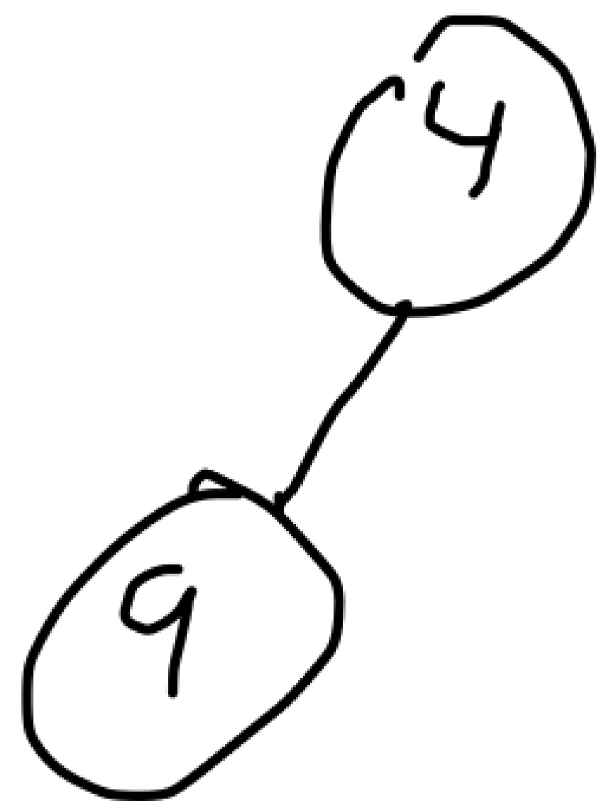
As we add, the heap is the first half of the array and the unsorted portion is the right

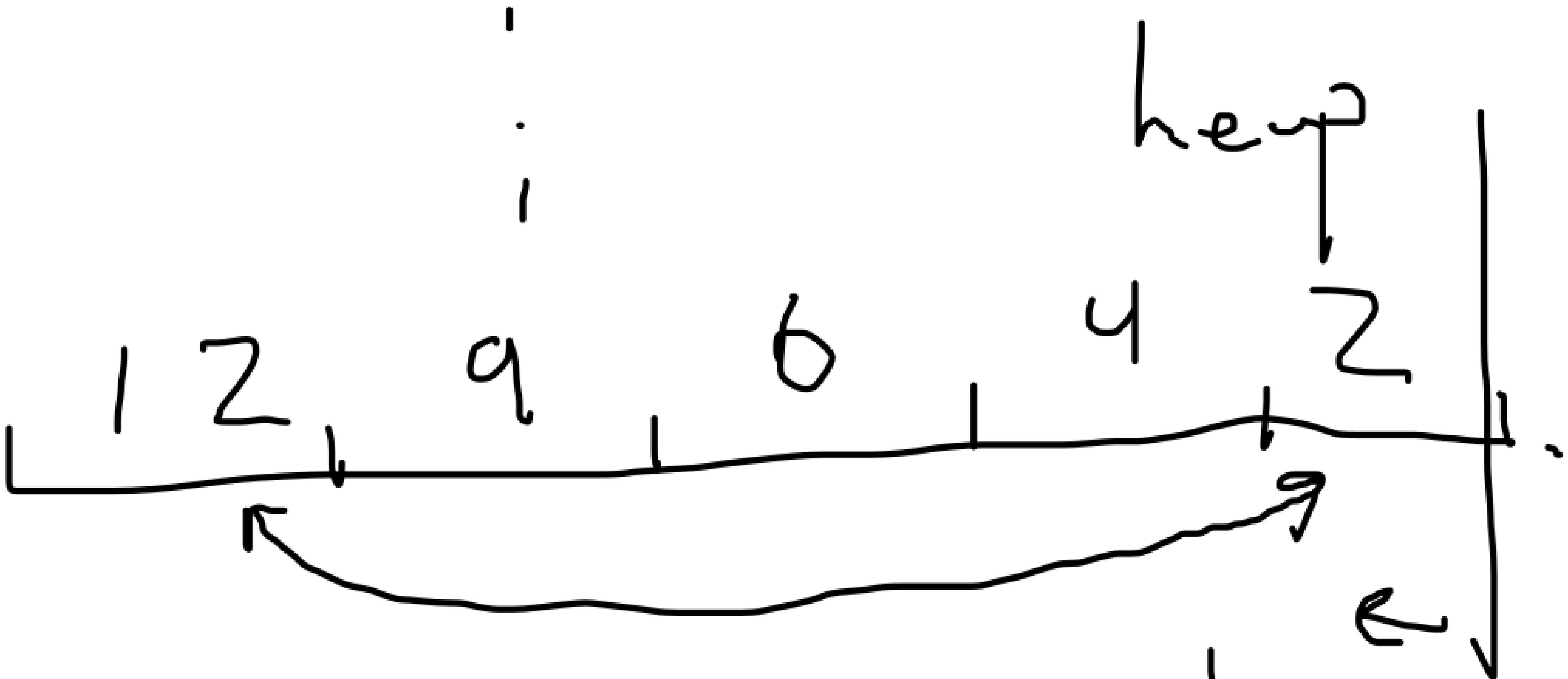


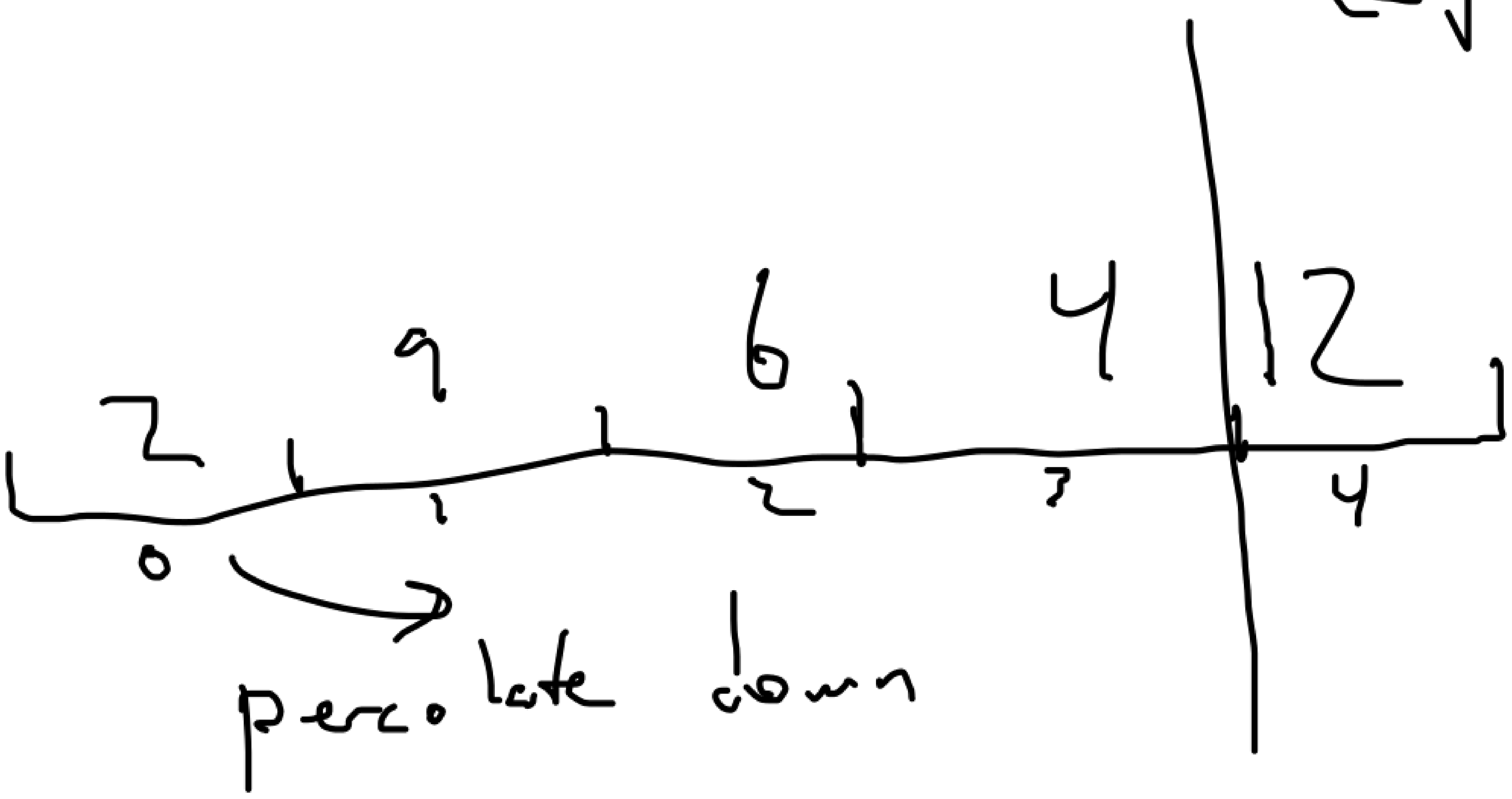
As we remove, the right side will be sorted and the left side will be the heap


heap
unsorted
4 9 2 6 ...
0 1 2 3


4 9 2 6 ...
0 1 2 3
percolate up


4
9


heap
12 9 6 4 2


2 9 6 4 12
0 1 2 3 4
percolate down