8.3)

1. Min: $2^{n-1} - 1 + 1 = 2^{n-1}$

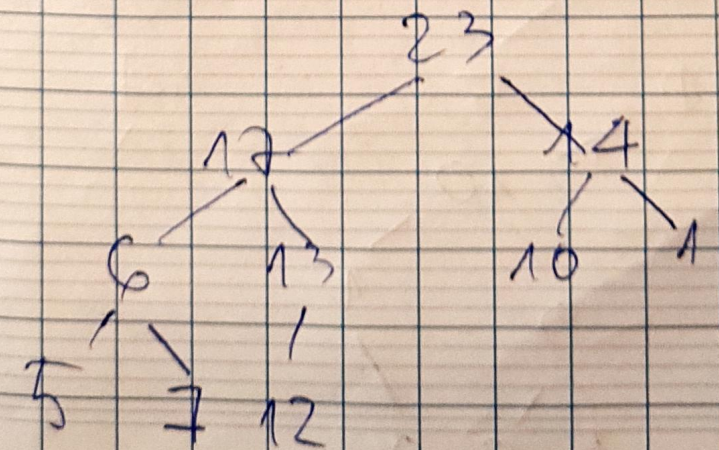
Max: $2^{n} - 1$

2. The smallest element stays at the bottom layer of the heap

3. ~~The array is sorted~~

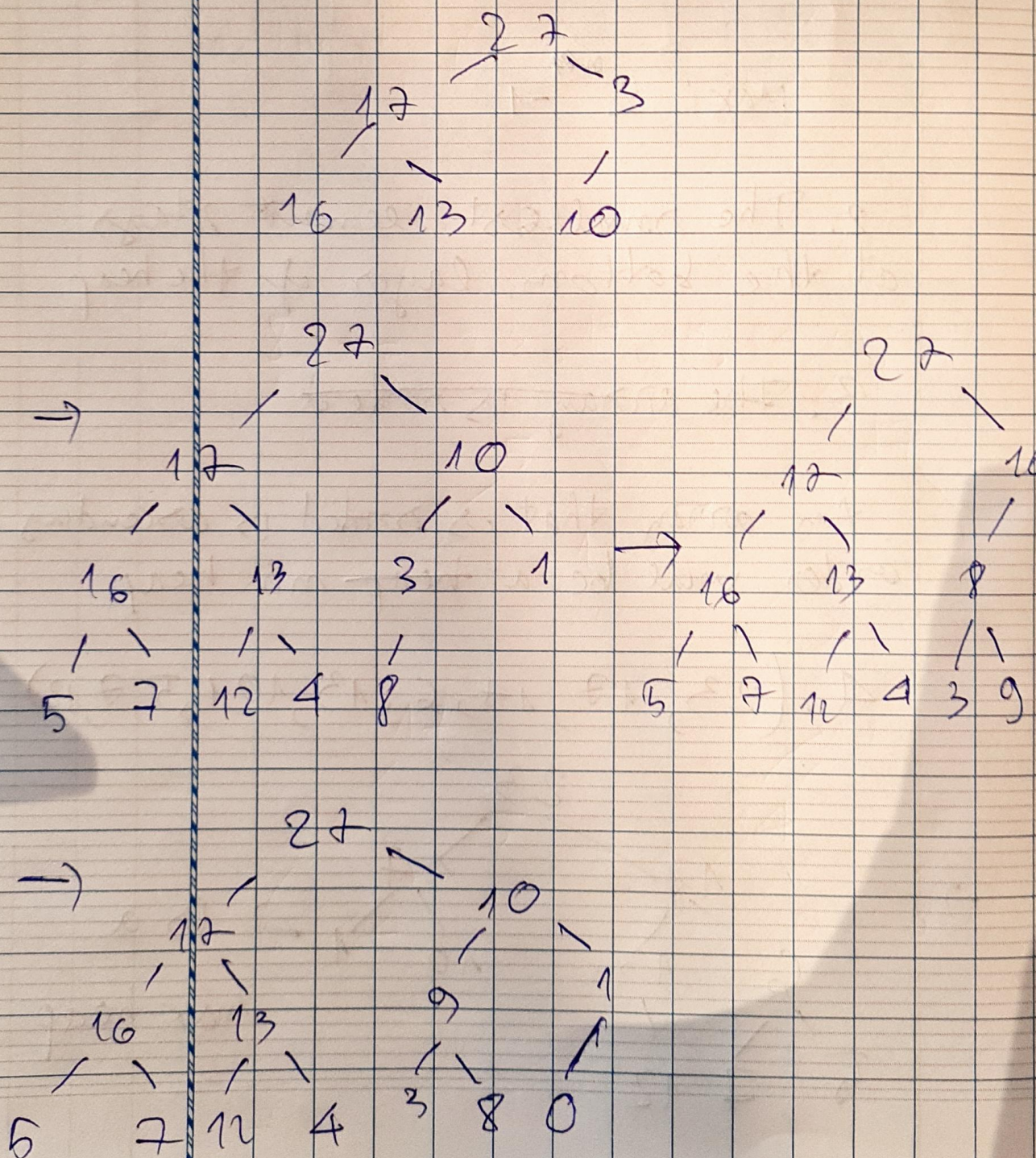
An array that is sorted in ascending order will be a ~~heap~~ min heap

4. (23, 17, 14, 6, 13, 10, 1, 5, 7, 12)



⇒ It is a max heap.

5. [27, 17, 3, 16, 13, 10, 1, 5, 7, 12, 4, 8, 2, 0]



6. No effect, A[i] > children so no need to swap ~~nodes~~ elements.

7. ~~Consider a h^{th} height heap.~~

minimum height ~~of heap is~~

~~a h^{th} ht~~

A heap with h-height will have at least 2^{n-1} elements.

$\Rightarrow$ size(A) $= 2^{n-1}$ ∴

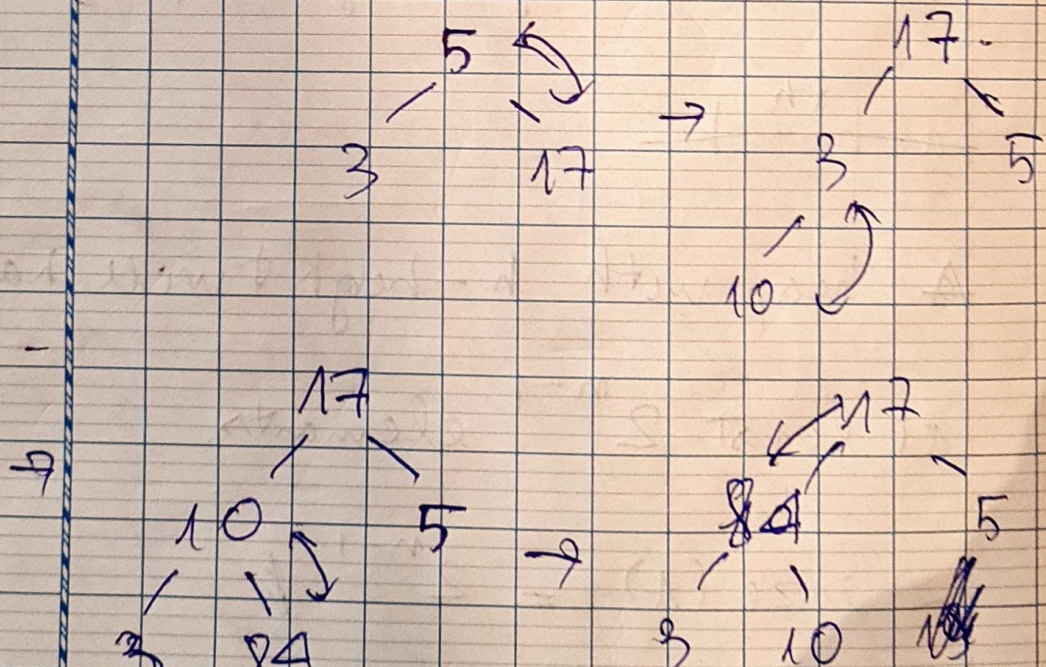
$\Rightarrow \frac{size(A)}{2} = 2^{n-2}$ the

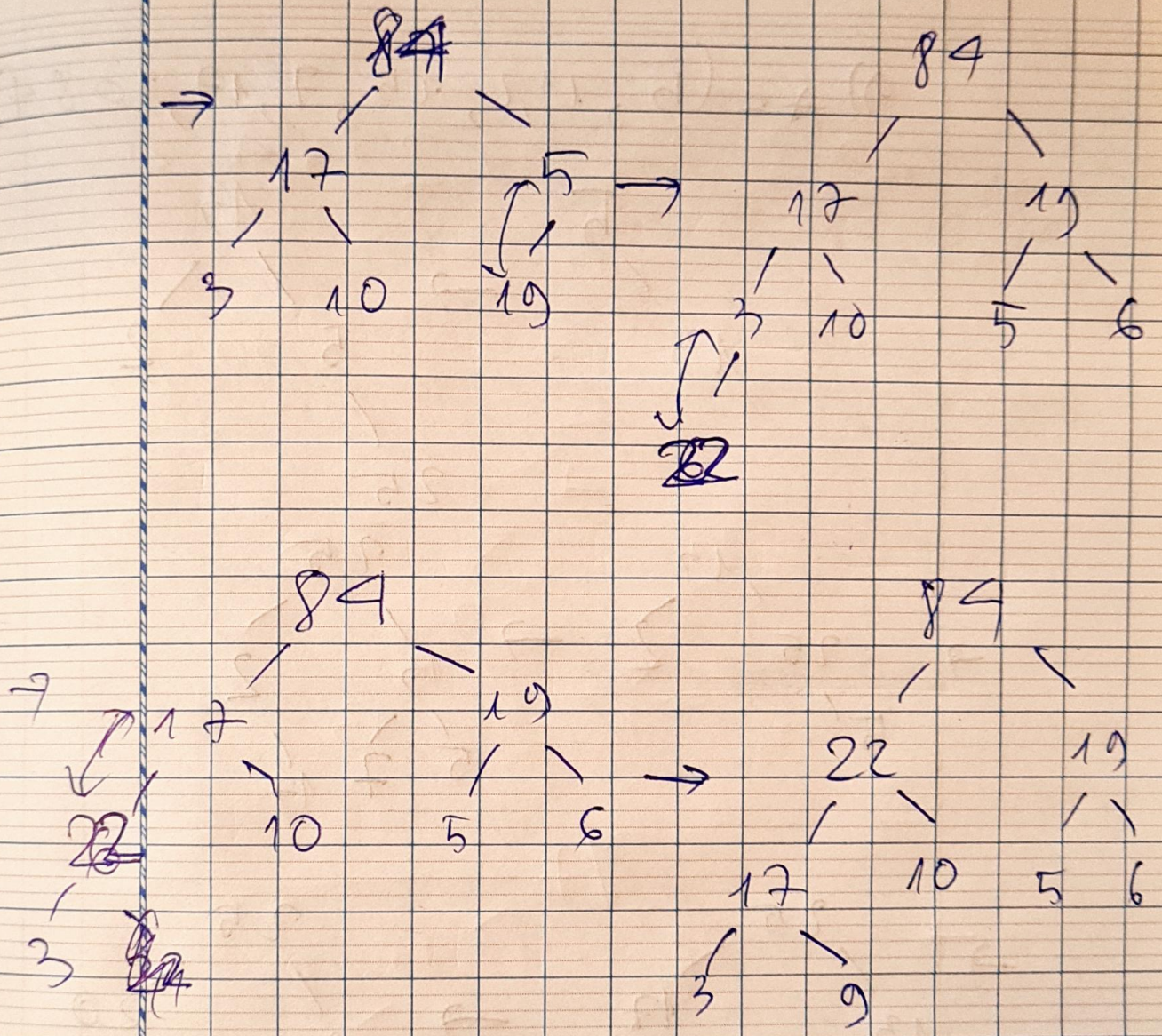
$\Rightarrow$ ~~at most $\frac{size(A)}{2}$ nodes~~

→ $\frac{size(A)}{2}$ ⇒ the 2nd lowest layer of the heap ⇒ if $i > \frac{size(A)}{2}$ →

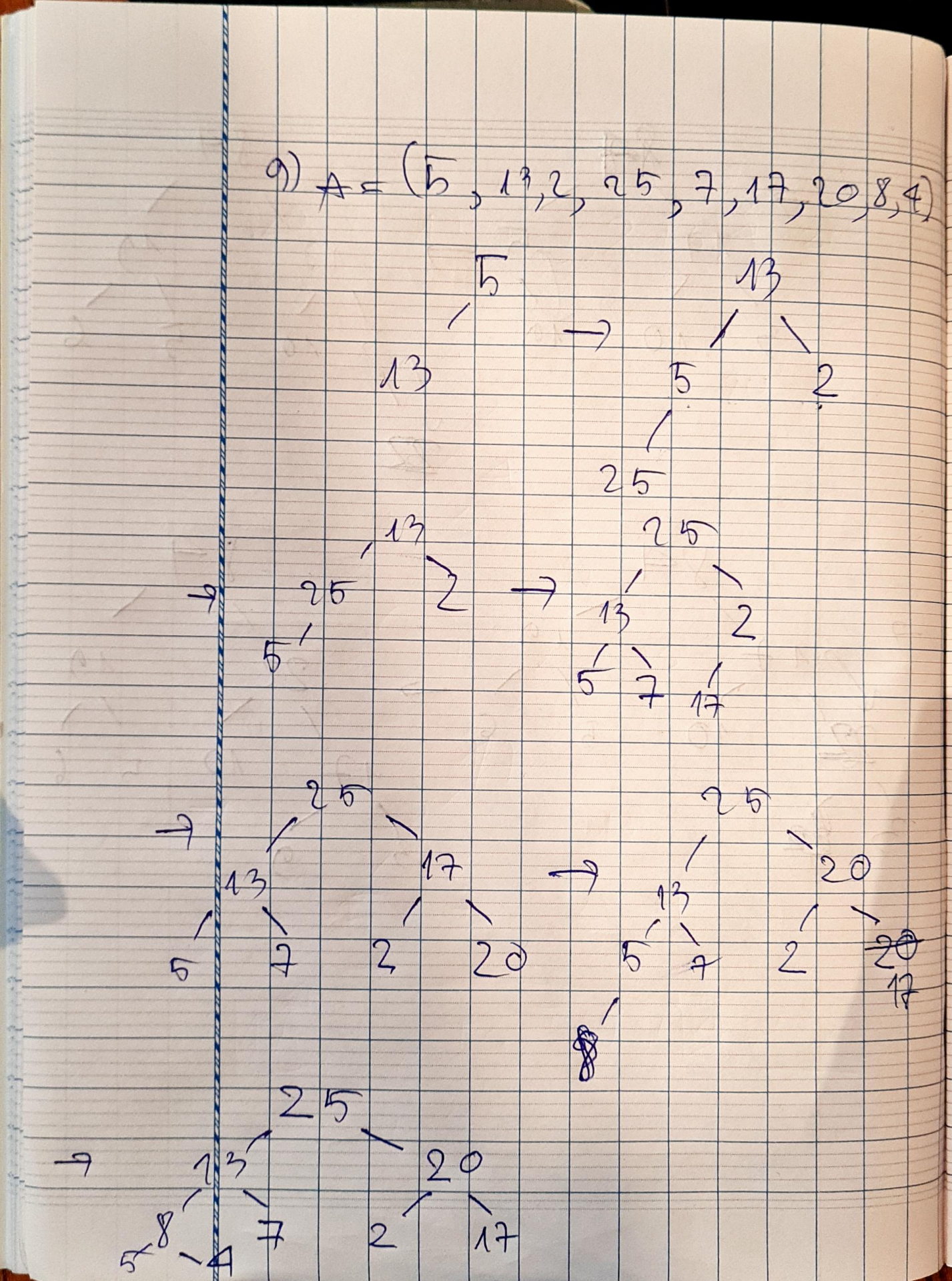
all elements there are leaves

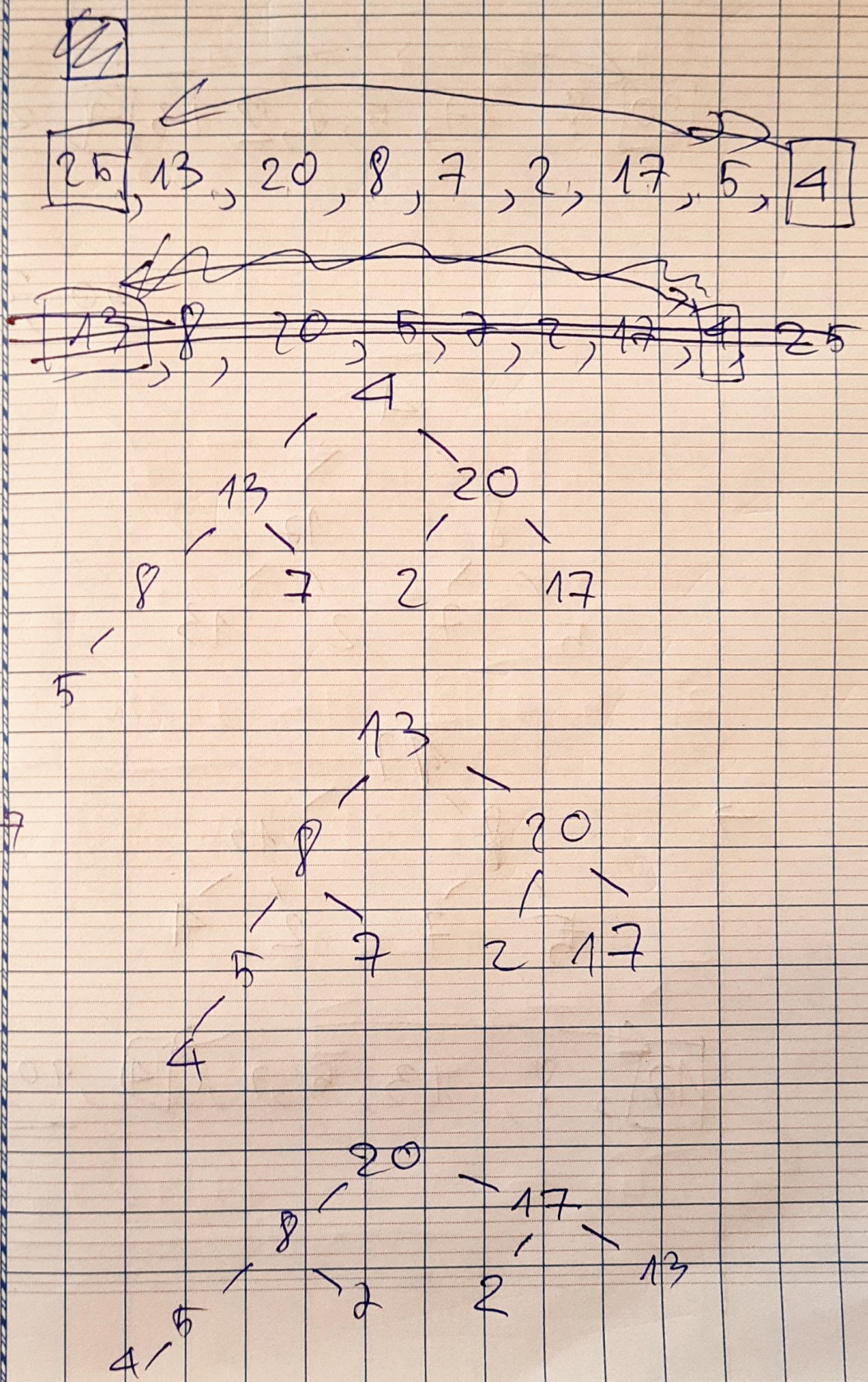
8. A = (5, 3, 17, 10, 84, 19, 6, 22, 9)

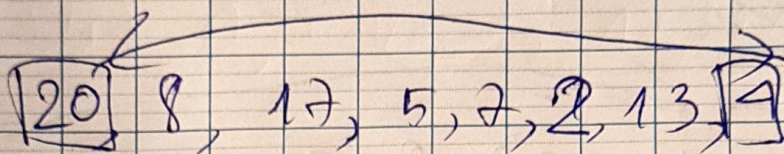



84
17
5
3
10
19
84
17
19
3
10
5
6
22
84
7
17
19
22
10
5
6
84
22
19
22
10
17
5
6
3
9

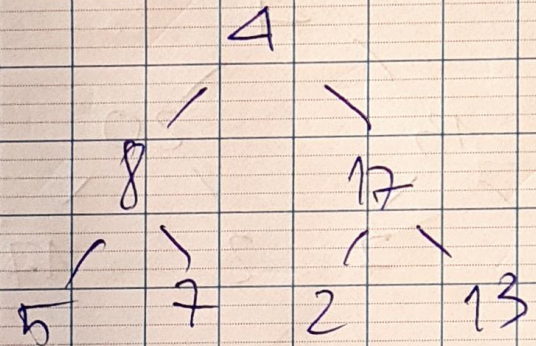
9) A = (5, 13, 2, 25, 7, 17, 20, 8, 4)

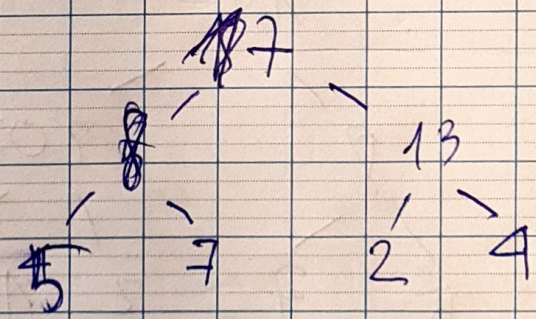


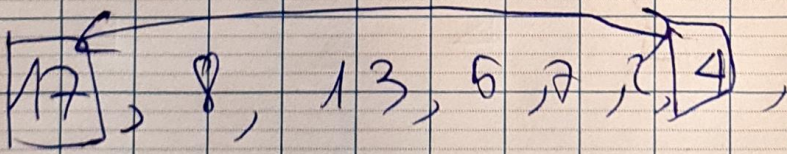


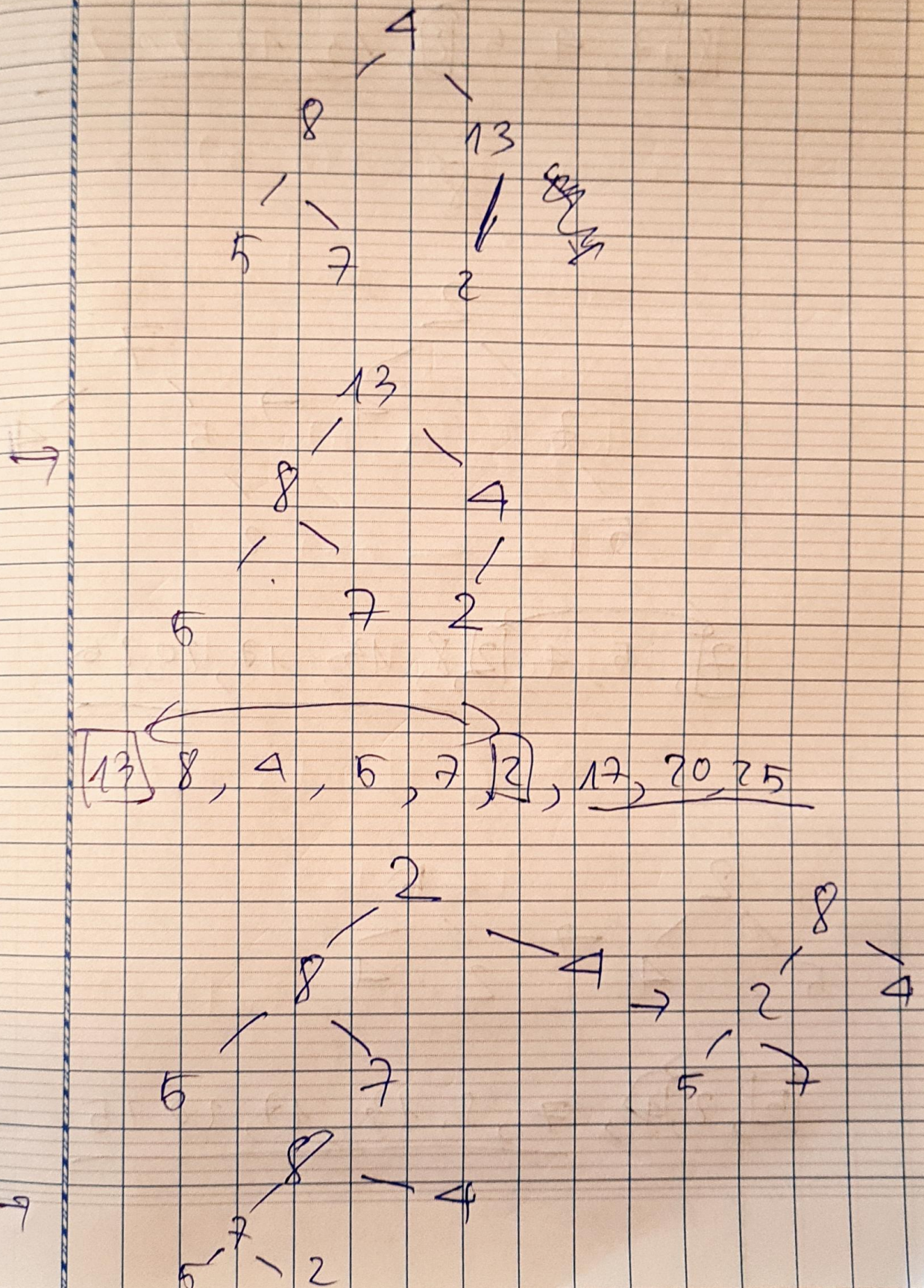


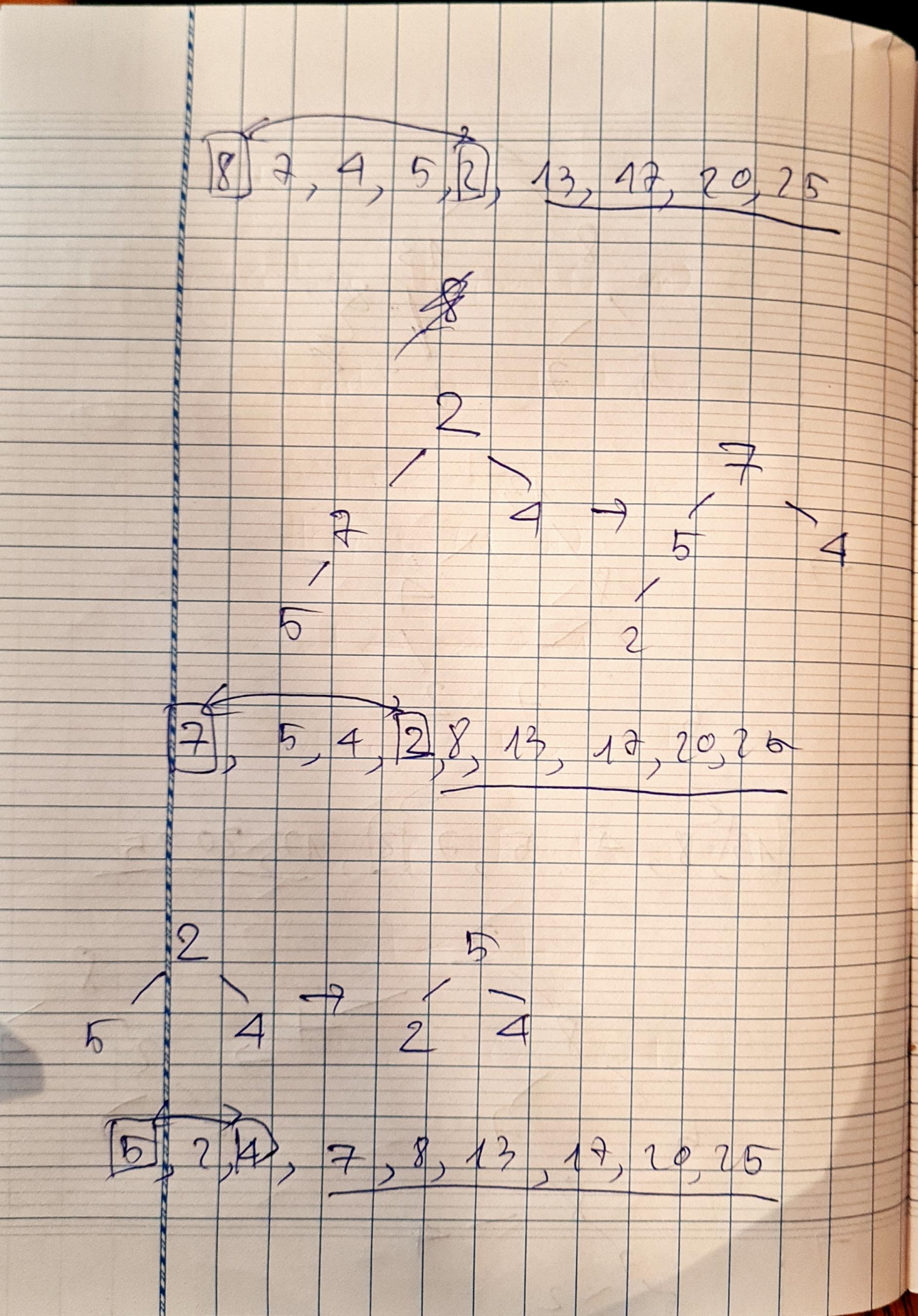
~~20, 25~~

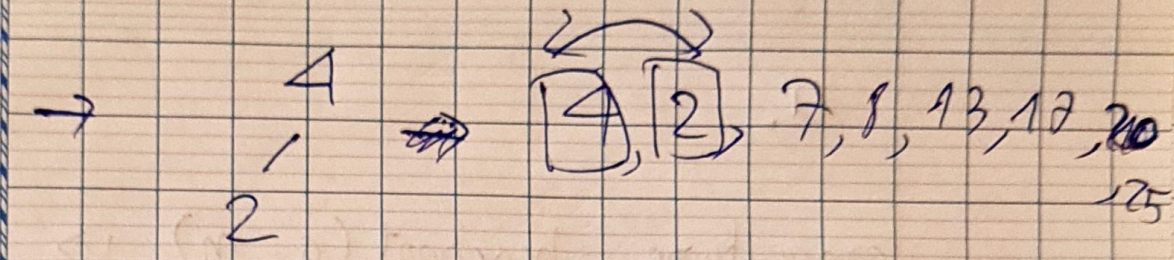







4
8
13
5
7
2
13
8
4
5
7
2
13, 8, 4, 5, 7, 2, 17, 20, 25
2
8
4
5
7
8
4
5
7
8
4
5
7
2


8, 7, 4, 5, 2, 13, 17, 20, 25
2
7
4
5
7
5
4
2
7, 5, 4, 2, 8, 13, 17, 20, 25
2
5
5
4
2
4
5, 2, 4, 7, 8, 13, 17, 20, 25



→ 2, 4, 7, 8, 13, 17, 20, 25.

10.

if sorted ascending, heap will be built again ⇒ $O(n \log n)$

if sorted decreasing, max element when removed will still result in $O(n \log n)$

11.

```
procedure heapsort(a, n):
  input: array a, length n

  Build-Heap(a, n).

  end ← n-1
  while end > 0 do
    swap(a[end], a[0])
    end ← end-1
    Heapify(a, end).
```