CHAPTER 6

Pn 6.1-3

~~by contract~~

Let us suppose that root does not have the max element in a max-heap ⇒ there exists a child of root ⊗ such that

$$A[\text{root}] < A[\text{child}(\text{root})]$$

$$\text{root} = \text{parent}(\text{child}(\text{root}))$$

$$\Rightarrow A[\text{~~root~~ parent}(\text{child}(\text{root}))] < A[\text{child}(\text{root})]$$

but $A[\text{parent}(i)] \geq A[i]$

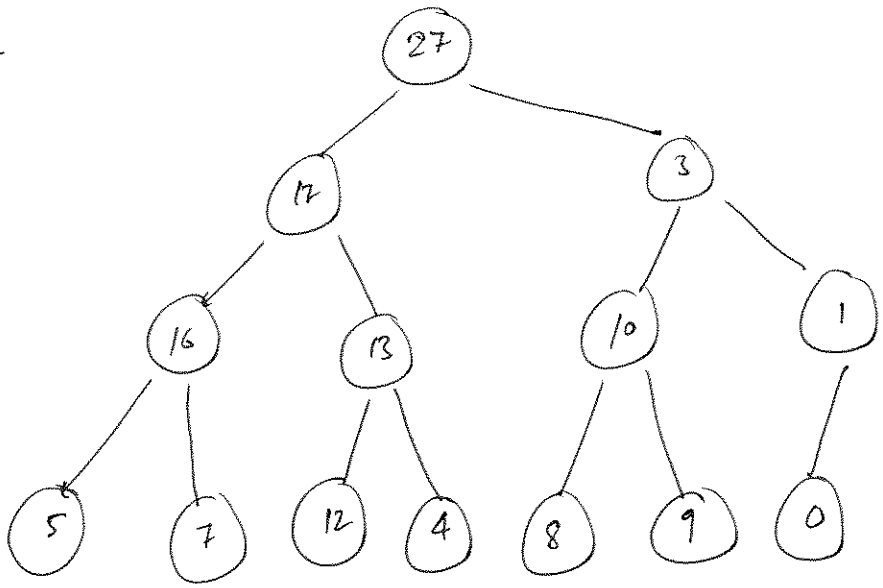
which is a property of max-heap

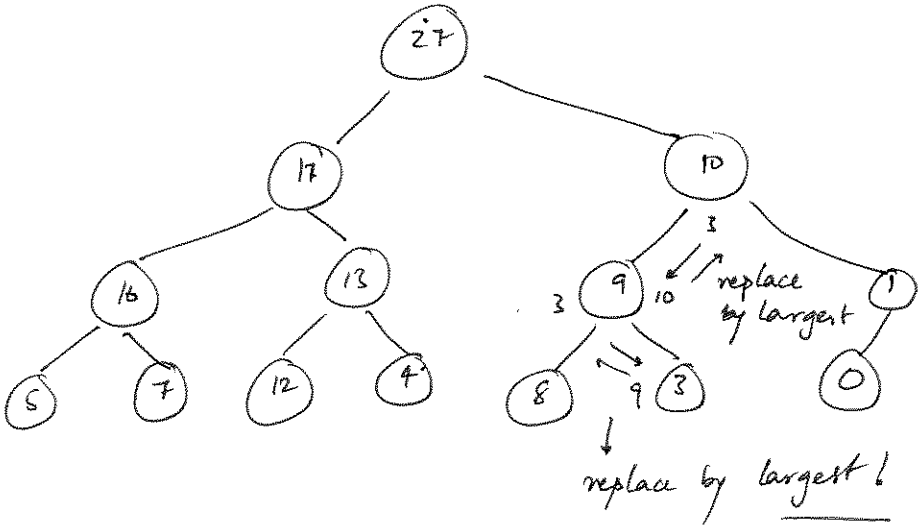
⇒ the heap is not completely heapified.

⇒ when the max-heap is made

$$A[\text{root}] \geq A[\text{child}(\text{root})]$$

Qn 6.2 -1





Max-Heapify

Qn 6.2-2

$T(n)$.

```
Min_Heapify (A, i)
    l = left(i)
    r = right(i)
    if l ≥ A.heap-size and A[l] ≮ A[i]
        ~~largest~~ smallest = l
    else smallest = i
    if r ≥ A.heap-size and A[r] ≮ A[~~largest~~ smallest]
        smallest = r
    if smallest ≠ i
        exchange A[i] and A[smallest]
        Min-Heapify (A, smallest)
```

(The first two `if` statements are bracketed together and marked $\Theta(1)$. The last `if` statement is marked $O(1)$, with an arrow pointing to $T(2n/3)$? HOW?)

RT Analysis: $T(n) = T\left(\frac{2n}{3}\right) + \Theta(1)$.

$\Rightarrow a = 1,\ b = 3/2,\ \log_b a = 0,\ k = 0.$

$\Theta(n^0 \log n) = \Theta(\log n)$.

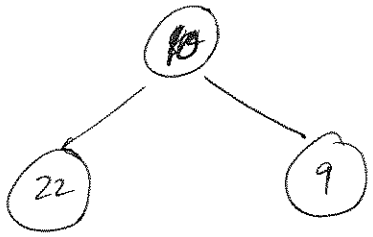
Proof: ~~PBSK~~: Proof by induction.

Qn 6.2-3

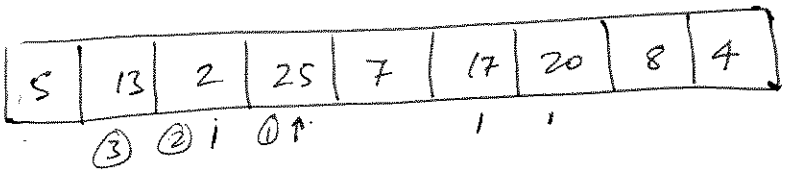
```
Max_heapify(A, i)
l = LEFT(i)
r = RIGHT(i).

While i ≤ A.length
    l = LEFT(i)
    r = RIGHT(i)
    if l ≤ A.heapsize and A[l] ≥ A[i]
        largest = l
    else largest = i
    if r ≤ A.heapsize and A[r] ≥ A[largest]
        largest = r
    if largest ≠ i
        exchange A[largest] and A[i]
        i = largest.
    else
        i++.
```

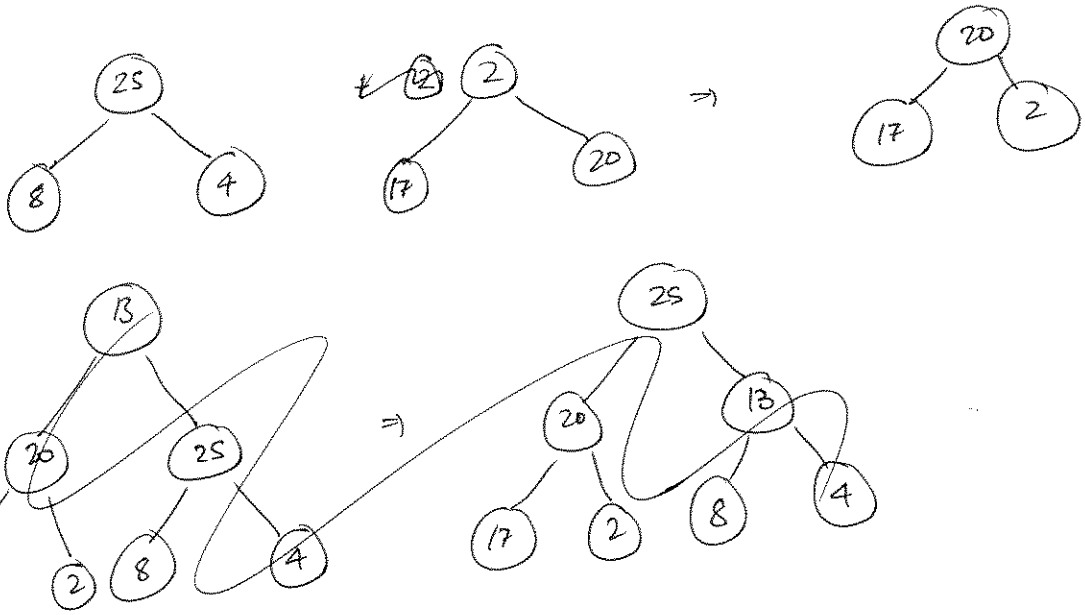
On 6.3-4

5	3	17	10	84	19	6	22	9

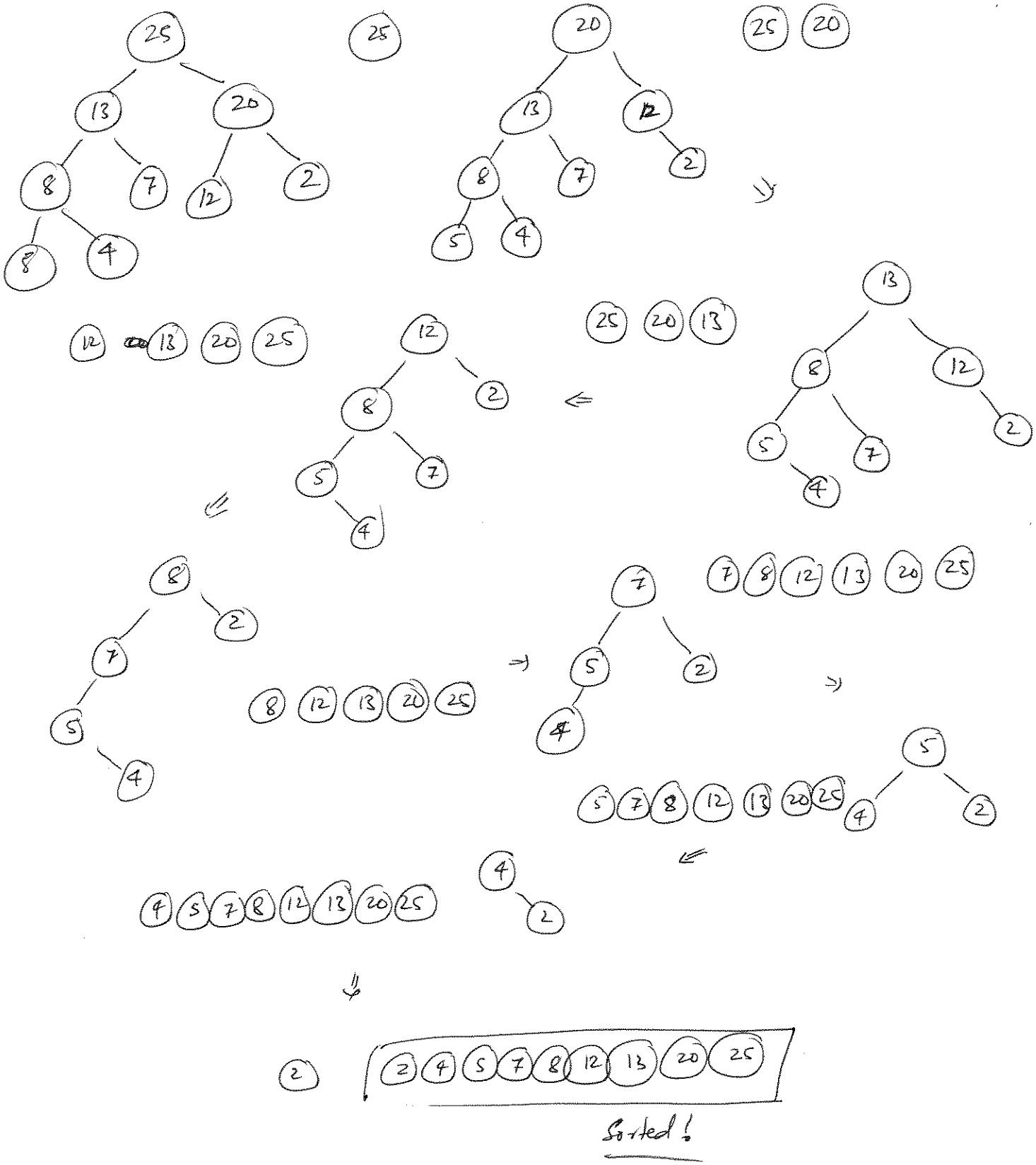


Qn 6.4-1





Build max heap


25
13 20
8 7 12 2
8 4
25
20
13 12
8 7 2
5 4
25 20
20 25
12 13 20 25
12
8 2
5 7
4
25 20 13
13
8 12
5 7 2
4
8
7 2
5
4
8 12 13 20 25
7
5 2
4
7 8 12 13 20 25
5
4 2
5 7 8 12 13 20 25
4
2
4 5 7 8 12 13 20 25
2
2 4 5 7 8 12 13 20 25
Sorted!

6.4-2

HEAPSORT(A)

1. BUILD-MAX-HEAP(A)
2. for i = A.length down to 2
3. exchange $A[1]$ and $A[i]$
4. A.heapsize = A.heap-size − 1
5. MAX-HEAPIFY(A, 1)

Post: Sorted Array $A[1 \ldots n]$

LI: $A[1 \ldots i]$ contains a max-heap of i smallest elements of $A[1 \ldots n]$ and the subarray $A[i+1 \ldots n]$ contains $n-i$ largest elements in sorted order

Initialisation: i = A.length $= n$

$A[1..i) = A[1 \ldots n]$ = Max-heap from Build-Max-heap!

$A[i+1..n] = A[n+1..n] = \phi$ sorted ✓

Maintainance: $A[1..i]$ = max-heap from "max-heapify(1)" when the first element is removed and moved to the ith place and then the heap is heapified after reducing the heap-size.

$A[i+1 \ldots n]$ = largest elements from the root of the heap moved from the nth position to i+1th position. Hence, sorted.

Termination: $i = 1$

$A[1 \ldots 1]$ = max-heap → single element, smallest element since all the roots have been remo

$A[2..n]$ = sorted largest elements.

⇒ $A[1..n]$ = sorted! ⇒ Post

Qn 6.4-5

Best case running time when all distinct elements. → Running time is independent of the type of elements!

~ Average case

$$T(n) = T(2n/3) + \Theta(1)$$

$$= \Theta(n \log n) \geq \boxed{\Omega(n \log n)}$$