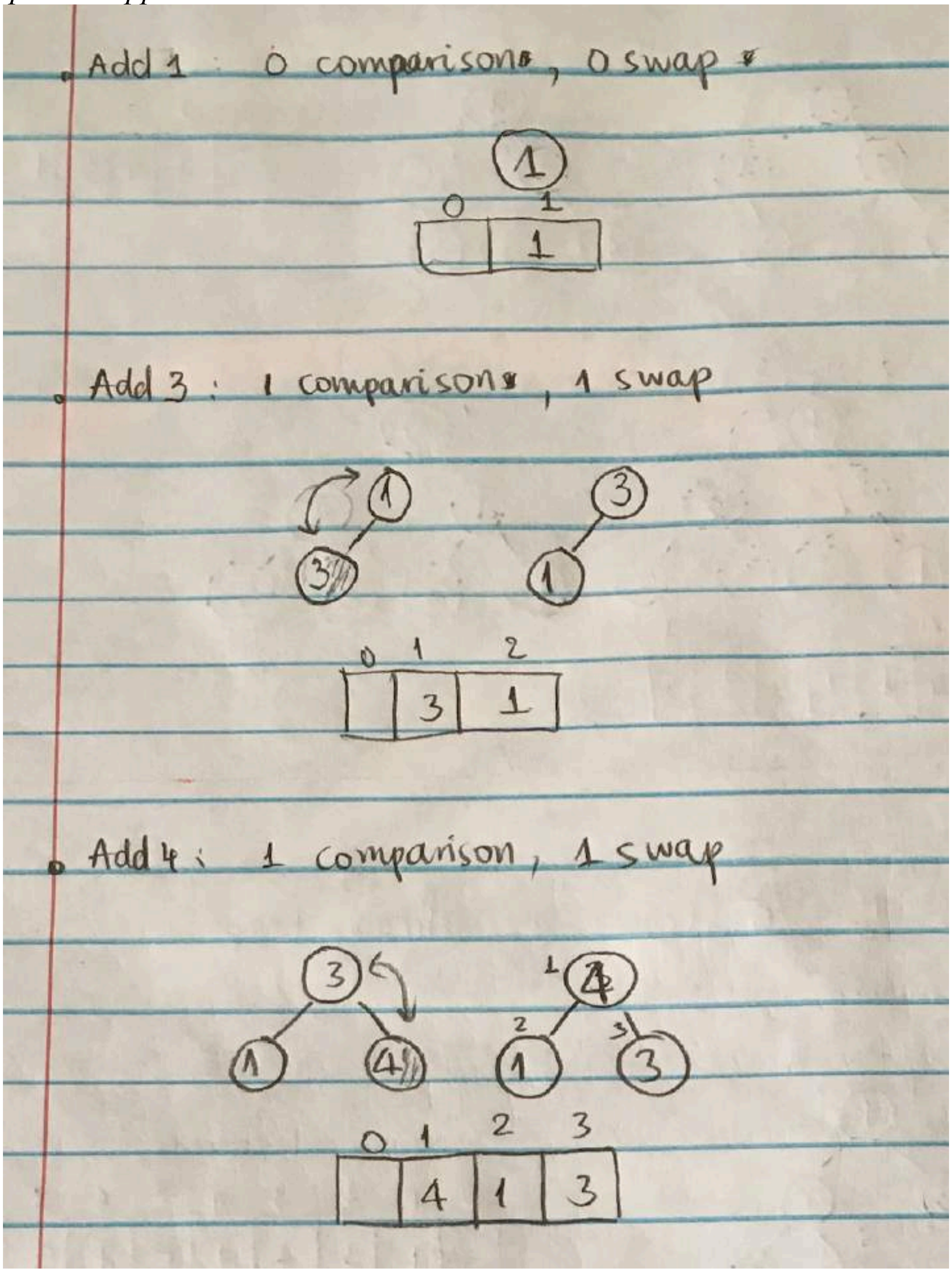
Course: Algorithm
Prof. Prem Nair
Student: Binh Van Tran
ID: 986648
Homework: Lab 11

1. **Question 1 – Starting with the values** 1, 3, 4, 4, 5, 7, 9, 11, 13, 13, 17, do the following
   a. *Top down approach*



- Add 1 : 0 comparisons, 0 swap

0	1
	1

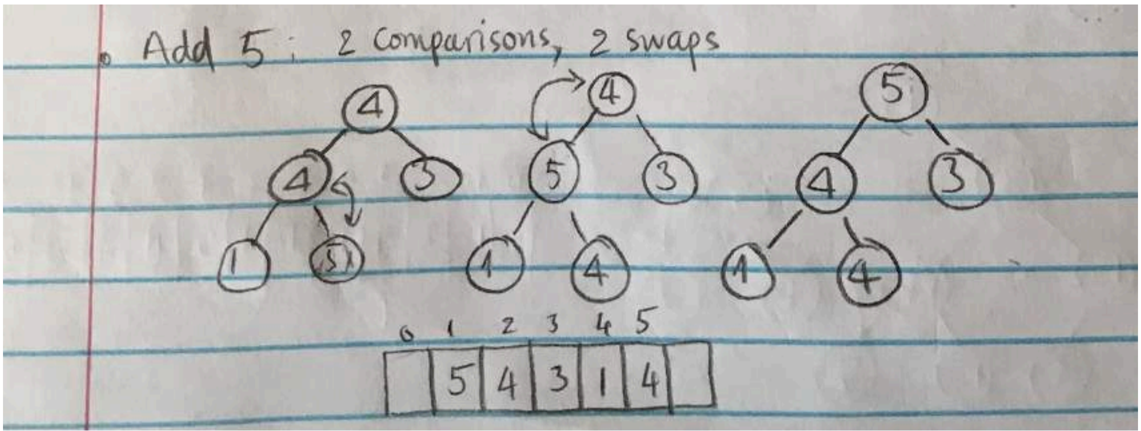
- Add 3 : 1 comparisons, 1 swap

0	1	2
	3	1

- Add 4 : 1 comparison, 1 swap

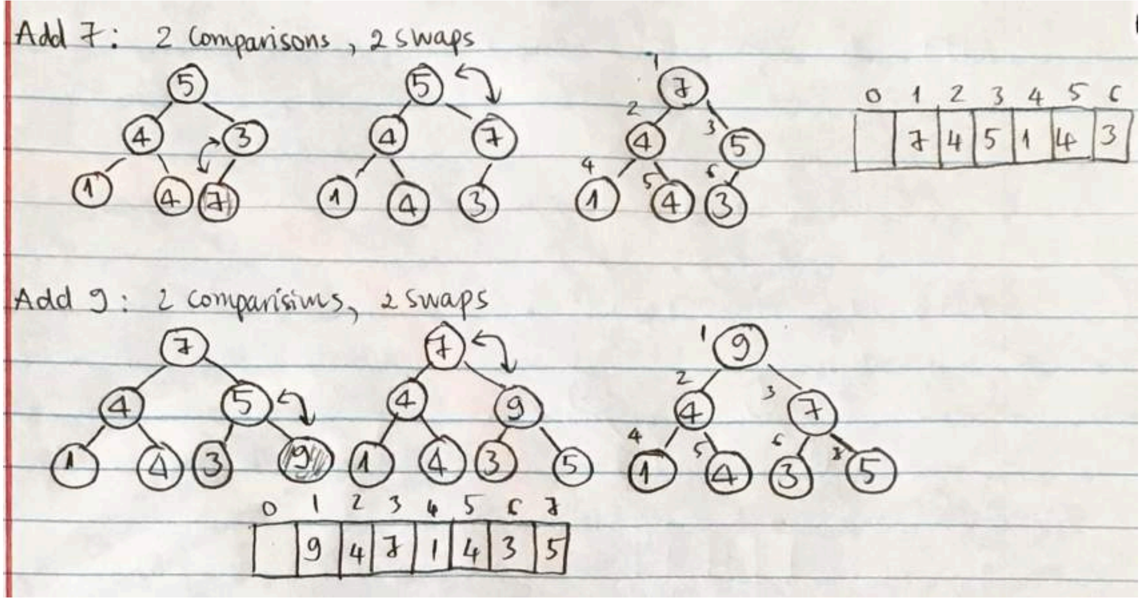
0	1	2	3
	4	1	3

Add 5: 2 comparisons, 2 swaps



0	1	2	3	4	5	
	5	4	3	1	4	

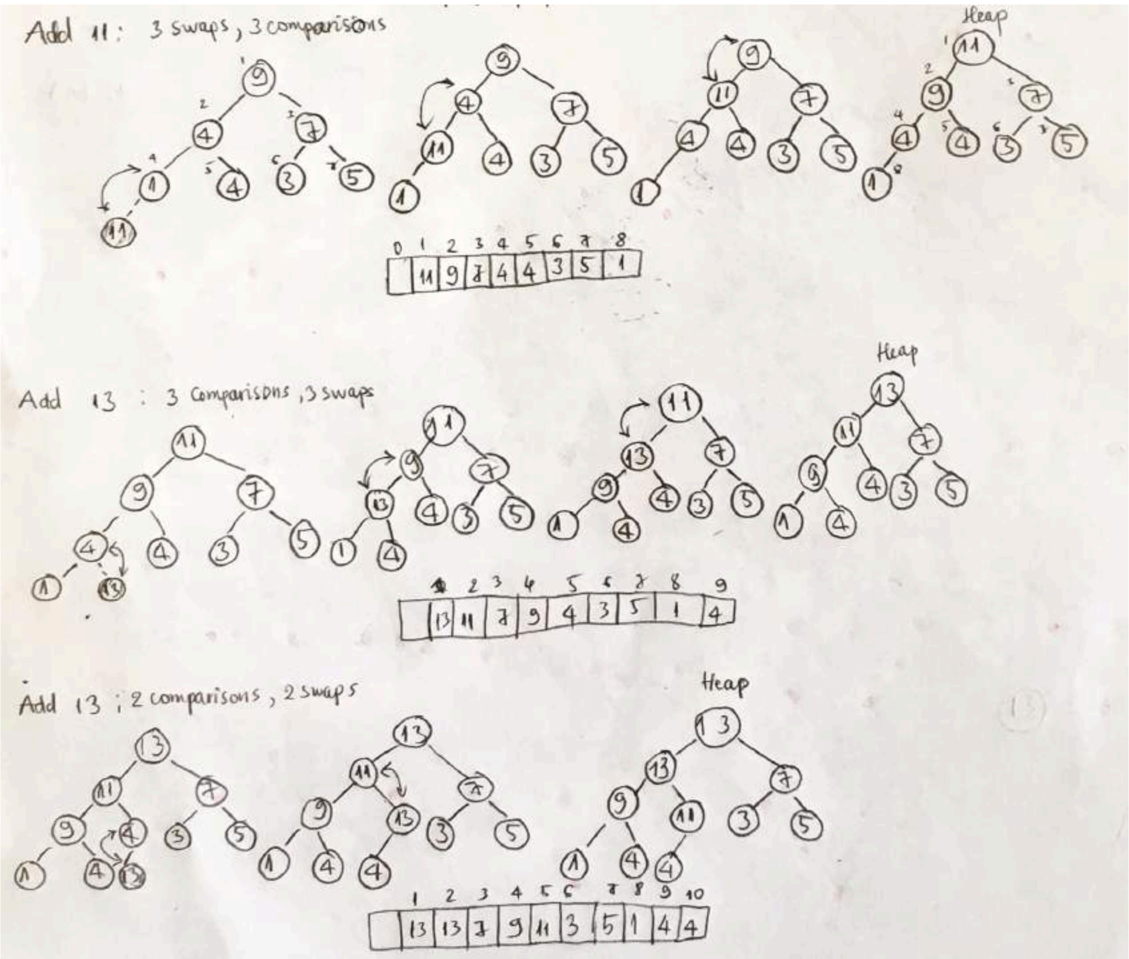
Add 7: 2 Comparisons, 2 swaps



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

Add 9: 2 comparisims, 2 swaps

0	1	2	3	4	5	6	7
	9	4	7	1	4	3	5



Add 11 : 3 swaps, 3 comparisons

0	1	2	3	4	5	6	7	8
	11	9	7	4	4	3	5	1

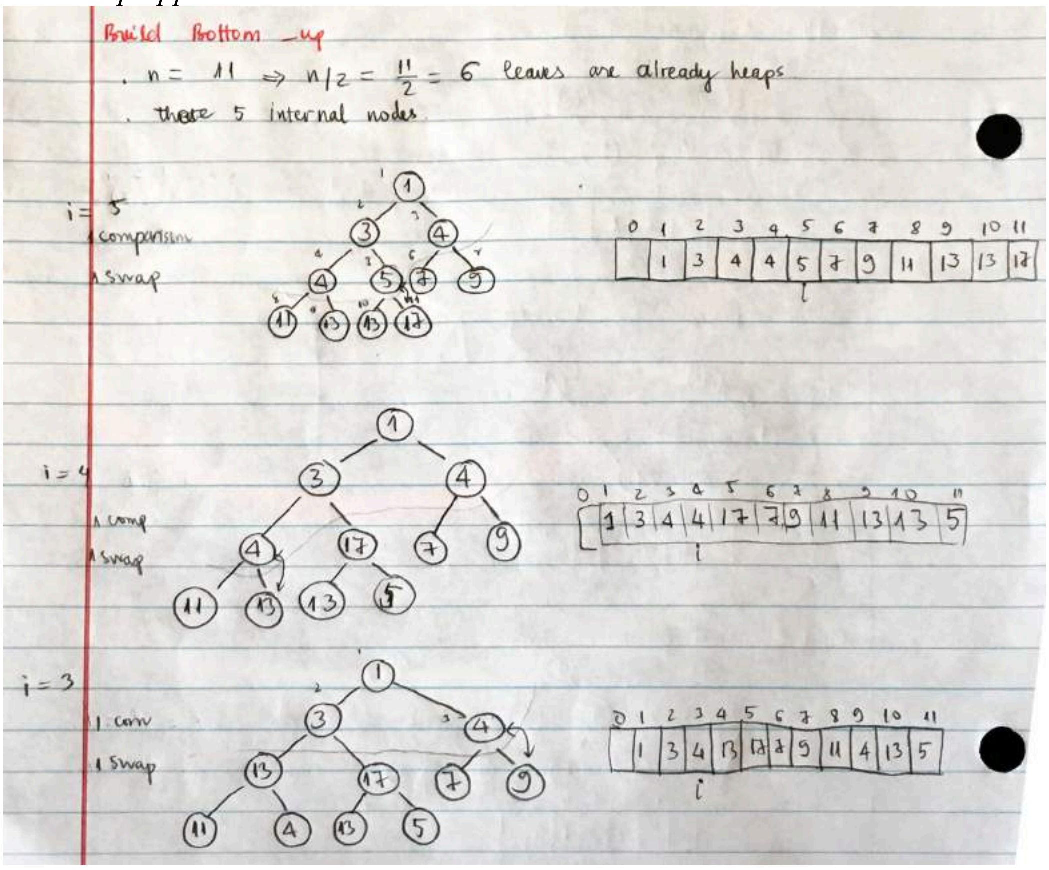
Add 13 : 3 Comparisons, 3 swaps

	1	2	3	4	5	6	7	8	9
	13	11	7	9	4	3	5	1	4

Add 13 ; 2 comparisons, 2 swaps

	1	2	3	4	5	6	7	8	9	10
	13	13	7	9	11	3	5	1	4	4

b. *Bottom up approach*



Build Bottom _up

. n = 11 ⇒ n/2 = $\frac{11}{2}$ = 6 leaves are already heaps

. there 5 internal nodes

i = 5
1 comparison
1 swap

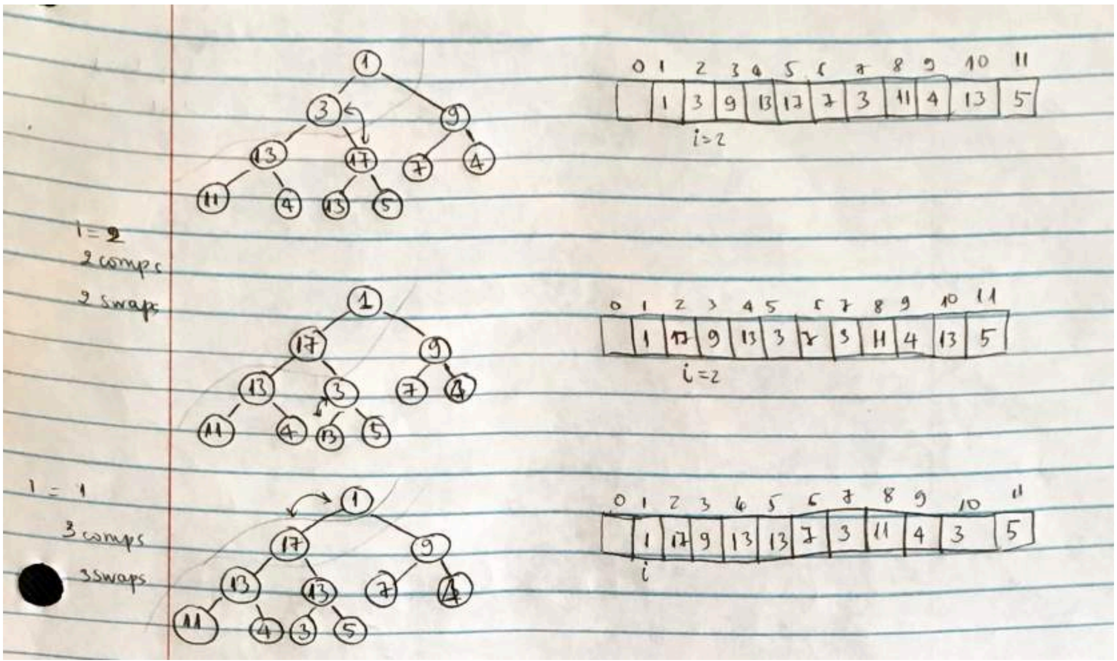
0	1	2	3	4	5	6	7	8	9	10	11
	1	3	4	4	5	7	9	11	13	13	17
					i						

i = 4
1 comp
1 swap

0	1	2	3	4	5	6	7	8	9	10	11
	1	3	4	4	17	7	9	11	13	13	5
				i							

i = 3
1 com
1 swap

0	1	2	3	4	5	6	7	8	9	10	11
	1	3	4	13	17	7	9	11	4	13	5
			i								



1=2
2 comps
2 swaps

0	1	2	3	4	5	6	7	8	9	10	11
	1	3	9	13	17	7	3	11	4	13	5

i=2

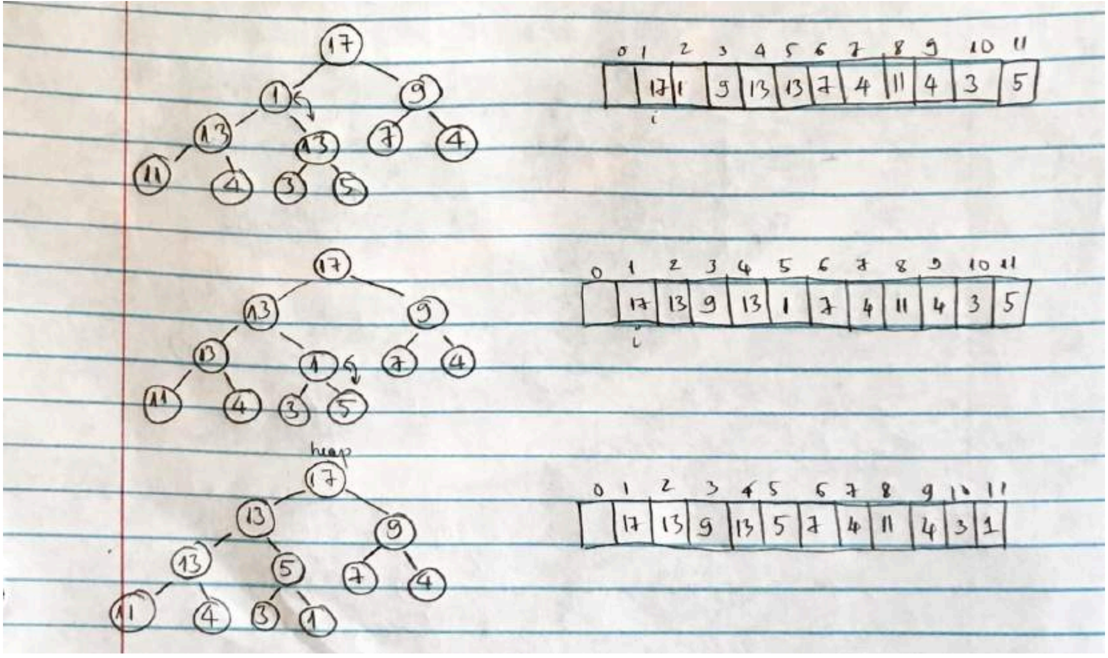
0	1	2	3	4	5	6	7	8	9	10	11
	1	17	9	13	3	7	3	11	4	13	5

i=2

1 = 1
3 comps
3swaps

0	1	2	3	4	5	6	7	8	9	10	11
	1	17	9	13	13	7	3	11	4	3	5

i



0	1	2	3	4	5	6	7	8	9	10	11
	17	1	9	13	13	7	4	11	4	3	5

i

0	1	2	3	4	5	6	7	8	9	10	11
	17	13	9	13	1	7	4	11	4	3	5

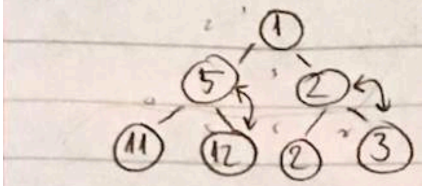
i

0	1	2	3	4	5	6	7	8	9	10	11
	17	13	9	13	5	7	4	11	4	3	1

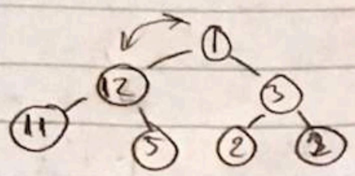
c. *Count of comparisons made in (a):* 16
d. *Count of comparisons made in (b):* 8
e. *Count of swaps made in (a):*16
f. *Count of swaps made in (b):* 8

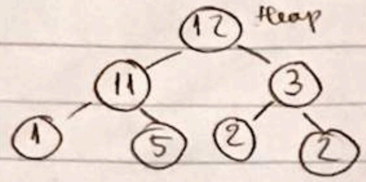
2. **Question 2 – HeapSort array [1,5,2,11,12,2,3]**

Heap sort array [1, 5, 2, 11, 12, 2, 3]



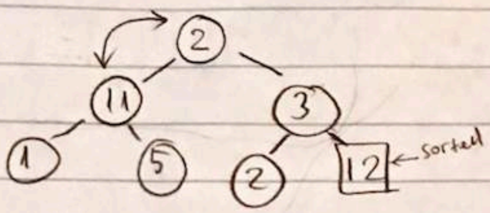
0	1	2	3	4	5	6	7
	1	5	2	11	12	2	3



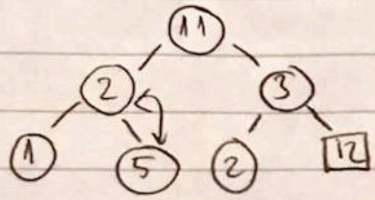


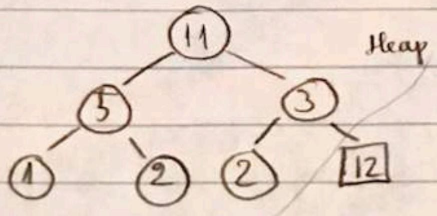
0	1	2	3	4	5	6	7	
	12	11	3	1	5	2	2	

swap



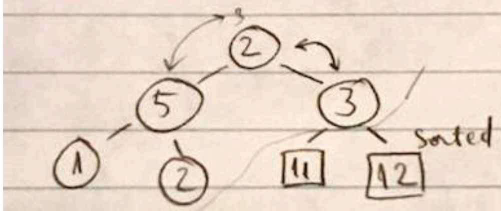
0	1	2	3	4	5	6	7
	2	11	3	1	5	2	12



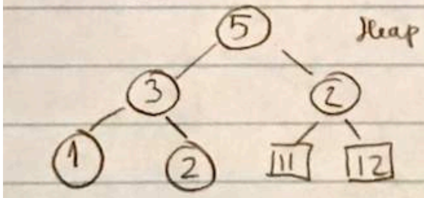


0	1	2	3	4	5	6	7
	11	5	3	1	2	2	12

swap

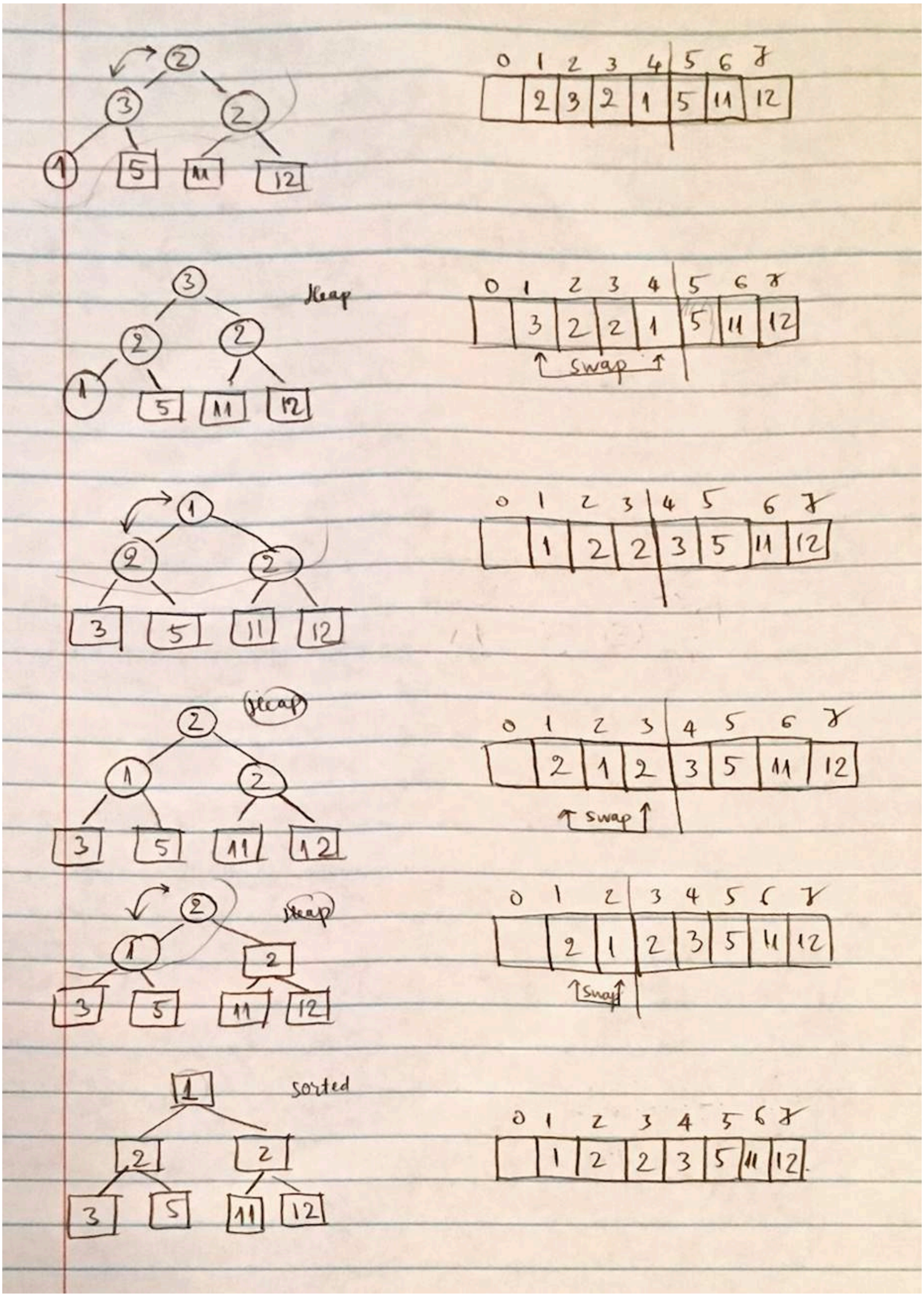


0	1	2	3	4	5	6	7
	2	5	3	1	2	11	12



0	1	2	3	4	5	6	7
	5	3	2	1	2	11	12

swap



0	1	2	3	4	5	6	7
	2	3	2	1	5	11	12

0	1	2	3	4	5	6	7
	3	2	2	1	5	11	12

↑ swap ↑

0	1	2	3	4	5	6	7
	1	2	2	3	5	11	12

0	1	2	3	4	5	6	7
	2	1	2	3	5	11	12

↑ swap ↑

0	1	2	3	4	5	6	7
	2	1	2	3	5	11	12

↑swap↑

0	1	2	3	4	5	6	7
	1	2	2	3	5	11	12

3. **Question 3** – Give two Heaps H1 and H2 of the same height, give an efficient algorithm to "*fuse together*" to make one heap

Algorithm: (for Max Heap)

Which Heap has bigger root node is considered as bigger Heap.

Let say H1 is bigger heap.

Extract H2 elements one by one from the root and append into H1

Heapify H1

a. *What is the best case?*

   If every element in H2 is smaller than or equals to the smallest element in H1 then no cost is needed for heapify. That means just basically append all elements in H2 into H1.

b. *What is the worst case?*

   All elements in H2 is greater than H1. Need to re-heapify

c. *What is the worst-case time complexity?*

   Say, H1 has n elements, H2 has m elements

   Time to append H2 into H1 $O(m)$

   Time to re-heapify H1: $O(n + m)$ with bottom-up approach.