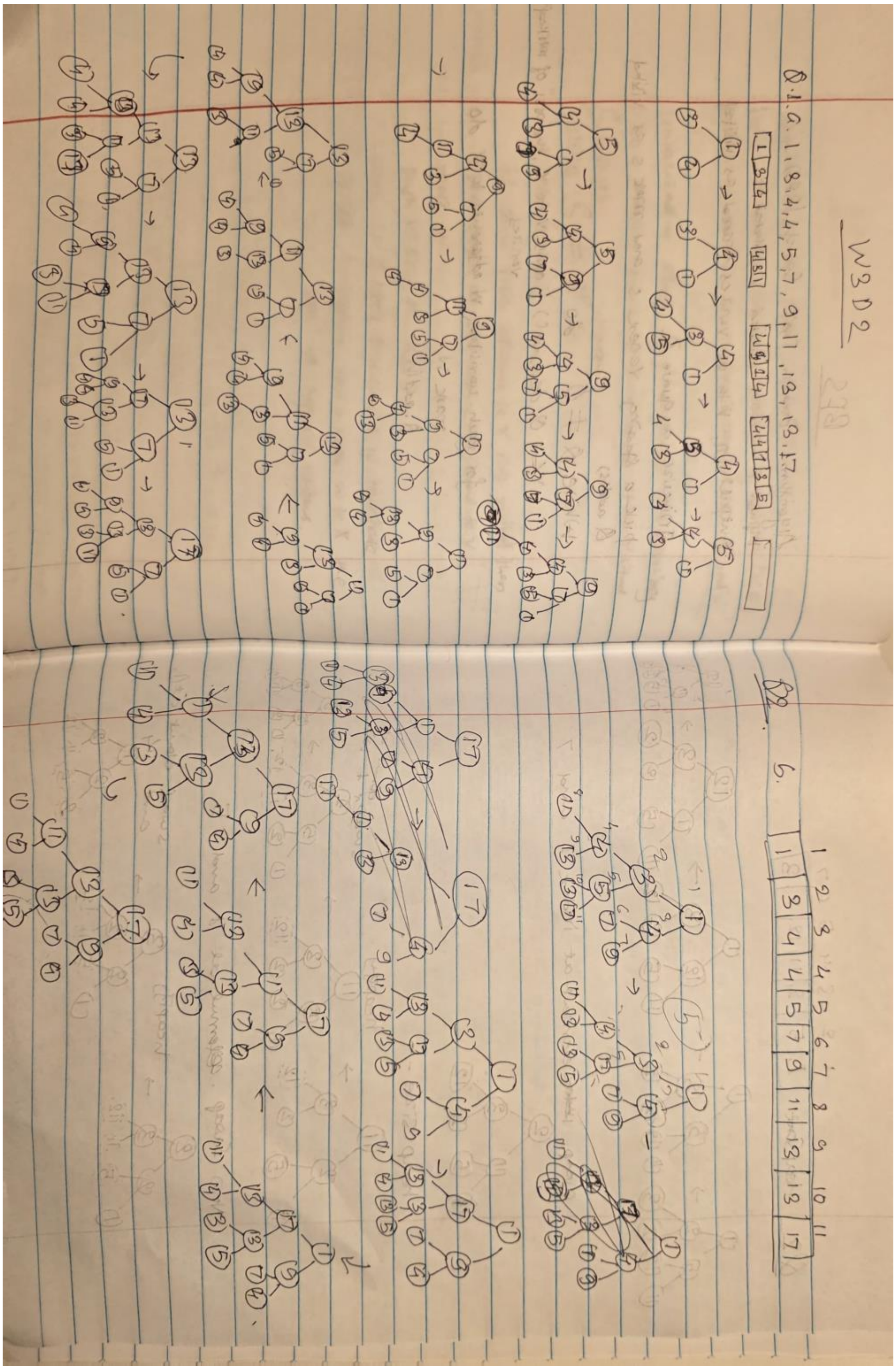
W3D2_solution

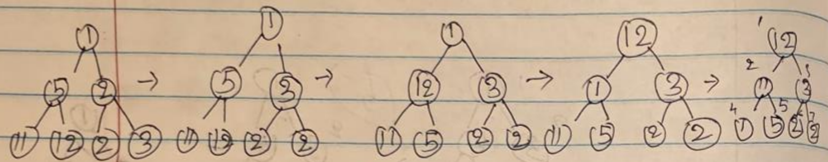
Q1



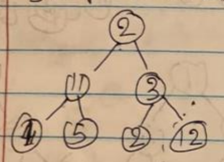
Q2

Q2 HeapSort

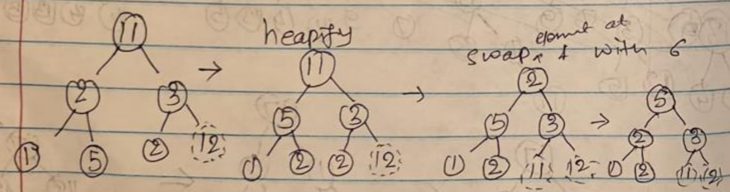
1	2	3	4	5	6	7
1	5	2	11	12	2	3



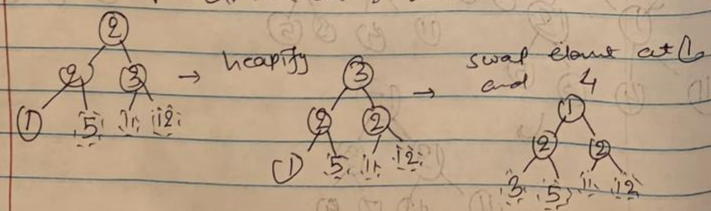
swap ~~index~~ element at index 1 and 7



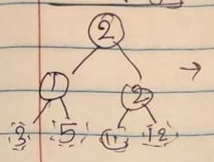
heap size --



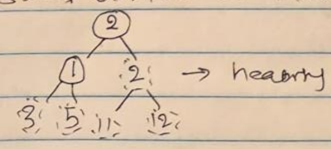
Now Swap element at 1 and 5



heapify



Swap element at index 1 and 3



Q3

Efficient Algorithm to Fuse Two Heaps of the Same Height

Assume:

- You have two **binary heaps** H_1 and H_2.
- Both heaps have the **same height** h.
- You want to merge them into a single valid heap.

Efficient Approach

The standard efficient method is:

1. **Concatenate** the arrays of the two heaps (placing all elements into one array).
2. **Run BUILD-HEAP** on the combined array.

BUILD-HEAP works in linear time, $O(n)$, where n is the number of items.

Since the heaps have the same height,
each has roughly $2^{h+1} - 1$ nodes ⇒ total is about $2^{h+2} - 2$.

(a) Best Case

In the *best case*, one of the heaps' elements can simply be attached as a child subtree without violating the heap property.

Example best case:

- All elements in H_1 are ≤ root of H_2 (for min-heap).
  Then you may attach H_1 as a subtree of H_2 without further fixing.

Best-case time

Only a constant number of pointer/array operations.

➡️ **Best case time: Θ(1)**

(b) Worst Case

In the *worst case*, every element of one heap violates the heap property when placed with the other.
Thus the combined structure must be **completely rebuilt** as a heap.

This happens when, for example:

- In a min-heap, all keys of H_1 are large while all keys of H_2 are small, or vice-versa.
- After concatenation, nearly every node must be sifted down.

(c) Worst-Case Time Complexity

The worst case requires reconstructing the heap, done by:

BUILD-HEAP on $2n$ elements

The BUILD-HEAP algorithm has time:

$$O(n)$$

where n is the number of elements in the combined heap.

Let each heap have size n.
Total size after fusion: $2n$.

So the time remains:

➡️ **Worst-case time complexity: Θ(n)**

Part	**Answer**
(a) Best case	$\Theta(1)$— simply attach one heap to the other with no violations
(b) Worst case	Must rebuild the entire heap (full reheapification)
(c) Worst-case time complexity	$\Theta(n)$ where n is the number of elements in the final heap

Algorithm in Pseudocode

FUSE-HEAPS(H1, H2):

1. Let A1 be the array representation of heap H1

2. Let A2 be the array representation of heap H2

3. Create a new array A of size |A1| + |A2|

4. Copy all elements of A1 into the beginning of A
5. Copy all elements of A2 into the remainder of A

6. BUILD-HEAP(A)

7. return A   // A is now a valid heap containing all elements

---

BUILD-HEAP(A) (for reference)

```
BUILD-HEAP(A):

1. n = length(A)

2. for i = floor(n/2) down to 1:
     HEAPIFY(A, i)

3. return A
```

HEAPIFY(A, i) (standard)

```
HEAPIFY(A, i):

1. left  = 2*i
2. right = 2*i + 1
3. smallest = i

4. if left  ≤ n and A[left]  < A[smallest]:
     smallest = left

5. if right ≤ n and A[right] < A[smallest]:
     smallest = right

6. if smallest != i:
     swap A[i], A[smallest]
     HEAPIFY(A, smallest)
```

- Concatenation: **Θ(n)**
- BUILD-HEAP: **Θ(n)**