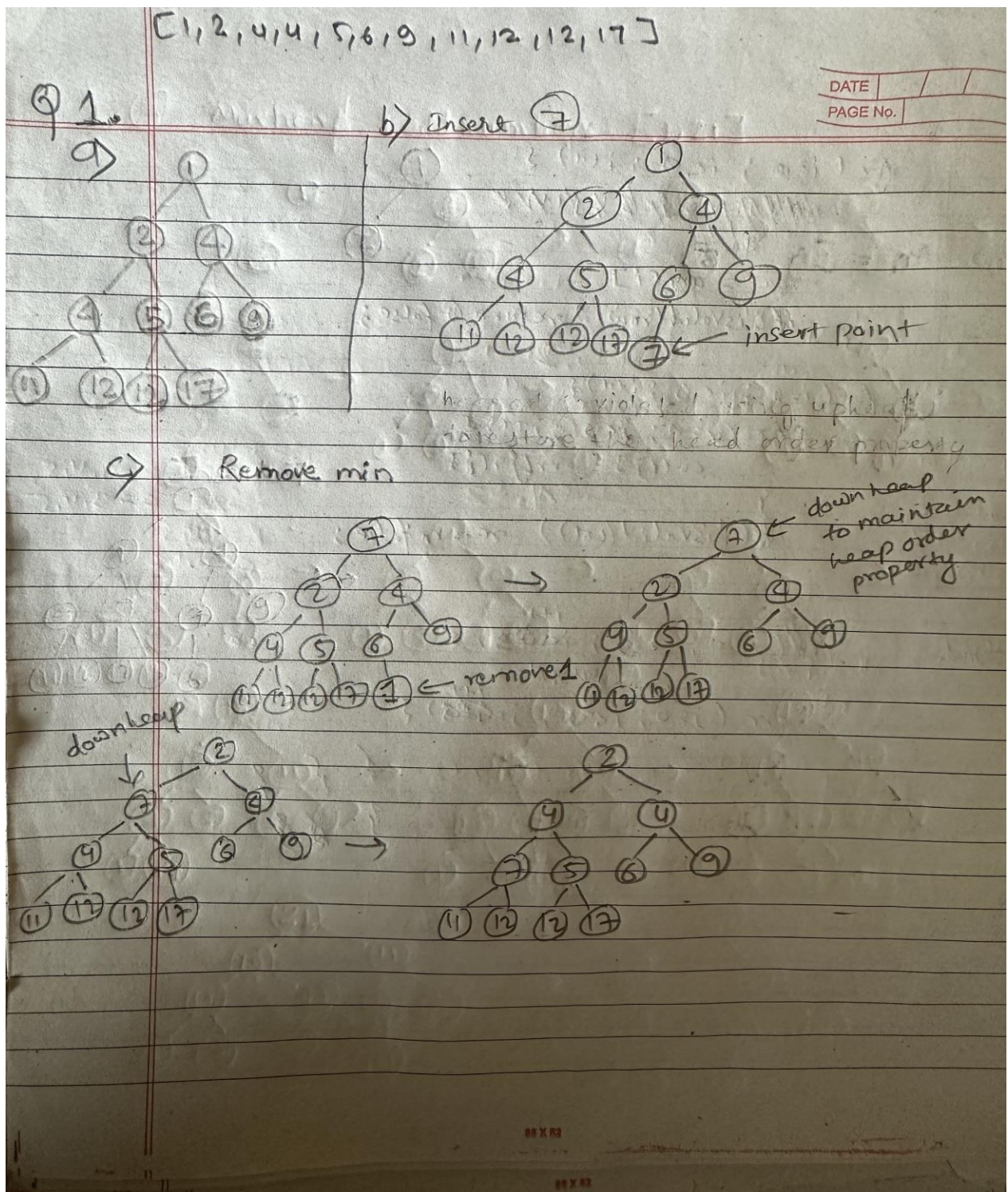


Q1.



Q2.

Minheap using BottomUpHeap

DATE
 PAGE No.

②. $[11, 5, 2, 3, 17, 24, 1]$

↑
key

$Ar_1 = [5, 2, 3]$
 $Ar_2 = [17, 24, 1]$

$[5, 2, 3]$

↑
key

$Ar_1 = [2]$
 $Ar_2 = [3]$

← downheap

$[17, 24, 1]$

↑
key

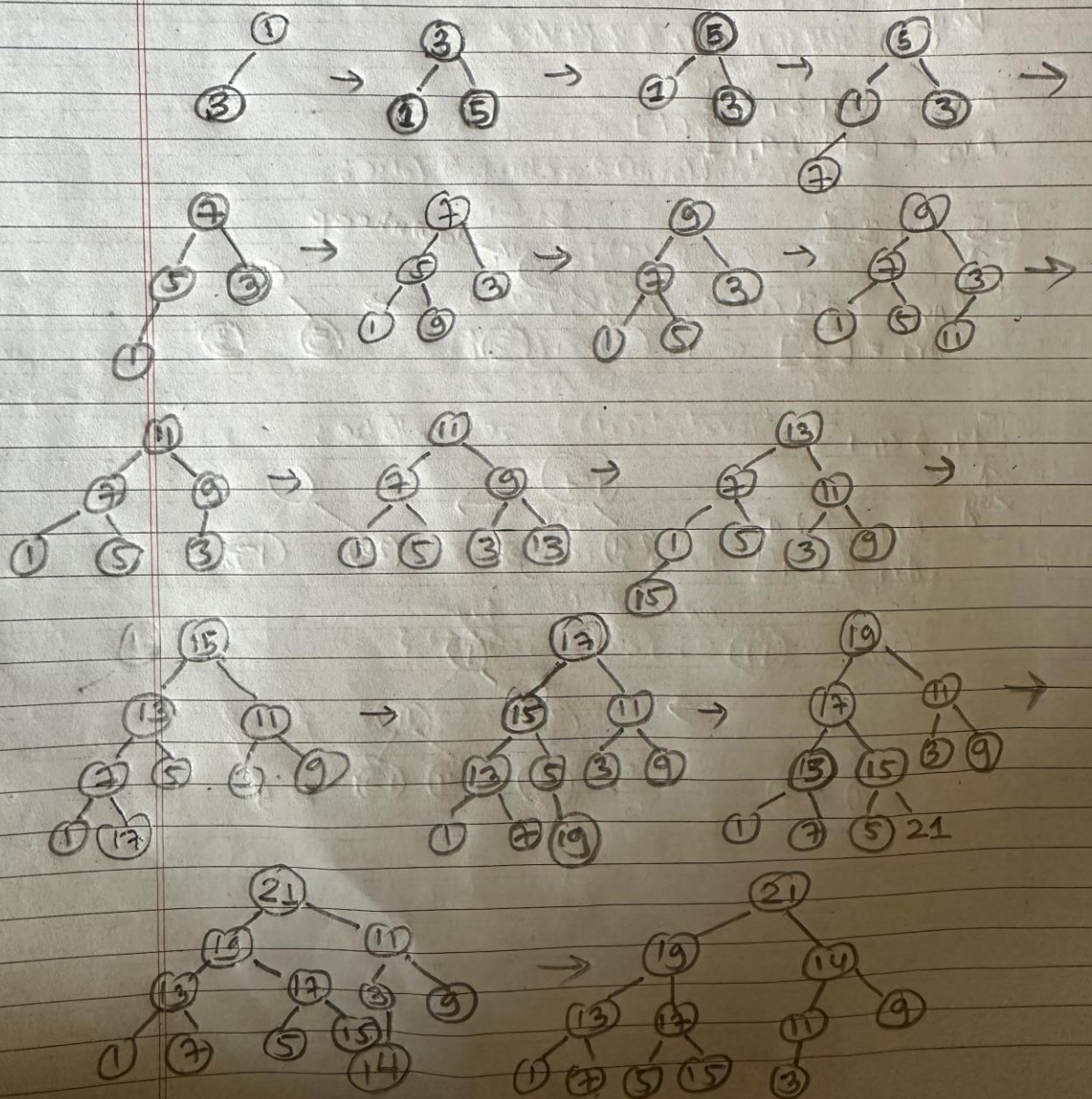
$Ar_1 = [24]$
 $Ar_2 = [1]$

The diagrams illustrate the bottom-up heap construction process. For the array $[5, 2, 3]$, the initial tree has root 5 and children 2 and 3. A downheap operation is shown where 5 moves to the right child position, resulting in root 2 with children 5 and 3. For the array $[17, 24, 1]$, the initial tree has root 17 and children 24 and 1. A downheap operation is shown where 17 moves to the right child position, resulting in root 1 with children 24 and 17. The final step shows the merging of these two heaps into a larger tree with root 11, which has children 2 and 17. Node 2 has children 5 and 3, and node 17 has children 24 and 13.

Q3.

DATE

PAGE No.



Q4.

```
class Solution {
    static class Node implements Comparable<Node> {
        int[] point;
        double distance;
        Node(int[] point){
            this.point = point;
            this.distance = (Math.pow(point[0],2) + Math.pow(point[1],2));
        }

        public int compareTo(Node obj){
            return Double.compare(this.distance, obj.distance);
        }
    }

    PriorityQueue<Node> maxHeap = new PriorityQueue<Node>();
    public int[][] kClosest(int[][] points, int k) {
        for(int[] arr: points){
            maxHeap.add(new Node(arr));
        }

        int[][] result = new int[k][2];
        int length = points.length;
        int pos = 0;
        for(int i=0;i<k;i++){
            result[i] = maxHeap.poll().point;
        }
        return result;
    }
}
```

Q5.

```
class Solution {

    public String frequencySort(String s) {
        HashMap<Character,Integer> dict = new HashMap<>();
        for(char c: s.toCharArray()){
            dict.put(c,dict.getOrDefault(c,0)+1);
        }

        PriorityQueue<Map.Entry<Character,Integer>> queue = new PriorityQueue<>((a,b)->
            b.getValue()-a.getValue());
        queue.addAll(dict.entrySet());

        StringBuilder sb = new StringBuilder();
        while(!queue.isEmpty()){
            Map.Entry<Character,Integer> item = queue.poll();
            for(int i=0;i<item.getValue();i++){
                sb.append(item.getKey());
            }
        }
        return sb.toString();
    }
}
```