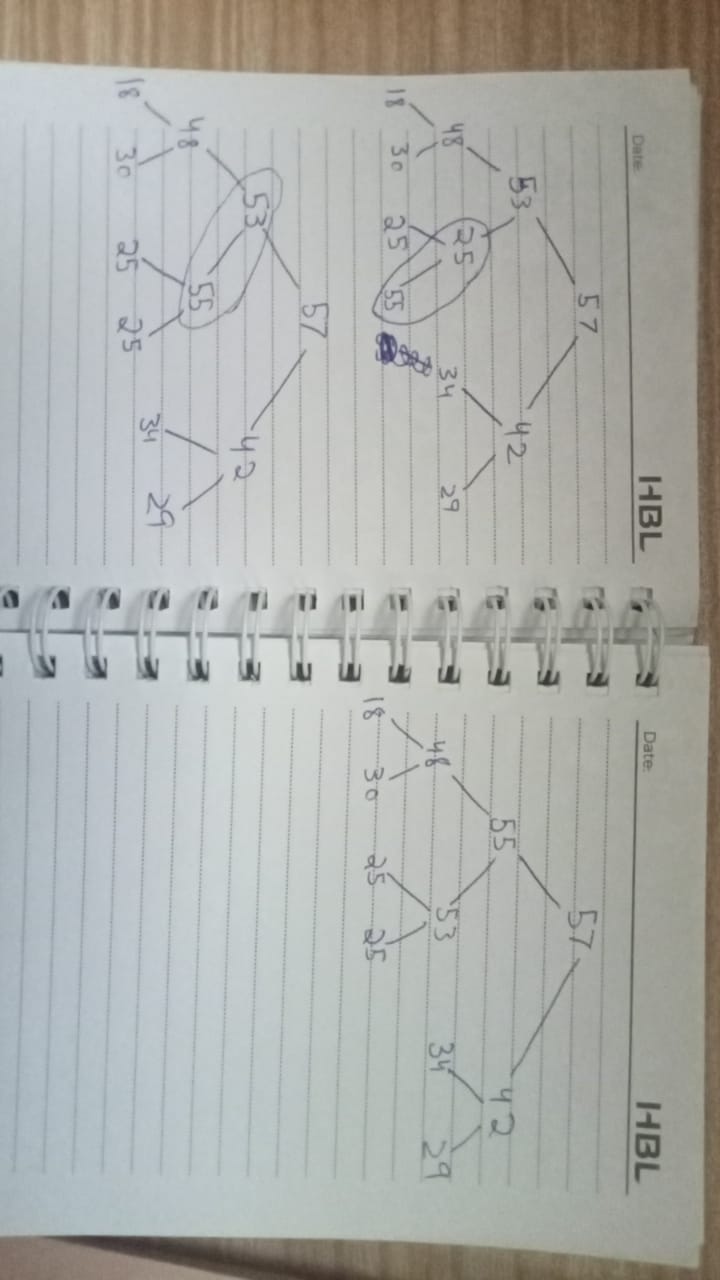
Q1 A) Since the algorithm halfs the array at each step, it calls T(n/2) for every T(n). Hence its time complexity is O(logn)

B) The array has a higher number at the start, and then it keeps decreasing until it reaches the minimum number inside the array. After that, the numbers will keep on increasing.

Algorithm: Use the binary search technique by going to the middle and checking if A[i] < A[i+1]. If True, go towards the left half of the array. Otherwise, go towards the right half. When the number is such that the both sides (left and right) are higher than it, then return that number. Its time complexity is same as the binary search, the only difference is the checking condition.

2)



3)

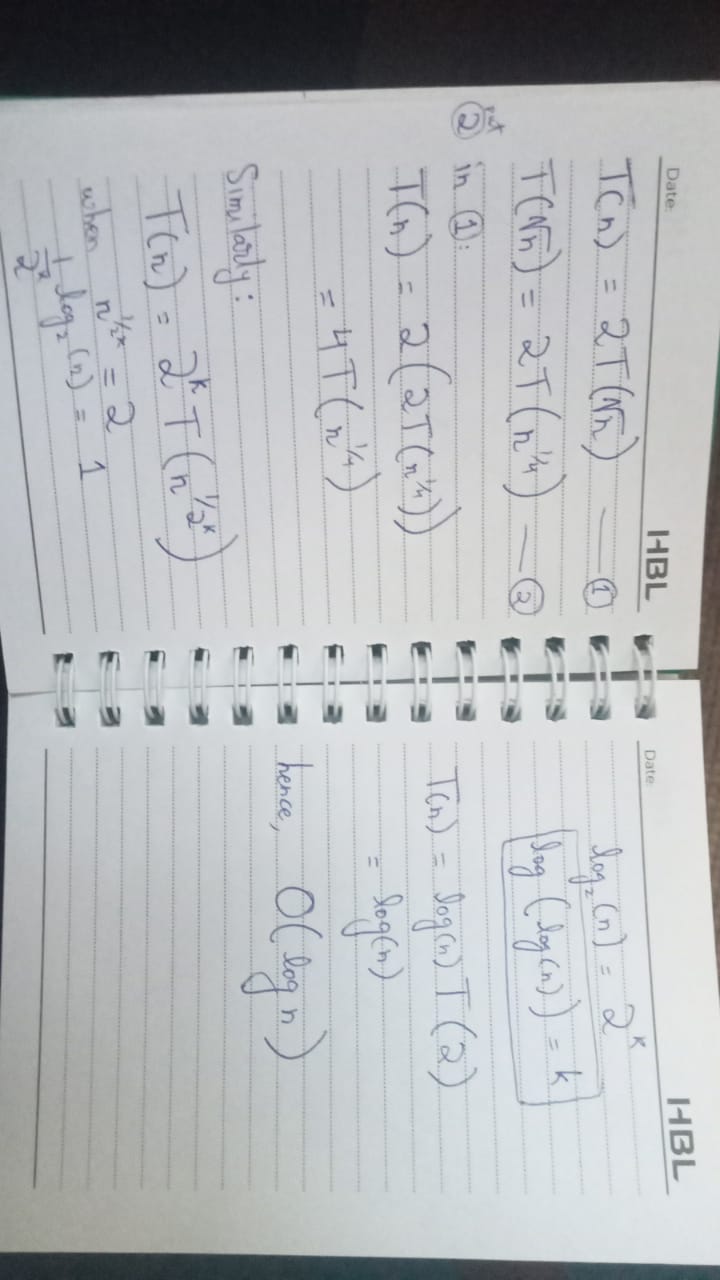
[57, 53, 42, 48, 25, 34, 29, 18, 30, 25]

[57, 53, 42, 48, 25, 34, 29, 18, 30, 25, 55]

[57, 53, 42, 48, 55, 34, 29, 18, 30, 25, 25]

[57, 55, 42, 48, 53, 34, 29, 18, 30, 25, 25]

4)



5)

T(n) = 3T(n/2) + n^2

a = 3, b = 2, k = 2, p = 0

O(n^2 \* log^0 n) = O(n^2)

T(n) = 64T(n/8) + n^2 \* logn

a = 64, b = 8, k = 2, p = 1

O(n^2 \* log^2 n) = O(n^2 \* log^2 n)

T(n) = 4T(n/2) + n^2

a = 4, b = 2, k = 2, p = 0

O(n^2 \* log n) = O(n^2 \* log n)

T(n) = (1/2)T(n/2) + n^2

a = 1/2, b = 2, k = 2, p = 0

O(n^2 \* log^0 n) = O(n^2)