almost perfect tree = nearly complete tree

1-) a-) $\log n, \log n^{100} \rightarrow \Theta(\log n)$

b-) $n, 2^n \rightarrow \Omega(n), O(2^n)$

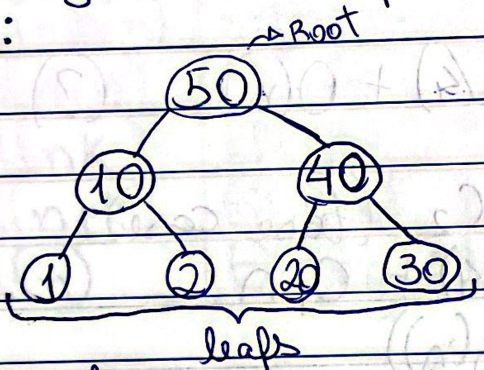
c-) $\frac{n^4 - n^3}{n}, 2022n^3 + \log n \rightarrow \Omega(n), O(n^3)$

2-)

14	8	12	2	5	1	10

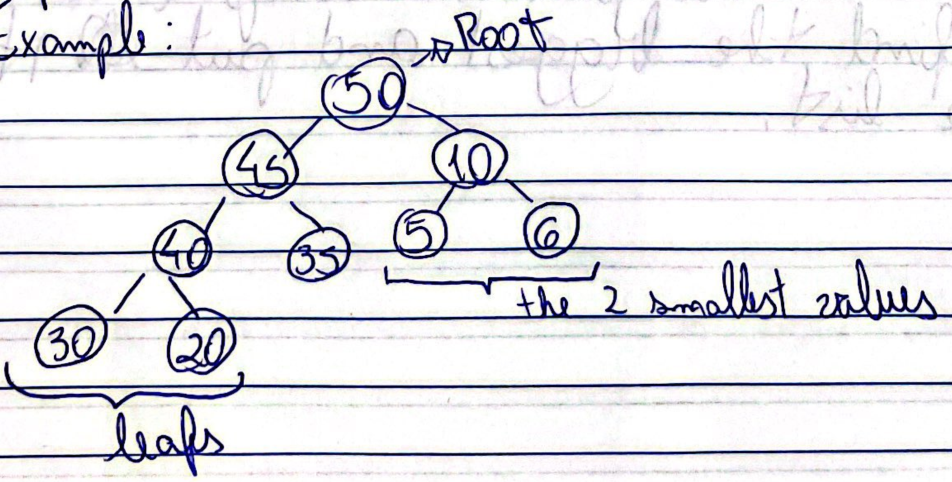
3-) a-) False; in an almost perfect tree, all the smallest values are not always at the leafs.

For Example:



In this case we have 20 and 30 being leafs, but they are still bigger than 10 that's not a leaf, so this proof that not always the smallest elements will be in the leafs. Actually, in an almost perfect tree, it's possible that the Kth smallest element is not even in the leafs.

For Example:



b.) The only order of values we get with heaps is between "parents" and "Kids". this means that for a Minimum heap the "parent" is always smaller than his "Kids" (also for the Kids of his Kids) and the root will be the smallest value.

4.) a.)

b.)
```
def check_numbers(A,m):
    counter = 0
    for e in A:
        if e=m or e=4m:
            counter += 1
    if counter >= 2:
        return True
```

Bonus Question:

d.) This is a MergeSort algorithm. It works by dividing the array in two minimal list (each one with one element) in a recursive way, than aftar that it return a sorted array from two minimal array, until it's all full, by using Selection Sort.