Problem Set 2. Qn. 2

Algorithm.

Input : An array $A[1 \dots n]$

Output : An element $x \in A$ (if exists) s.t

x appears more than $\frac{n}{2}$ times in A.

else, null

Maj-Element(A)

if $|A| = 1$, return $A[1]$.

else

$$A_1 = A\left[1 \dots \left\lfloor \frac{n}{2} \right\rfloor\right]$$

$$A_2 = A\left[\left\lfloor \frac{n}{2} \right\rfloor + 1, \dots, n\right]$$

$$x_1 = \text{Maj\_Element}(A_1)$$

$$x_2 = \text{Maj\_Element}(A_2)$$

```
if x_1 ≠ null
    n_1 = count of x_1 in A
    if n_1 > n/2,
        return n
if x_2 ≠ null
    n_2 = count of x_2 in A
    if n_2 > n/2,
        return n_2
return null
```

Running Time:

The algorithm involves 2 recursive calls on inputs of half the size. Further, computing the value of n_1, n_2 takes $O(n)$ time.

$\therefore$ Total running time can be expressed as

$$T(n) = 2T\left(\frac{n}{2}\right) + n$$

which solves to $T(n) = \Theta(n \log n)$ using Master Theorem

Proof of correctness:

If the algorithm returns a non-null value x, it is easy to see that x appears more than $\frac{n}{2}$ times in A, as it is explicitly checked by the algorithm.

Now, we will show that it is not possible that there exists $x \in A$ that appears more than $\frac{n}{2}$ times and algorithm returns null.
We will prove by induction on n.

It is easy to prove the base case for $n=1$.
Now, assume the claim holds for smaller values of n.

If x appears more than $\frac{n}{2}$ times in A, x appears more than $\frac{n}{4}$ times in A_1 and/or A_2, since $A = A_1 \uplus A_2$.

Assume, without loss of generality, that x appears more than $\frac{n}{4}$ times in A_1. This implies that x is the only such element in A_1.

$\therefore$ By induction hypothesis, $x_1 = x$. and algorithm counts the number of times x appears in A. and algorithm returns x.

$\square$