Divide & Conquer #3 - Majority Element

- **What is the majority element problem?**

  If we have an array A of size n, a **majority element** is a value that appears in that array more than $\frac{n}{2}$ times.

  > **More than half the array is made up by the value**.

- **Majority element algorithm**

  We'll use two phases:

  - **Phase 1** - Use divide-and-conquer to find candidate value of M.

    This candidate value isn't guaranteed to be the majority element, so we have to check to make sure.

  - **Phase 2** - Check if M really is a majority element in $\theta(n)$ time.

- **Phase 1 - Divide-and-Conquer approach**

  Here's the algorithm for the divide-and-conquer part (phase 1):

  - Divide

    1. Group the elements of the array into $\frac{n}{2}$ pairs (think tuples).
    2. Compare each pair (y, z):
       - If $y = z$, then keep y and discard z.
       - If $y \neq z$, then discard both y and z.
    3. If n is odd, there will be one unpaired element.

       We handle this with brute force.

       - Check if this element is a majority element.
       - If it is, return x, but otherwise discard the element.

4. We end up keeping $\leq \frac{n}{2}$ elements.

▼ Conquer

One recursive call on subarray of size $\leq \frac{n}{2}$. Repeat the 'divide' step on all of these subarrays.

▼ Combine

Nothing remains to be done, so omit this step.

Eventually we get back a candidate element. Then we need to check to make sure that the candidate element is actually the majority element.

▼ Why does this algorithm work?

If M is a majority element in the array and has more than $\frac{n}{2}$ occurrences in the array, then there will be at least one (M, M) pair in the array somewhere. It is inevitable.

This also means that if M is a majority in the initial array, it will also be a majority in the newly produced array. It'll have the *most possible pairs*.

▼ Runtime Analysis

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

- Number of recursive subproblems = 1
- Size of each subproblem = $\frac{n}{2}$
- Time for all of the non-recursive steps = $\theta(n)$ (because we might have to check for a majority and we have to go over all of the stuff).

Solution:

$$a = 1, b = 2, k = 1$$
$$log_2 1 = 0 < 1$$
$$T(n) = \theta(n^k) = \theta(n)$$
