# Alturfi's Majority Spacing Lemma: A Structural Insight into the Majority Element Problem

Mohamed Fuad Alturfi

December 27, 2024

**Abstract**

We present *Alturfi's Majority Spacing Lemma*, which states that if an array of length $n$ has a majority element $x$ (i.e. occurring more than $n/2$ times), then $x$ must either occupy two consecutive positions or appear in the final position of the array. Although this result follows intuitively from a pigeonhole argument, explicit articulation of this lemma provides fresh educational clarity and novel possibilities for streaming and online majority-check scenarios. We demonstrate how it can simplify proofs of correctness for well-known algorithms like Boyer–Moore, offer practical insights for real-time data analysis, and serve as a stepping stone for more complex combinatorial arguments.

## 1 Introduction

Finding a majority element in an array of $n$ integers is a classical problem. An element $x$ is called a *majority* if it appears more than $\frac{n}{2}$ times in the array. The famous Boyer–Moore voting algorithm [1] finds such an element in linear time and constant space. Despite the algorithm's elegance, many first-time learners struggle to intuitively understand *why* a true majority element cannot be "canceled out" by the pairwise technique Boyer–Moore uses.

*Alturfi's Majority Spacing Lemma* provides a straightforward structural insight into the arrangement of a majority element. Specifically, it states that if a majority element $x$ truly appears more than $\frac{n}{2}$ times, then $x$ *must* either appear in two adjacent positions at least once or occupy the final position of the array. Although this follows from a short pigeonhole argument, formally stating and highlighting it helps illuminate the correctness of majority-element algorithms.

We begin by defining the lemma and providing a rigorous proof. We then elaborate on its implications for algorithms and pedagogy, where its clarity can strengthen the explanation of key proofs. Finally, we examine how the lemma might extend to streaming or sliding-window contexts, and point to possible directions for generalization to other combinatorial settings.

### 1.1 Contributions

- **Alturfi's Majority Spacing Lemma:** We formalize and prove a useful structural property of majority elements.
- **Algorithmic Clarity:** We show how this lemma unifies and reinforces the intuition behind Boyer–Moore's cancellation approach, and discuss potential use in streaming or partial-array checks.

- **Pedagogical Strength:** For teaching and textbooks, the lemma offers a clear, compact handle on why a majority element can't be systematically "spread out" to avoid detection.

## 2 Preliminaries and Notation

Let $A = [a_1, a_2, \ldots, a_n]$ be an array of $n$ elements. We say $x$ is a *majority element* of $A$ if the count of $x$ in $A$ exceeds $\frac{n}{2}$.

**Definition 2.1** (Majority Element)**.** An element $x$ is a *majority element* of an array $A$ of length $n$ if

$$\text{count}(x, A) > \frac{n}{2}.$$

We shall refer to $\text{count}(x, A)$ as the number of times $x$ appears in $A$. Examples of majority-finding algorithms include Boyer–Moore [1] and its generalizations like Misra–Gries [2], which track frequent elements using limited memory.

## 3 Alturfi's Majority Spacing Lemma

We now state the core structural insight. Although the proof is straightforward, the lemma itself provides a clear vantage point for understanding majority-element concentration.

**Lemma 3.1** (Alturfi's Majority Spacing Lemma)**.** *Let $A$ be an array of length $n$ containing a* majority *element $x$. Then $x$ must either:*

1. *appear consecutively in at least one pair of adjacent positions in $A$, or*
2. *occupy the last position of $A$.*

### 3.1 Proof of the Lemma

Let $k = \text{count}(x, A)$, and note that $k > \frac{n}{2}$. Assume for contradiction that $x$ never appears in two adjacent positions throughout the array *and* the last position is not $x$. This means we must distribute all $k$ occurrences of $x$ in such a way that between every pair of occurrences there is at least one different element. Hence we need at least $k - 1$ *other* elements to separate the $k$ copies of $x$. Consequently, the array length must accommodate

$$k + (k - 1) = 2k - 1$$

distinct positions. But since $k > \frac{n}{2}$, it follows that

$$2k - 1 \;\geq\; n.$$

Moreover, we assumed the last position $a_n$ is not $x$, effectively eliminating one more slot for placing $x$. This exacerbates the contradiction: we cannot fit all $k$ occurrences of $x$ without forcing two of them to be adjacent or placing one in the final position. Hence our assumption is false, and the lemma is proved. □

*Remark* 3.2. While conceptually simple, Lemma 3.1 explicitly *codifies* the typical pigeonhole argument in a manner that is convenient for referencing in proofs related to majority-finding problems.

# 4 Algorithmic and Educational Implications

## 4.1 Connecting to Boyer–Moore

The Boyer–Moore voting algorithm [1] makes a single pass through the array, pairing off distinct elements and discarding them until only one candidate remains. A second pass checks whether the candidate indeed exceeds $n/2$. The key puzzle for new learners is *why* a true majority cannot be fully "canceled out."

**Insight via Lemma 3.1.** If a majority element $x$ must at some point occupy two consecutive positions or appear in the last position, it guarantees that in the pairwise cancellation process, there will always be a point in the array where $x$ can't be matched for cancellation—because it clusters or forces a final unmatched slot. This structural perspective often feels more concrete than the pairwise argument alone.

## 4.2 Streaming and Online Variation

In standard offline settings, you can always do a quick second pass to confirm the Boyer–Moore candidate. In *streaming* or *online* settings, storage or re-scanning may be limited. While the usual majority vote approach still finds a candidate in a single pass, confirming that candidate meets the threshold might not be straightforward without storing the entire stream.

**Real-Time Check.** Lemma 3.1 suggests that in any sufficiently large prefix of the stream that truly contains a majority, that majority must have "given itself away" either by consecutive adjacency or by claiming the final position. One could:

- Keep track of a minimal "adjacency detector" to note any point where the candidate element (from Boyer–Moore's logic) appears twice in a row.
- Observe if the candidate consistently appears in the *current* last slot over time.

These cues can bolster partial confirmation in practical streaming systems where memory is constrained.

## 4.3 Pedagogical Strength

Instructors often face the challenge of explaining Boyer–Moore's survival guarantee. The lemma clarifies that scattering a majority element thinly enough to avoid detection is fundamentally impossible. This single, short proof can be more intuitive than rehashing the details of the cancellation mechanism. Students see how a combinatorial principle (pigeonhole-style reasoning) ensures forced adjacency or forced final-slot occupation.

# 5 Further Generalizations

## 5.1 Pluralities vs. Majorities

For an element that is merely the *plurality* (outnumbering every other individual element but not exceeding $n/2$), forced adjacency is no longer guaranteed. The lemma relies crucially on $k > n/2$. For pluralities, it is possible to intersperse occurrences more evenly. Still, we can investigate looser forms of this lemma that quantify "partial clustering" for near-majorities.

### 5.2 Sliding Windows and Subarrays

If one considers all subarrays or fixed-size windows of the array, we can ask: does a majority in each window require adjacency within that window? While Lemma 3.1 addresses the entire array, analyzing how its logic behaves *locally* in sub-windows might yield interesting extensions, especially for real-time analytics in sensor networks or financial data streams.

### 5.3 Higher-Dimensional Spaces

One might ask if there's a 2D or multi-dimensional analogue: if a color in a 2D image occupies more than half the pixels, must there be a minimal rectangular block or contiguous patch reflecting that dominance? Proving analogous "forced adjacency" statements in higher dimensions often involves more complex combinatorial geometry but follows a similar intuition: sufficiently large coverage can't be scattered entirely.

## 6 Conclusion and Outlook

We have proposed **Alturfi's Majority Spacing Lemma** as a simple and powerful structural statement about majority elements. While at heart it is a direct pigeonhole argument, explicitly naming and framing it in algorithmic and pedagogical contexts opens the door to:

- Stronger *educational* presentations of majority-element algorithms, especially Boyer–Moore.
- Potential uses in *streaming* and *online* majority checks, where forced adjacency or end-position constraints can be tracked in real time.
- Extensions to sliding windows, partial arrays, and higher dimensions.

Although the lemma by itself does not replace standard majority-count verification, its clarity contributes to both teaching and incremental algorithmic insights. By recognizing that a true majority cannot endlessly evade consecutive repetition or final placement, we gain an intuitive guarantee that underpins various majority-finding approaches.

## Acknowledgments

This work was inspired by the desire to clarify the intuitive underpinnings of the Boyer–Moore voting approach. We thank the broader community for their foundational work on frequency counting in streaming and offline settings.

## References

[1] R. S. Boyer and J. S. Moore. A fast majority vote algorithm. *Technical Report*, University of Texas at Austin, 1981.

[2] J. Misra and D. Gries. Finding repeated elements. *Science of Computer Programming*, 2(2):143–152, 1982.

[3] G. Cormode and S. Muthukrishnan. An improved data stream summary: The Count-Min sketch and its applications. *Journal of Algorithms*, 55(1):58–75, 2005.

[4] W. Feller. *An Introduction to Probability Theory and Its Applications*, Vol. I, 3rd Edition. Wiley, 1968.