

Boyer-Moore Voting Algorithm (C++)

Boyer-Moore Voting Algorithm Logic

The Boyer-Moore Voting Algorithm is used to find the majority element in an array.

It works in two passes:

1. Candidate Selection:

- Traverse the array.
- If count is 0, assign current element as candidate.
- If element == candidate, increment count.
- Else, decrement count.

2. Candidate Verification:

- Count occurrences of the candidate.
- If it appears more than $n/2$ times, return it.
- Otherwise, return -1 (no majority element).

Example Dry Run (arr = [1, 2, 3, 5, 2, 2, 2])

Step-by-step Candidate Selection:

Index | Element | Candidate | Count | Action

0	1	1	1	Start with 1
1	2	1	0	2 != 1 -> count-- -> 0 -> new candidate = 2
2	3	3	1	count was 0 -> new candidate = 3
3	5	3	0	5 != 3 -> count-- -> 0 -> new candidate = 5
4	2	2	1	count was 0 -> new candidate = 2
5	2	2	2	2 == 2 -> count++
6	2	2	3	2 == 2 -> count++

Final candidate = 2. It appears 4 times. Since $4 > 7/2$, it's the majority.

Boyer-Moore Voting Algorithm (C++)

C++ Code

```
#include <iostream>
using namespace std;

int majorityElement(int arr[], int n) {
    int candidate = -1, count = 0;

    // Phase 1: Find candidate
    for(int i = 0; i < n; i++) {
        if(count == 0) {
            candidate = arr[i];
            count = 1;
        } else if(arr[i] == candidate) {
            count++;
        } else {
            count--;
        }
    }

    // Phase 2: Verify candidate
    count = 0;
    for(int i = 0; i < n; i++) {
        if(arr[i] == candidate)
            count++;
    }

    if(count > n / 2)
        return candidate;
    return -1;
}

int main() {
    int arr[] = {1, 2, 3, 5, 2, 2, 2};
    int n = sizeof(arr) / sizeof(arr[0]);
    int result = majorityElement(arr, n);
    if(result != -1)
        cout << "Majority Element: " << result << endl;
    else
        cout << "No Majority Element found." << endl;
    return 0;
}
```

Boyer-Moore Voting Algorithm (C++)

```
}
```