

## Majority Element

**Majority Element:** A majority element in an array A[] of size n is an element that appears more than n/2 times (and hence there is at most one such element).

Write a function which takes an array and emits the majority element (if it exists), otherwise prints NONE as follows:

```
I/P : 3 3 4 2 4 4 2 4 4
O/P : 4
```

```
I/P : 3 3 4 2 4 4 2 4
O/P : NONE
```

### METHOD 1 (Basic)

The basic solution is to have two loops and keep track of maximum count for all different elements. If maximum count becomes greater than n/2 then break the loops and return the element having maximum count. If maximum count doesn't become more than n/2 then majority element doesn't exist.

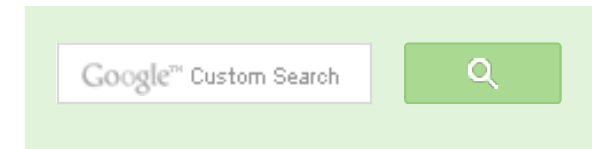
**Time Complexity:**  $O(n^2)$ .

**Auxiliary Space :**  $O(1)$ .

### METHOD 2 (Using Binary Search Tree)

Thanks to [Sachin Midha](#) for suggesting this solution.

Node of the Binary Search Tree (used in this approach) will be as follows.



53,521 people like [GeeksforGeeks](#).



[Interview Experiences](#)

[Advanced Data Structures](#)

[Dynamic Programming](#)

[Greedy Algorithms](#)

[Backtracking](#)

[Pattern Searching](#)

[Divide & Conquer](#)

[Mathematical Algorithms](#)

[Recursion](#)

```
struct tree
{
    int element;
    int count;
}BST;
```

Insert elements in BST one by one and if an element is already present then increment the count of the node. At any stage, if count of a node becomes more than  $n/2$  then return.

The method works well for the cases where  $n/2+1$  occurrences of the majority element is present in the starting of the array, for example {1, 1, 1, 1, 1, 2, 3, 4}.

**Time Complexity:** If a binary search tree is used then time complexity will be  $O(n^2)$ . If a **self-balancing-binary-search** tree is used then  $O(n \log n)$

**Auxiliary Space:**  $O(n)$

### METHOD 3 (Using Moore's Voting Algorithm)

This is a two step process.

1. Get an element occurring most of the time in the array. This phase will make sure that if there is a majority element then it will return that only.
2. Check if the element obtained from above step is majority element.

#### 1. Finding a Candidate:

The algorithm for first phase that works in  $O(n)$  is known as Moore's Voting Algorithm. Basic idea of the algorithm is if we cancel out each occurrence of an element  $e$  with all the other elements that are different from  $e$  then  $e$  will exist till end if it is a majority element.

```
findCandidate(a[], size)
```

1. Initialize index and count of majority element  
`maj_index = 0, count = 1`
2. Loop for  $i = 1$  to  $size - 1$   
 (a) If `a[maj_index] == a[i]`  
`count++`  
 (b) Else  
`count--;`  
 (c) If `count == 0`  
`maj_index = i;`



### Popular Posts

[All permutations of a given string](#)

[Memory Layout of C Programs](#)

[Understanding "extern" keyword in C](#)

[Median of two sorted arrays](#)

[Tree traversal without recursion and without stack!](#)

[Structure Member Alignment, Padding and Data Packing](#)

[Intersection point of two Linked Lists](#)

[Lowest Common Ancestor in a BST.](#)

[Check if a binary tree is BST or not](#)

[Sorted Linked List to Balanced BST](#)

```
count = 1
3. Return a[maj_index]
```

Above algorithm loops through each element and maintains a count of `a[maj_index]`. If next element is same then increments the count, if next element is not same then decrements the count, and if the count reaches 0 then changes the `maj_index` to the current element and sets count to 1.

First Phase algorithm gives us a candidate element. In second phase we need to check if the candidate is really a majority element. Second phase is simple and can be easily done in  $O(n)$ . We just need to check if count of the candidate element is greater than  $n/2$ .

Example:

`A[] = 2, 2, 3, 5, 2, 2, 6`

Initialize:

`maj_index = 0, count = 1` → candidate '2'?

2, 2, 3, 5, 2, 2, 6

Same as `a[maj_index]` => `count = 2`

2, 2, 3, 5, 2, 2, 6

Different from `a[maj_index]` => `count = 1`

2, 2, 3, 5, 2, 2, 6

Different from `a[maj_index]` => `count = 0`

Since `count = 0`, change candidate for majority element to 5 => `maj_index = 3, count = 1`

2, 2, 3, 5, 2, 2, 6

Different from `a[maj_index]` => `count = 0`

Since `count = 0`, change candidate for majority element to 2 => `maj_index = 4`

2, 2, 3, 5, 2, 2, 6

Same as `a[maj_index]` => `count = 2`

2, 2, 3, 5, 2, 2, 6

Different from `a[maj_index]` => `count = 1`

Finally candidate for majority element is 2.

# Deploy Early. Deploy Often.

DevOps from  
Rackspace:

## Automation

FIND OUT HOW ►



First step uses Moore's Voting Algorithm to get a candidate for majority element.

2. Check if the element obtained in step 1 is majority

```
printMajority (a[], size)
1. Find the candidate for majority
2. If candidate is majority. i.e., appears more than n/2 times.
   Print the candidate
3. Else
   Print "NONE"
```

### Implementation of method 3:

```
/* Program for finding out majority element in an array */
# include<stdio.h>
# define bool int

int findCandidate(int *, int);
bool isMajority(int *, int, int);

/* Function to print Majority Element */
void printMajority(int a[], int size)
{
    /* Find the candidate for Majority*/
    int cand = findCandidate(a, size);

    /* Print the candidate if it is Majority*/
    if(isMajority(a, size, cand))
        printf(" %d ", cand);
    else
        printf("NO Majority Element");
}

/* Function to find the candidate for Majority */
int findCandidate(int a[], int size)
{
    int maj_index = 0, count = 1;
    int i;
    for(i = 1; i < size; i++)
    {
        if(a[maj_index] == a[i])
            count++;
        else
            count--;
    }
```

## Recent Comments

**Aman** Hi, Why arent we checking for conditions...

Write a C program to Delete a Tree. · 18 minutes ago

kzs please provide solution for the problem...

Backtracking | Set 2 (Rat in a Maze) · 22 minutes ago

**Sanjay Agarwal** bool

tree::Root\_to\_leaf\_path\_given\_sum(tree...  
Root to leaf path sum equal to a given number · 47 minutes ago

**GOPI GOPINATH** @admin Highlight this sentence "We can easily...

Count trailing zeroes in factorial of a number · 48 minutes ago

**newCoder3006** If the array contains negative numbers also. We...

Find subarray with given sum · 1 hour ago

**newCoder3006** Code without using while loop. We can do it...

Find subarray with given sum · 1 hour ago

AdChoices 

[▶ C++ Vector](#)

[▶ C++ Code](#)

```

        if(count == 0)
        {
            maj_index = i;
            count = 1;
        }
    }
    return a[maj_index];
}

/* Function to check if the candidate occurs more than n/2 times */
bool isMajority(int a[], int size, int cand)
{
    int i, count = 0;
    for (i = 0; i < size; i++)
        if(a[i] == cand)
            count++;
    if (count > size/2)
        return 1;
    else
        return 0;
}

/* Driver function to test above functions */
int main()
{
    int a[] = {1, 3, 3, 1, 2};
    printMajority(a, 5);
    getchar();
    return 0;
}

```

**Time Complexity:**  $O(n)$

**Auxiliary Space :**  $O(1)$

Now give a try to below question

Given an array of  $2n$  elements of which  $n$  elements are same and the remaining  $n$  elements are all different. Write a C program to find out the value which is present  $n$  times in the array. There is no restriction on the elements in the array. They are random (In particular they not sequential).

► [C++ Array](#)

AdChoices ►

► [Java Array](#)

► [Java Algorithm](#)

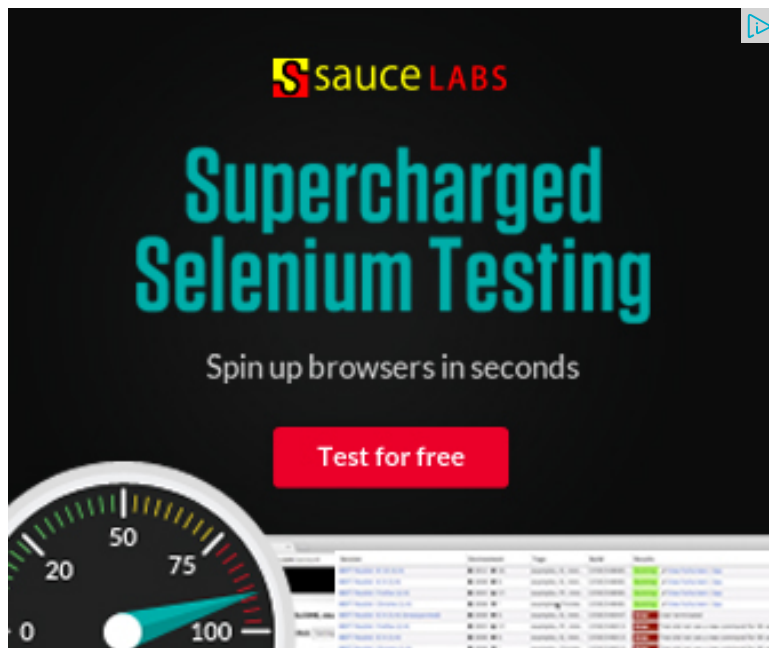
► [C++ Java](#)

AdChoices ►

► [Java Algorithm](#)

► [C Algorithm](#)

► [C++ Example](#)



## Related Topics:

- Remove minimum elements from either side such that  $2 \times \text{min}$  becomes more than max
- Divide and Conquer | Set 6 (Search in a Row-wise and Column-wise Sorted 2D Array)
- Bucket Sort
- Kth smallest element in a row-wise and column-wise sorted 2D array | Set 1
- Find the number of zeroes
- Find if there is a subarray with 0 sum
- Divide and Conquer | Set 5 (Strassen's Matrix Multiplication)
- Count all possible groups of size 2 or 3 that have sum as multiple of 3



30

 Tweet

0



4

**Writing code in comment?** Please use [ideone.com](https://www.ideone.com) and share the link here.

@geeksforgeeks, **Some rights reserved**

**Contact Us!**

Powered by **WordPress & MooTools**, customized by geeksforgeeks team