Write a program to find the majority element in an array in Python

In this article, TipsMake.com will learn with you how to write a program to determine the majority element in Python programming language, Write a program to find the majority element in an array in Python

Problem : Find the majority element in an array. An element that dominates in array A[] with array size n is the element that occurs more than n/2 times.

Example :

```
Input : {3, 3, 4, 2, 4, 4, 2, 4, 4} Output : 4 Gi?i thích: T?n su?t xu?t hi
?n c?a 4 là 5 l?n h?n m?t n?a kích th??c c?a m?
ng. Input : {3, 3, 4, 2, 4, 4, 2, 4} Output : Không có ph?n t? chi?m ?a s?
 Gi?i thích: Không có ph?n t? nào có t?n su?t xu?t hi?n l?n h?n m?t n?
a kích th??c c?a m?ng.
```

In this article, TipsMake.com will learn with you how to write a program to determine the majority element in the Python programming language.

Method 1: The simplest way to handle it

The most basic solution is to use two loops to count the maximum number of occurrences of all different elements. If the maximum number of occurrences is greater than n/2, stop the loop and return the element with the maximum number of occurrences. If the maximum number of occurrences is not greater than n/2 then the majority element does not exist.

Illustrated solution:

```
arr[] = {3, 4, 3, 2, 4, 4, 4, 4}, n = 8 For i = 0: count = 0 - L?
p qua toàn b? m?ng, khi m?t ph?n t? b?ng v?i arr[i] (là 3), t?
ng count - count c?a arr[i] là 2, s? này nh? h?n n/2, do v?y nó không ph?
i là ph?n t? chi?m ?a s? For i = 1: count = 0 - L?p qua toàn b? m?ng, khi m
?t ph?n t? b?ng v?i arr[i] (là 4), t?ng count - count c?a arr[i] là 5, s?
 này l?n h?n n/2 (là 4), do v?y nó chính là ph?n t? chi?m ?a s? K?t lu?
n, 4 là ph?n t? chi?m ?a s?.
```

Follow the steps below to resolve the issue:

1. Create a variable to store the maximum count, count = 0.
2. Browse the table from start to finish.
3. With an element in the array, run another loop to find the frequency of occurrence of similar elements in the given array.

4. If the count is greater than the maximum, update the maximum count and store the index in another variable.
5. If the maximum frequency is greater than half the size of the array, print the element. Otherwise, returns the message that the array has no majority element.

Here is the sample code for your reference:

```
# Python3 program to find Majority # element in an array # Function to find Majo
```

Method 2: Use a binary search tree (BST)

Add each element to the BTS and if that element is present, increment the node count. At any stage, if the count of a node is greater than n/2 then print the element.

Follow the steps below to resolve the issue:

1. Create a binary search tree, if the same element is entered in the binary search tree, the node frequency will be increased.
2. Traverse the array and add the element to the binary search tree.
3. If the maximum frequency value of any node is greater than half the size of the array, perform an in-order traversal and find that node.
4. Otherwise, returns the array message without the majority element.

Here is the sample code for your reference:

```
# Python3 program to demonstrate insert operation in binary # search tree. # cla
```

Method 3: Use Moore's Voting algorithm

This is a 2 step process:

1. The first step gives the element that can be the majority element in the array. If there is a majority element in the array then this step will definitely return the element with the majority. Otherwise, it returns the candidate for the majority element.
2. The second step is to check whether the majority element obtained from the previous step is really the majority element. This step is necessary because there exists a case where there is no majority element.

Illustrated solution:

```
arr[] = {3, 4, 3, 2, 4, 4, 4, 4}, n = 8 maj_index = 0, count = 1 ?
 i = 1: arr[maj_index] != arr[i] - count = count – 1 = 1 – 1 = 0 - now count == 0
?
 i = 2: arr[maj_index] != arr[i] - count = count – 1 = 1 – 1 = 0 - now count == 0
?
 i = 3: arr[maj_index] != arr[i] - count = count – 1 = 1 – 1 = 0 - now count == 0
?
 i = 4: arr[maj_index] != arr[i] - count = count – 1 = 1 – 1 = 0 - now count == 0
? i = 5: arr[maj_index] == arr[i] - count = count + 1 = 1 + 1 = 2 ?
 i = 6: arr[maj_index] == arr[i] - count = count + 1 = 2 + 1 = 3 ?
 i = 7: arr[maj_index] == arr[i] - count = count + 1 = 3 + 1 = 4 Do ?
ó, arr[maj_index] có kh? n?ng là ?ng viên ph?n t? chi?m ?a s?. Bây gi?, duy
```

```
?t l?i toàn b? m?ng ?? ki?m tra xem arr[maj_index] có ph?i là ph?n t? chi?
m ?a s? hay không. arr[maj_index] là 4 4 xu?t hi?n 5 l?n trong m?
ng nên 4 là ph?n t? chi?m ?a s?.
```

Follow the steps below to resolve the issue:

1. Iterates over each element and maintains the count of the majority element (count) and the majority index maj_index.
2. If the next element is similar then increase count and if the next element is not similar then decrease count.
3. If count is 0, then change maj_index to the current element and set count back to 1.
4. Now continue to traverse the array and find the count of the found majority element.
5. If count is more than half the array size, return the element.
6. Otherwise, returns an array without a majority element.

Here is the sample code for your reference:

```
# Program for finding out majority element in an array # Function to find the ca
```

Method 4: Use Hashing

In a Hashtable (key-value pair), at value maintains the count for each element (key) and whenever count is greater than half the length of the array, returns that key (the element that is the majority).

Illustrated solution:

```
arr[] = {3, 4, 3, 2, 4, 4, 4, 4}, n = 8 T?o m?t hashtable cho m?
ng 3 -> 2 4 -> 5 2 -> 1 Duy?t qua toàn b? hashtable - Count cho 3 là 2, nh?
 h?n n/2 (4) do v?y nó không th? là ph?n t? chi?m ?a s?
. - Count cho 4 là 5, l?n h?n n/2 (4) do v?y 4 là ph?n t? chi?m ?a s?. K?
t lu?n 4 là ph?n t? chi?m ?a s?.
```

Follow these steps to resolve the issue:

1. Create a hashtable containing key-value pairs. In this case key is the element while value is the frequency of occurrence of that element in the array.
2. Traverse the entire array from beginning to end.
3. For each element in the array, add the element to the hashtable if the element does not already exist as a key, otherwise load the value of the key (array[i]) and increment the value by 1.
4. If count is greater than half of the array size, return the majority element and terminate the program.
5. If not, return the message that the majority element was not found.

Here is the sample code for your reference:

```
# Python3 program for finding out majority # element in an array def findMajorit
```

Method 5: Use Sorting

The idea of ??this method is to rearrange the array. Sorting causes similar elements in the array to be contiguous so iterate over the array and update the count until the current element is still the same as the previous one. If the frequency of an element is greater than half the size of the array, return the element that has the majority.

Illustrated solution:

```
arr[] = {3, 4, 3, 2, 4, 4, 4, 4}, n = 8 M?
ng sau khi Sorting => arr[] = {2, 3, 3, 4, 4, 4, 4, 4}, count = 1 ?
 i = 1: - arr[i] != arr[i – 1] => arr[1] != arr[0] - count không l?n h?
n n/2, do ?ó thi?t l?p l?i count, count = 1 ?
 i = 2: - arr[i] == arr[i – 1] => arr[2] == arr[1] = 3 - count = count + 1 = 1 +
? i = 3 - arr[i] != arr[i – 1] => arr[3] != arr[2] - count không l?n h?
n n/2, do ?ó thi?t l?p l?i count, count = 1 ?
 i = 4 - arr[i] == arr[i – 1] => arr[4] == arr[3] = 4 - count = count + 1 = 1 + 1
?
 i = 5 - arr[i] == arr[i – 1] => arr[5] == arr[4] = 4 - count = count + 1 = 2 + 1
?
 i = 6 - arr[i] == arr[i – 1] => arr[6] == arr[5] = 4 - count = count + 1 = 3 + 1
?
 i = 7 - arr[i] == arr[i – 1] => arr[7] == arr[6] = 4 - count = count + 1 = 4 + 1
?a 4 l?n h?n n/2 nên 4 là ph?n t? chi?m ?a s?.
```

Follow these steps to resolve the issue:

1. Sort the array and create a variable count and temp with temp = INT_MIN.
2. Iterate over all elements from beginning to end.
3. If the current element is equal to the previous element, increment count.
4. Otherwise, set count back to 1.
5. If count is more than half the size of the array, return the majority element and stop the program.
6. If not, return the message that there is no majority element.

Here is the sample code for your reference:

```
# Python3 program to find Majority # element in an array # Function to find Majo
```

TipsMake.com hopes that this article will be useful to you.

You finished reading the article "**Write a program to find the majority element in an array in Python**" edited by the **TipsMake** team. We hope this article has provided you with many useful tech tips and tricks. You can search for similar articles on tips and guides. Thank you for reading and for following us regularly.