Find the two non-repeating elements in an array of repeating elements

Asked by SG

Given an array in which all numbers except two are repeated once. (i.e. we have 2n+2 numbers and n numbers are occurring twice and remaining two have occurred once). Find those two numbers in the most efficient way.

Method 1(Use Sorting)
First sort all the elements. In the sorted array, by comparing adjacent elements we can easily get the non-repeating elements. Time complexity of this method is O(nLogn)

Method 2(Use XOR)
Let x and y be the non-repeating elements we are looking for and arr[] be the input array. First calculate the XOR of all the array elements.

```
     xor = arr[0]^arr[1]^arr[2].....arr[n-1]
```

All the bits that are set in xor will be set in one non-repeating element (x or y) and not in other. So if we take any set bit of xor and divide the elements of the array in two sets – one set of elements with same bit set and other set with same bit not set. By doing so, we will get x in one set and y in another set. Now if we do XOR of all the elements in first set, we will get first non-repeating element, and by doing same in other set we will get the second non-repeating element.

```
Let us see an example.
   arr[] = {2, 4, 7, 9, 2, 4}
1) Get the XOR of all the elements.
     xor = 2^4^7^9^2^4 = 14 (1110)
2) Get a number which has only one set bit of the xor.
   Since we can easily get the rightmost set bit, let us use it.
     set_bit_no = xor & ~(xor-1) = (1110) & ~(1101) = 0010
   Now set_bit_no will have only set as rightmost set bit of xor.
3) Now divide the elements in two sets and do xor of
   elements in each set, and we get the non-repeating
   elements 7 and 9. Please see implementation for this
   step.
```

Implementation:

```
#include <stdio.h>
#include <stdlib.h>

/* This finction sets the values of *x and *y to nonr-epeating
```

```
 elements in an array arr[] of size n*/
void get2NonRepeatingNos(int arr[], int n, int *x, int *y)
{
  int xor = arr[0]; /* Will hold xor of all elements */
  int set_bit_no;  /* Will have only single set bit of xor */
  int i;
  *x = 0;
  *y = 0;

  /* Get the xor of all elements */
  for(i = 1; i < n; i++)
   xor ^= arr[i];

  /* Get the rightmost set bit in set_bit_no */
  set_bit_no = xor & ~(xor-1);

  /* Now divide elements in two sets by comparing rightmost set
   bit of xor with bit at same position in each element. */
  for(i = 0; i < n; i++)
  {
    if(arr[i] & set_bit_no)
     *x = *x ^ arr[i]; /*XOR of first set */
    else
     *y = *y ^ arr[i]; /*XOR of second set*/
  }
}

/* Driver program to test above function */
int main()
{
  int arr[] = {2, 3, 7, 9, 11, 2, 3, 11};
  int *x = (int *)malloc(sizeof(int));
  int *y = (int *)malloc(sizeof(int));
  get2NonRepeatingNos(arr, 8, x, y);
  printf("The non-repeating elements are %d and %d", *x, *y);
  getchar();
}
```

Run on IDE

Time Complexity: O(n)

Auxiliary Space: O(1)

GATE CS Corner   Company Wise Coding Practice

Bit Magic Bit Magic XOR

Related Posts:

- Bitwise and (or &) of a range
- Calculate XOR from 1 to n.
- Multiples of 4 (An Interesting Method)
- Multiply a number with 10 without using multiplication operator
- Equal Sum and XOR
- Length of the Longest Consecutive 1s in Binary Representation
- Check whether a given number is even or odd
- Pairs of complete strings in two sets of strings

Logged in as **mohitingale5** (Logout)

3.7 Average Difficulty : **3.7/5.0**
Based on **63** vote(s)

Basic Easy Medium Hard Expert

Add to TODO List

Mark as DONE

Writing code in comment? Please use code.geeksforgeeks.org, generate link and share the link here.

Load Comments