Find Missing And Repeating

Given an unsorted array of size n. Array elements are in the range of 1 to n. One number from set {1, 2, ...n} is missing and one number occurs twice in the array. Find these two numbers.

Examples:

Input: arr[] = {3, 1, 3}
Output: Missing = 2, Repeating = 3
Explanation: In the array, 2 is missing and 3 occurs twice

Input: arr[] = {4, 3, 6, 2, 1, 1}
Output: Missing = 5, Repeating = 1

Method 1 (Use Sorting)
Approach:

- Sort the input array.
- Traverse the array and check for missing and repeating.

Time Complexity: O(nLogn)

Method 2 (Use count array)
Approach:

- Create a temp array temp[] of size n with all initial values as 0.
- Traverse the input array arr[], and do following for each arr[i]
    - if(temp[arr[i]] == 0) temp[arr[i]] = 1;
    - if(temp[arr[i]] == 1) output "arr[i]" //repeating
- Traverse temp[] and output the array element having value as 0 (This is the missing element)

Time Complexity: O(n)
Auxiliary Space: O(n)

Method 3 (Use elements as Index and mark the visited places)
Approach:
Traverse the array. While traversing, use the absolute value of every element as an index and make the value at this index as negative to mark it visited. If something is already marked negative then this is the repeating element. To find missing, traverse the array again and look for a positive value.

Time Complexity: O(n)

```
static void printTwoElements(int arr[], int size)
  {
     int i;
     System.out.print("The repeating element is ");

     for (i = 0; i < size; i++) {
```

```
        int abs_val = Math.abs(arr[i]);
        if (arr[abs_val - 1] > 0)
            arr[abs_val - 1] = -arr[abs_val - 1];
        else
            System.out.println(abs_val);
    }

    System.out.print("and the missing element is ");
    for (i = 0; i < size; i++) {
        if (arr[i] > 0)
            System.out.println(i + 1);
    }
}
```

Method 4 (Make two equations using sum and sum of squares)
Approach:

- Let x be the missing and y be the repeating element.
- Let N is the size of array.
- Get the sum of all numbers using formula **S = N(N+1)/2**
- Get the sum of square of all numbers using formula **Sum_Sq = N(N+1)(2N+1)/6**
- Iterate through a loop from i=1....N
- **S -= A[i]**
- **Sum_Sq -= (A[i]*A[i])**
- It will give two equations
  x-y = S - (1)
  x^2 - y^2 = Sum_sq
  x+ y = (Sum_sq/S) - (2)

Time Complexity: O(n)

```
static Vector<Integer> repeatedNumber(int[] a)
  {

    BigInteger n=BigInteger.valueOf(a.length);

    BigInteger s=BigInteger.valueOf(0);
    BigInteger ss=BigInteger.valueOf(0);

    for(int x : a)
```

```
        {
            s=  s.add(BigInteger.valueOf(x));
            ss= ss.add(BigInteger.valueOf(x).multiply(BigInteger.valueOf(x)));
        }


        BigInteger as= n.multiply(n.add(BigInteger.valueOf(1))).divide(BigInteger.valueOf(2));
        BigInteger ass=
as.multiply(BigInteger.valueOf(2).multiply(n).add(BigInteger.valueOf(1))).divide(BigInteger.valu
eOf(3));


        BigInteger sub=as.subtract(s);
        BigInteger add=(ass.subtract(ss)).divide(sub);
        //(ass-ss)/sub;


        int b = sub.add(add).divide(BigInteger.valueOf(2)).intValue();
        //(sub+add)/2;
        int A = BigInteger.valueOf(b).subtract(sub).intValue();
        Vector<Integer> ans = new Vector<>();
        ans.add(A);
        ans.add(b);
        return ans;
    }
```

Method 5 (Use XOR)

Approach:

- Let x and y be the desired output elements.
- Calculate XOR of all the array elements.

xor1 = arr[0]^arr[1]^arr[2].....arr[n-1]

- XOR the result with all numbers from 1 to n

xor1 = xor1^1^2^.....^n

In the result *xor1*, all elements would nullify each other except x and y. All the bits that are set in *xor1* will be set in either x or y. So if we take any set bit (We have chosen the rightmost set bit in code) of *xor1* and divide the elements of the array in two sets – one set of elements with the 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 x, and by doing same in other set we will get y.

Time Complexity: O(n)

```
int[] findTwoElement(int arr[], int n) {
      int x=0;
      for(int i=0;i<n;i++)
      {
         x=x^arr[i];
         x=x^(i+1);
      }
      int pos=x&(-x);
      int repeat=0,missing=0;

      for(int i=0;i<n;i++)
      {

         if((arr[i]&pos)!=0)
         {
            missing= missing^arr[i];
         }

         if(((i+1)&pos)!=0)
         {
             missing= missing^(i+1);
         }
      }
     repeat=missing^x;
     boolean flag=true;
```

```
  for(int i=0;i<n;i++)
  {
      if(repeat==arr[i])
      {
          flag=false;
          break;
      }
  }
  if(flag)
  {
      int t=repeat;
      repeat=missing;
      missing=t;
  }
   int[] res=new int[2];
   res[0]=repeat;
   res[1]=missing;
   return res;
}
```