Find the Missing Element

Problem

Consider an array of non-negative integers. A second array is formed by shuffling the elements of the first array and deleting a random element. Given these two arrays, find which element is missing in the second array.

Here is an example input, the first array is shuffled and the number 5 is removed to construct the second array.

Input:

```
finder([1,2,3,4,5,6,7],[3,7,2,1,4,6])
```

Output:

```
5 is the missing number
```

Solution

The naive solution is go through every element in the second array and check whether it appears in the first array. Note that there may be duplicate elements in the arrays so we should pay special attention to it. The complexity of this approach is O(N^2), since we would need two for loops.

A more efficient solution is to sort the first array, so while checking whether an element in the first array appears in the second, we can do binary search (we'll learn about binary search in more detail in a future section). But we should still be careful about duplicate elements. The complexity is O(NlogN).

If we don’t want to deal with the special case of duplicate numbers, we can sort both arrays and iterate over them simultaneously. Once two iterators have different values we can stop. The value of the first iterator is the missing element. This solution is also O(NlogN). Here is the solution for this approach:

In [1]:
```python
def finder(arr1,arr2):
    
    # Sort the arrays
    arr1.sort()
    arr2.sort()
    
    # Compare elements in the sorted arrays
    for num1, num2 in zip(arr1,arr2):
        if num1!= num2:
            return num1
    
    # Otherwise return last element
    return arr1[-1]
```

```
In [2]: arr1 = [1,2,3,4,5,6,7]
        arr2 = [3,7,2,1,4,6]
        finder(arr1,arr2)

Out[2]: 5
```

In most interviews, you would be expected to come up with a linear time solution. We can use a hashtable and store the number of times each element appears in the second array. Then for each element in the first array we decrement its counter. Once hit an element with zero count that's the missing element. Here is this solution:

```
In [3]: import collections

        def finder2(arr1, arr2):

            # Using default dict to avoid key errors
            d=collections.defaultdict(int)

            # Add a count for every instance in Array 1
            for num in arr2:
                d[num]+=1

            # Check if num not in dictionary
            for num in arr1:
                if d[num]==0:
                    return num

                # Otherwise, subtract a count
                else: d[num]-=1
```

```
In [4]: arr1 = [5,5,7,7]
        arr2 = [5,7,7]

        finder2(arr1,arr2)

Out[4]: 5
```

One possible solution is computing the sum of all the numbers in arr1 and arr2, and subtracting arr2's sum from array1's sum. The difference is the missing number in arr2. However, this approach could be problematic if the arrays are too long, or the numbers are very large. Then overflow will occur while summing up the numbers.

By performing a very clever trick, we can achieve linear time and constant space complexity without any problems. Here it is: initialize a variable to 0, then XOR every element in the first and second arrays with that variable. In the end, the value of the variable is the result, missing element in array2.

```
In [6]: def finder3(arr1, arr2):
            result=0

            # Perform an XOR between the numbers in the arrays
            for num in arr1+arr2:
                result^=num
                print(result)

            return result
```

```
In [7]: finder3(arr1,arr2)
```

```
5
0
7
0
5
2
5
5
```

Out[7]:

Test Your Solution

In [8]:

```python
"""
RUN THIS CELL TO TEST YOUR SOLUTION
"""
from nose.tools import assert_equal

class TestFinder(object):

    def test(self,sol):
        assert_equal(sol([5,5,7,7],[5,7,7]),5)
        assert_equal(sol([1,2,3,4,5,6,7],[3,7,2,1,4,6]),5)
        assert_equal(sol([9,8,7,6,5,4,3,2,1],[9,8,7,5,4,3,2,1]),6)
        print('ALL TEST CASES PASSED')

# Run test
t = TestFinder()
t.test(finder)
```

```
ALL TEST CASES PASSED
```

Good Job!