Coding Exercise #1

Write a function:
  def smallest_absent_int(a)
that, given an array A of N integers, returns the smallest positive integer (greater than 0) that does not occur in A.

Assume that:
* N is an integer within the range [1..100,000];
* each element of array A is an integer within the range [–1,000,000..1,000,000].

So, for example:
smallest_absent_int([6,5,1,2,4]) => 3
smallest_absent_int([1,3,2,6,3,5,4]) => 7
smallest_absent([-5,0,-3,]) => 1

The method should pass the following tests (for example - you are encouraged to write your own as well):

```
class TestArray < Test::Unit::TestCase
  def test_arrays
    # simple cases
    assert_equal(  4, smallest_absent_int([1,2,3]))
    assert_equal(  4, smallest_absent_int([3,2,1]))
    assert_equal(  2, smallest_absent_int([1,3,4]))
    assert_equal(  2, smallest_absent_int([4,1,3]))
    assert_equal(  1, smallest_absent_int([-1,-5]))

    # handle duplicate values
    assert_equal(  1, smallest_absent_int([3,4,4,6,3]))
    assert_equal(  2, smallest_absent_int([-1,0,-1,0,1,0]))

    # larger datasets
    assert_equal(1001, smallest_absent_int((-1000..1000).to_a))
    assert_equal( 101, smallest_absent_int((-10..100).to_a + (102..150).to_a))
  end
end
```

Bonus: You might present a few different implementations and discuss pros and cons - also in terms of speed and efficiency. Benchmark gem is your friend!