Nicholas White

PlayWithArraysImpl.class

Class which contains work on these methods:

- find size
- find Largest Value
- find Second Largest Value
- Intersect

Question 2] Let N be the number of values stored in the array; how many comparisons does your algorithm require?

O(N)

Question 3] Let N be the number of values stored in the array; how many comparisons does your algorithm require?

O(N^2)

Question 4] Let N1, N2, and N3 be the number of values stored in arrays A1, A2, and A3 respectively; what is the worst-case number of comparisons your algorithm requires?

O(N^3)

```
package ds_arrays_assignment;


import ds_arrays.MyArrayIF;

import ds_arrays.MyUnorderedArray;



public class PlayWithArraysImpl<E extends Comparable<E>> implements
PlayWithArraysIF<E> {


   public PlayWithArraysImpl() {

   }


   @Override

   public int findSize(MyArrayIF<E> a) {

      int counter = 0;

      try{


         while (a.getElementAt(counter) != null){

            counter++;

         }


      }

      catch (IndexOutOfBoundsException e){

         //System.err.println("IndexOutOfBoundsException: " + e.getMessage());

      }

      return (counter);

   }
```

```
@Override
public E findLargestValue(MyArrayIF<E> a) {
    E largest = a.getElementAt(0);
    try{
        int i =1;
        while (a.getElementAt(i) != null){
            if (largest.compareTo(a.getElementAt(i)) < 0){
                largest = a.getElementAt(i);
            }
            i++;
        }

    }
    catch (IndexOutOfBoundsException e){
        //System.err.println("IndexOutOfBoundsException: " + e.getMessage());
    }
    return largest;
}

@Override
public E findSecondLargestValue(MyArrayIF<E> a) {
    E largest = this.findLargestValue(a);
    E secondLargest = a.getElementAt(0);
    try{
        int i =1;
```

```
            while (a.getElementAt(i) != null){
                if (secondLargest.compareTo(largest) == 0){
                    secondLargest = a.getElementAt(1);
                }
                if ((secondLargest.compareTo(a.getElementAt(i)) < 0) &&
(a.getElementAt(i).compareTo(largest) != 0)){
                    secondLargest = a.getElementAt(i);
                }
                i++;
            }
        }
        catch(IndexOutOfBoundsException e){

        }
        return secondLargest;
    }

    @Override
    public MyArrayIF<E> intersect(MyArrayIF<E> a1, MyArrayIF<E> a2,
MyArrayIF<E> a3) {
        int size_a1 = this.findSize(a1);
        try{
            for (int i=0;i<size_a1;i++){
                if ((a2.find(a1.getElementAt(i)) != -1) && (a3.find(a1.getElementAt(i)) != -
1)){
                }
                else{
                    a1.delete(a1.getElementAt(i));
                i--;
            }
        }

    }

    catch (IndexOutOfBoundsException e){
}
    return a1;
 }
}
```