CSC220 Assignment05

Searching and Recursion

The goal of this week's assignment is:

1. Practice searching
2. Continue learning the significance of special cases!
3. Learning how to write test to check your implementation

Things you must do:

1. There are many details in this assignment. Make sure you read the whole thing carefully before writing any code and closely follow this instruction.
2. You must complete your assignment **individually**.
3. Always remember Java is case sensitive.
4. Your file names, class names, and package name must match exactly as they are specified here.
5. Your project must include the methods you implemented in the lab.

Things you must not do:

1. You must not change the file names, class names, package names.
2. You must not change the signature of any of these methods (name, parameters, …). Just fill in the missing code inside them. However, you are more than welcome to create any helper methods you need as necessary.

Part 0

- You first must make sure that you have already finished the lab successfully and have all methods in the lab instruction working properly.
- **DO NOT** start your assignment unless you have all the features in the lab working. The tests we run on your assignment may fail if you have an incorrect lab implementation.

Part 1 – `SortedBinarySet` Class Implementation

After making sure the methods you implemented during lab work properly, we will extend the functionality of our class by adding new methods. **Please implement the methods in the order given** and pay close attention to the notes:

- `public boolean remove(double value)`
    - This method should first make a call to `findIndex()` to find the location of value, which for now still calls the private sequentialFind (this will change in the next step). If the element is not present in the list (how do we know?), return **false**. Otherwise, remove the element and return **true** because the list contained the given value.
    - Be careful about which **member variables** should be updated before returning true.
    - Be sure to test your remove method before moving forward.

- `public boolean binaryFind(double target)`
    - this method will return the index where the element target occurs. This method **must** make use of binary search. If `target` is not present in the list, it should return the index where it should be added. Use the same formula we learned during the lab.
    - Change the boolean `usesBinarySearch` to be **`false`** so that findIndex now calls this method instead. insert() and remove() now make use of binary search :-)

- `public boolean contains(double value)`
    - this function returns true if this list contains the value
    - otherwise, it returns false
    - this method must take advantage of the list being sorted and use a binary search to determine if an item is in the array. (Hint: try reusing your other methods.)

- `public boolean containsAll(double[] elements)`
    - this function returns **true** if all the input values (stored in elements) are in the list
    - otherwise, return **false**
    - this method must take advantage of the list being sorted and use a binary search to determine if an item is in the array. (**Hint**: try reusing your other methods.)

- `public SortedBinarySet(double[] input)`
    - This is another constructor for your class that will accept an array and create an object of SortedBinarySet that includes the values in the input array
    - Be careful of the following cases
        - the input might have duplicates (remember your list is not allowed to contain duplicates)
        - the input might not be sorted. **Hint:** think whether you can reuse any of the methods you have implemented to make your job easier.

Part 2 – Testing

Unlike previous assignments you are not given any tests as a starting point. You **must** create your own tests to examine each method you implemented in the previous part.

Part 3 – `sequentialFind` or `binaryFind`? 🤯

As we have seen this week during lecture, some algorithms are more efficient than others. For Lab05 we implemented a *search* routine two ways: `sequentialFind`(), and `binaryFind`().

- For the last part of this assignment, we would like you to think about which one is more efficient. Remember that `findIndex()` will use either method based on the boolean variable `usesBinarySearch`.
- When you think you have an answer, set the variable `usesBinarySearch` to the appropriate value (either **true** or **false**) so findIndex() makes use of that method.
    - For example, setting `usesBinarySearch` to false would indicate that `sequentialFind()` is faster than `binaryFind()`.

To test this out in your Tester, you may consider inserting 100,000 elements (using a loop) into the sorted set and comparing the time it takes (using `System.nanoTime()`) to complete the operation when using `sequentialFind` or `binaryFind`.

Remarks

- Make sure to submit your assignment by (re-)uploading your **Lab05** folder into your **csc220-cXXXX** folder by the deadline **(Tuesday @ 11:59pm)**
- Be sure to include **Tester.java** in your submission to BOX. Remember that we will **not** run your testing code; we will only confirm that you have done testing.
- **For all your assignments, please start early and seek help early (either from the instructor or the TAs).**