Retain All Complexity Worksheet

In this worksheet we will analyze the complexity of the `retainAll` method on various Collections. Assume that variable names tell you the Collection (e.g. treeSet2 is a `TreeSet`) and that Collections ending in “1” contain ***n*** elements and the lists ending in “2” contain ***m*** elements.

`ArrayList` implements `retainAll` as follows (this code is simplified a little from the actual implementation, but the changes do not affect the complexity of the code). Note that `Object[] elementData` and `int size` are fields of `ArrayList` whose meaning should be obvious.

```java
public boolean retainAll(Collection<?> c) {
      boolean modified = false;
      int w = 0;
      for (int r = 0; r < size; r++) {
            if (c.contains(elementData[r])) {
                  elementData[w++] = elementData[r];
            }
      }
      if (w != size) {
            // clear to let GC do its work
            for (int i = w; i < size; i++)
                  elementData[i] = null;
            size = w;
            modified = true;
      }
      return modified;
}
```

Based on this implementation, and what you know about the complexity of the relevant operations on the other Collections, give a tight bound on the complexity of `retainAll` for the following cases:

1. `arrayList1.retainAll(arrayList2);`

2. `arrayList1.retainAll(linkedList2);`

3. `arrayList1.retainAll(treeSet2);`

4. `arrayList1.retainAll(hashSet2);`

Here is `retainAll` as implemented by `LinkedList`, `TreeSet`, and `HashSet`:

```java
public boolean retainAll(Collection<?> c) {
      boolean modified = false;
      Iterator<E> iter = iterator();
      while (iter.hasNext()) {
            if (!c.contains(iter.next())) {
                  iter.remove();
                  modified = true;
            }
      }
      return modified;
}
```

Give a tight bound on the complexity of `retainAll` for the following cases:

```java
// ------------------------------------------------------------
5. treeSet1.retainAll(arrayList2);


6. treeSet1.retainAll(linkedList2);


7. treeSet1.retainAll(treeSet2);


8. treeSet1.retainAll(hashSet2);


// ------------------------------------------------------------
9.  hashSet1.retainAll(arrayList2);


10. hashSet1.retainAll(linkedList2);


11. hashSet1.retainAll(treeSet2);


12. hashSet1.retainAll(hashSet2);


// ------------------------------------------------------------
13.  linkedList1.retainAll(arrayList2);


14.  linkedList1.retainAll(linkedList2);


15.  linkedList1.retainAll(treeSet2);


16.  linkedList1.retainAll(hashSet2);
```