Algorithms, Running Time of Sets and Lists

Consider the following function:

```python
def count_duplicates(small_list, big_container):
    """Count the number of items in small_list that also appear in big_container."""
    total_duplicates = 0
    for item in small_list:
        if item in big_container:
            total_duplicates += 1
    return total_duplicates
```

Consider the two following uses of the `count_duplicates` function (`r()` is a function that returns a random integer and \(m\) and \(n\) are the lengths of the two data structures where \(m\) is much smaller than \(n\)).

```python
small_list = [r() for _ in range(m)]
big_list = [r() for _ in range(n)]
big_set = set(big_list)  # Convert the list to a set (hash table)

# Run the function with a list
list_total = count_duplicates(small_list, big_list)

# Run the function with a set
set_total = count_duplicates(small_list, big_set)
```

The only differences between these two uses of the `count_duplicates` function is in the creation of the second argument (`big_list` vs `big_set`). The variable `big_list` is a list and the variable `big_set` is a set (a hash table type data structure).

Answer the following questions while paying particular attention to the if statement in the `count_duplicates` function.

(a) How do you check if an object exists in an unsorted list, and what is the asymptotic running time?

(b) How do you check if an object exists in a set (hash table), and what is the asymptotic running time?

(c) What is the asymptotic running time of `count_duplicates` when it is called with a list?

(d) What is the asymptotic running time of `count_duplicates` when it is called with a set?

(e) Do you expect the function to run faster the first time or the second time?