*Table 1: The time complexities of this assignment*

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | **Resizable Array Bag** | | | **Linked Bag** | | |
| **union** | **intersection** | **difference** | **union** | **intersection** | **difference** |
| **Best case** | O(n) | O(n^2) | O(n) | O(n) | O(n^2) | O(n) |
| **Worst case** | O(n) | O(n^2) | O(n^2) | O(n) | O(n^2) | O(n^2) |

**Explanation:**

Resizable Array Bag

* Best Case
  + Union: Union simply joins the two bags together. The items of the first bag are iterated over and added to a new bag (n+n operations), and the input bag is converted to an array (n operations). Each add operation takes a constant time as the array is presized to guarantee no resizing. The elements of this array are iterated over, and each is added to the bag (n+n ops). Total operations: 5n, so on the order O(n). Best and worst cases are identical as there is no dependence on the ordering of elements within the bag and the array is never resized.
  + Intersection: The items of the first bag are traversed over in a loop which runs n times. Inside of this loop, calls are made to getFrequencyOf, which must traverse the entire bag each time it’s called (O(n) time complexity in all cases). So the total time complexity is O(n\*n) or O(n^2). This is the same for both best and worst case.
  + Difference: toArray is first used to convert the input bag to an array (n operations). A for loop is then employed to clone the bag (n ops). A second for loop iterates over the items in the array, making calls to contains and remove on each iteration. In the best case the time complexity of contains and remove is O(1) (entry being searched for/removed is at the beginning). So the total time complexity is O(n).
* Worst Case
  + Union: Union simply joins the two bags together. The items of the first bag are iterated over and added to a new bag (n+n operations), and the input bag is converted to an array (n operations). Each add operation takes a constant time as the array is presized to guarantee no resizing. The elements of this array are iterated over, and each is added to the bag (n+n ops). Total operations: 5n, so on the order O(n). Best and worst cases are identical as there is no dependence on the ordering of elements within the bag and the array is never resized.
  + Intersection: The items of the first bag are traversed over in a loop which runs n times. Inside of this loop, calls are made to getFrequencyOf, which must traverse the entire bag each time it’s called (O(n) time complexity in all cases). So the total time complexity is O(n\*n) or O(n^2). This is the same for both best and worst case.
  + Difference: toArray is first used to convert the input bag to an array (n operations). A for loop is then employed to clone the bag (n ops). A second for loop iterates over the items in the array, making calls to contains and remove on each iteration. In the worst case the time complexity of contains and remove is O(n) (entry being searched for/removed is at the end). So the total time complexity is O(n^2).

Linked Bag:

* Best Case
  + Union: (same as ArrayBag) Union simply joins the two bags together. The items of the first bag are iterated over by traversing the linked list and added to a new bag (n+n operations), and the input bag is converted to an array (n operations). The elements of this array are iterated over (n ops), and each is added to the bag (each addition is constant time). Total operations: 4n, so on the order O(n). Best and worst cases are identical.
  + Intersection: The items of the first bag are traversed over in a loop which runs n times. Inside of this loop, calls are made to getFrequencyOf, which must traverse the entire bag each time it’s called (O(n) time complexity in all cases). So the total time complexity is O(n\*n) or O(n^2). This is the same for both best and worst case.
  + Difference: toArray is first used to convert the input bag to an array (n operations). A for loop is then employed to clone the bag (n ops). A second for loop iterates over the items in the array, making calls to contains and remove on each iteration. In the worst case the time complexity of contains and remove is O(n) (entry being searched for/removed is at the end). So the total time complexity is O(n^2).
* Worst Case
  + Union: (same as ArrayBag) Union simply joins the two bags together. The items of the first bag are iterated over by traversing the linked list and added to a new bag (n+n operations), and the input bag is converted to an array (n operations). The elements of this array are iterated over (n ops), and each is added to the bag (each addition is constant time). Total operations: 4n, so on the order O(n). Best and worst cases are identical.
  + Intersection: The items of the first bag are traversed over in a loop which runs n times. Inside of this loop, calls are made to getFrequencyOf, which must traverse the entire bag each time it’s called (O(n) time complexity in all cases). So the total time complexity is O(n\*n) or O(n^2). This is the same for both best and worst case.
  + Difference: Worst case would be O(n^2) because for (T t : secondBag) O(n) has an if block within. There are two for loops. The first for loop calls the add method, which has one if block O(n). Then the contains method returns a call from the getIndexOf which has a while loop with an if statement O(n^2). Next, the remove method is a O(1) as it does not contain any for loops or other constructs. At the end, the time complexity is O(n^2).