|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | Resizable Array Bag |  |  | Linked Bag |  |  |
| Time Complexity | Union | Intersection | Difference | Union | Intersection | Difference |
| BestCase: | O(n+m) | O(n+m) | O(n+m) | O(n+m) | O(n+m) | O(n+m) |
| WorstCase: | O(n+m) | O(m\*n) | O(m\*n) | O(n+m) | O(m\*n) | O(m\*n) |

Leonardo Davalos - LinkedBag

Union Worst & Best Case -

Cloning the caller of the method will take O(n) always as it has to pass through every element. It will take O(m) always to pass through each of the elements in the second bag and add them to the first. In total it will take O(m+n)

Intersection Worst Case -

The method goes through every element of the second bag as it attempts to find a matching element in the first bag. Worse case would be that when its looking for the element in the first bag what it is searching for is at the end. It would compare every element in bag 2 to every element in bag 1. O(m\*n)

Intersection Best Case -

For the best case the method will involve it not needing to search through the caller's elements to remove any intersection. The best case would be that the caller is ordered in such a way that they line up to what intersections are being checked. For exampled if A was 0,1 and B was 0 then as it passed through the elements of B it would not search A for the element because it is the first element. O(m\*n) would become O(m).The remove method shufles data for LinkedBags so that would have to be accounted for. Cloning first bag takes O(n).

Difference Worst Case -

The method goes through every element of the second bag as it attempts to find a matching element in the first bag. Worse case would be that when its looking for the element in the first bag what it is searching for is at the end. It would compare every element in bag 2 to every element in bag 1. O(m\*n)

Difference Best Case -

The best case would have the method not need to search through the first bag to find a match. If the given bag was ordered in a way where each element would align with its match, there would be no need to search through the bag. O(m\*n) would become O(m). Only the size of one data set matters. With the loop not dominating the method O(n) is added for creating the clone of first bag.

Dean Mah – Resizeable Array Bags

Union Worst & Best Case -

Cloning the first bag will take O(n) since it has to copy every entry in the bag. It will take O(m) to add the entries from the second bag of size m to the clone bag. Therefore, it will take a total of O(n+m) since the bags can have different lengths.

Intersection Wost Case -

The method will have to look at all entries in bag 2 for each of the entries in bag 1 making it O(n) from the first bag times the O(m) of the second bag therefore resulting in O(m\*n) as it copies these over to a clone bag.

Intersection Best Case -

The method will only have to look at the first entry in bag 2 for each entry in bag 1 making the best case O(n+m) because it only has to look at each entry once and takes O(n) to clone them to the cloned return bag.

Difference Worst Case -

The method will take O(n) and O(m) to clone the bags. Then it will take the clone of the socond bag and look at each entry and remove it from the clone of the first bag if it contains the same entry before also removing it from the second bag. This process takes O(m\*n) because it will have to look through every entry in the first bag looking to see if any are the same as the entry in the second bag which repeats for every entry in bag 2.

Difference Best Case -

If each entry in bag 2 matches the first entry of bag 1 it will only go over each entry in each bag once resulting in O(m+n) since it takes the time from going over the entries in bag 2 once O(m) and the time to clone the entries from the bags O(n).

GITHUB:

<https://github.com/leothecrz/Project0>

NAMES:

Leonardo Davalos

Dean Mah