Thet Naing Soe **1.Assignment/8.Task** 26th March 2023

Neptun Code**: X0FLXY**
x0flxy@inf.elte.hu

Task

Implement the bag type which contains integers. Represent the bag as a sequence of (element, frequency) pairs. Implement as methods: inserting an element, removing an element, returning the frequency of an element, returning the most frequent element from the bag (suggestion: store the most frequent element and update it when the bag changes), printing the bag.

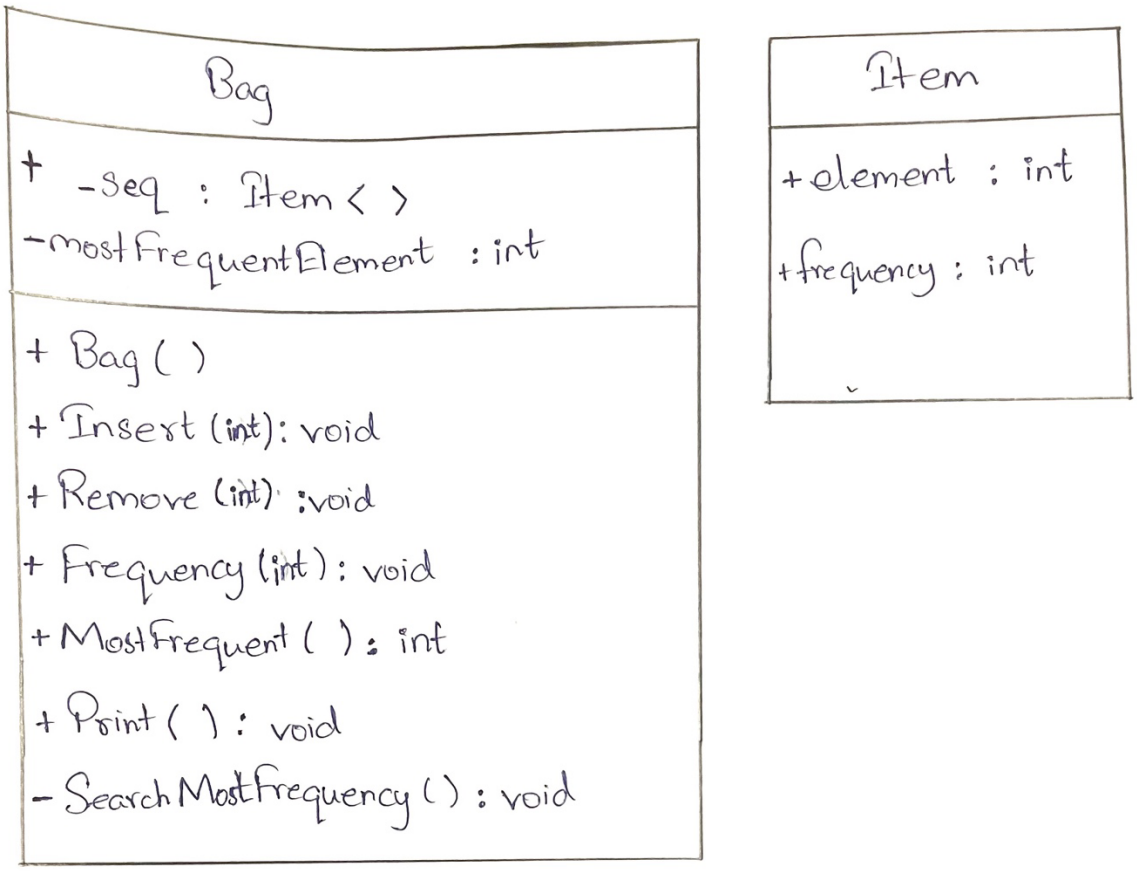
Bag Type

Set of Values

The values of that a Bag-type variable can have, together form the set of values of a Bag.

Bag(n,m) = { $b \in \mathbb{Z}^{nxm}$ | $n \in \mathbb{Z} \wedge m \in \mathbb{Z} \rightarrow$ b <(n,m)> }

The UML diagram of the class methods which has all the operations are as follows.



Operations

1. Inserting an element and store the mostFrequentElement.

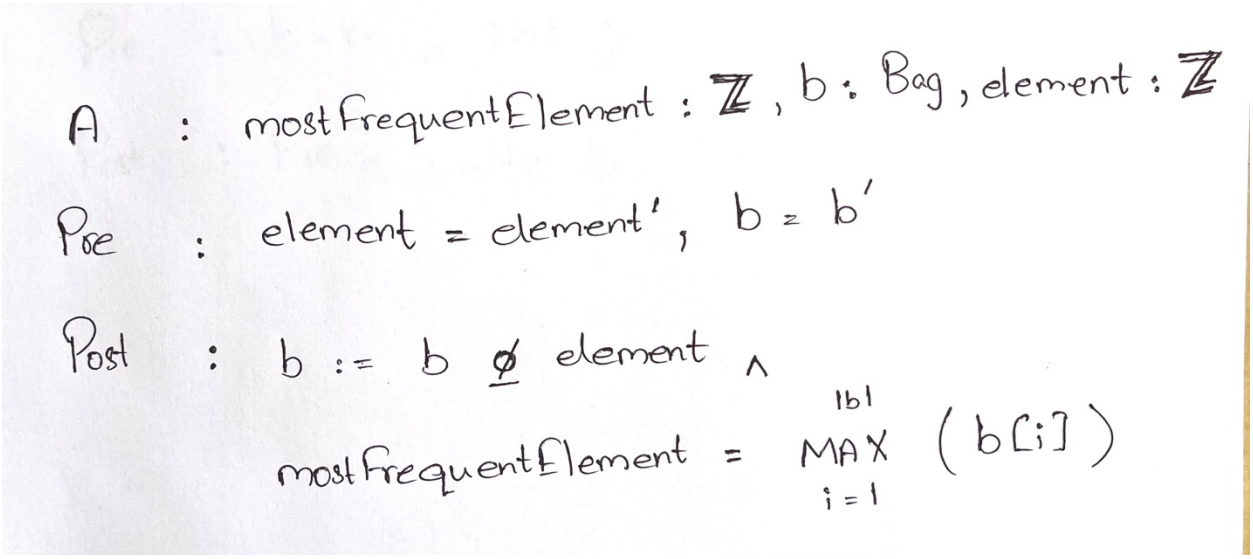
A : mostFrequentElement : $\mathbb{Z}$, b : Bag , element : $\mathbb{Z}$

Pre : element = element' , b = b'

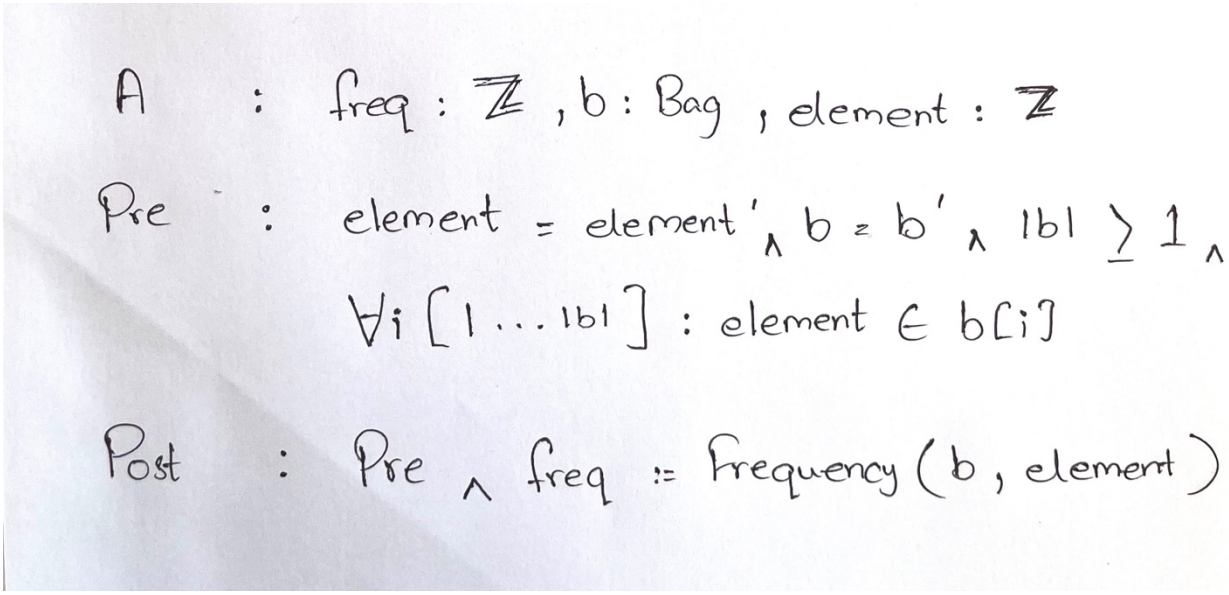
Post : $b := b \cup element \wedge$

$$mostFrequentElement = \max_{i=1}^{|b|} (b[i])$$

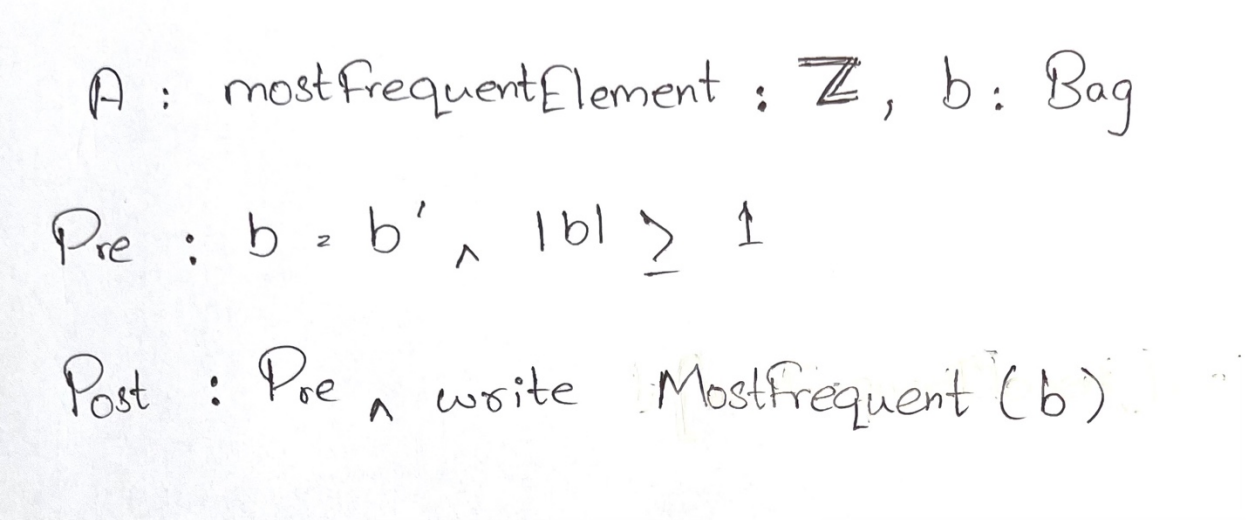
2. Removing an element and modify the mostFrequentElement.



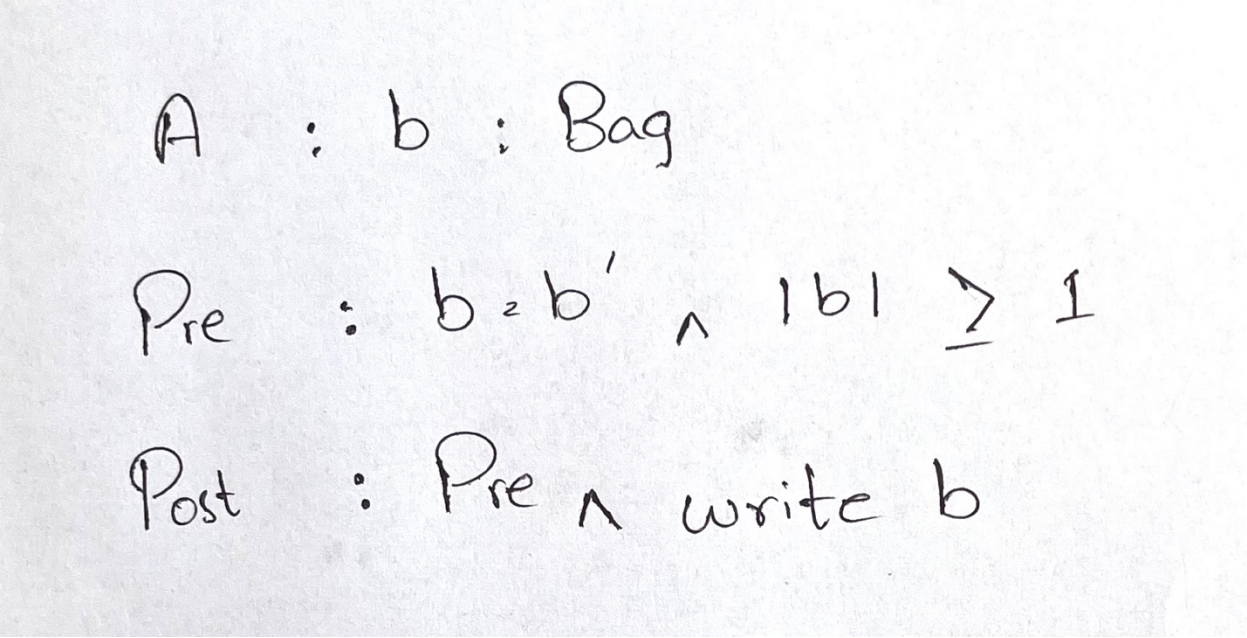
3. Getting the frequency of an element.



4. Getting the most frequent element.



5. Printing the bag.



Representation

Here is an example representation of the Bag and the program.

0	1	2	3	4

_seq = < (2,4) , (3,2) , (7,5) , (4,1) , (1,3) >

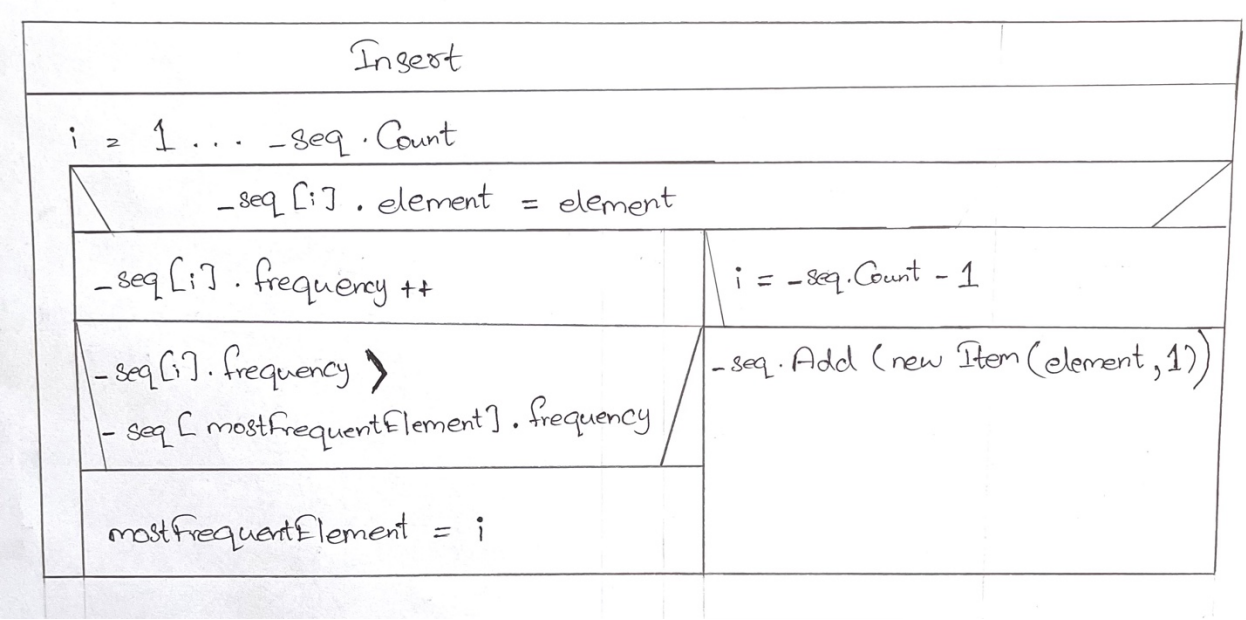
mostFrequentElement = 2 (index of the mostFrequentElement)

Since the program says we need to store (element , frequency) format, we will constantly check what is the index of mostFrequentElement after inserting and removing elements. So that we can easily give out what is the mostFrequentElement since we knows the index of it by _seq[mostFrequentElement].element which is 7 in this demonstration.

Implementation

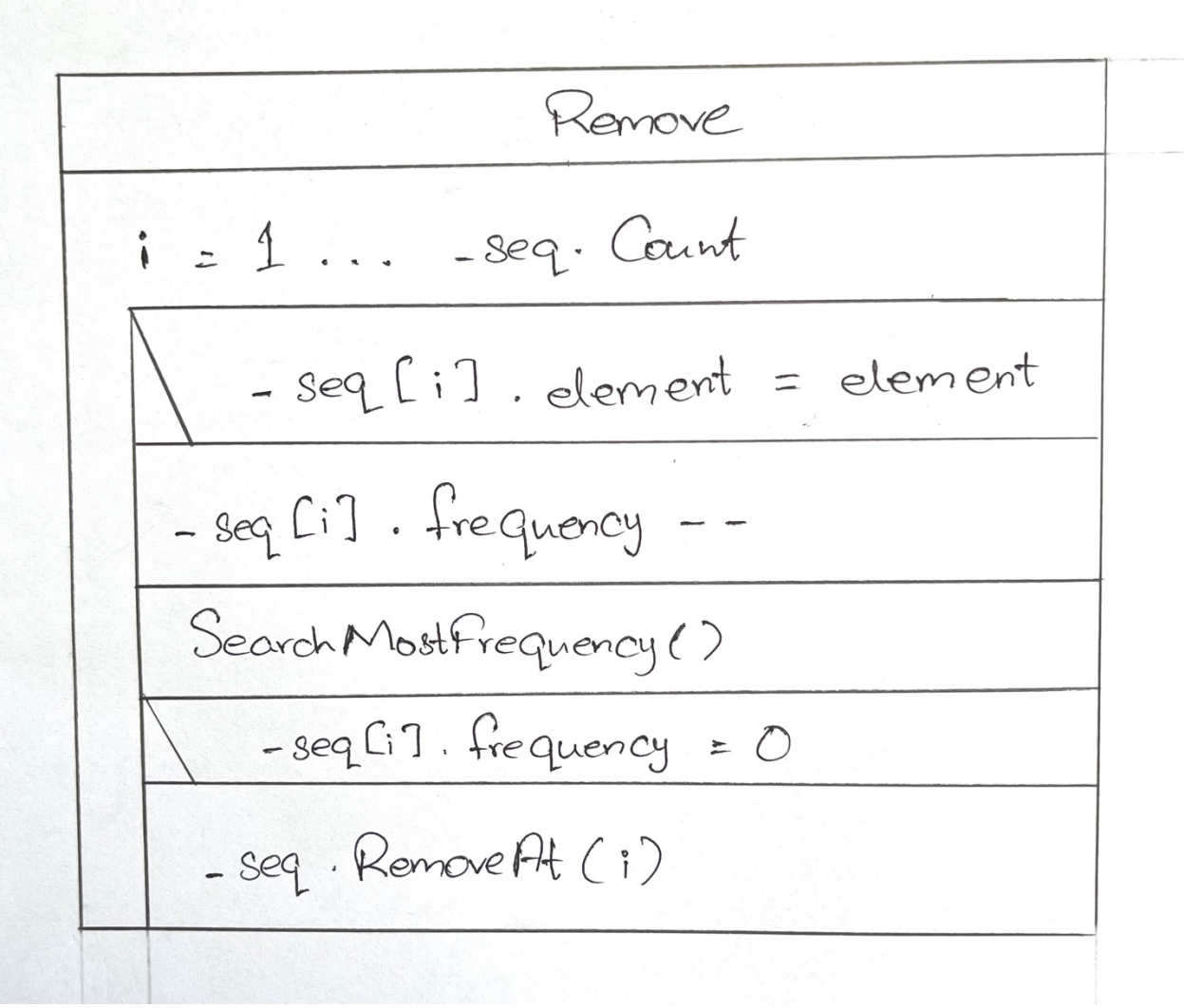
1. Inserting an element

Insert an element to the bag and store the mostFrequentElement by SearchMostFrequency method. If the inserted element is new, then its frequency is 1. If it is already existed in the bag, we will just add its frequency by 1.



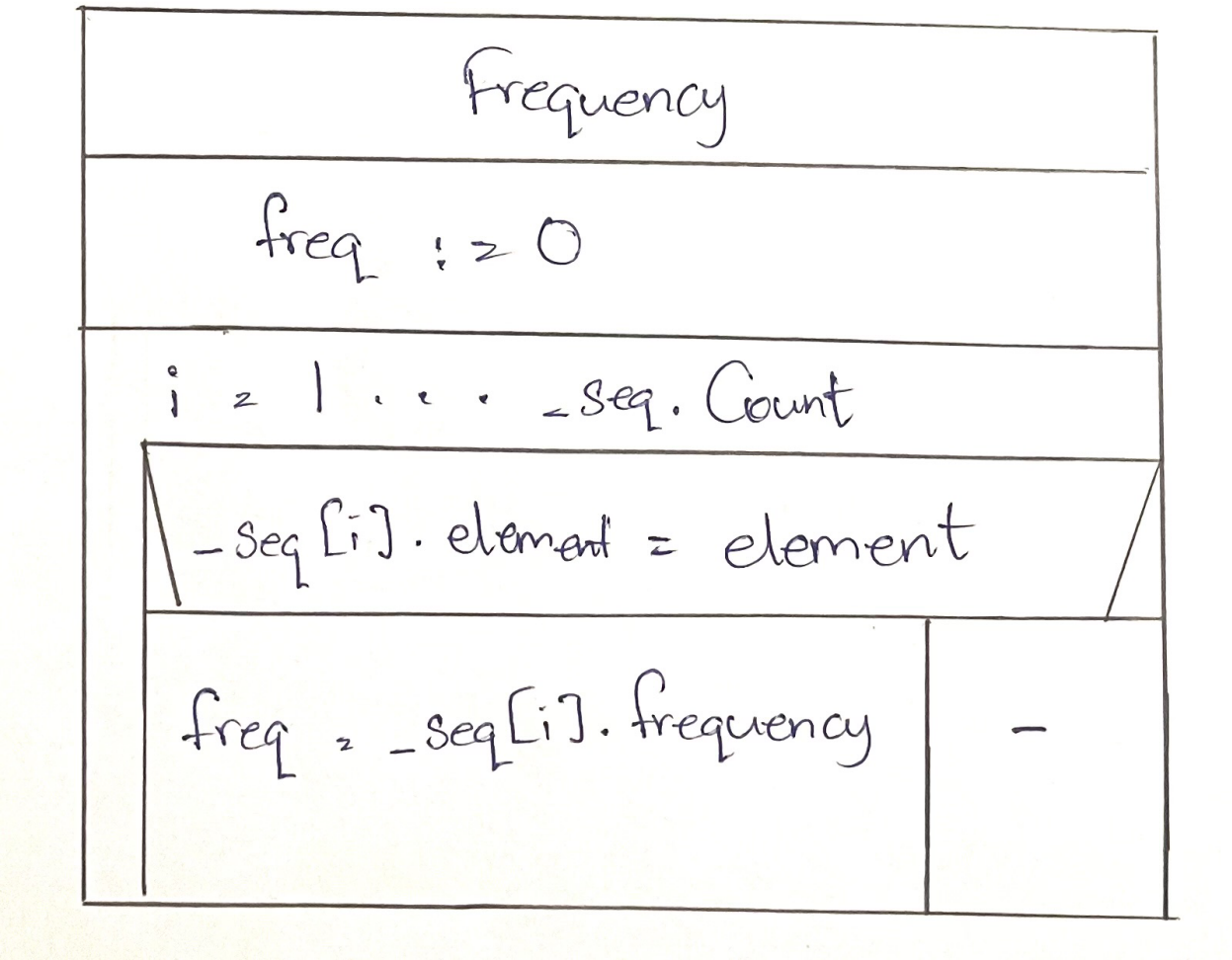
2. Removing an element

Remove an element in the bag if we can find it in the bag and reduce its frequency by 1. And then, we will do SearchMostFrequency method to find the mostFrequentElement again and if the frequency of the element that got deleted is zero, we will remove it from the bag.



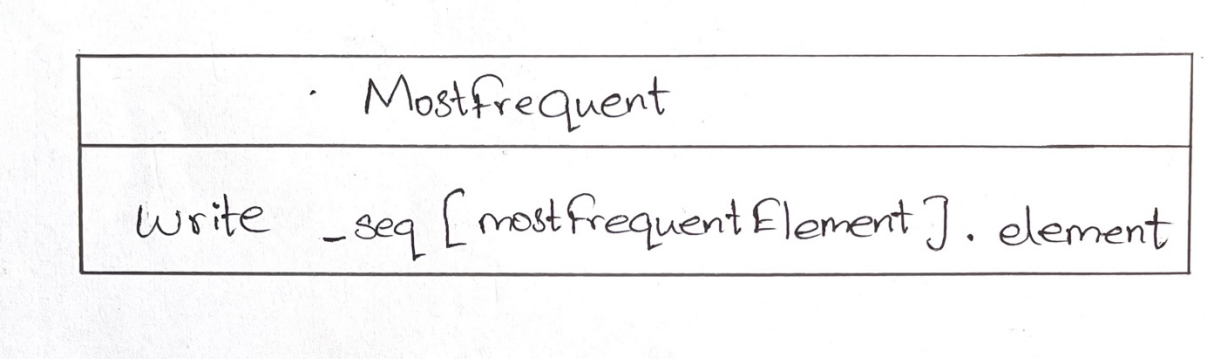
3. Getting the frequency of an element

Get the newly added element and check if it is the same as one of the elements in the bag and if it is, we will give out the corresponding frequency.

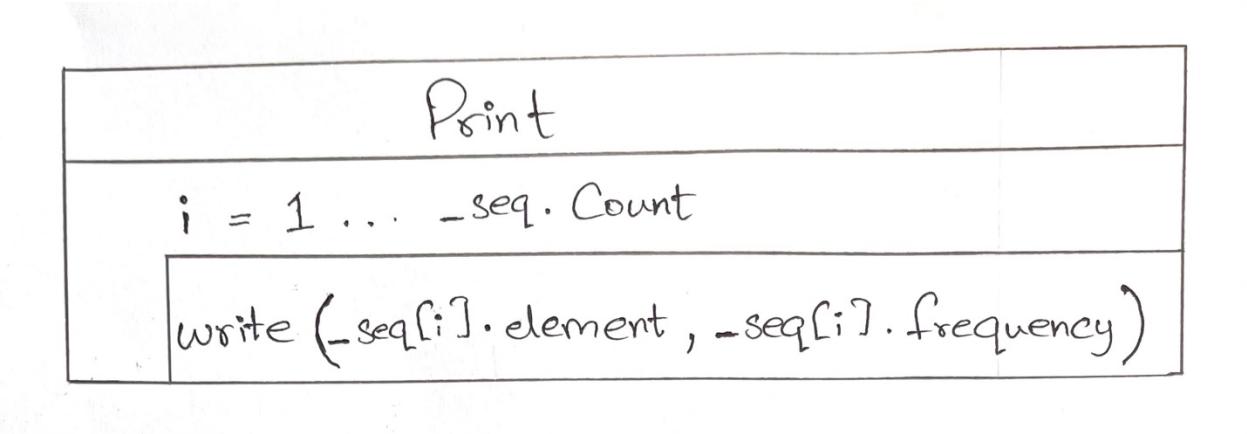


4. Getting the most frequent element

We did all the operation for it in the first two methods so we will simply just return it with the index of mostFrequentElement.

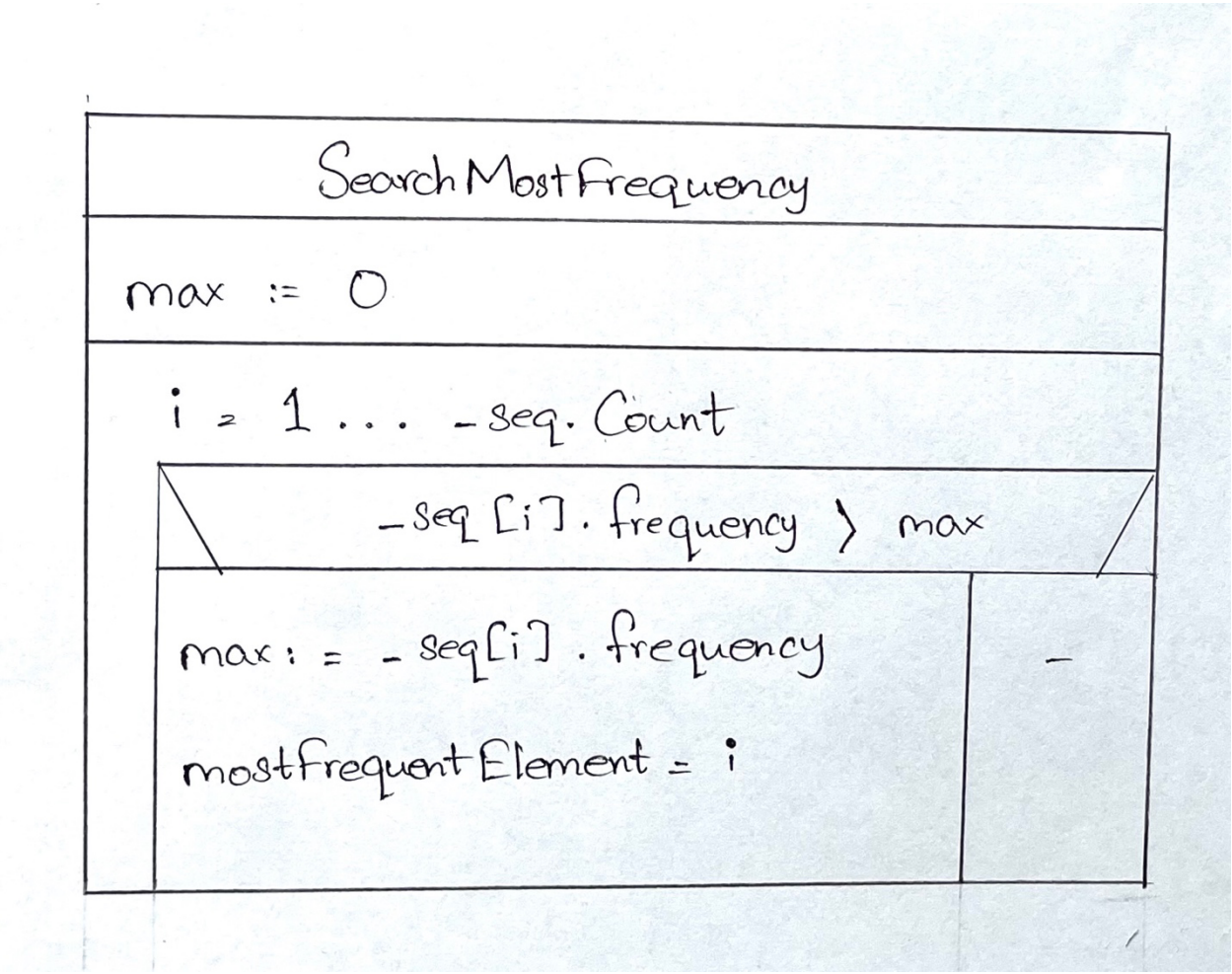


5. Printing the bag



Printing the bag with the pair of element and frequency.

6. SearchMostFrequent : the method that is used to find the mostFrequentElement in Insert and Remove methods.



Tesing

Testing the operations (black box testing)

1. Create a struct to pair element and frequency. And then, create a list of structs.
2. Checking if we can insert an element to an empty bag. If so, that element will be the value of index mostFrequentElement.
   Check If we can insert an element which is already existing in the bag. If so, the corresponding frequency of that element should also be incremented.
   Check if we can insert an element that is not in the bag. If so, then the frequency of that element is 1.

3. Check if we can remove an element from the empty bag and then there should be exception handling.
   Check that if the removed element is the mostFrequentElement. If so, we should implement another method to search for it again.
   Check if the element to be removed is not in the bag, an error handling should occur to indicate that the element is not in the bag.
4. Check the element to get the frequency if it is in the bag. If yes, just return the frequency of it. If not, we should give an error.
5. Check there is a mostFrequentElement of the bag. If yes, just return it and if it is an empty bag, we should give an error. Insert elements, find the mostFrequentElement and then remove elements to change it and see if the value of mostFrequentElement is updated or not.
6. Printing the whole bag if there are elements in it. If not, it is an empty bag.

Testing based on the code (white box testing)

1. Generating and catching exceptions.