Saifeddin Al-kurdi O9ffdc	OOP Assignment Number 8	Group 5

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), and printing the bag. The lecture code cannot be submitted.

Plan

We have a bag class that has multiple elements each element has a number and a frequency if it is, we need to create four methods that work on this data type, to add and remove elements to find an element by its frequency and we have to find an element's number by finding the highest frequency

Bag Data Type

Set of values

Bag = {(elem,freq)∈ Tuple | element ∈ Z, frequency ∈ N }

Operations

1-AddElem Function/Method

When we first add a new tuple it checks if there are any other tuples with the same element if not it adds smoothly if there is a similar element it ends up adding their frequencies together e.g. AddElem(a,2) and there exists (a,3) it ends up equating to (a,5)

A = {x: Tuple, x.elem: Z, x.freq: N ,l: L}

Pre = {x=x'∧ x.elem=x.elem' ∧ x.freq=x.freq'}

Post={Pre ∧ (*l*,x.elem) = *SEARC*H i=1..|x| *Cond(Element=x.elem)* →*Tuple(Element, Frequency+x.freq*)otherwise *Tuple (x.elem, x.freq)*

***2*-RemElem Function/Method**

When we want to remove a new tuple it checks if there are any other tuples with the same element if not it adds smoothly if there is a similar element it ends up adding their frequencies together e.g. AddElem(a,2) and there exists (a,3) it ends up equating to (a,5)

A = {x: Tuple*, x.elem: Z, x.freq: N , l: L}

Pre = {x=x'∧ x.elem=x.elem' ∧ x.freq=x.freq'}

Post={Pre ∧ (*l*,x.elem) = *SEARC*H i=1..|x| *Cond(Element=x.elem)* →*Tuple(Element, Frequency-x.freq*)otherwise *Remove Tuple(x.elem, x.freq)*

***3*-FreqOfElem Function/Method**

When we want to find the frequency of a certain element we check if the bag is empty if not then we search for the tuple with the element we want then print its frequency

A = {x: Tuple*, x.elem: Z}

Pre = { x=x'∧ x.elem=x.elem'}

Post={Pre ∧ (x.elem) = *SEARC*H i=1..|x| *Cond(Element=x.elem)* → *x.freq*

***4*-MaxFreqElem Function/Method**

When we want to find the highest frequency of a certain element we check if the bag is empty if not we check if it has only one element and IF not we do a max search to find the element with the highest frequency we make two variables to save the mostFrequentElement and maxFrequency as zeros then we loop in the tuples and replace the biggest to be in the mostFrequentElement and maxFrequency

A = {x: Tuple*, x.freq: N , mostFrequentElement = Z , maxFrequency = N}

Pre = {x=x'∧ x.freq=x.freq' ∧ mostFrequentElement = mostFrequentElement' ∧ maxFrequency= maxFrequency'}

Post={Pre ∧ (mostFrequentElement, maxFrequency) = MAX I=1..|bag| cond(maxFrequency < x.freq) → x.elem ,otherwise mostFrequentElement

Representation

The Bag consists of pairs represented as **List<(int, int)>**, **List<item1, item2...>** class, where item1, item2 represent the types of the data elements. Each pair contains two integers, element and frequency,respectively.

Implementation

1- Adding an element

If the BAG has a tuple (x,y) where x = element, and y= frequency, then we add (2, 1) if the element doesn't exist, or it becomes (2, y+1) if the element already exists.

```
AddElem(x,y)

Exists = false
    for each (elem, freq) tuple in bag:

        if elem == Element:
        bag[indexOf(tuple)] = (element, freq + 1)

        Exists = true
        break

    if not Exists:

        bag.add((elem,freq ))
```

2- Removing an element

If the BAG has a pair (x,y) where x = element, and y= frequency, then remove (2,1) if the element doesnt exist ,or if it does exist we do this (2,y-1) if the fruency becomes zero then the tuple is removed from the bag.

```
RemElem(x,y):
If Bag not empty
    tupleToRemove = for each (elem, freq) tuple in bag: if elem == Element:
    if tupleToRemove!= null
        bag[indexOf(tuple)] = (Elemet,Frequency-freq)
        if tupleToRemove.Frequency <= 0
            Remove.tupleToRemove
    else
        return false
```

3- Getting the frequency of an element

If the BAG has a tuple (x,y) where a = element, and b= frequency. And b ≥ 1 (*element exists*), then return b. can be implemented as:

```
FreqOfElem(x)
for each tuple in bag
    if tuple.element == elem
    return tuple.freq
```

4-Getting the most frequent element

If the BAG has a tuple (x,y) where x = element, and y= frequency. And bag isnt empty, then return most frequent element. can be implemented as:

```
MaxFreqElem()

If |Elements| = 0

        then bag is empty

If |Elements| = 1

        then we have one element it doesnt work like that

mostFrequentElement = 0

maxFrequency = 0

for each element in bag:
        freq = FreqOfElem(element)

         if freq > maxFrequency:

                maxFrequency = freq

                mostFrequentElement  = elem

return mostFrequentElement
```

Testing

(Black box testing)

1- (TestAddElem())

Adding the element 4 with frequency 6 , since it doesn’t exist it will be (4,6) , then adding the element 4 with frequency 4 , since element 4 already exists then it becomes (4,10). And to make sure of that we use the method.AreEqual(10, freqOfElem(4)) and it should be correct

2- (TestRemElem())

We add an element with its frequency for example (2,1) then we call the built-in function IsTrue(bag.RemElem(2,1)) and it should be true because we removed the element and it is a successful process.

3-(TestfreqOfElem())

We add an element 1 with its frequency 2 then to make sure it’s correct we use the built-in function.AreEqual(2, bag.FreqOfElem(1)), so since the frequency of element 1 is 2, they are equal so it is true.

4-(TestMostFrequentElement())

We add three elements with their frequency (1,2),(2,1),(1,1) , so here they end up being (1,3),(2,1) so when we call the function MaxFreqElem on them and equal with the built-in function AreEqual like this.AreEqual(1, bag.MaxFreqElem()); is it true because the highest frequency is element 1's frequency then we add an element (2,3) so the bag with the tuples becomes (1,3),(2,4) and then we call this again and it's the 2 so it's true.AreEqual(2, bag.MaxFreqElem());

(White box Testing)

1. Creating a bag with 1000 elements

2. Entering a null value and catching it with exceptions.

3.Putting a number of a method that doesnt exist.