Bag

Written and Implemented

by

Zayar Htet (DRCVG2)

OOP ELTE

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

Bag = { (x,y) $\in \mathbb{Z}$ }

Bag(n) = (x, y) where x, y $\in \mathbb{Z}$, x is the value and y is the occurence frequency.

Operations

1. Insert
   Inserting an element into a bag. If the element exists in the bag, then increase its frequency. If the element does not exist, just insert a new pair with frequency 1.

   A: Bag(n), Pair = (element x frequency), element, frequency $\in \mathbb{Z}$, e $\in \mathbb{Z}$, c $\in$ [1..n]
   Pre: a = a' $\wedge$ e = e' $\wedge$ c = c'
   Post: Pre $\wedge$ Exists = $\exists$i $\in$ [1..n]: Bag[i].element = e $\rightarrow$ c := e
   Exists $\rightarrow$ Bag[c].frequency := Bag[c].frequency+1
   Otherwise $\rightarrow$ Bag[n+1] := (e,1)

2. Remove
   Removing an element from a bag. If the element exists in the bag, then reduce its frequency. After reducing, if the frequency is zero (0), then erase that element from a bag. If the element does not exist, throw an exception.

   A: Bag(n), Pair = (element x frequency), element, frequency $\in \mathbb{Z}$, e $\in \mathbb{Z}$, c $\in$ [1..n]
   Pre: a = a' $\wedge$ e = e' $\wedge$ c = c'
   Post: Pre $\wedge$ Exists = $\exists$i $\in$ [1..n]: Bag[i].element = e $\rightarrow$ c := e
   Exists $\rightarrow$ Bag[c].frequency := Bag[c].frequency – 1
   Bag[c].frequency = 0 $\rightarrow$ erase Bag[c]
   Otherwise $\rightarrow$ throw NonExistingElementException

3. getFrequency
   Getting the frequency of the element. If the element does not exist, then just return zero (0). If the element exists, return the frequency of that element.

   A: Bag(n), Pair = (element x frequency), element, frequency $\in \mathbb{Z}$, e $\in \mathbb{Z}$, c $\in$ [1..n], result $\in$ N
   Pre: a = a' $\wedge$ e = e' $\wedge$ c = c'
   Post: Pre $\wedge$ Exists = $\exists$i $\in$ [1..n]: Bag[i].element = e $\rightarrow$ c := e
   Exists $\rightarrow$ result := Bag[c].frequency

Otherwise → result := 0

4. mostFrequentElement
   Getting the most frequent element in a bag. If the bag is empty, just throw an exception. If the bag is not empty, then return the element with the greatest number of frequency.

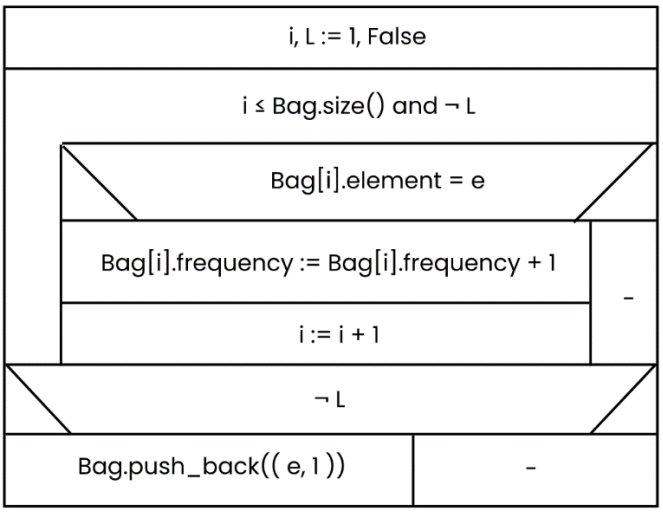
   A: Bag(n), Pair = (element x frequency), element, frequency ∈ $\mathbb{Z}$,
   most ∈ Bag, result ∈ $\mathbb{Z}$
   Pre: -
   Post: n = 0 → throw EmptyBagException
   otherwise → ∃i ∈ [1..n]: ∀j∈[1..n]: Bag[j].frequency <= Bag[i].frequency
   most := Bag[i] ∧ result := most.element

Representation

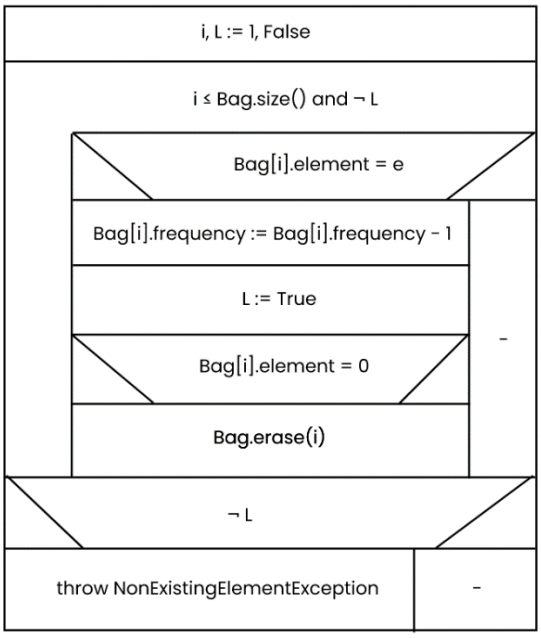
(e_1, f_1) (e_2, f_2) (e_3, f_3) (e_4, f_4) (e_5, f_5)

Implementation

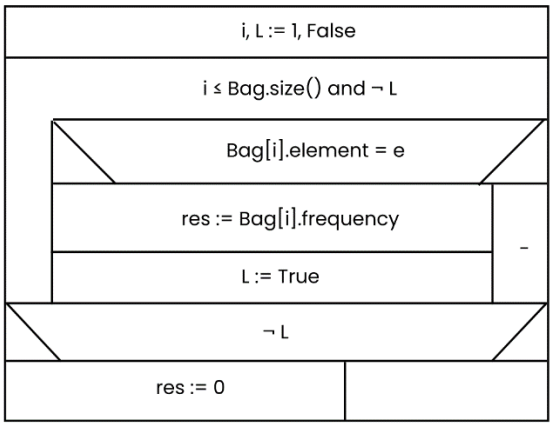
1. Insert
   Inserting an element into a bag. If the element exists in the bag, then increase its frequency. If the element does not exist, just insert a new pair with frequency 1.



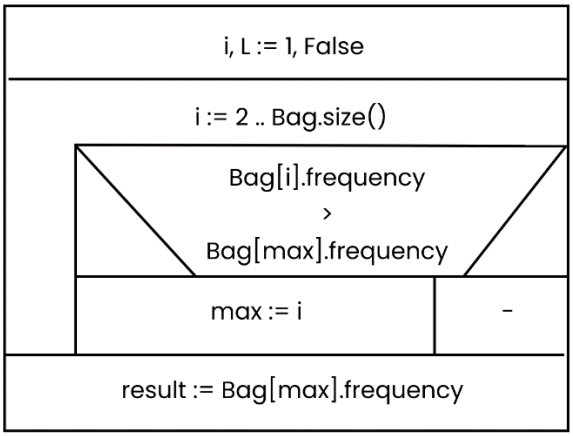
2. Remove
   Removing an element from a bag. If the element exists in the bag, then reduce its frequency. After reducing, if the frequency is zero (0), then erase that element from a bag. If the element does not exist, throw an exception.



3. getFrequency
   Getting the frequency of the element. If the element does not exist, then just return zero (0). If the element exists, return the frequency of that element.



4. mostFrequentElement
   Getting the most frequent element in a bag. If the bag is empty, just throw an exception. If the bag is not empty, then return the element with the greatest number of frequency.



Testing

Testing the operations (Black box Testing)

1. Inserting an element
   a. inserting an element
   b. inserting an existing element
   c. checking the frequency of the element
   d. inserting an illegal value
2. Removing an element
   a. removing an existing element
   b. removing a non-existing element
   c. checking the exception
3. Checking getFrequency
   a. Getting the frequency of an existing element
   b. Getting the frequency of a non-existing element
   c. checking the exception
4. Checking the mostFrequencyElement
   a. Getting the mostFrequencyElement
   b. Getting the mostFrequencyElement while the bag is empty