Tuguldur Sosorbaram
EERDEY
Sutu0318@gmail.com

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 number of elements which occur only once in the bag (suggestion: store the number of these elements and update it when the bag changes), printing the bag. Lecture code cannot be submitted.

Bag type set of values

Element(e, f) = {e ∈ $\mathbb{Z}$, f ∈ $\mathbb{N}$ }

BagType() = { seq ∈ $\mathbb{Z}^n$, SingleElemCount ∈ $\mathbb{N}$ }

Operations

1. *Inserting*
   Inserting new element to a bag.
   A = (b:Bag, e: $\mathbb{Z}$)
   Pre = (b=b` ∧ e=e`)
   Post = (b = b ∪ e)

2. *Removing*
   Removing given existing element from the bag.
   A = (b:Bag, e: $\mathbb{Z}$)
   Pre = (b=b` ∧ e=e` ∧ b ≠ ∅ ∧ e ∈ b)
   Post = (b = b / e)

3. *Frequency of an element*
   Number of given element in the bag.
   A = (b:Bag, e: $\mathbb{Z}$, f: $\mathbb{N}$)
   Pre = (b=b` ∧ e=e` ∧ b ≠ ∅ ∧ e ∈ b)
   Post = (Pre ∧ f = b.frequent(e))

4. *Single elements count*
   Number of elements occurring only once in the bag.
   A = (b:Bag)
   Pre = (b=b` ∧ b ≠ ∅)
   Post = (Pre ∧ f = b.getSec())

Tuguldur Sosorbaram
EERDEY
Sutu0318@gmail.com

Representation

Bag:

Element:	E1	E2	E3	…
SEC	0			

Sec = Single element count

Type invariant:

SEC : $\mathbb{N} \wedge \mathrm{SEC} = \sum_{\substack{e \in b \\ e.f=1}} 1$

Element:

e	1	-5	7	…
f	2	1	4	…

e = element key

f = element frequency

Implementation

Before implementation of the operations here is implementation of helper function isin()

This function uses linear search algorithm

A = (b:Bag, e:Z, l:L, ind:Z)

Pre = (b = b` ∧ e =e`)

Post = (Pre ∧ l,ind = $\mathrm{SEARCH}_{i \in b}$ e=i.e)

START

Input integer e

Set l,ind = false, 0

Loop while l is false and ind is less of equal than bag length

l,ind equals (b[ind].e = e)

i = i + 1

end loop

1. *Inserting*

Sutu0318@gmail.com

START
Input integer e
Set l, ind = isin(e)
If l is true
Increase frequency of b[ind]
If l is false
Add e as a new element to b

END

But inserting a new element might change the single element count. So, on both execution of if statement single element count should be checked.

START
Input integer e
Set l, ind = isin(e)
If l is true
If b[ind] frequency is 1 decrease Single Element Count by 1
Increase frequency of b[ind] by 1
If l is false
Add e as a new element to b
Increase Single Element Count by 1

END

2. *Removing*

START
Input integer e
Set l, ind = isin(e)
If l is true
Decrease frequency of b[ind] by 1
If b[ind] frequency is 1 increase Single Element Count by 1
If frequency of b[ind] is 0
decrease Single Element Count by 1
Remove from the bag
END
Highlighted part is for updating single element count

3. *Frequency*

START
Input integer e
Set l, ind = isin(e)
If l is true
Output frequency of b[ind]
END

Sutu0318@gmail.com

4. *Single element count*
   Single element count is always updated whenever new element inserted or any element removed.
   START
   Output Single element count.

   END

Used algorithmic patterns

Linear search for helper function isin()

Other operations are very simple with the help of isin() functions. No need to use algorithmic patterns.

But I used the counting algorithm in another helper function. But that function is only used for testing.

Testing

Black box testing:

1. Inserting test
   a. Inserting to empty bag
   b. Inserting existing element to non-empty bag
   c. Inserting non-existing element to non-empty bag
2. Remove test
   a. Try removing from empty bag
   b. Try removing non-existing element from non-empty bag
   c. Removing existing element from bag with only 1 element
   d. Removing existing element from bag with multiple element
3. Frequent test
   a. Getting count from empty bag
   b. Getting count of non-existing element from non-empty bag
   c. Getting count of existing element from bag with only 1 element
   d. Getting count of existing element from bag with multiple elements
4. Getting Single element count
   a. SEC from empty bag
   b. SEC from bag with no single element
   c. SEC from bag with only 1 single element
   d. SEC from bag with multiple elements
   e. SEC before and after inserting
   f. SEC before and after removing

White box testing:

Generating and catching all the exceptions.

Tuguldur Sosorbaram
EERDEY
Sutu0318@gmail.com

Test cases with expected outputs:

1. **Inserting test**
   a. **Inserting to empty bag**
      Insert e1
      Bag.count = 1
   b. **Inserting existing element to non-empty bag**
      Insert e1
      Bag.count = 1
      Insert e1
      Bag.count = 2
   c. **Inserting non-existing element to non-empty bag**
      Insert e1
      Bag.count = 1
      Insert e2
      Bag.count = 2
2. **Remove test**
   a. **Try removing from empty bag**
      Remove e1
      Expect EmptyBagException
   b. **Try removing non-existing element from non-empty bag**
      Remove e1
      Expect NotElementException
   c. **Removing existing element from bag with only 1 element**
      Insert e1
      Remove e1
      Bag.count = 0
   d. **Removing existing element from bag with multiple element**
      Insert e1
      Insert e2
      Remove e1
      Bag.count = 1
3. **Frequent test**
   a. **Getting count from empty bag**
      Count e1
      Expect EmptyBagException
   b. **Getting count of non-existing element from non-empty bag**
      Count e1
      Expect NotElementException
   c. **Getting count of existing element from bag with only 1 element**
      Insert e1
      Count e1 = 1
   d. **Getting count of existing element from bag with multiple elements**

Sutu0318@gmail.com

Insert e1
Insert e1
Count e1 = 2

4. **Getting Single element count test**
    a. **SEC from empty bag**
    Expect EmptyBagException
    b. **SEC from bag with no single element**
    SEC = 0
    c. **SEC from bag with only 1 single element**
    SEC = 1
    d. **SEC from bag with multiple elements**
    SEC = 2
    e. **SEC increasing after inserting**
    Insert e1
    SEC = 1
    Insert e2
    SEC = 2
    f. **SEC decreasing after inserting**
    Insert e1
    Insert e2
    SEC = 2
    Insert e1
    SEC = 1
    g. **SEC increasing after removing**
    Insert e1
    Insert e1
    SEC = 0
    Remove e1
    SEC = 1
    h. **SEC decreasing after removing**
    Insert e1
    Insert e2
    SEC = 2
    Remove e1
    SEC = 1