Murad Huseynov **1. assignment/10. task** 27th March 2023
RX15MW
rx15mw@inf.elte.hu
Group 10

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 largest element in the bag (suggestion: store the largest element and update it when the bag changes), print the bag.

Bag type

Set of values

Bag = { Item : (*element, frequency)* $\in \mathbb{N}^{1..m} \mid$ *element* $\in \mathbb{N}$, *frequency* $\in \mathbb{N}$, *m* $\in \mathbb{N}$ }

Operations

1. Inserting an element into the bag
Inserting element e into the bag b: ($e \in [1..m] \Leftrightarrow |B|$): *b.InsertElement(e)*.

Formally: *A* : (*b : Bag* , *e* : $\mathbb{N}$)
Pre = (*b* = *b'*)
Post = (*b* $\underline{\cup}$ *{e}*)

2. Removing an element from the bag
Removing element e from the Bag b: *(*$e \in [1..m]$*)*: *b.RemoveElement(e))*. Elements which are not in the bag cannot be removed: ($e \notin [1..m]$)

Formally: *A* : (*b : Bag* , *e* : $\mathbb{N}$)
Pre = (*b* = *b'* ^ *e* $\in$ *b*)
Post = (($\forall i \in$ [1.. |*b*|]: *b.seq[i].element* = *e)* -> *b.RemoveElement(e)*)

3. Returning the frequency of an element from the bag
Returning the frequency of element e from the bag b: *b.GetFrequency()*

Formally: *A* : (*b : Bag* , *e* : $\mathbb{N}$, *f :* $\mathbb{N}$)
Pre = (*b* = *b'* ^ *b.Count* $\neq$ *0*)
Post = (Pre ^ ($\forall i \in$ [1.. |*b*|]: *b.seq[i].element* = *e)* -> *(f* = *b.GetFrequency(e))*)

4. Returning the largest element in the bag
Returning the largest element in the bag b: *b.GetLargest()*

Formally: *A* : (*b : Bag* , *l* : $\mathbb{N}$)
Pre = (*b* = *b'* ^ *b.Count* $\neq$ *0*)
Post = (Pre ^ $\forall i \in$ [1.. |*b*|]: *b.seq[i].element* < *b.seq[i+1].element* :
l = *b.seq[|b|-1].element)*

Representation

The elements in the bag are stored in a list in the form of pairs, together with their frequency. The pairs are sorted in increased order, based on the elements.

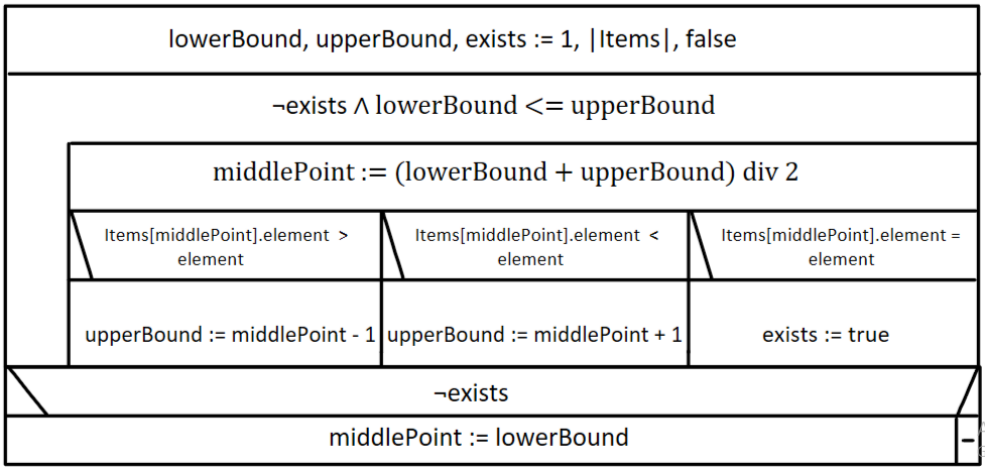
Item = rec (element: $\mathbb{N}$, *frequency:* $\mathbb{N}$)
Items: Item*

Type Invariants:

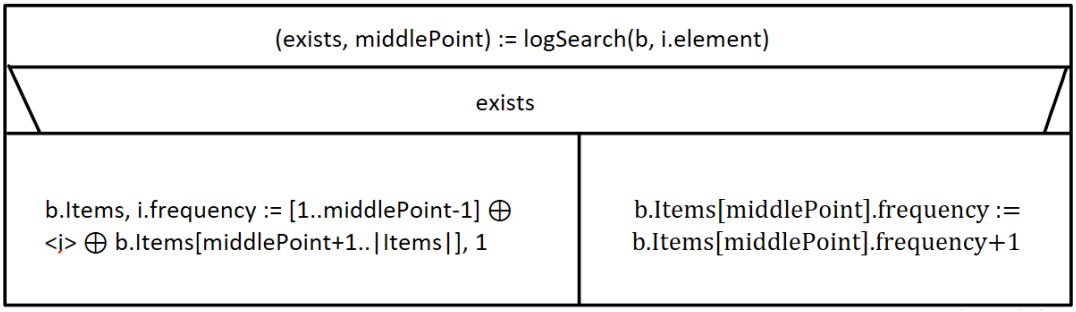
$\forall i \in [1..|\text{Items}|-1]$: Items[i].element < Items[i+1].element
$\text{max} \in [0..|\text{m}|]$: vec[max] = $\mathbf{max}^{m}_{i=0}$ vec[i]

Implementation

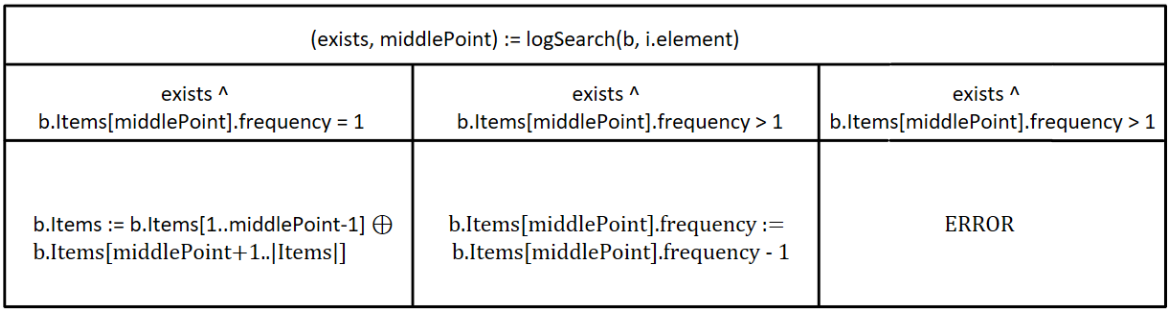
1. logSearch algorithm



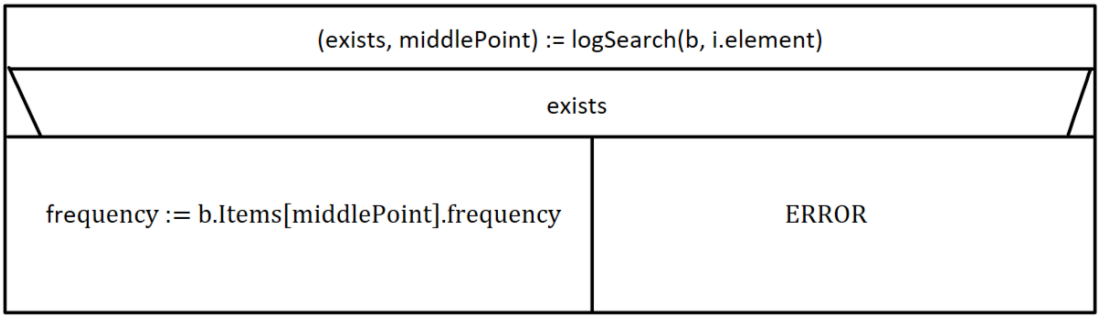
2. Inserting an element



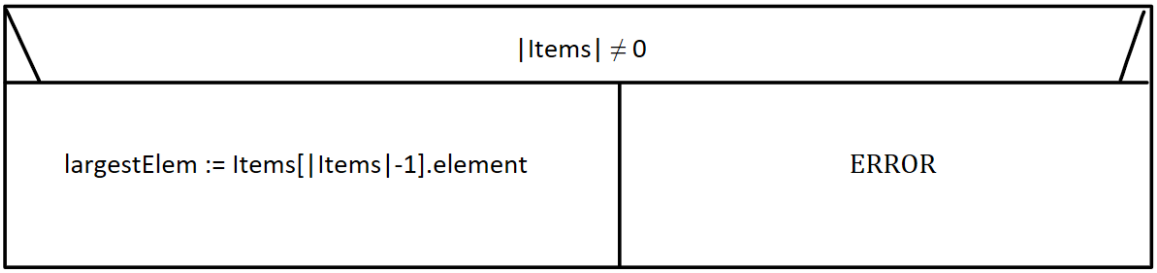
3. Removing an element



4. Getting frequency of an element



5. Getting the largest element



Testing

Testing the operations (black box testing)

1) Creating a bag. The bag is empty by default.
   a) Checking if the bag is empty.

2) Adding the same element into the bag
   a) Creating a bag and adding an element only once and checking whether the element is in the bag and the frequency of it is exactly 1.
   b) Adding the same element multiple times into the and checking if its frequency is increasing by 1 in each add.

3) Adding different elements into the bag
   a) Creating a bag and adding several elements into the bag only once and checking if the elements are in the bag and their frequencies are exactly 1.
   b) Adding these elements more than once and checking if their frequencies are increasing by 1 in each add, accordingly. The frequency of each added element is exactly the same as the number of relevant adds.

4) Adding the same element into an empty bag and removing it
   a) Creating a bag and adding an element into the bag only once and checking whether the element is in the bag and its frequency is exactly 1. Afterwards, removing the element and the bag is empty.
   b) Adding the same element 2 times and checking if the frequency of the element is 2. Removing element once and the frequency of the element is 1. Removing the element 2 times and the bag is empty.
   c) Adding the same element multiple times and then removing it once, and the bag is not empty, but the frequency of element decreases.

5) Adding different element into a bag and removing them
   a) Creating a bag and adding several elements into the bag only once and checking if their frequencies are one. Afterwards removing the elements and the size of the bag should decrease and the deleted elements should be gone from the bag.
   b) Adding these elements multiple times and checking if their frequencies are increasing in each add, accordingly. Afterwards, removing these elements and checking if their frequencies are decreasing in each remove and if their size is decreasing to zero, the elements should be gone from the bag.

6) Getting frequency of an element
   a) Creating a bag and adding several elements different times into the bag and checking if their frequencies are equal to the number of adds.
   b) Performing adds and removes and checking if the frequencies are changing accordingly.

7) Getting the largest element in the bag
   a) Creating a bag and adding an element into the bag, once or several times. Since it is the only element, it is the largest.
   b) Adding a second element into the bag that is greater than firstly added element. The second element is the largest one.
   c) Adding a third element which is greater than the first two elements. The third element is the largest.
   d) Adding a fourth element which is less than the latest largest element. The largest remains the same.
   e) Removing the largest element, and the new largest element is the previous one.

Testing based on the code (white box testing)

1. Generating and catching exceptions.
   a) Exception(s) while trying to remove an element in the bag.
   b) Exception(s) while trying to get frequency of an element in the bag.
   c) Exception(s) while trying to get the largest element in the bag.