Seminar 01

Thursday, March 3, 2022 2:52 PM

- Abstract Data Types = contain the list of operations and data structures (containers)
- Data Structures = implementation of the container;
- Containers = something that holds data

BAG (multiset) - ADT

- allows duplicates;
- doesn't care about order or positions. (informally)

TElem - the type of the elements it contains

B = {b | b is a bag with elements of type TElem}

I = {i | i is an iterator over elements of a bag}

init(b);
  preconditions: -
  postconditions: $b \in B$, b is empty

add(b, e);
  preconditions: e is of type TElem, $b \in B$
  postconditions: $b' \in B$, $b' = b \cup \{e\}$

remove(b, e);
  preconditions: e is of type TElem, $b \in B$
  postconditions: $b' \in B$, $b' = b \cup \{e\}$, remove $\leftarrow$ true, if $e \in b$
  remove $\leftarrow$ false, if $e \notin b$

search(b, e);
  preconditions: e is of type TElem, $b \in B$
  postconditions: search $\leftarrow$ true, if $e \in b$
  search $\leftarrow$ false, if $e \notin b$

destroy(b);
preconditions: $b \in B$
postconditions: b is destroyed

size(b);
preconditions: $b \in B$
postconditions: size ← number of elements in b

iterator(b, i);
preconditions: $b \in B$
postconditions: $i \in \hat{I}$, i is an iterator over b

number_of_ocurrences(b, e);
preconditions: e is of type TElem, $b \in B$
postconditions: number_of_ocurrences ← number of times e appears in b

ITERATOR:
init(i, b);
preconditions: $b \in B$
postconditions: $i \in \hat{I}$, i is an iterator over b, i refers to the first element in b or is invalid if b is empty

next(i, b);
preconditions: $i \in \hat{I}$, i is valid
postconditions: $i' \in \hat{I}$, i' refers to the next element refered by i
throws error if i is not valid (exception)
i' is invalid, if i refered to the last element

get_current(i, e);
preconditions: $i \in \hat{I}$, i is valid
postconditions: e is of type TElem, e is the element i refers to

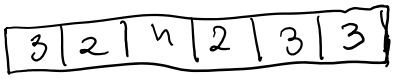
| throws exception if it is not valid.

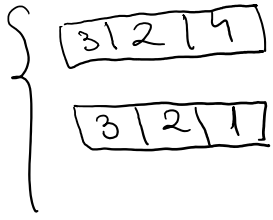
valid(i);
preconditions: $i \in \mathbb{N}$
postconditions: { valid ← true, if i refers to a valid element
valid ← false, otherwise.

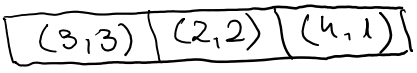
destroy(i);
preconditions: $i \in \mathbb{N}$
postconditions: i is destroyed

first(i, b);
preconditions: $i \in \mathbb{N}$
postconditions: { $i' \in \mathbb{N}$, i' refers to the first element
i' is invalid, if b is empty

Data Structure: Dynamic Array







[] (lists in Python) – dynamic arrays

Implementation (list of pairs (element, frequency of element)):

```
class Bag:
    def __init__(self):
```

```
        self.__data = []

    def add(self, e):
        for p in self.__data:
            if p[0] == e:
                p[1] += 1
            else:
                self.__data.append([e, 1])
    def remove(self, e):
        for p in self.__data:
            if p[0] == e:
                if p[1] > 1:
                    p[1] -= 1
                else:
                    del p # inefficient
                    return true
        return false

class BagIterator:
    def __init__(self, b):
        self.__b = b
        self.__i = 0
        self.__j = 0
```