Implementing a Deque with a Circular Array — X12385_en

The Doble-Ended Queue (Deque) Abstract Data Type

A **deque** (*double-ended queue*) is an abstract data type (ADT) such that an instance D supports the following methods:

D.add_first(e): Add element `e` to the front of deque D.

D.add_last(e): Add element `e` to the back of deque D.

D.delete_first(): Remove and return the first element from deque D; an error occurs if the deque is empty.

D.delete_last(): Remove and return the last element from deque D; an error occurs if the deque is empty.

Additionally, the deque ADT will include accessors:

D.first(): Return a reference to the element at the front of deque D, without removing it; an error occurs if the deque is empty.

D.last(): Return a reference to the element at the back of deque D, without removing it; an error occurs if the deque is empty.

D.is_empty(): Return `True` if deque D does not contain any elements.

len(D): Return the number of elements in deque D; in Python, we implement this with the special method `__len__`.

By convention, we assume that a newly created deque is empty, and that there is no a priory bound on the capacity of the deque. Elements added to the deque can have arbitrary type.

Implementing a Deque with a Circular Array

We can implement the `deque` ADT in much the same way as the `ArrayQueue` class provided in the **public_files** section of this problem statement implements the `Queue` ADT. The same instance variables, `_data`, `_size`, and `_front`, can be used. Whenever we need to know the index of the back of the deque, or the first available slot beyond the back of the deque, we can use modular arithmetic for the computation. For example, the implementation of the `last()` method uses the index

```
back = (self._front + self._size - 1) % len(self._data)
```

The implementation of the `ArrayDeque.add_last` method is essentially the same as that for `ArrayQueue.enqueue`, including the reliance on a `_resize` utility. Likewise, the implementation of the `ArrayDeque.delete_first` method is the same as that for `ArrayQueue.dequeue`. Implementations of `add_first` and `delete_last` use similar techniques. One sublety is that a call to `add_first` may need to wrap around the beginning of the array, which can be done using modular arithmetic to circularly *decrement* the index as follows.

```
self._front = (self._front - 1) % len(self._data)
```

Pqrogramming problem

Define an `ArrayDeque` class that implements the **double-ended queue** (*deque*) ADT as sketched above. You should also write a program that uses the class `ArrayDeque` to process a sequence of orders of the form `add_first element`, `add_last element`, `first`, `last`, `delete_first`, `delete_last`, `len`, `empty`; performs each order requested if it can be executed; and informs the user accordingly.
Hint: Define the `ArrayDeque` class as a subclass of the `ArrayQueue` class provided.

Sample input

```
add_last 5
add_first 3
add_first 7
first
delete_last
len
delete_last
delete_last
add_first 6
last
add_first 8
is_empty
last
```

Sample output

```
Size: 1; last element: 5
Size: 2; first element: 3
Size: 3; first element: 7
First element: 7
5 removed
Size: 2
3 removed
7 removed
Size: 1; first element: 6
Last element: 6
Size: 2; first element: 8
Deque is not empty
Last element: 6
```

Problem information

Author :
Generation : 2022-10-21 12:07:46