Double-ended Queue (II): The Doubly Linked List-based Implementation

CSCD 300 – Data Structures

Eastern Washington University

Goal

We will discuss how to implement the conceptual double-ended queue data structure by using a doubly linked list with dummy head and tail nodes.

The doubly linked list based implementation

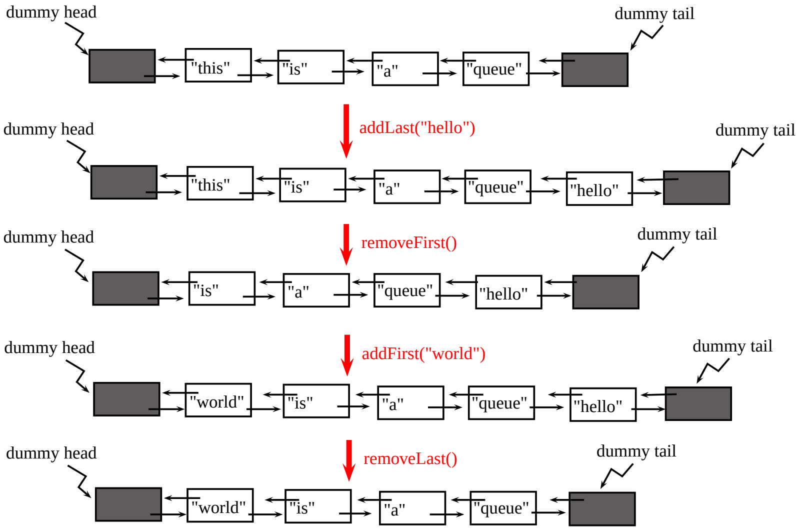
- We use a doubly linked list with dummy head and tail nodes to host the deque.
- The head side of the list is the head side of queue.
- The tail side of the list is the tail side of queue.

Basic API methods

- `getFirst()`: return the head item of the deque.
- `getLast()`: return the tail item of the deque.
- `addFirst(item)`: add the `item` into the head of the deque.
- `addLast(item)`: add the `item` into the tail of the deque.
- `removeFirst()`: remove and return the head item in the deque.
- `removeLast()`: remove and return the tail item in the deque.

An example follows ...

An example sequence of deque's operations



Pseudocode (See the attached Java code for the full implementation.)

Initialization

```
head = new dummy node; tail = new dummy node;
head.prev = null; head.next = tail;
tail.prev = head; tail.next = null;
size = 0;
--
Time cost: O(1)
```

getFirst()

```
if(size > 0)
   return head.next.element;
--
Time cost: O(1)
```

getLast()

```
if(size > 0)
   return tail.prev.element;
--
Time cost: O(1)
```

continue ...

addFirst(item)

```
v = new node(item);
v.prev = head;
v.next = head.next;
head.next.prev = v;
head.next = v;
size ++;
--
Time cost: O(1)
```

addLast(item)

```
v = new node(item);
v.prev = tail.prev;
v.next = tail;
tail.prev.next = v;
tail.prev = v;
size ++;
--
Time cost: O(1)
```

removeFirst()

```
if(size == 0) return null;
deleted = head.next;
head.next.next.prev = head;
head.next = head.next.next;
deleted.prev = null;
deleted.next = null;
size --;
return deleted.element;
--
Time cost: O(1)
```

removeLast()

```
if(size == 0) return null;
deleted = tail.prev;
tail.prev.prev.next = tail;
tail.prev = tail.prev.prev;
deleted.prev = null;
deleted.next = null;
size --;
return deleted.element;
--
Time cost: O(1)
```