FIFO Queue (III): The Singly Linked List-based Implementation

CSCD 300 – Data Structures

Eastern Washington University

EASTERN WASHINGTON UNIVERSITY
start something big

Goal

We will demonstrate how to implement the conceptual FIFO queue data structure by using a physical singly linked list data structure.

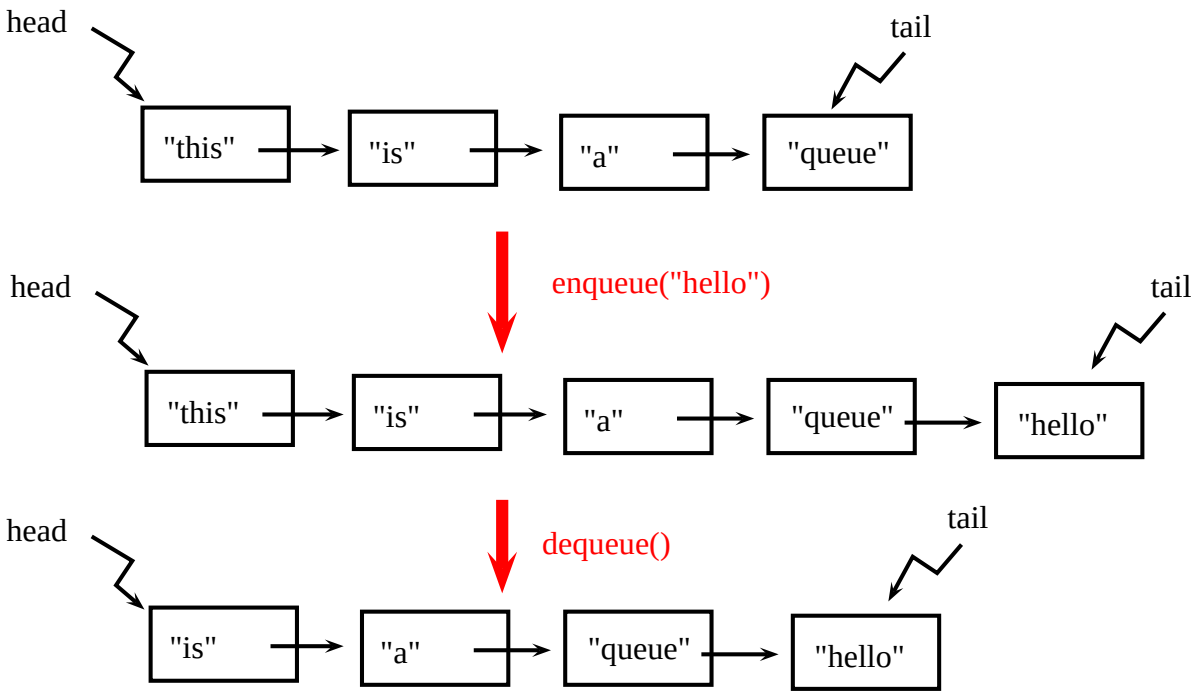
Outline

EASTERN WASHINGTON UNIVERSITY
start something big

The singly linked list-based implementation

- We use a singly linked list to host the FIFO queue.
- The head of the list is the head of queue.
- The tail of the list is the tail of queue.

Question: What if we set the head (tail, resp.) of the list as the tail (head, resp.) of the queue ? Why is that a bad choice ?



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

Initialization

```
head = tail = null;
size = 0;
--
Time cost: O(1)
```

enqueue(item)

```
new_node = new node(item);
if(size == 0)
   head = tail = new_node;
else
   tail.next = new_node;
   tail = new_node;
size++;
--
Time cost: O(1)
```

dequeue()

```
if(size == 0) return error;
ret = head.element;
if(size == 1)
   head = tail = null;
else
   head = head.next;
size --;
return ret;
--
Time cost: O(1)
```

Array-based implementation vs. Linked list-based implementation

The array-based implementation

- Good: fast access; no extra space cost for maintaining the "`next`" links.
- Bad: there is a pre-determined capacity, so the queue size cannot be larger than that capacity; The memory space of those array locations that are not being used is all wasted.

The singly linked list-based implementation

- Good: There is no predetermined capacity, meaning the queue can have as many items as we want, as long as the main memory space is available.
- Bad: Extra space cost for maintaining those "`next`" links; a bit slower than the array-based implementation because of those link change operations when we do the `enqueue` and `dequeue` operations.