CSC3100 Data Structures Tutorial 6: Queue

Yaomin Wang

Contents

- Queue Concept
- Queue Variants
- Queue Exercise

Queue

First-In, First-Out (FIFO)

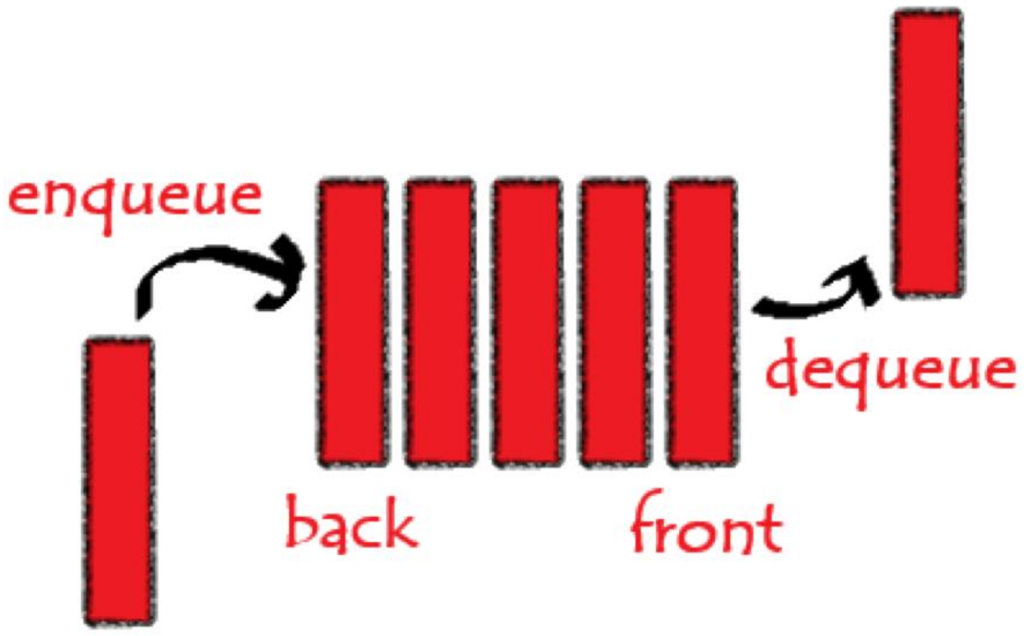


Figure: Queue analogy

Basic operations

- enqueue ()
  - Add elements to the back of the queue.
- dequeue ()
  - Remove elements from the front of the queue.

Double-ended Queue

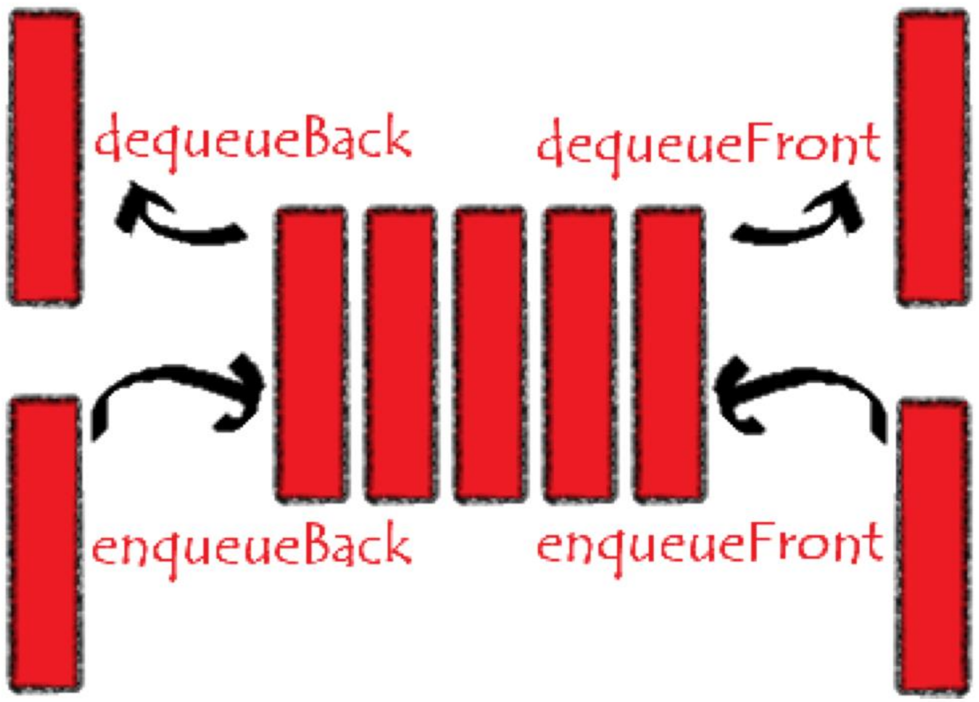


Figure: Double-ended Queue analogy

Double-ended queue is a more flexible queue that allows enqueue and dequeue at both ends.

Circular Queue

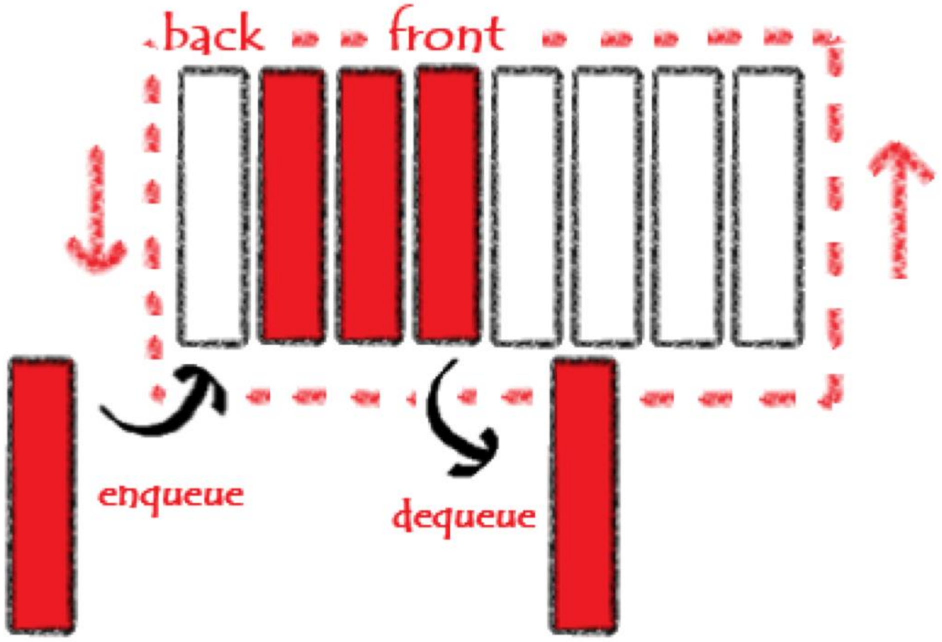
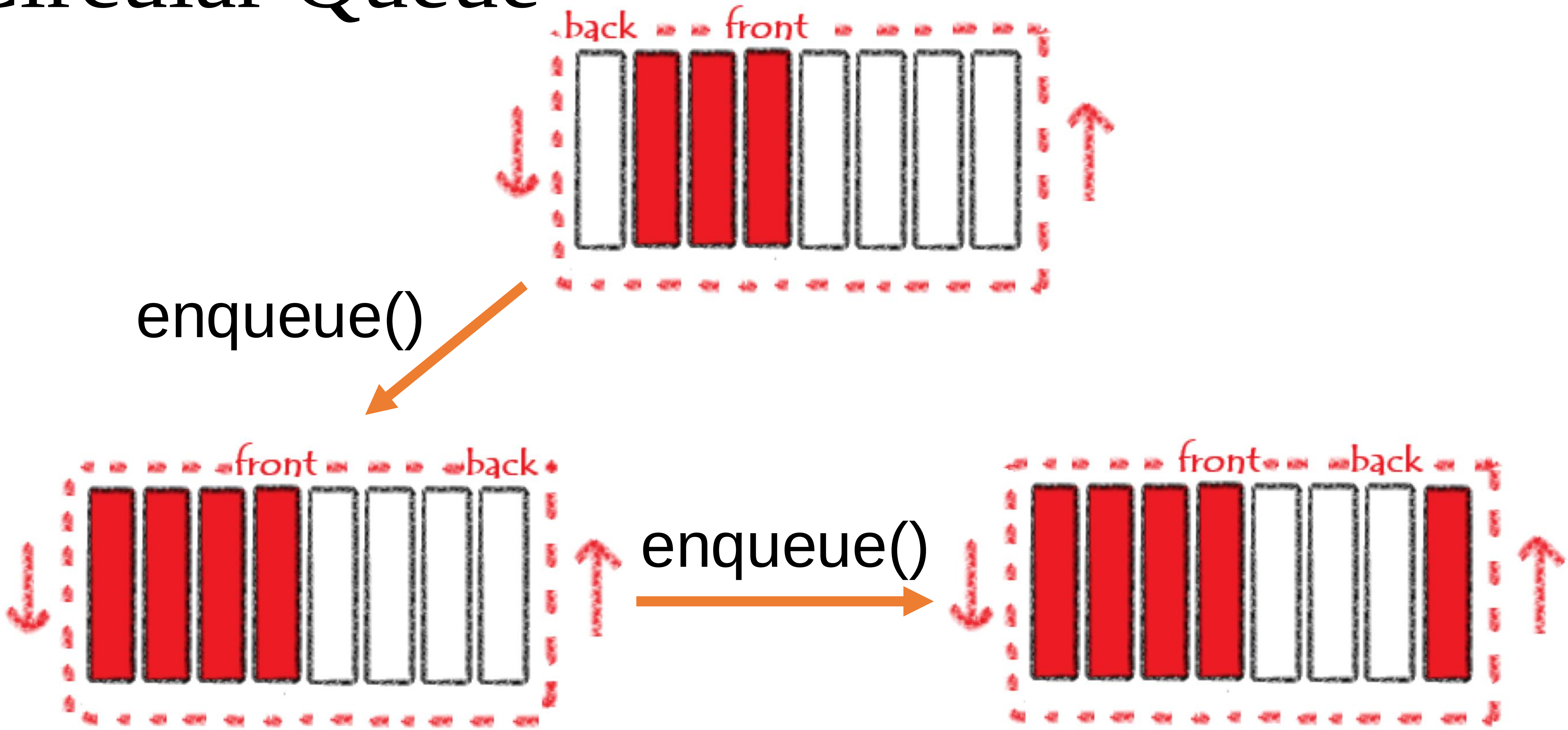


Figure: Circular Queue analogy

A queue in a fixed-size circular buffer, making it as if it were connected end-to-end. With fixed capacity but it is memory efficient.

Circular Queue



Priority Queue

A queue where elements are processed based on priority.
Unlike the standard queue, it does not necessarily implements a First-In, First-Out, policy.

Will be covered in more detail in Tutorial about heap.

Exercise 1: Implement stack using queues(LeetCode P225)

Implement a last-in-first-out (LIFO) stack using only two queues.

https://leetcode.com/problems/implement-stack-using-queues/

Exercise 2: Sliding Window Maximum(LeetCode P239)

Given an array of integers, there is a sliding window of size k which is moving from the very left of the array to the very right. You can only see the k numbers in the window. Each time the sliding window moves right by one position.

Example 1:

```
Input: nums = [1,3,-1,-3,5,3,6,7], k = 3
Output: [3,3,5,5,6,7]
Explanation:
Window position                Max
---------------               -----
[1  3  -1] -3  5  3  6  7       3
 1 [3  -1  -3] 5  3  6  7       3
 1  3 [-1  -3  5] 3  6  7       5
 1  3  -1 [-3  5  3] 6  7       5
 1  3  -1  -3 [5  3  6] 7       6
 1  3  -1  -3  5 [3  6  7]      7
```

https://leetcode.com/problems/sliding-window-maximum/

Solution 2: Sliding Window Maximum(LeetCode P239)

Method 1: Brute-Force

For a sequence with length n, there exists n-k+1 windows, the time complexity to find the maximum value in a window is $O(k)$. The overall time complexity is $O(nk)$.

We could save the time by ***double-end queue*** to reduce the time complexity of finding the maximum value in a window to $O(1)$.

https://leetcode.com/problems/sliding-window-maximum/

Solution 2: Sliding Window Maximum(LeetCode P239)

Method 2: Monotonic queue

Key observations:

- If the **current** element is **greater than or equal to** the element at the **end** of the data structure, then the latter no longer influences the maximum of future windows; hence, it **can be discarded**.
- If the element at the **beginning** of the data structure and the **current** element is *k* **distance away**, then the former no longer influences the maximum of future windows; hence, it **can also be discarded**.
- From the beginning to the end of the data structure, the elements are **sorted in descending order**; hence, the **maximum lies** at the **beginning** of the data structure.

https://leetcode.com/problems/sliding-window-maximum/

Solution 2: Sliding Window Maximum(LeetCode P239)

Method 2: Monotonic queue

								Queue	Maximum Item
1	3	-1	-3	5	3	6	7	3, -1	3
1	3	-1	-3	5	3	6	7	3, -1, -3	3
1	3	-1	-3	5	3	6	7	3, -1, -3, 5	5
1	3	-1	-3	5	3	6	7	5, 3	5
1	3	-1	-3	5	3	6	7	5, 3, 6	6
1	3	-1	-3	5	3	6	7	6, 7	7

https://leetcode.com/problems/sliding-window-maximum/

Solution 2: Sliding Window Maximum(LeetCode P239)

Method 2: Monotonic queue

Initialization:

Define a double-ended queue “deque”, a result list “res”, and the length of the array “n”.

Sliding Window:

The left boundary ranges from **i = 1 - k** to **i = n - k** ;
The right boundary ranges from **j = 0** to **j = n - 1**.

(a) If i > 0” and the front element of the `deque` is equal to the deleted element `nums[i - 1]`, then dequeue the front element.
(b) Remove all elements in the `deque` that are less than nums[j]
(c) Add nums[j] to the end of the `deque`.
(d) If a window is formed (i.e., `i >= 0`): Add the maximum value of the window (i.e., the front element of the `deque`) to the result list `res`.

Return value: Return the result list `res`.

https://leetcode.com/problems/sliding-window-maximum/

Thanks for your Attention!

Q&A