Theory: Deque

🕐 4 minutes   2 / 2 problems solved
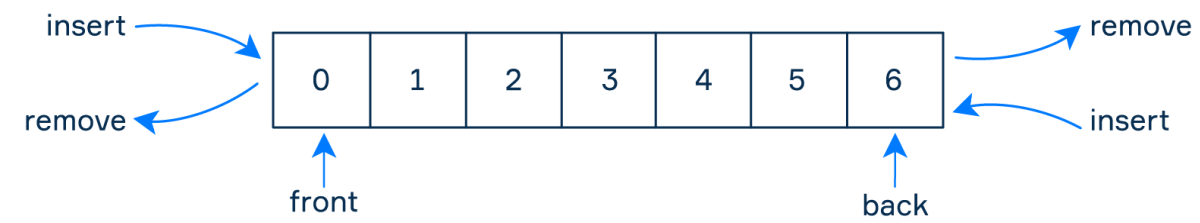
[Start practicing]

§1. Essentials

Deque is a generalization of the queue that allows elements to be inserted and removed from both ends of it. The term deque comes from "double-ended queue". It combines access rules provided by queue (FIFO) and stack (LIFO) together.

The following picture demonstrates a deque with seven elements:
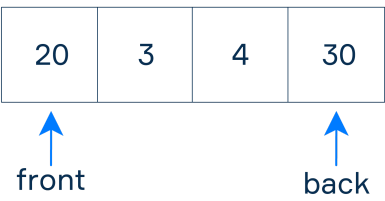

§2. Operations

There are four basic operations on it:

- insert an element to the front;
- insert an element to the back;
- remove an element from the front;
- remove an element from the back.

Suppose we execute the following operations on the deque presented above:

```
1    remove an element from the front
2    remove an element from the front
3    remove an element from the front
4    insert 20 to the front
5    remove an element from the back
6    remove an element from the back
7    insert 30 to the back
```

The result is shown below:

A deque often has some additional operations that allow us to see the first and the last elements without removing them (examine the front and examine the back) as well as obtaining the number of elements in it. Usually, deques do not support indexing, but some implementations may provide it.

§3. Application

Developers use deque less often than regular queue. The most common examples of deque are:

- **undo-redo** operations in software like graphic editors or IDE;
- **steal task scheduling algorithms:** a processor takes the first task from its deque and performs it; when one of the processors completes execution of its own tasks, it steals the last task from the deque of another processor and executes it.

Yet, of course, there are still many other cases where deque can help you.

📄 Report a typo

200 users liked this theory. **3** didn't like it. **What about you?**

Current topic:

✓ Deque ···

Topic depends on:

✓ Stack ···

✓ Queue ···

Topic is required for:

Queue and Stack ···

✓ Queue in Python ···
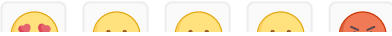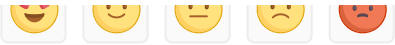
Start practicing

Comments (3)   Hints (0)   Useful links (0)   Show discussion

Comments (3)   Hints (0)   Useful links (0)   Show discussion