Deque

They say it takes both sides to build a bridge. So far, we have considered the two sides separately, namely **First-In, First-Out (FIFO)** and **Last-In, First-Out (LIFO)** principles. What if we combine them? We end up with a new data structure called **deque**, with full access to both the first and the last elements.

Deque is a generalization of the queue that allows us to insert and remove elements from both ends of it. Deques combine access rules provided by queues (FIFO) and stacks (LIFO). The term deque is an abbreviation for "double-ended queue". However, some sources use the terms **dequeue** or **head-tail linked list** instead.

It is not difficult to guess that there are two main sub-types of deques that can be useful in specific situations:

1. **Input-restricted deque** — insertion is restricted at a single end, meanwhile, deletion can be performed from both sides.
2. **Output-restricted deque** — insertion can be done from both ends, but deletion is restricted at a single end.

insert_front(d, a) — inserts the element at front of queue.

insert_back(d, a) — inserts the element at the back of queue.

remove_front(d) — deletes the first element of the deque

remove_back(d) — deletes the last element of the deque

Usually, deques do not support indexing, but some implementations may provide it.

All the operations mentioned above can obviously be performed in $O(1)$.

Programmers widely use deques in many areas including computer software to:

- implement other data structures, like stacks, queues, etc. You might come across the topic of this on your journey around our platform someday.
- store web browser's history: newly visited URLs are inserted at the top, and older links are removed from the list.
- undo-redo operations in software, like graphic editors, IDEs, etc. They need access to only the first and the last element, therefore, deque is ideal for this task.
- steal task scheduling algorithms: a processor takes the first task from its deque and performs it;