1756. Design Most Recently Used Queue Premium

Solved ●

Medium Topics Companies Hint

Design a queue-like data structure that moves the most recently used element to the end of the queue.

Implement the `MRUQueue` class:

- `MRUQueue(int n)` constructs the `MRUQueue` with `n` elements: `[1,2,3,...,n]`.
- `int fetch(int k)` moves the k^{th} element **(1-indexed)** to the end of the queue and returns it.

Example 1:

> **Input:**
> ["MRUQueue", "fetch", "fetch", "fetch", "fetch"]
> [[8], [3], [5], [2], [8]]
> **Output:**
> [null, 3, 6, 2, 2]
>
> **Explanation:**
> MRUQueue mRUQueue = new MRUQueue(8); // Initializes the queue to [1,2,3,4,5,6,7,8].
> mRUQueue.fetch(3); // Moves the 3rd element (3) to the end of the queue to become [1,2,4,5,6,7,8,3] and returns it.
> mRUQueue.fetch(5); // Moves the 5th element (6) to the end of the queue to become [1,2,4,5,7,8,3,6] and returns it.
> mRUQueue.fetch(2); // Moves the 2nd element (2) to the end of the queue to become [1,4,5,7,8,3,6,2] and returns it.
> mRUQueue.fetch(8); // The 8th element (2) is already at the end of the queue so just return it.

Constraints:

- `1 <= n <= 2000`
- `1 <= k <= n`
- At most `2000` calls will be made to `fetch`.

Follow up: Finding an `O(n)` algorithm per `fetch` is a bit easy. Can you find an algorithm with a better complexity for each `fetch` call?

Seen this question in a real interview before? 1/5

Yes No

Accepted 21.5K Submissions 27.9K Acceptance Rate 76.9%

Topics

Companies

Hint 1

Hint 2

Hint 3

Similar Questions

Discussion (7)