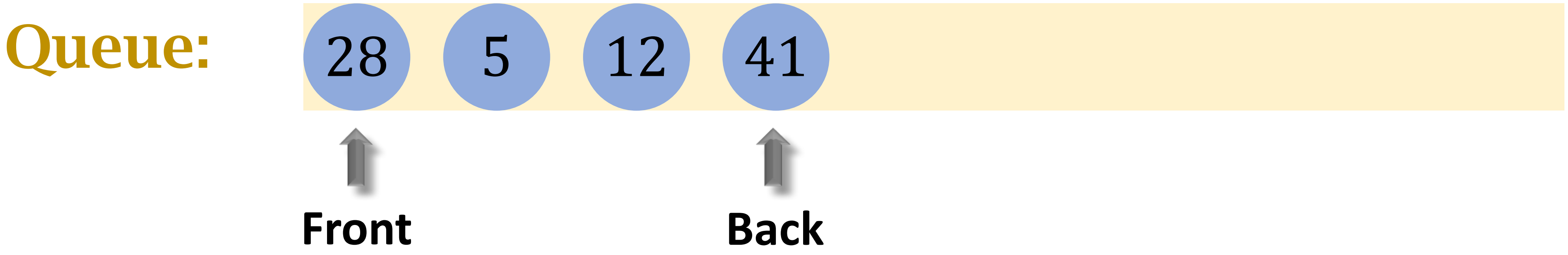
Priority Queues

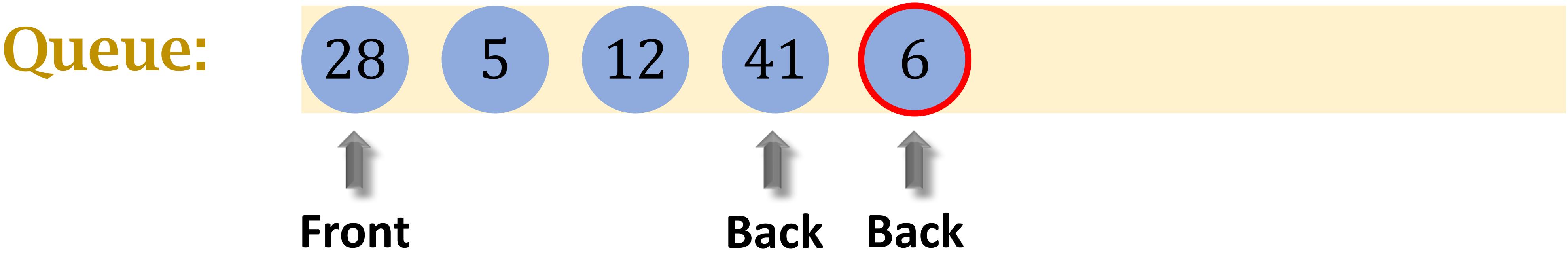
Shusen Wang

Standard Queues

Current State



Enqueue(6)



Queue:

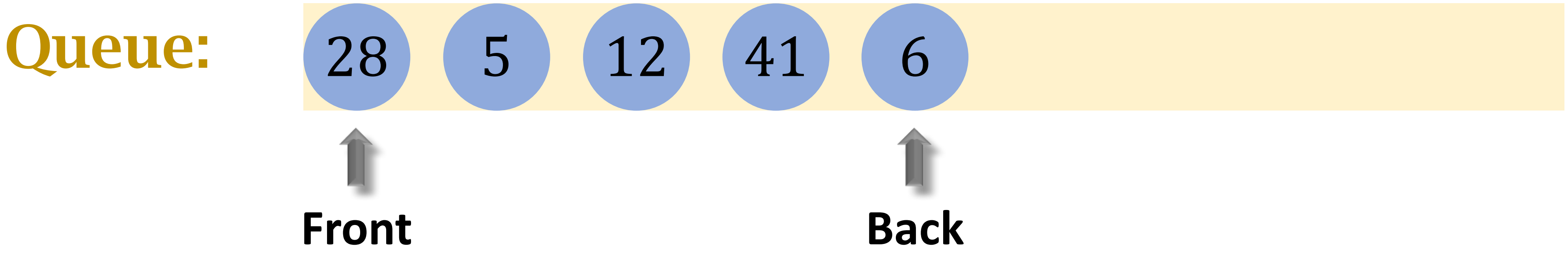
28 5 12 41 6

Front

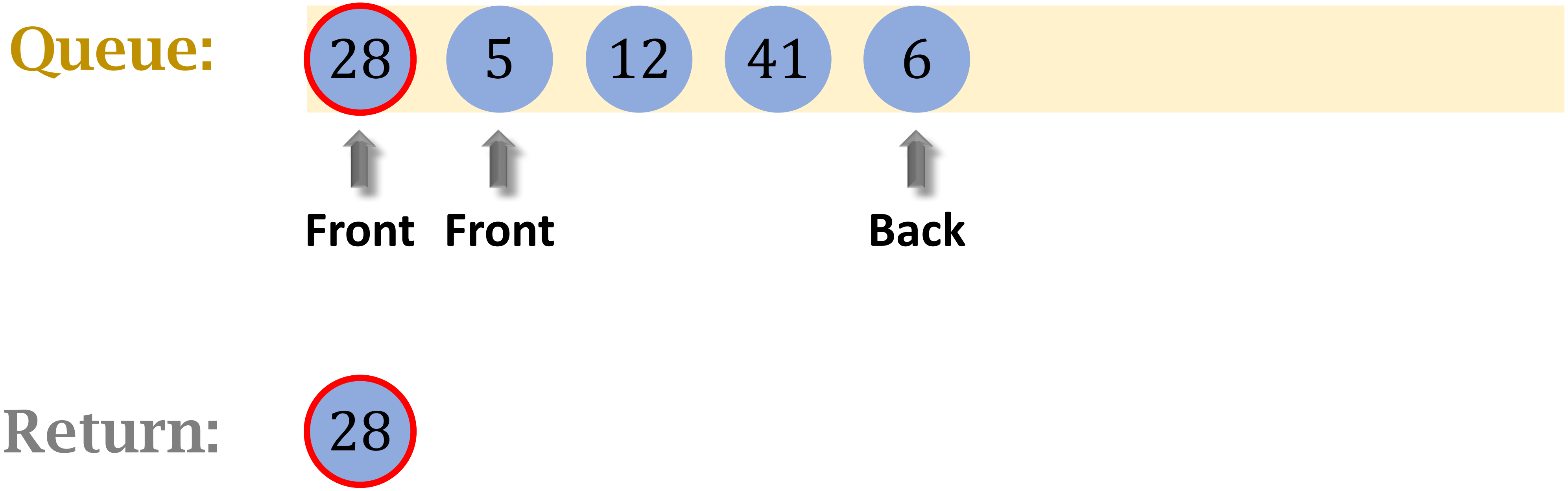
Back

Back

After Enqueue(6)



Dequeue()

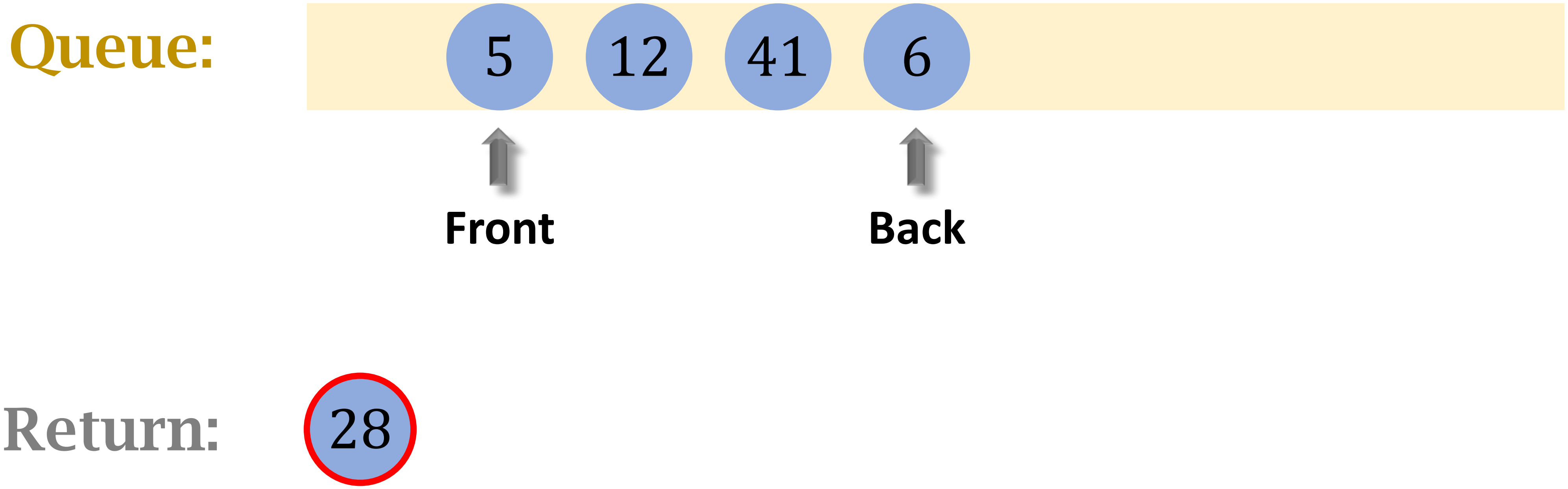


Queue:

Front Front Back

Return:

After Dequeue()



Queue:

5

12

41

6

Front

Back

Return:

28

Priority Queues

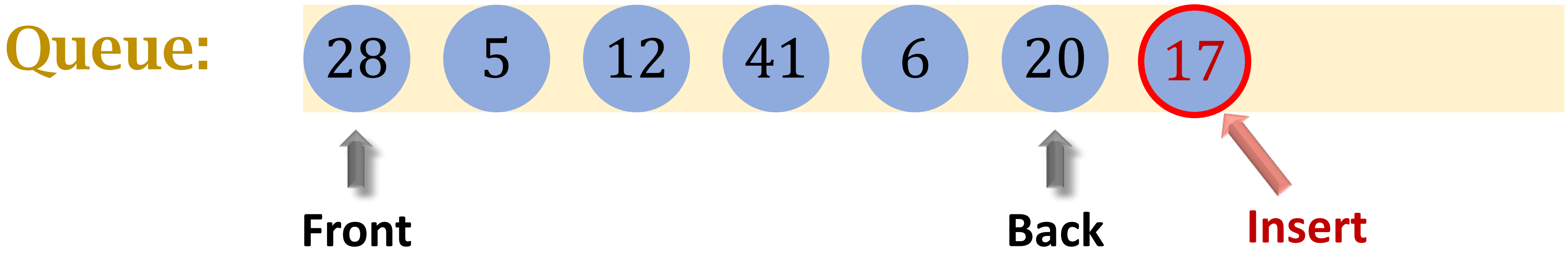
Priority Queues

Priority queues support two operations:

- **`insert(i)`** : insert a new element **`i`** into the queue.
- **`deleteMin()`** : Find, return, and delete the minimum.

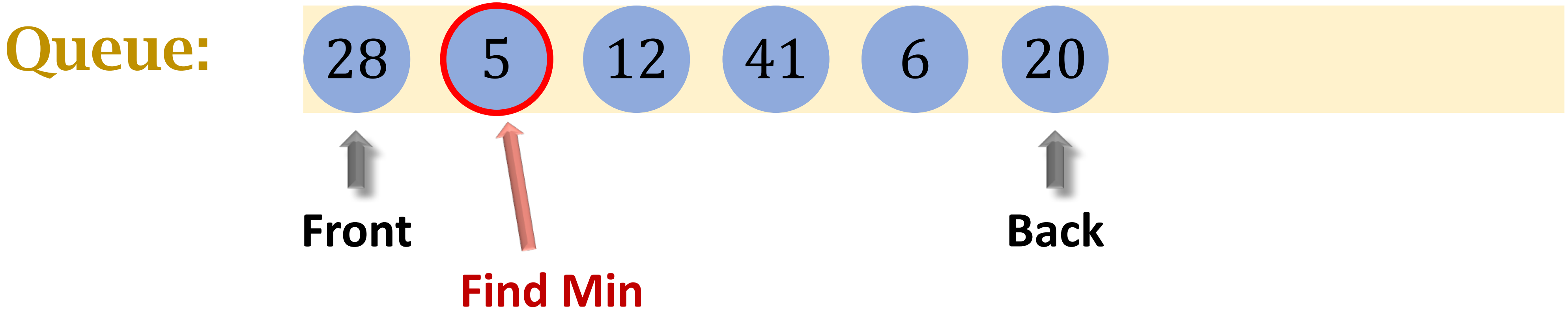
Question: How to implement priority queue?

Naïve Solution 1: Standard Queue



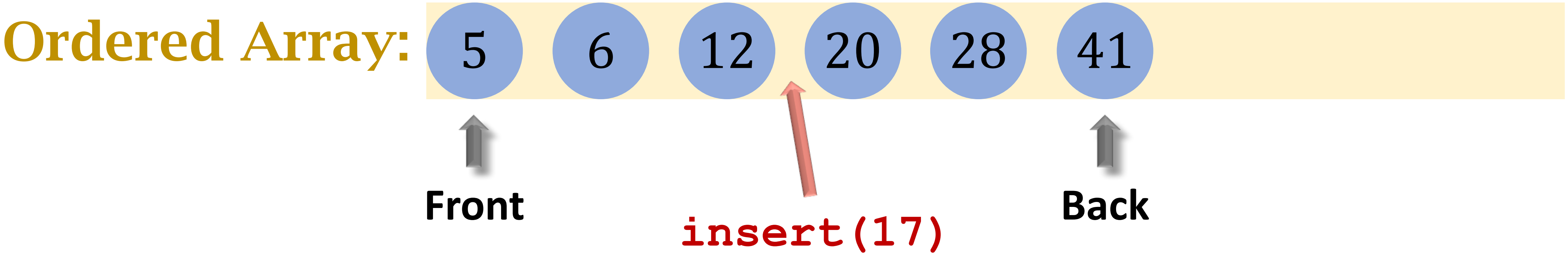
- **insert(i)**: $O(1)$ time.

Naïve Solution 1: Standard Queue



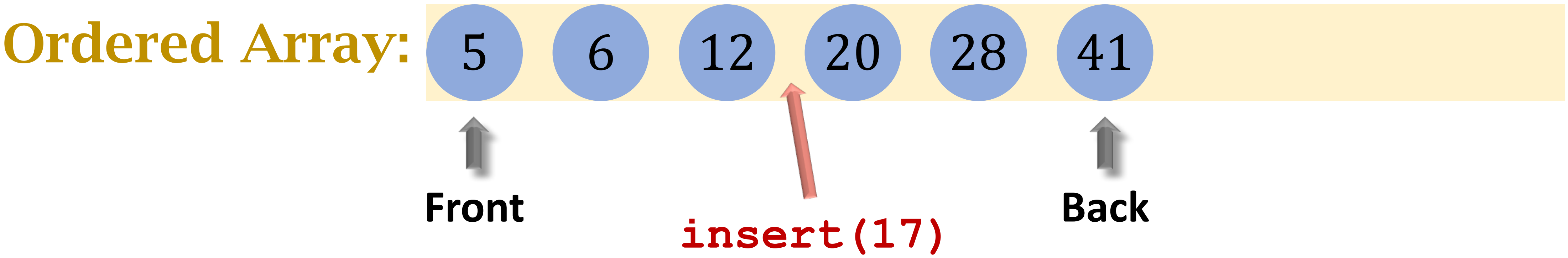
- `insert(i)`: $O(1)$ time.
- `deleteMin()`: $O(n)$ time (due to the search of the minimum.)

Naïve Solution 2: Ordered Array



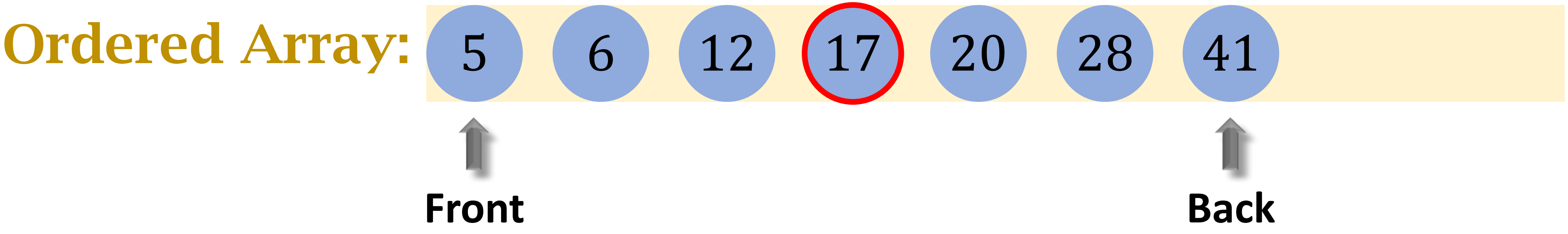
- **`insert(i)`**: $O(n)$ time.
  - $O(\log n)$ time for searching the position.
  - $O(n)$ time for moving the bigger elements backward.

Naïve Solution 2: Ordered Array



- **insert(i)**: $O(n)$ time.
  - $O(\log n)$ time for searching the position.
  - $O(n)$ time for moving the bigger elements backward.

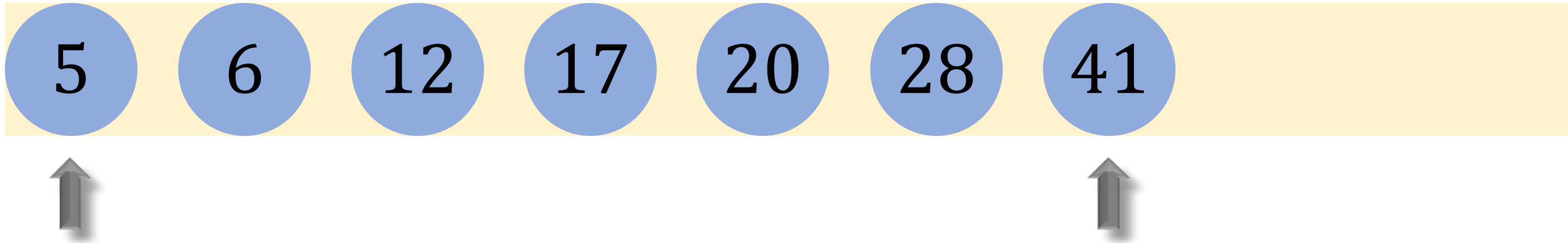
Naïve Solution 2: Ordered Array



- **`insert(i)`**: $O(n)$ time.
  - $O(\log n)$ time for searching the position.
  - $O(n)$ time for moving the bigger elements backward.

Naïve Solution 2: Ordered Array

Ordered Array:



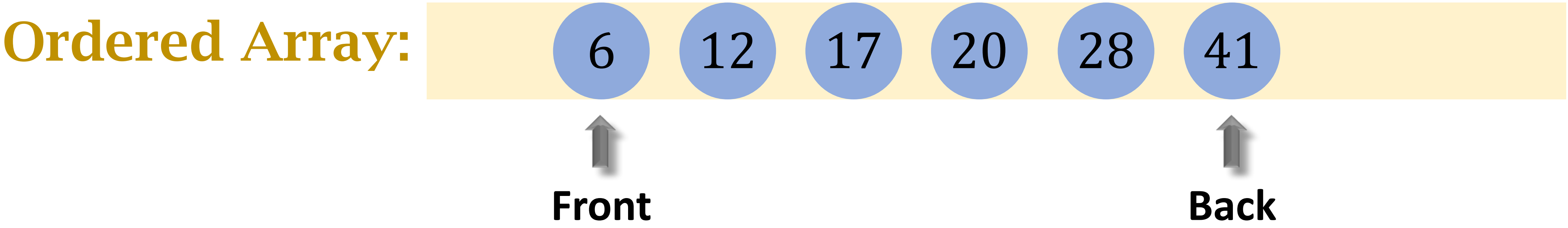
Front

Back

The min is at the front.

- `insert(i)`: $O(n)$ time.
- `deleteMin()`: $O(1)$ time.

Naïve Solution 2: Ordered Array



Ordered Array:

- `insert(i)`: $O(n)$ time.
- `deleteMin()`: $O(1)$ time.

A Better Solution: Binary Heap

Binary heaps support two operations:

- `insert(i)`: $O(\log n)$ time.
- `deleteMin()`: $O(\log n)$ time.

Thank You!