Engineering Queuing Solutions

Zoran Horvat
CEO at Coding Helmet

@zoranh75 https://codinghelmet.com

Introducing Queues and Stacks

`Queue<T>`

Exposes `Enqueue()` **and** `Dequeue()`

Operates in $O(1)$ **time and space per item**

A.k.a. FIFO (first-in, first-out)

FIFO problems are comparatively rare

Often used as temporary storage

Queues are common in multithreading

Decouple producers from processors

`Stack<T>`

Exposes `Push()` **and** `Pop()`

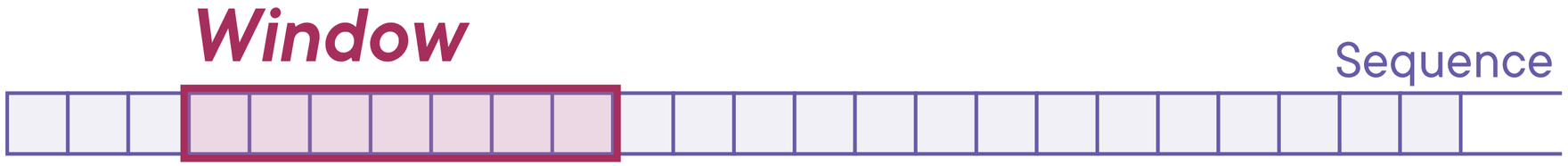
Operates in $O(1)$ **time and space per item**

A.k.a. LIFO (last-in, first-out)

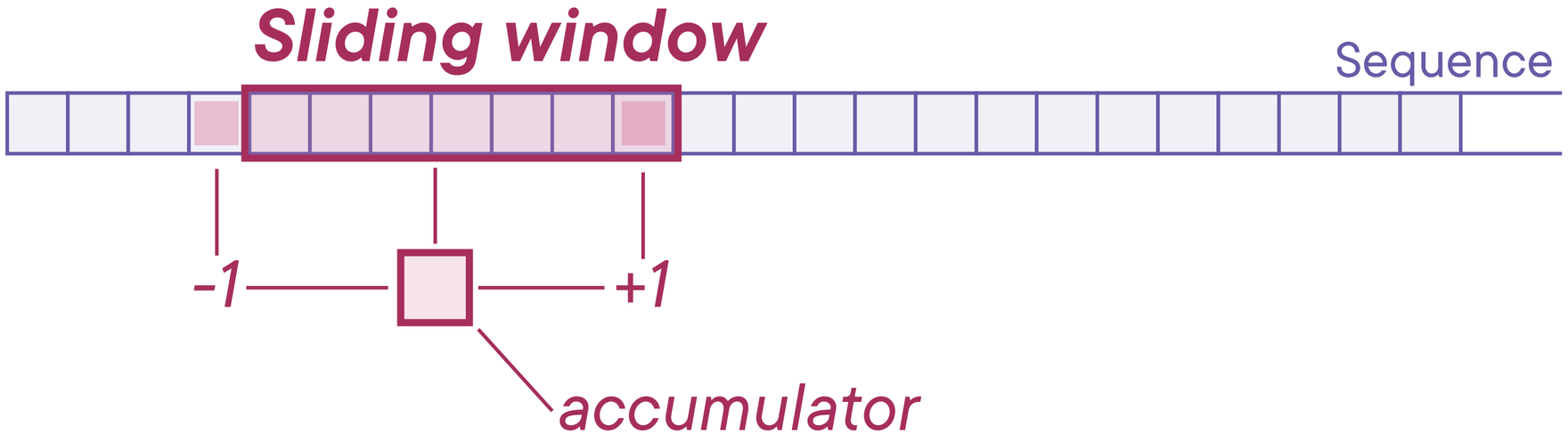
LIFO problems are rare *indeed*

Often used as temporary storage

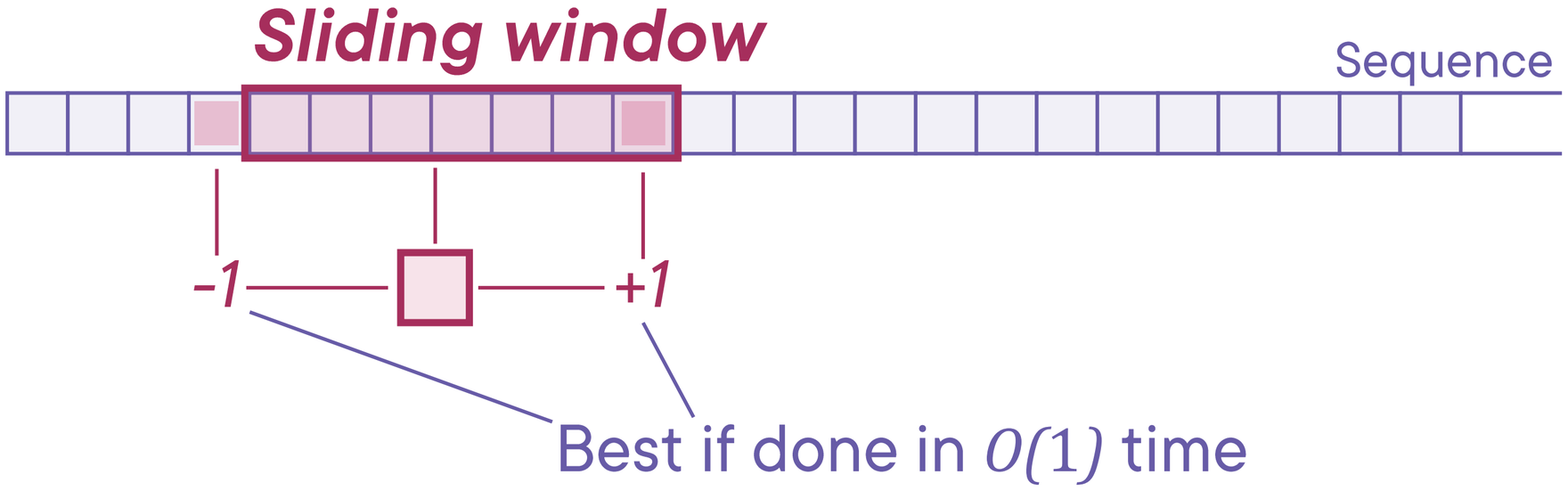
Introducing the Sliding Window Technique



Introducing the Sliding Window Technique



Introducing the Sliding Window Technique



Program.cs | SlidingWindow.cs | WorkItemsTimeWindow.cs ●

Models > Collections > WorkItemsTimeWindow.cs > { } Models.Collections > Models.Collections.WorkItemsTimeWindow > Tail

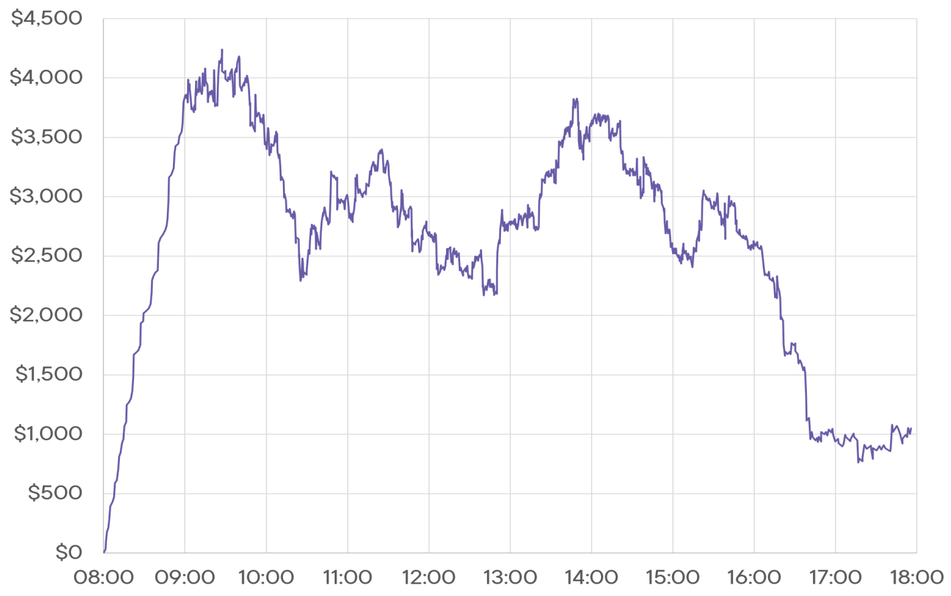
```
1  namespace Models.Collections;
2
3  public class WorkItemsTimeWindow
4  {
5      public WorkItemsTimeWindow(TimeSpan interval)
6
7
8
9
10
11
12     public void Add(WorkItem item)
13     public void SetCurrentTime(DateTime time)
14
15     public int Count =>
16     public MoneyBag TotalCost =>
17     public WorkItem Head =>
18     public WorkItem Tail =>
19 }
```

The public API

- .vscode
- ConsoleDemo
  - ConsoleDemo.csproj
  - Operators.cs
  - Program.cs
- Models
  - Collections
    - DelegatingAccumul...
    - FullySortedList.cs
    - IAccumulator.cs
    - IOrderedList.cs
    - OrderedListProjectio...
    - SlidingWindow.cs
    - WorkItemsTimeWin...
  - Common
  - Currency.cs
  - CurrencyCodeEqualit...
  - Employee.cs
  - FinanceExtensions.cs
  - IWorkersService.cs
  - IWorkItemsService.cs
  - Models.csproj
  - Money.cs
  - MoneyBag.cs
  - PayRate.cs
  - Worker.cs
  - WorkItem.cs
- Models.Tests
  - Data
  - Services
  - Models.Tests.csproj
- ContractorsCo.sln

0 0 ContractorsCo.sln Ln 18, Col 65 Spaces: 4 UTF-8 CRLF C#


Add support staff
Plan downtime
$4,500
$4,000
$3,500
$3,000
$2,500
$2,000
$1,500
$1,000
$500
$0
08:00
09:00
10:00
11:00
12:00
13:00
14:00
15:00
16:00
17:00
18:00



Customer's request:

At any moment,
display the highest cost
of any work item
assigned during the last hour

Analyzing the Request

We have the working sliding window algorithm

- We shall apply it to solve the maximum problem

There is no such accumulator that can quickly report a maximum

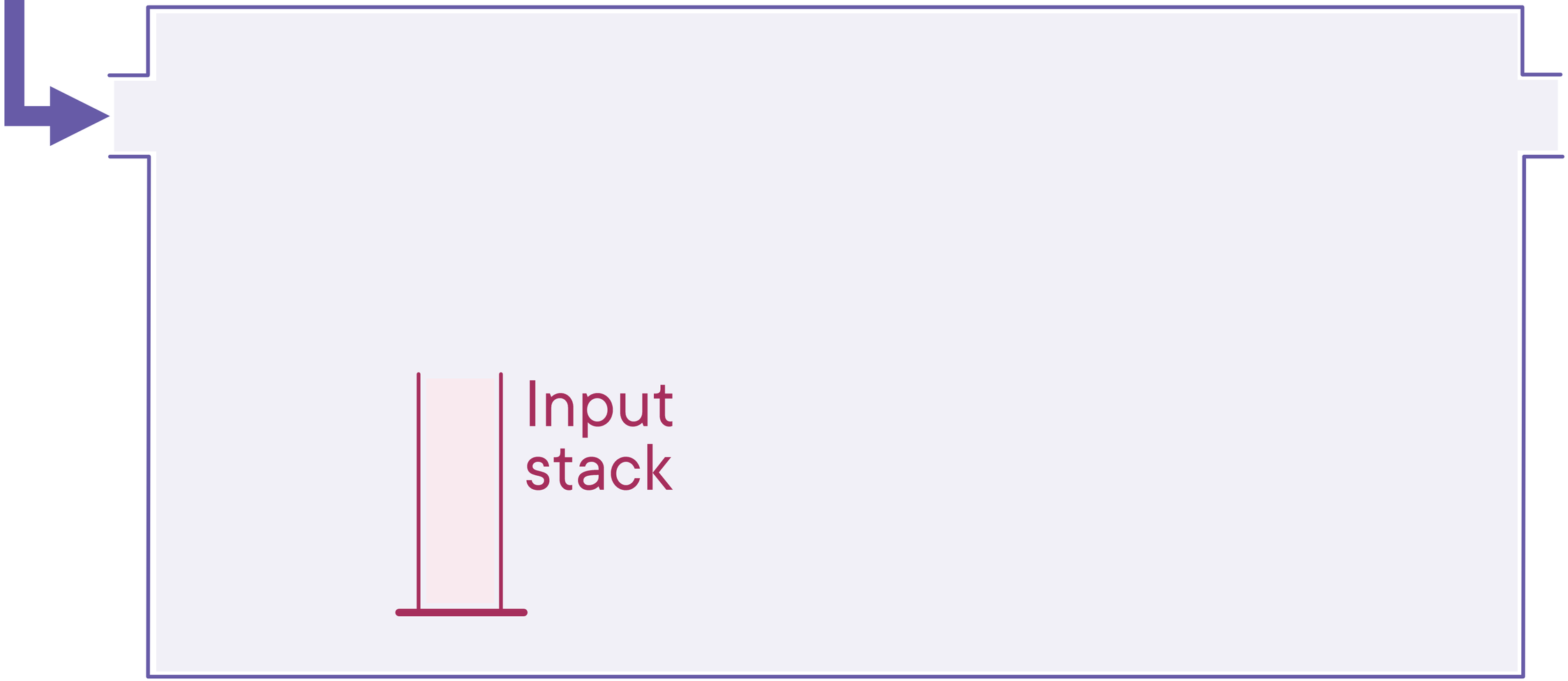
- When current maximum is removed, we do not know its successor

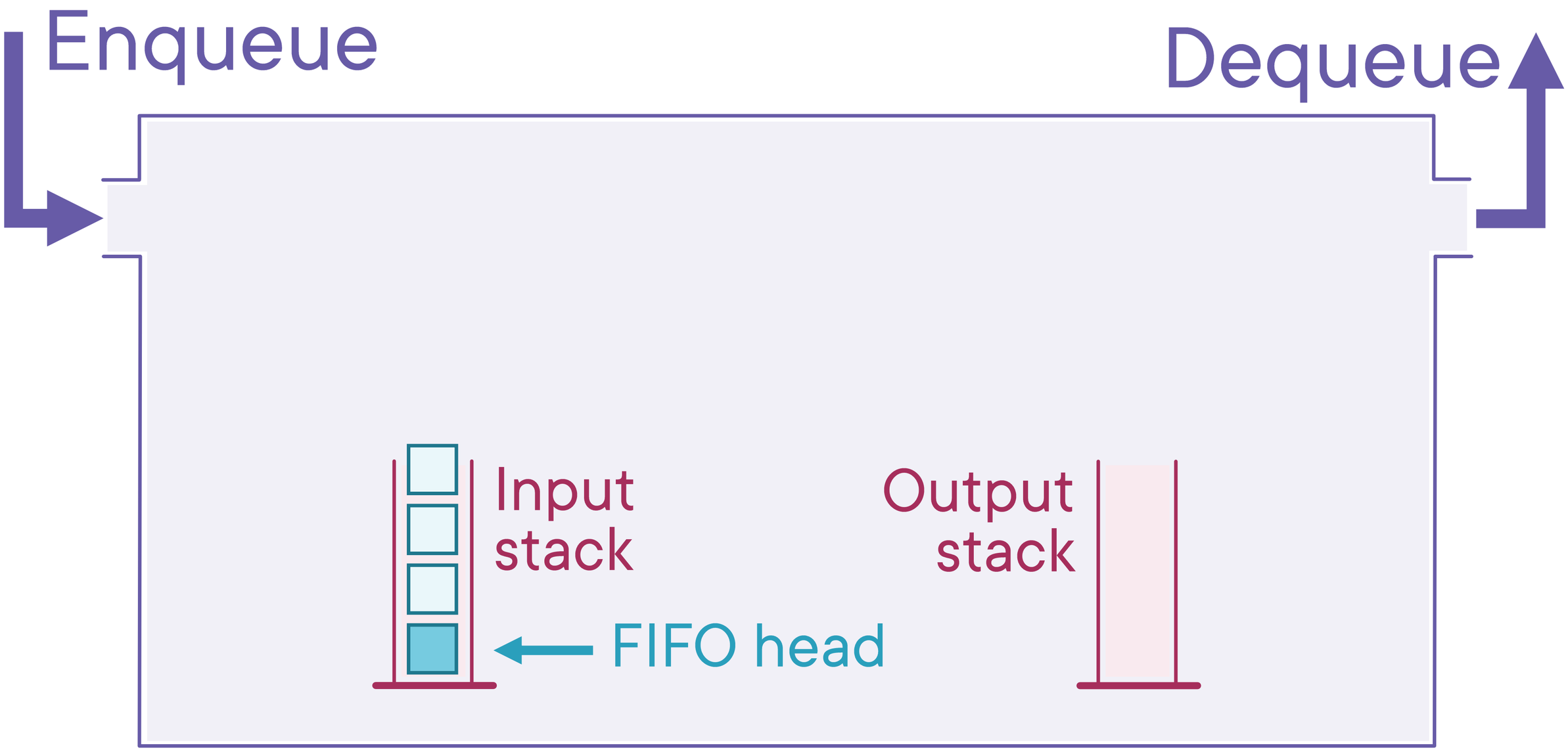
The sliding window algorithm is fine

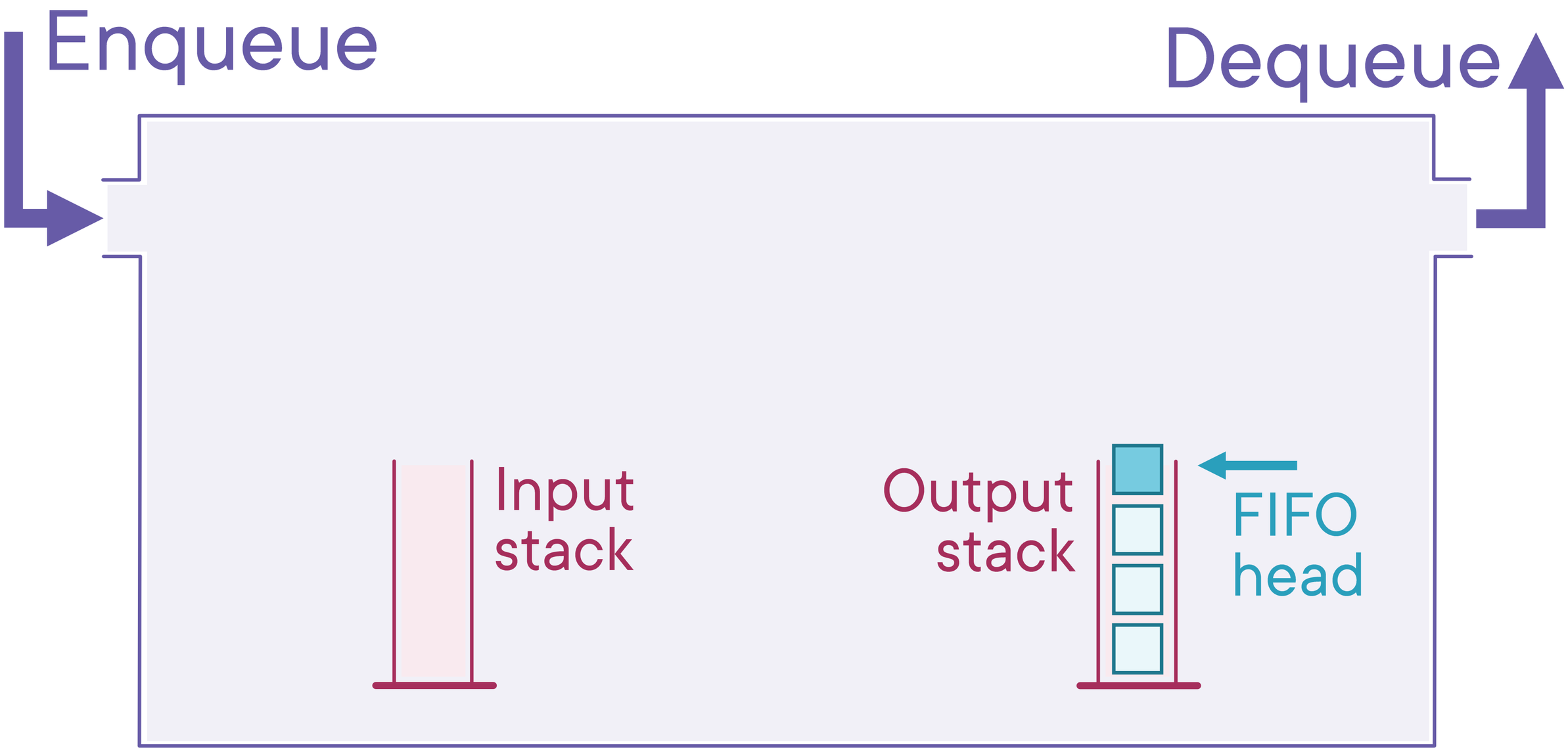
- The accumulators are not

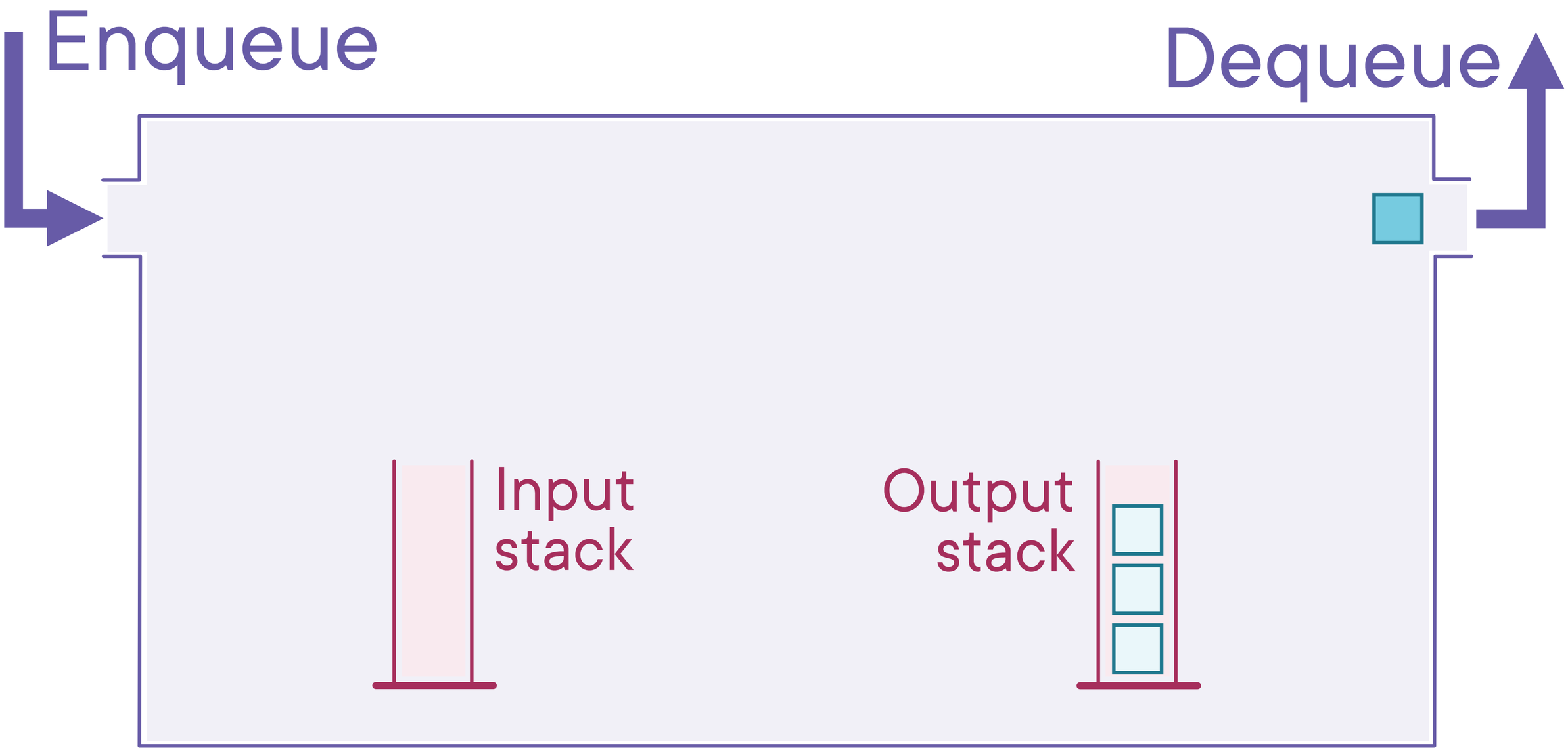
Constructing a queue using two stacks

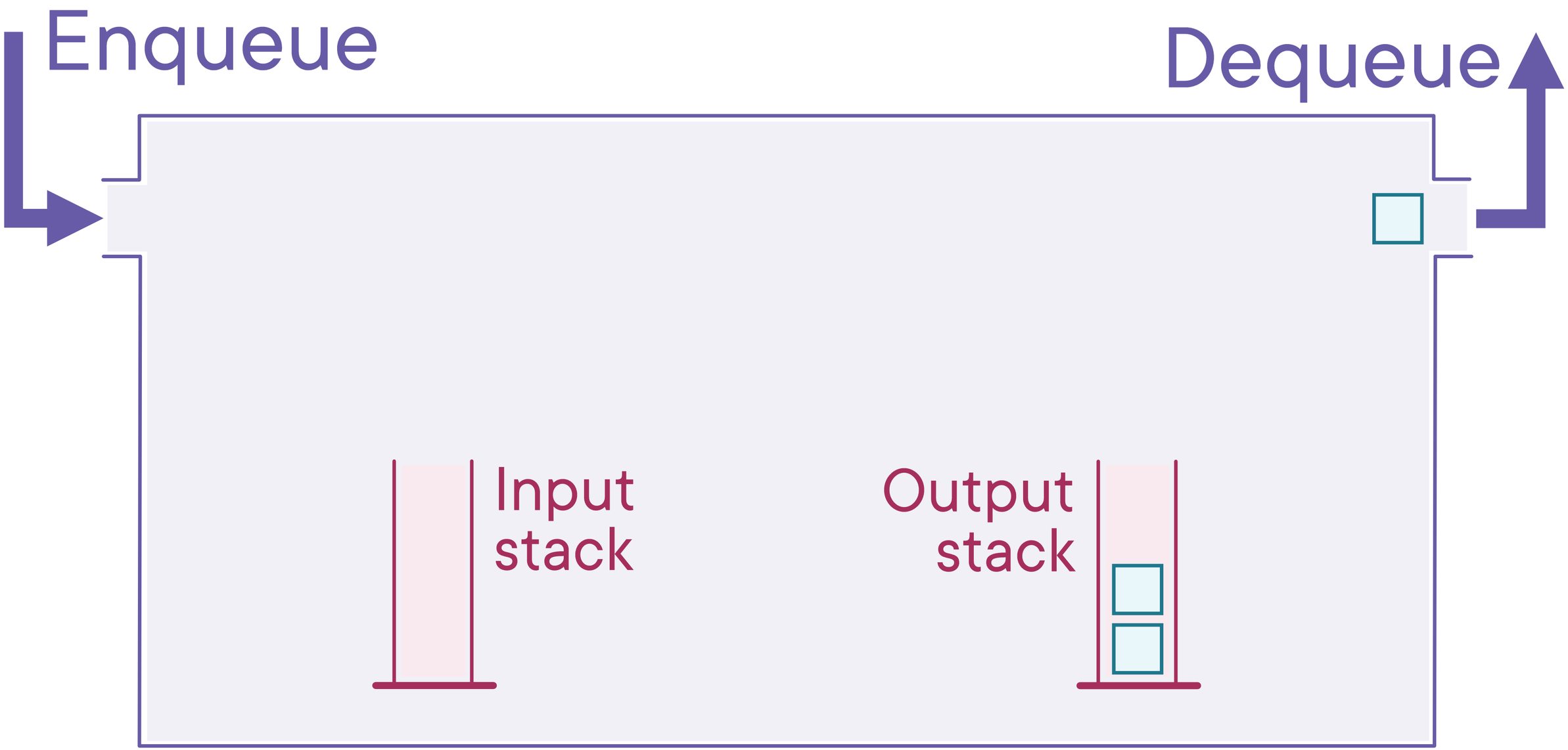
Enqueue

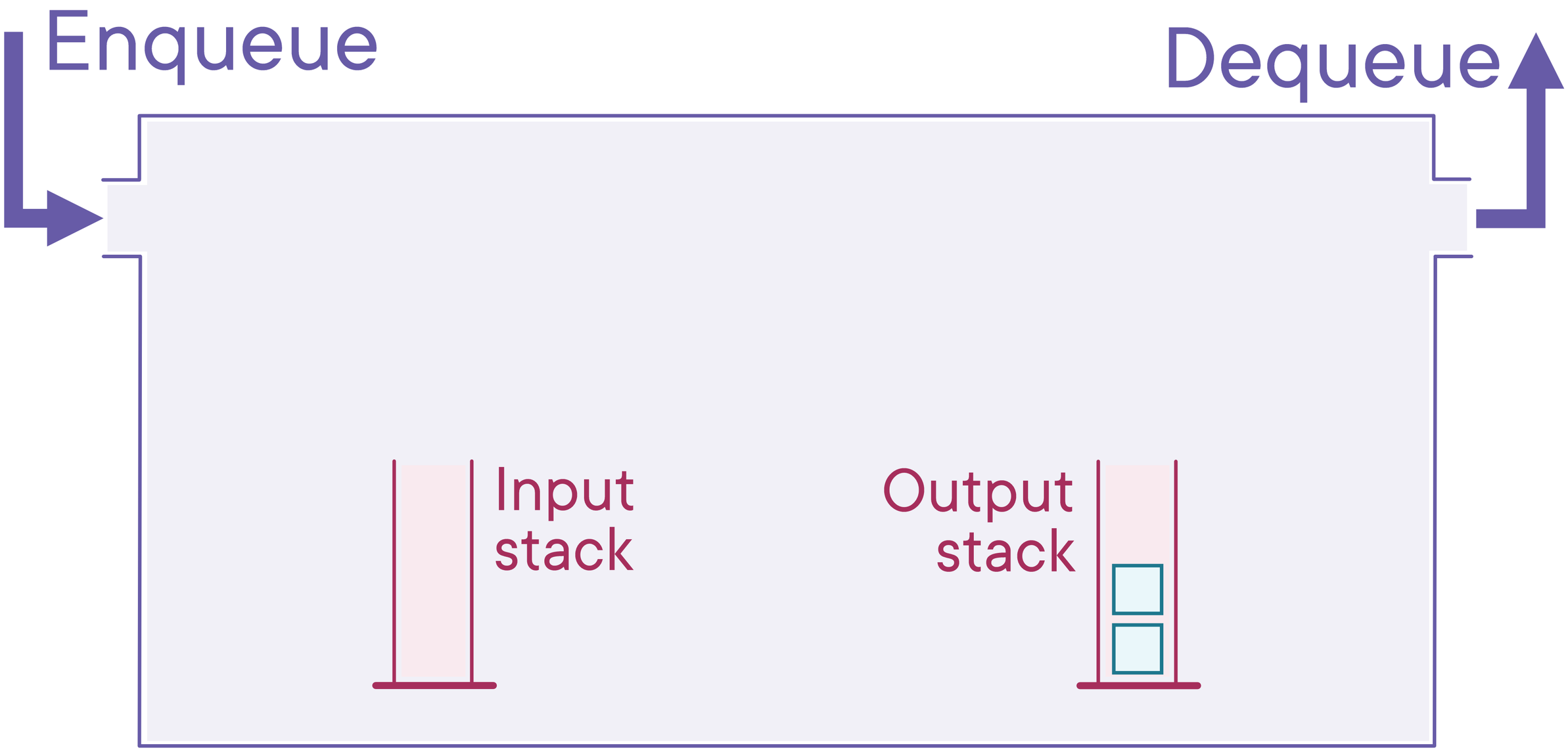


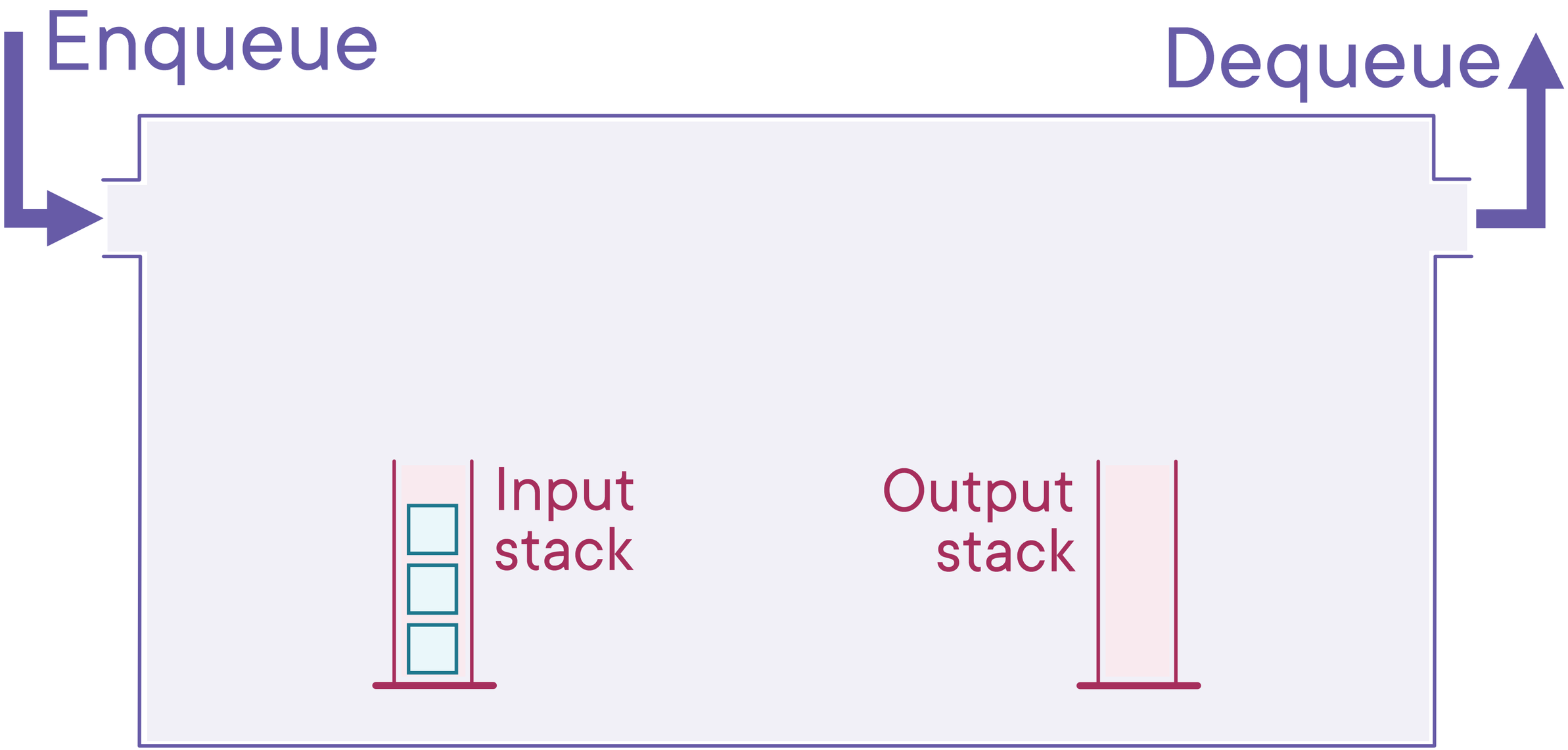

Enqueue
Dequeue
Input stack
FIFO head
Output stack

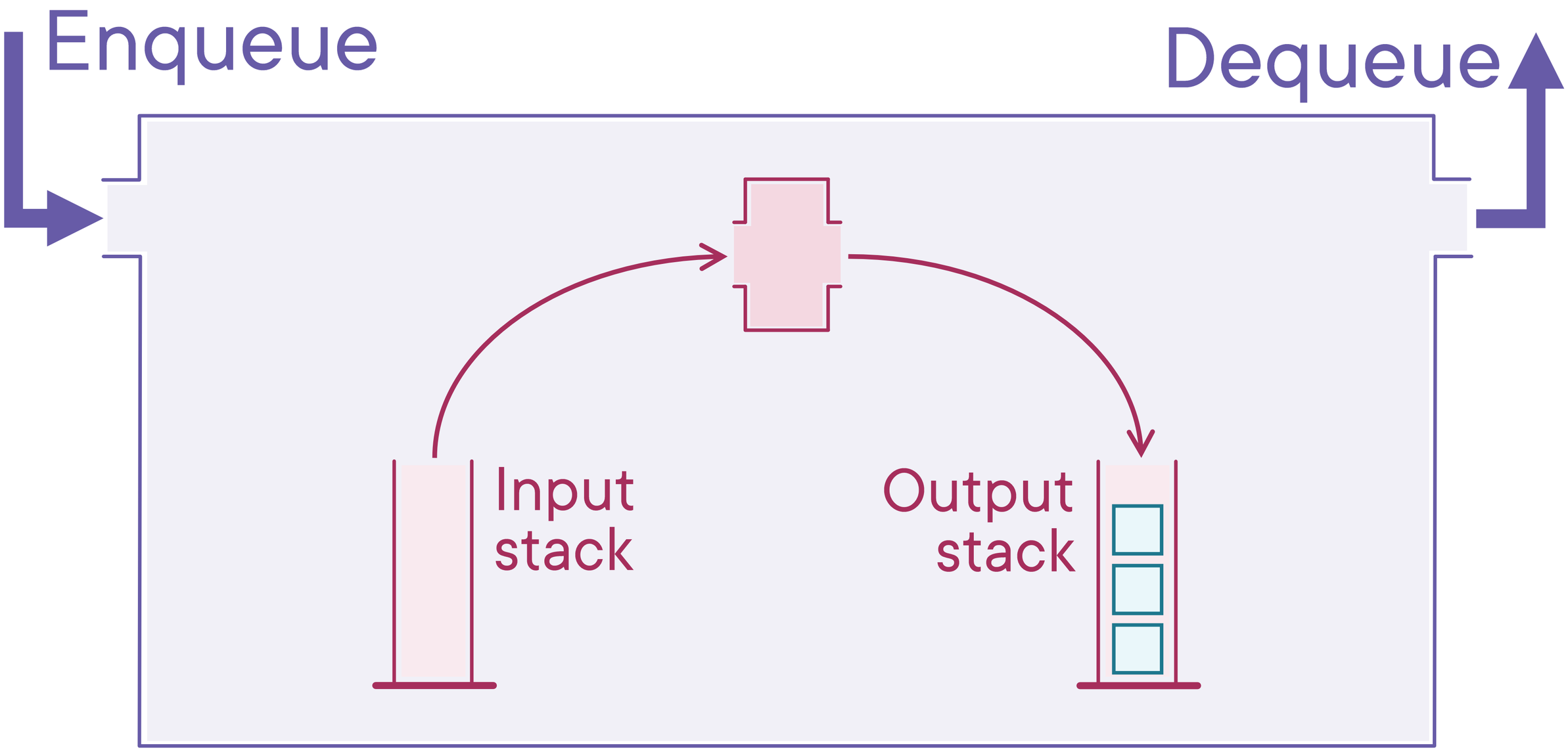

Enqueue
Dequeue
Input stack
Output stack
FIFO head

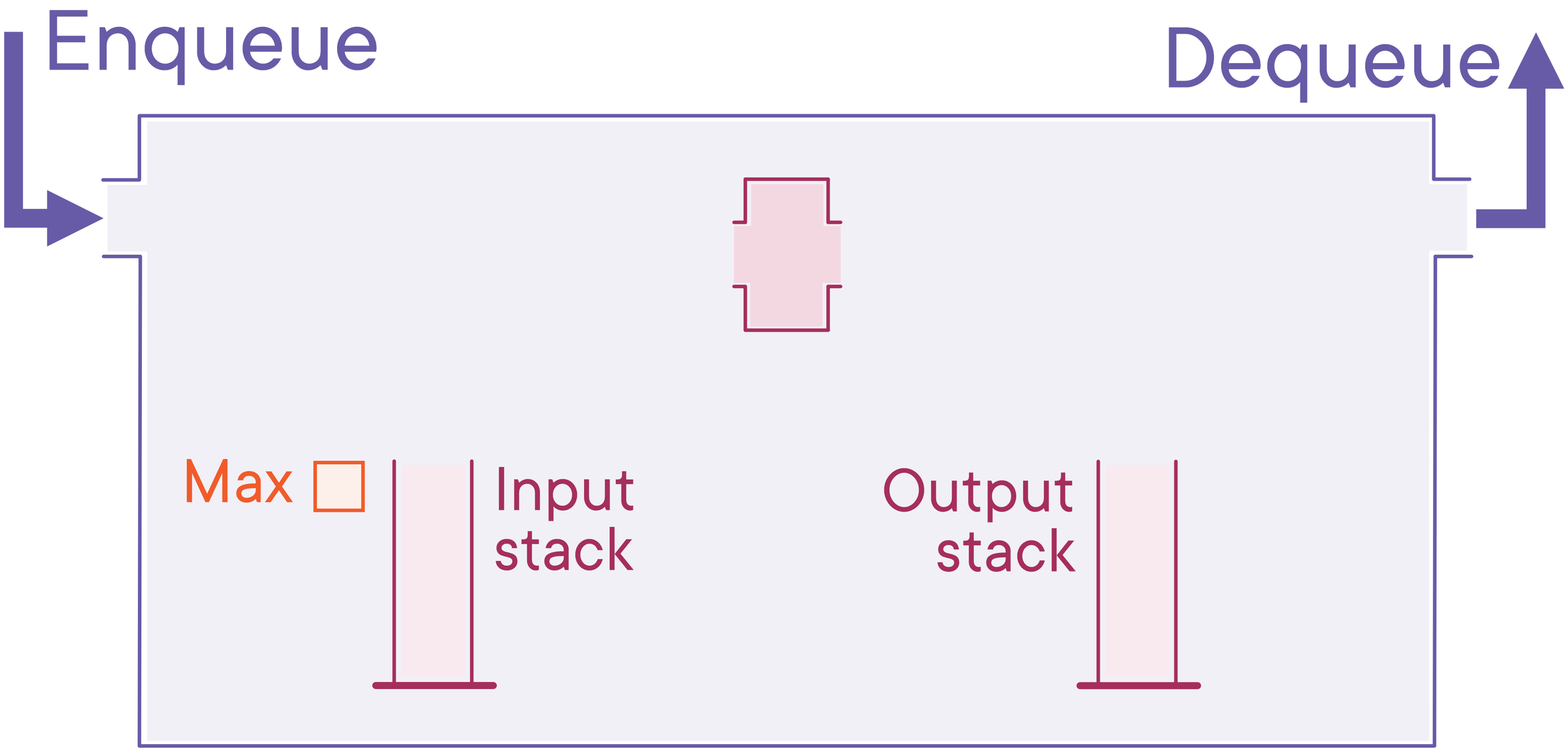

Enqueue
Dequeue
Input stack
Output stack

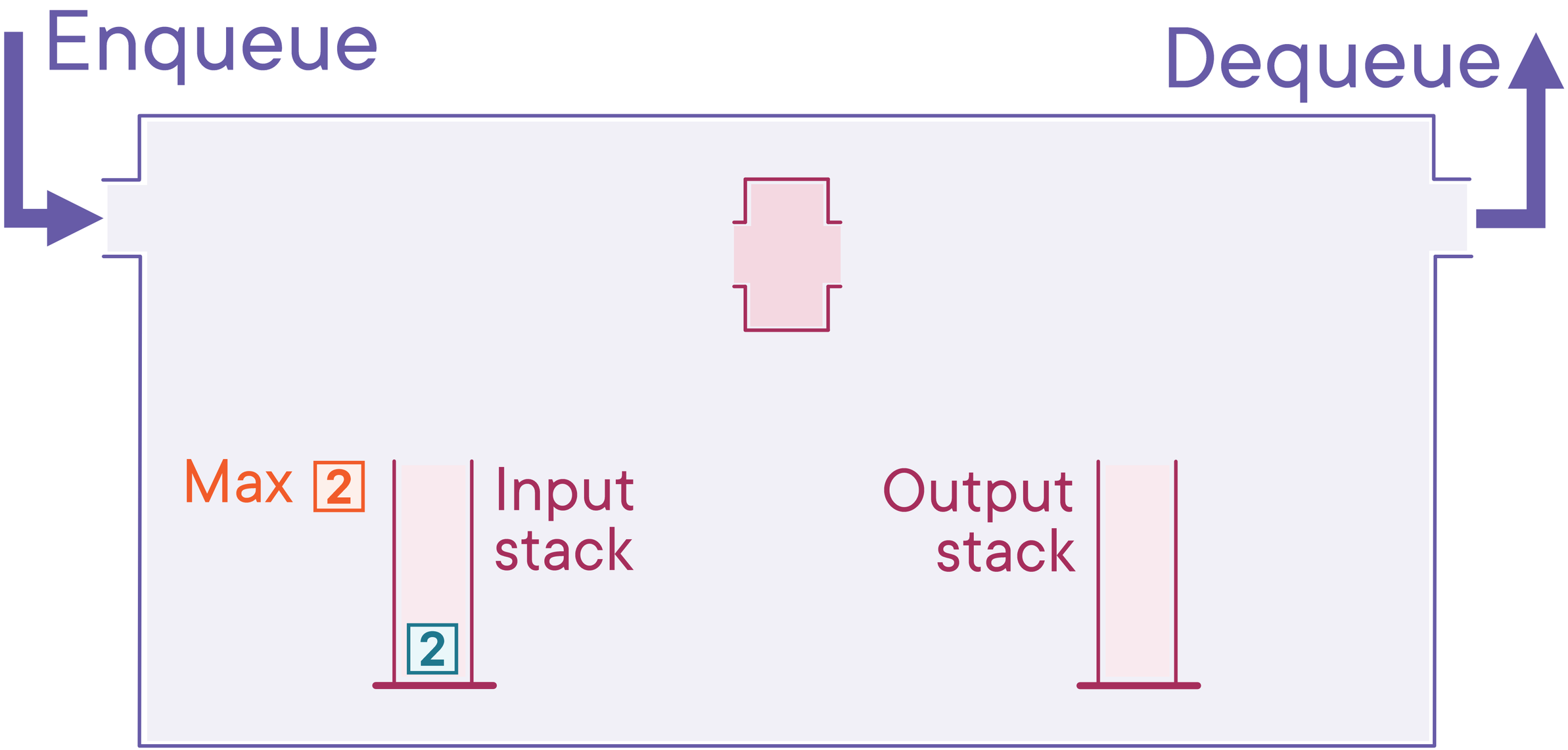

Enqueue
Dequeue
Input stack
Output stack

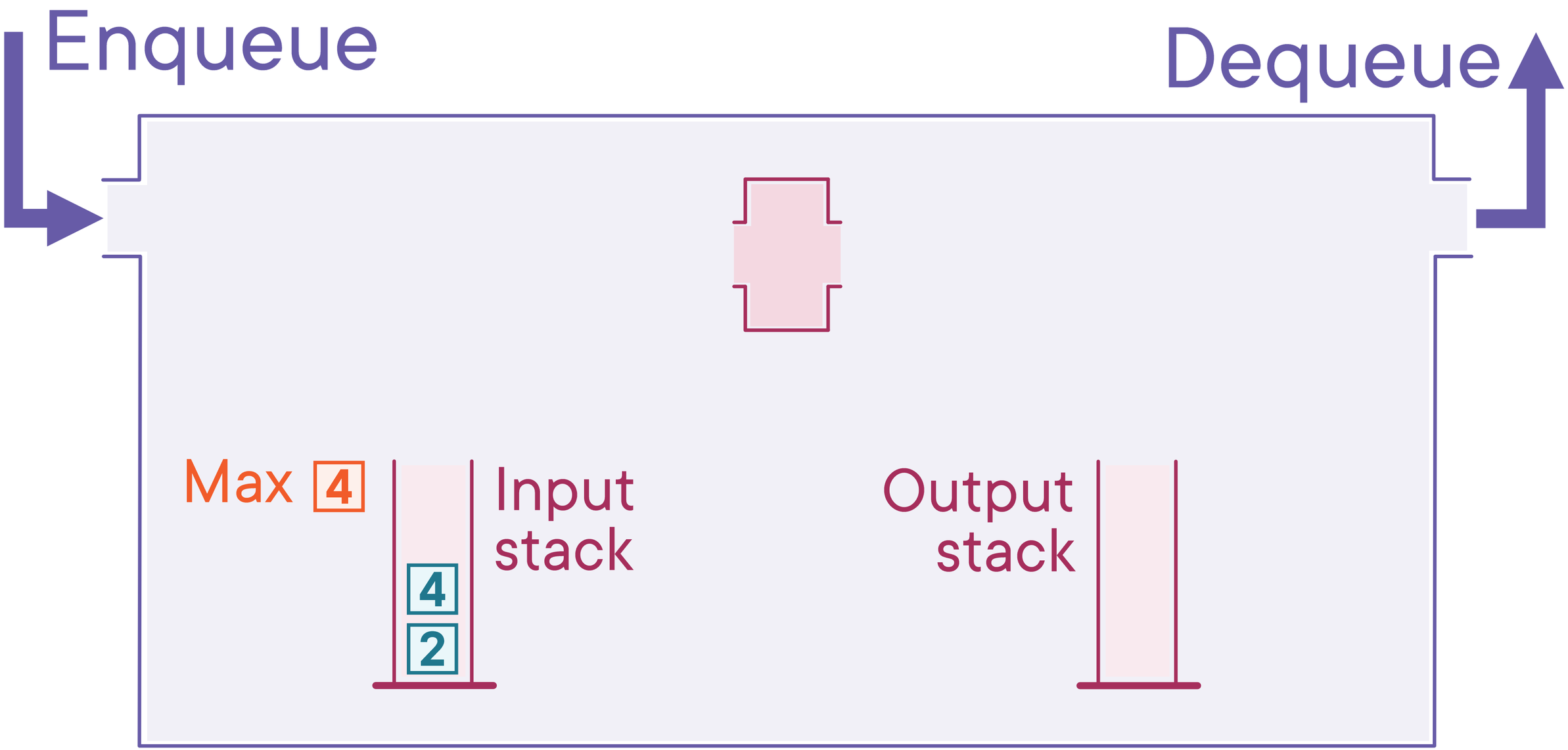

Enqueue
Dequeue
Input stack
Output stack

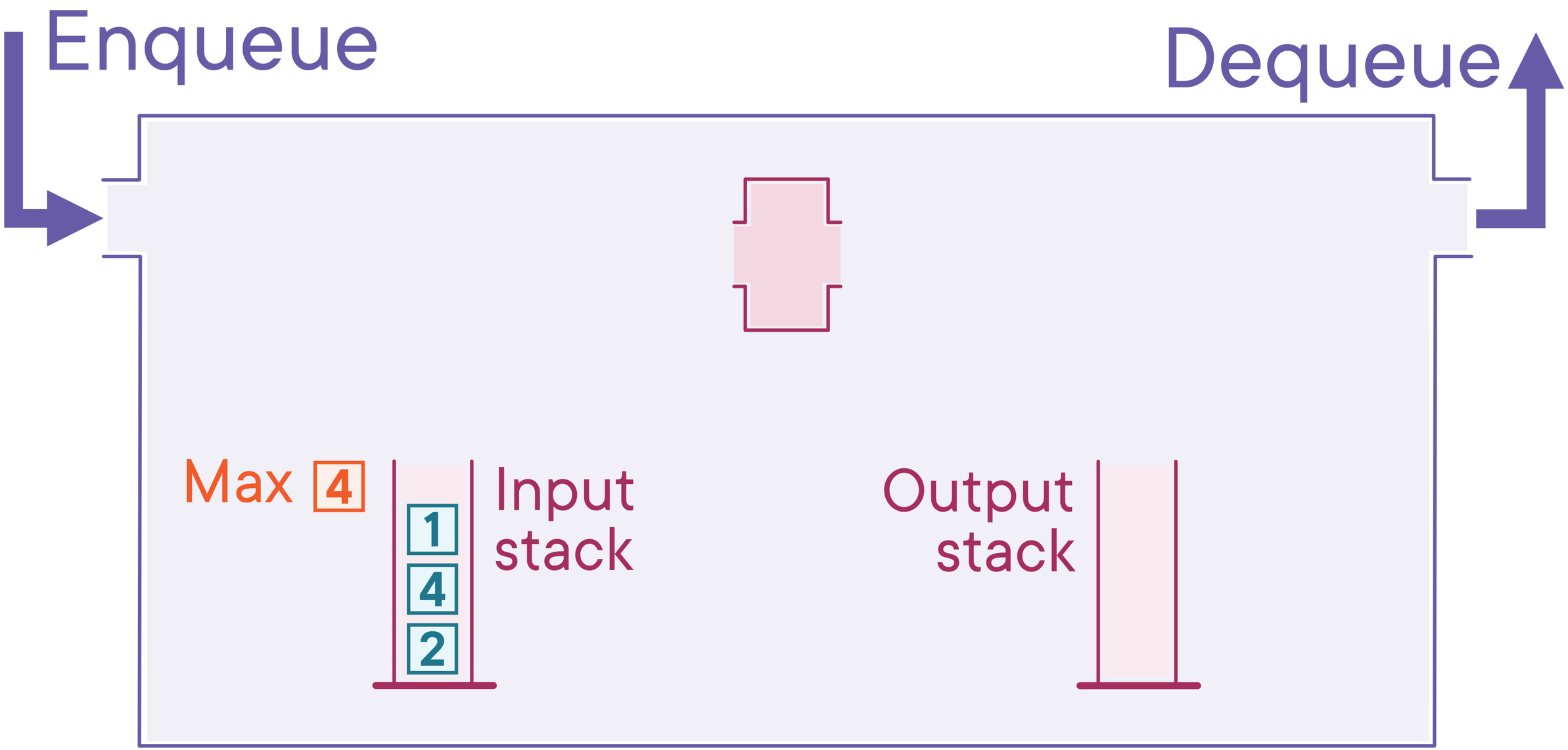

Enqueue
Dequeue
Input stack
Output stack

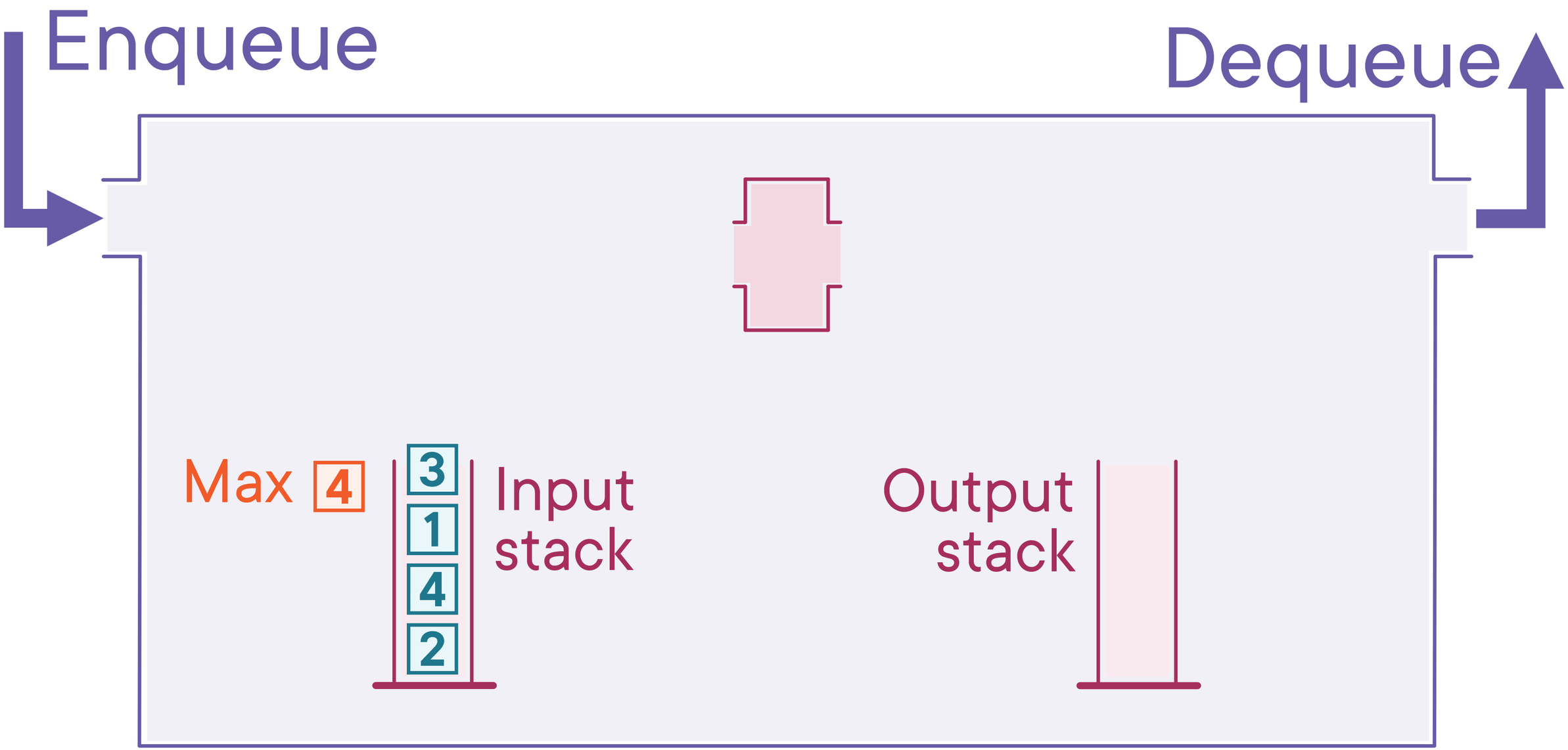

Enqueue
Dequeue
Input stack
Output stack

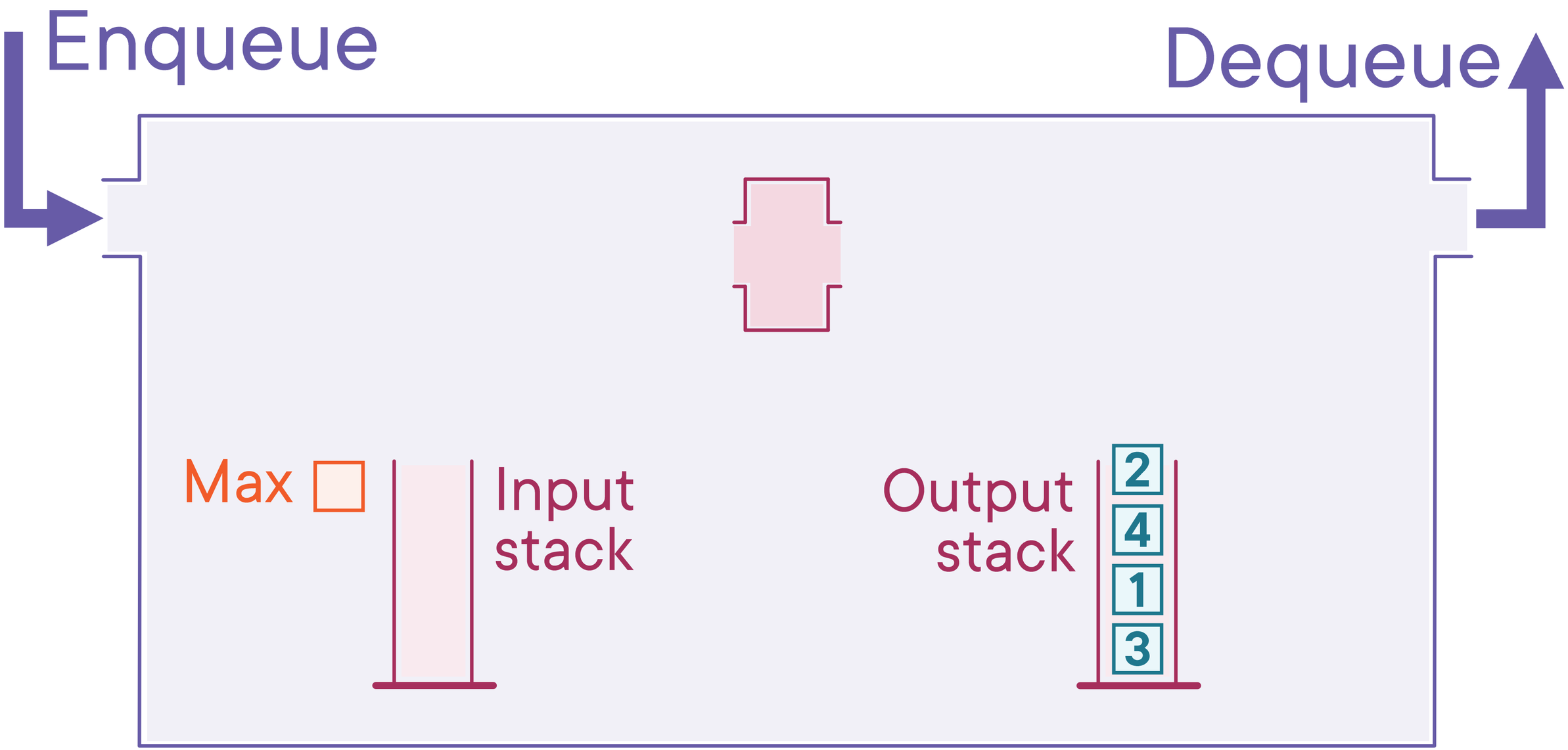

Enqueue
Dequeue
Max
Input stack
Output stack

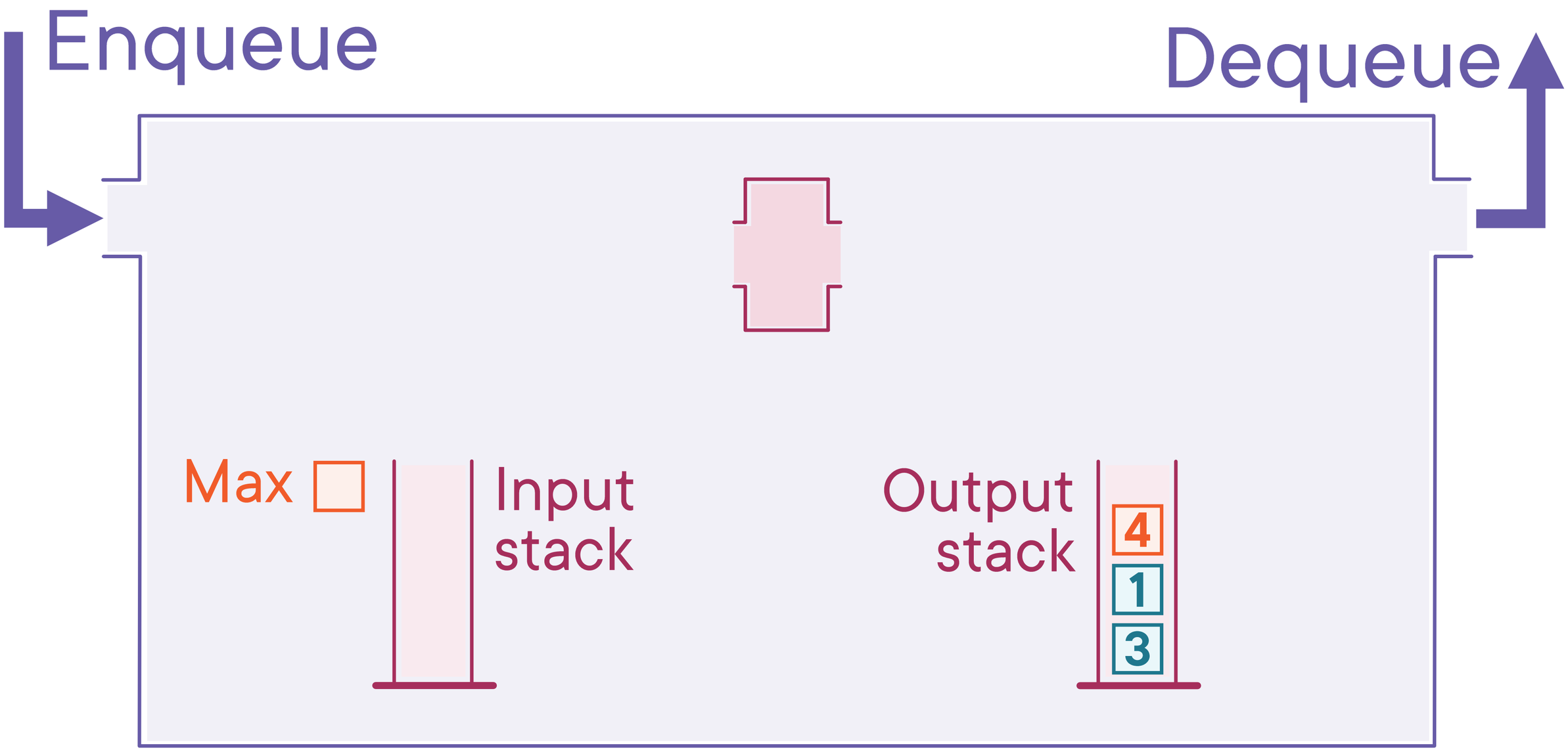

Enqueue
Dequeue
Max 2
2
Input stack
Output stack

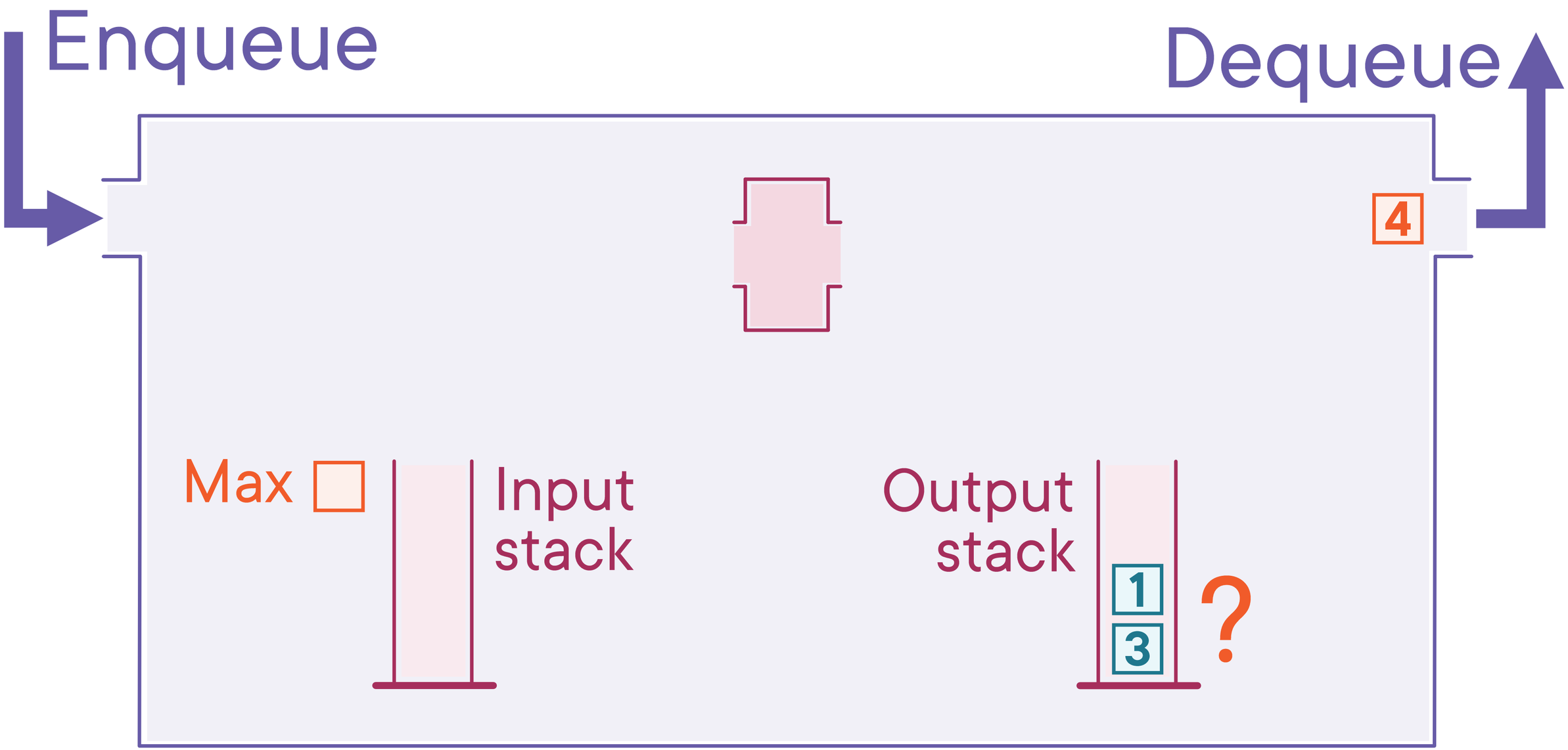

Enqueue
Dequeue
Max 4
4
2
Input stack
Output stack

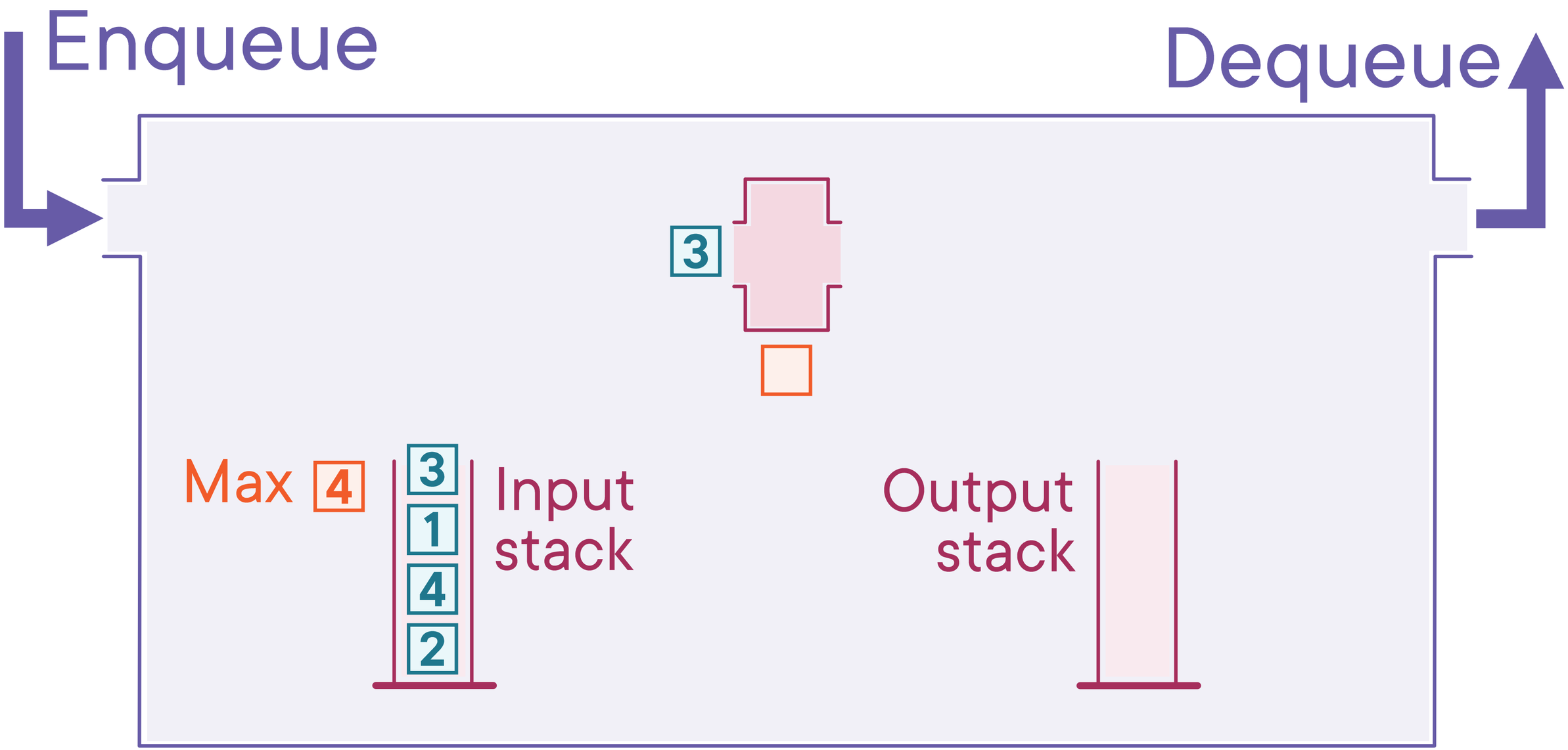

Enqueue
Dequeue
Max 4
1
4
2
Input stack
Output stack

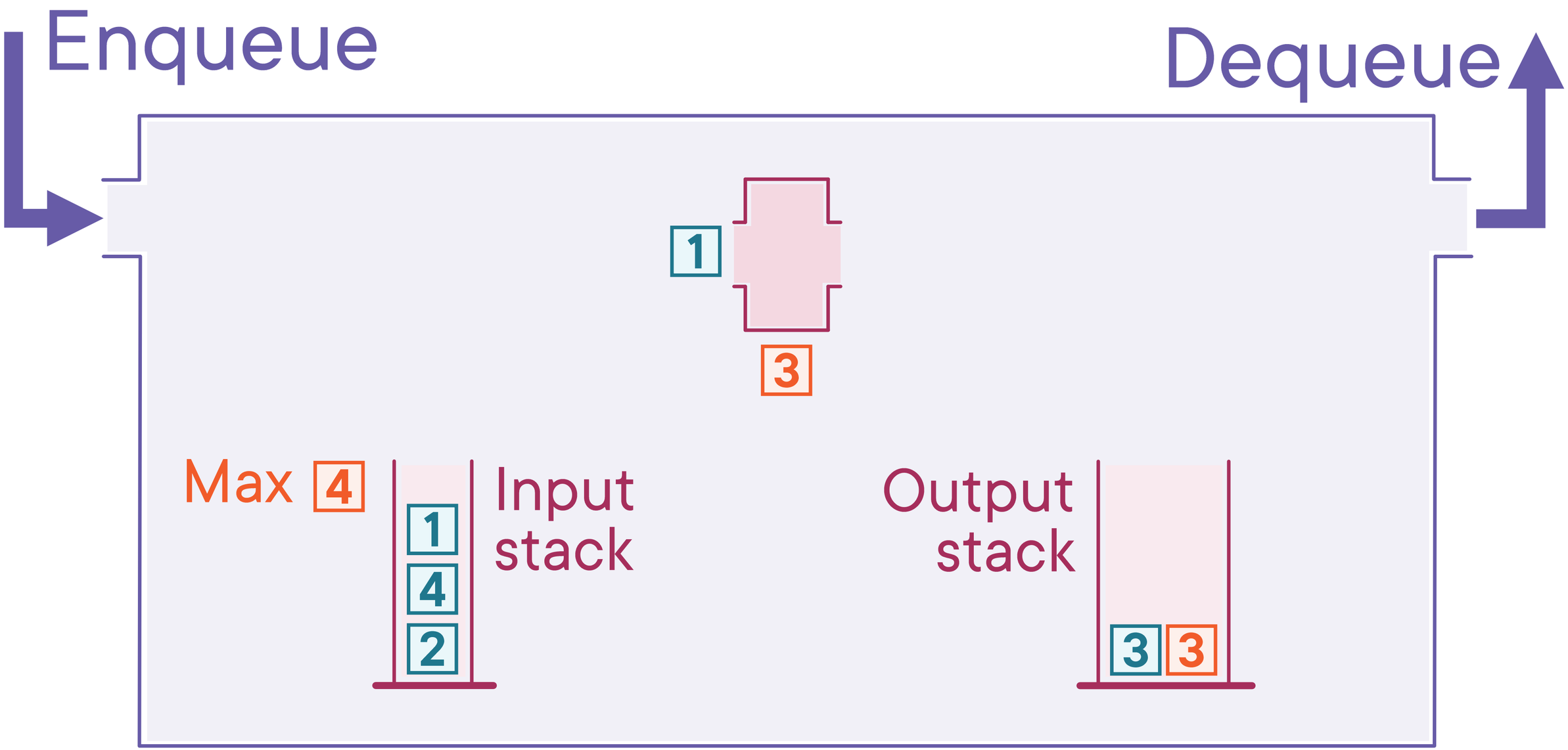

Enqueue
Dequeue
Max 4
3
1
4
2
Input stack
Output stack

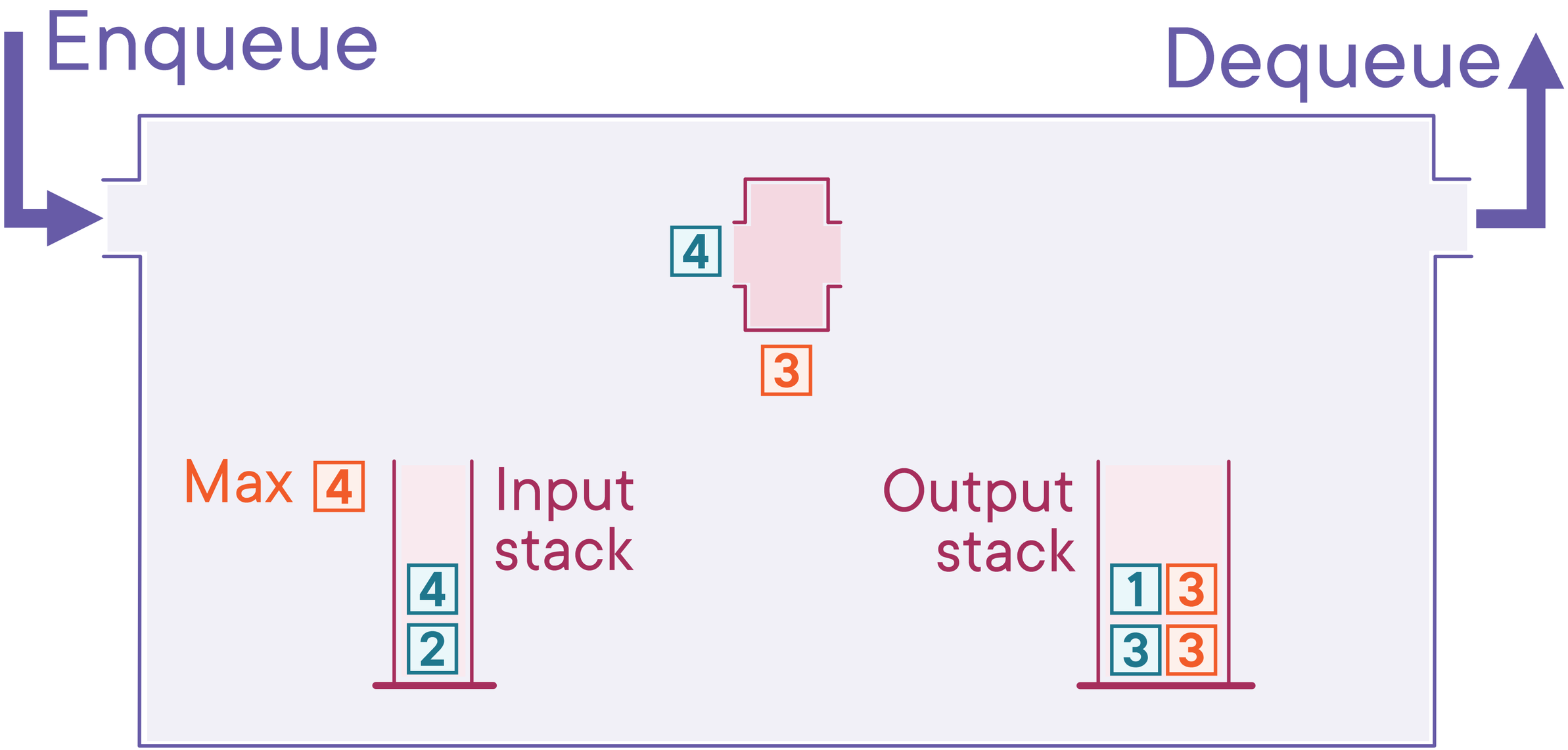

Enqueue
Dequeue
Max
Input stack
Output stack
2
4
1
3

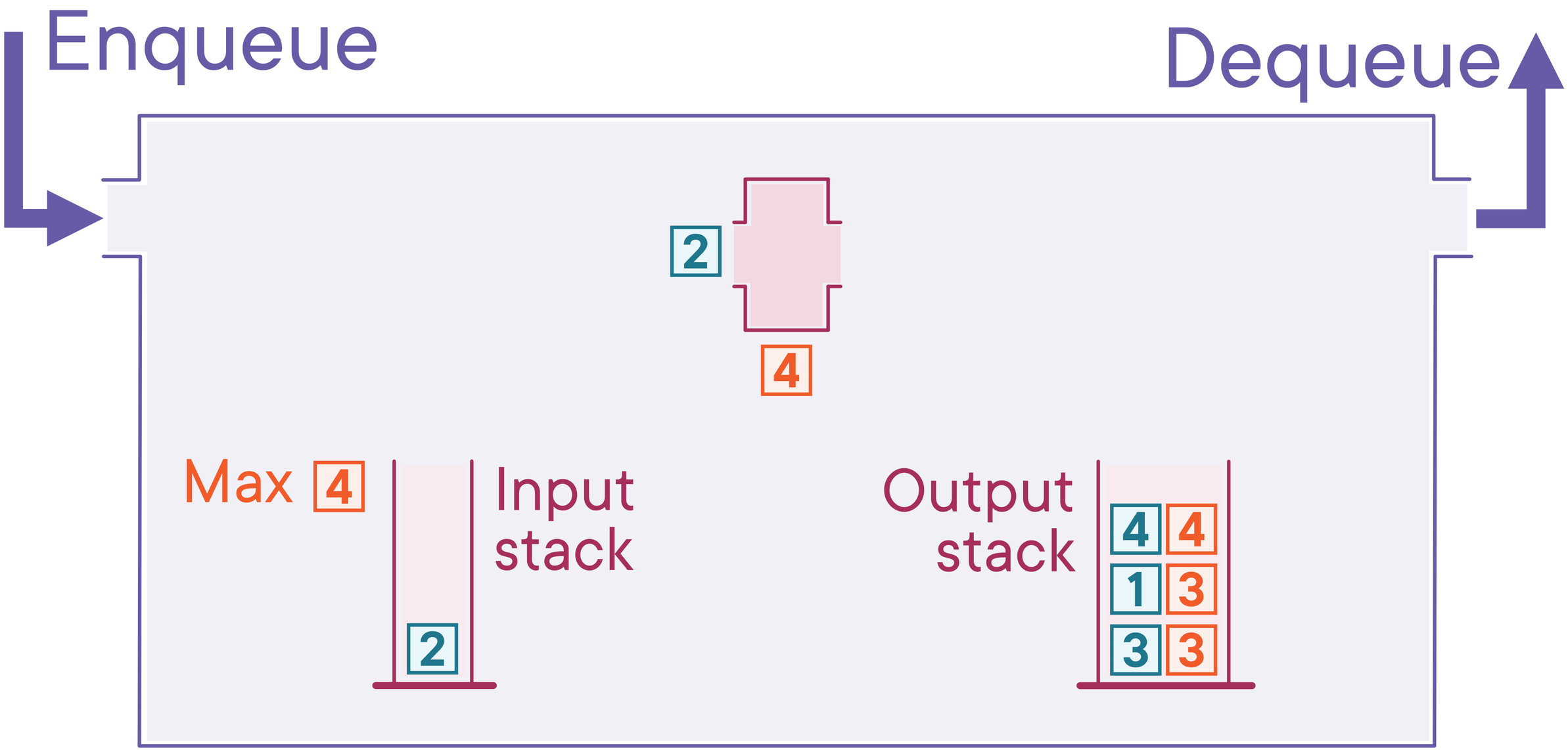

Enqueue
Dequeue
Max
Input stack
Output stack
4
1
3

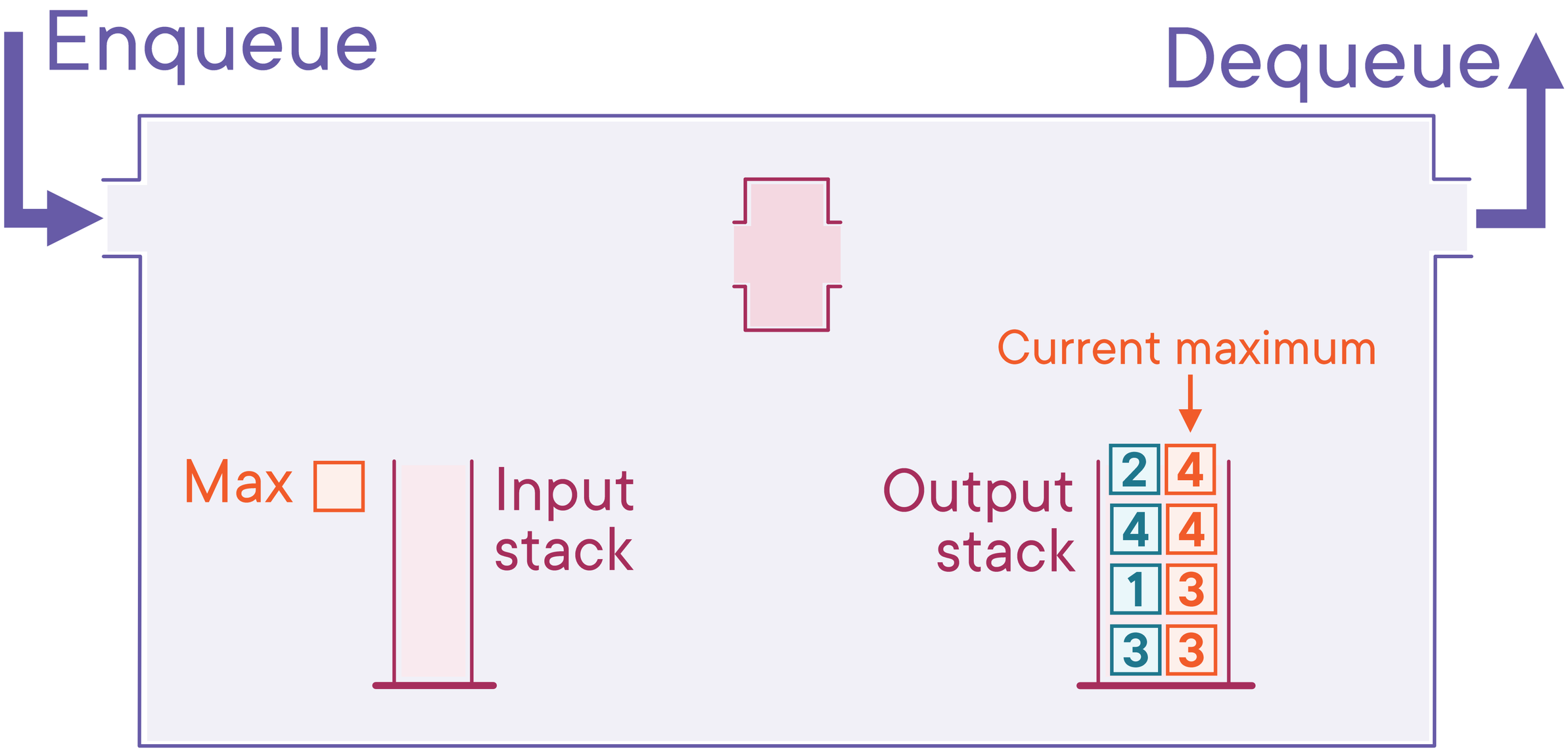

Enqueue
Dequeue
4
Max
Input stack
Output stack
1
3
?

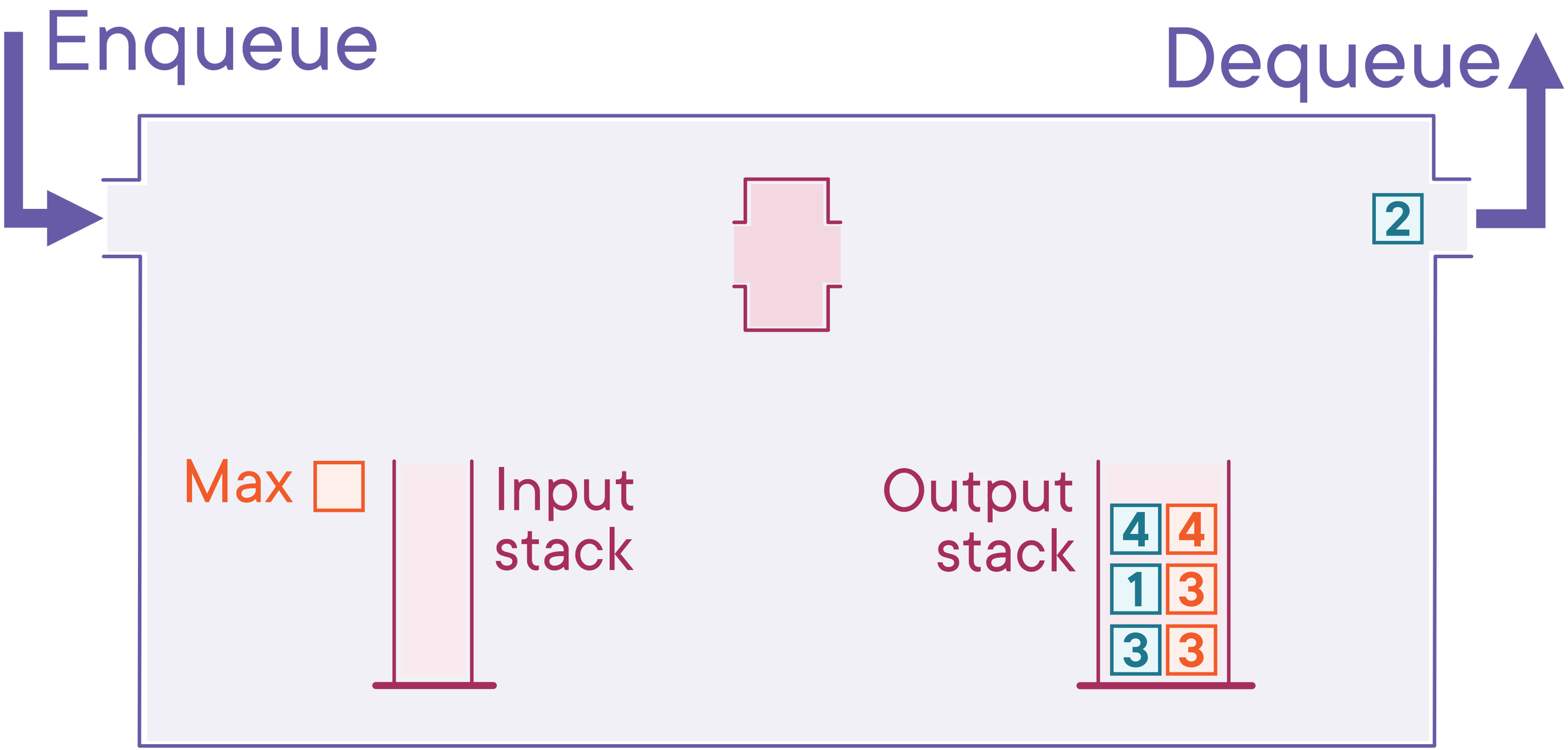

Enqueue
Dequeue
3
Max
4
3
1
4
2
Input stack
Output stack

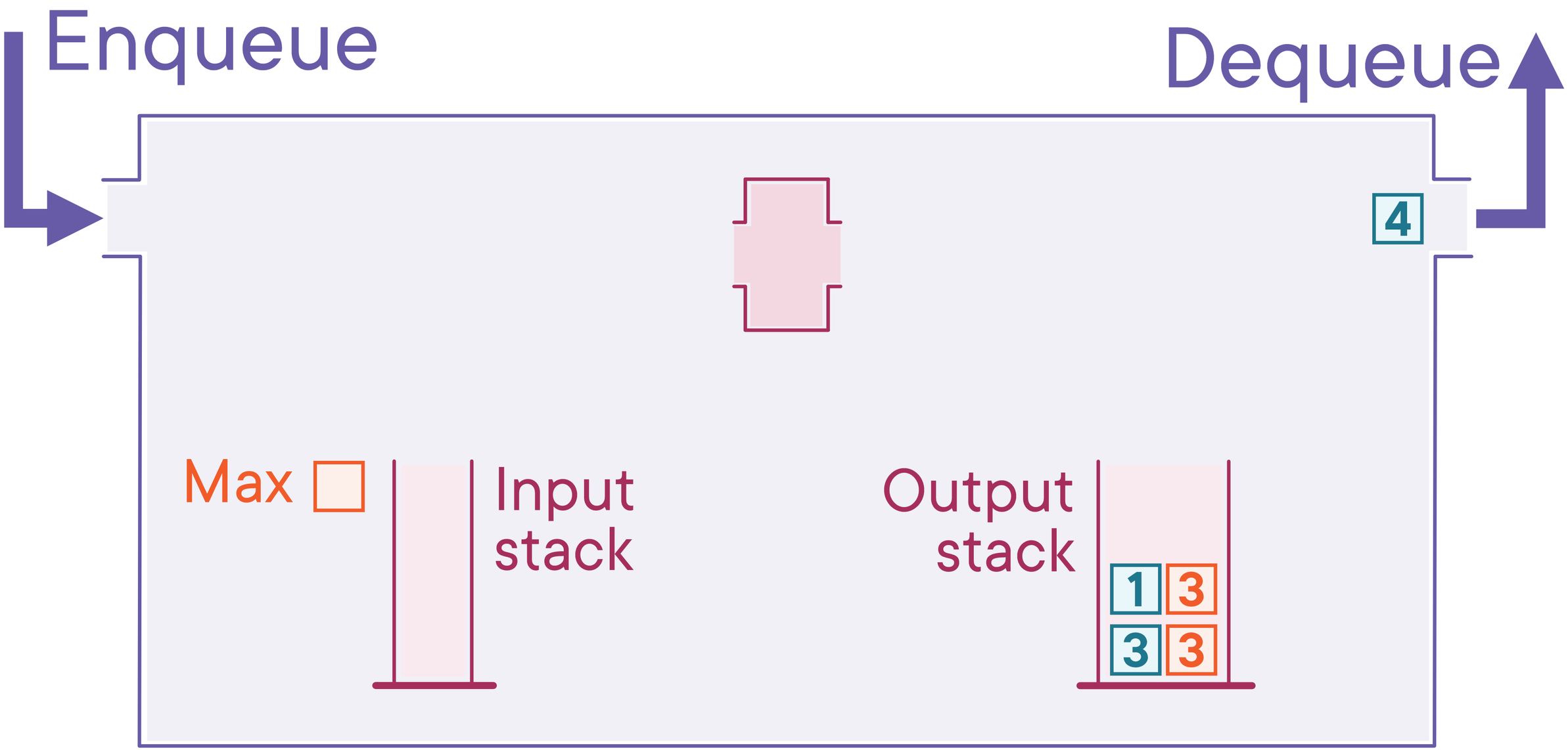

Enqueue
Dequeue
1
3
Max 4
1
4
2
Input stack
Output stack
3
3

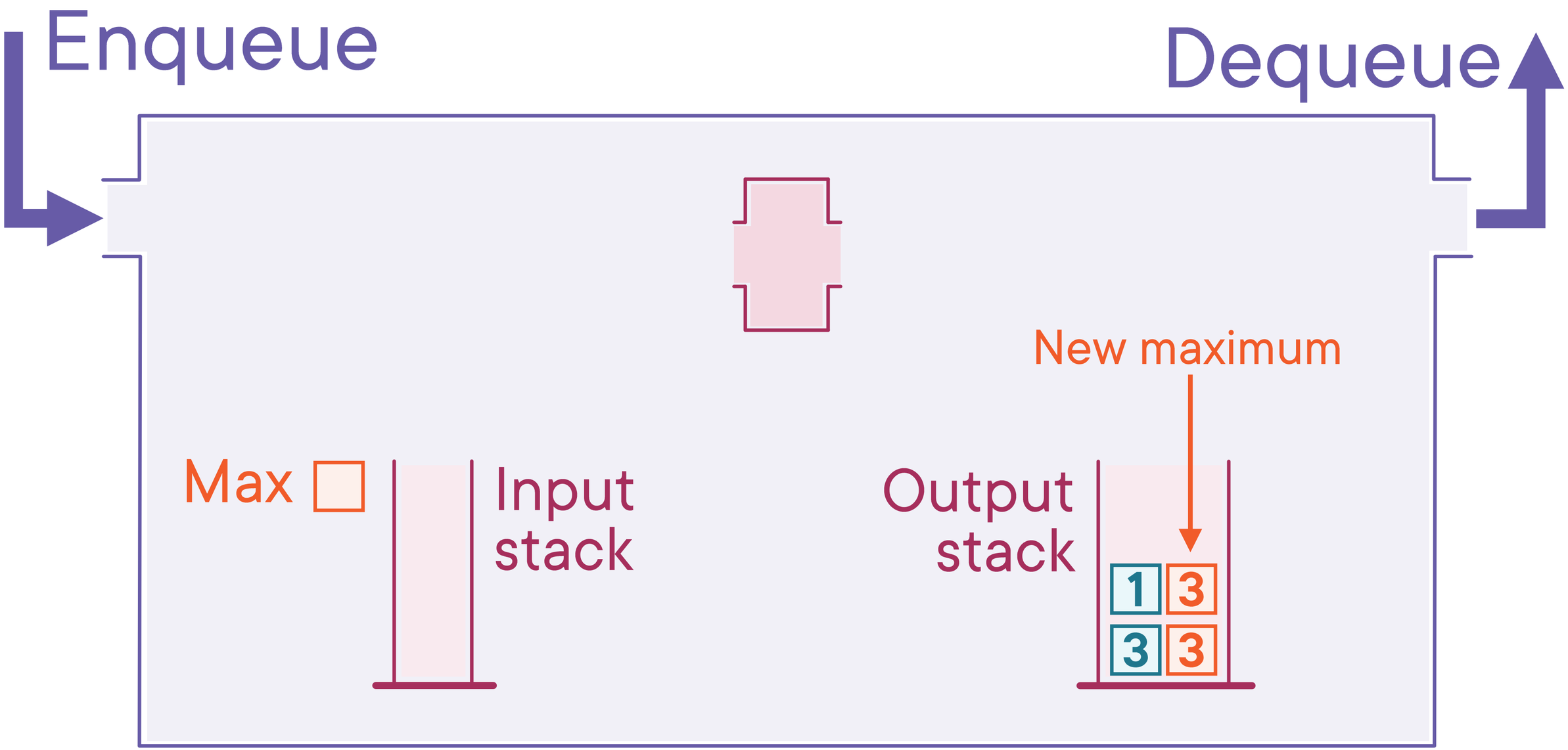

Enqueue
Dequeue
4
3
Max 4
Input stack
4
2
Output stack
1
3
3
3

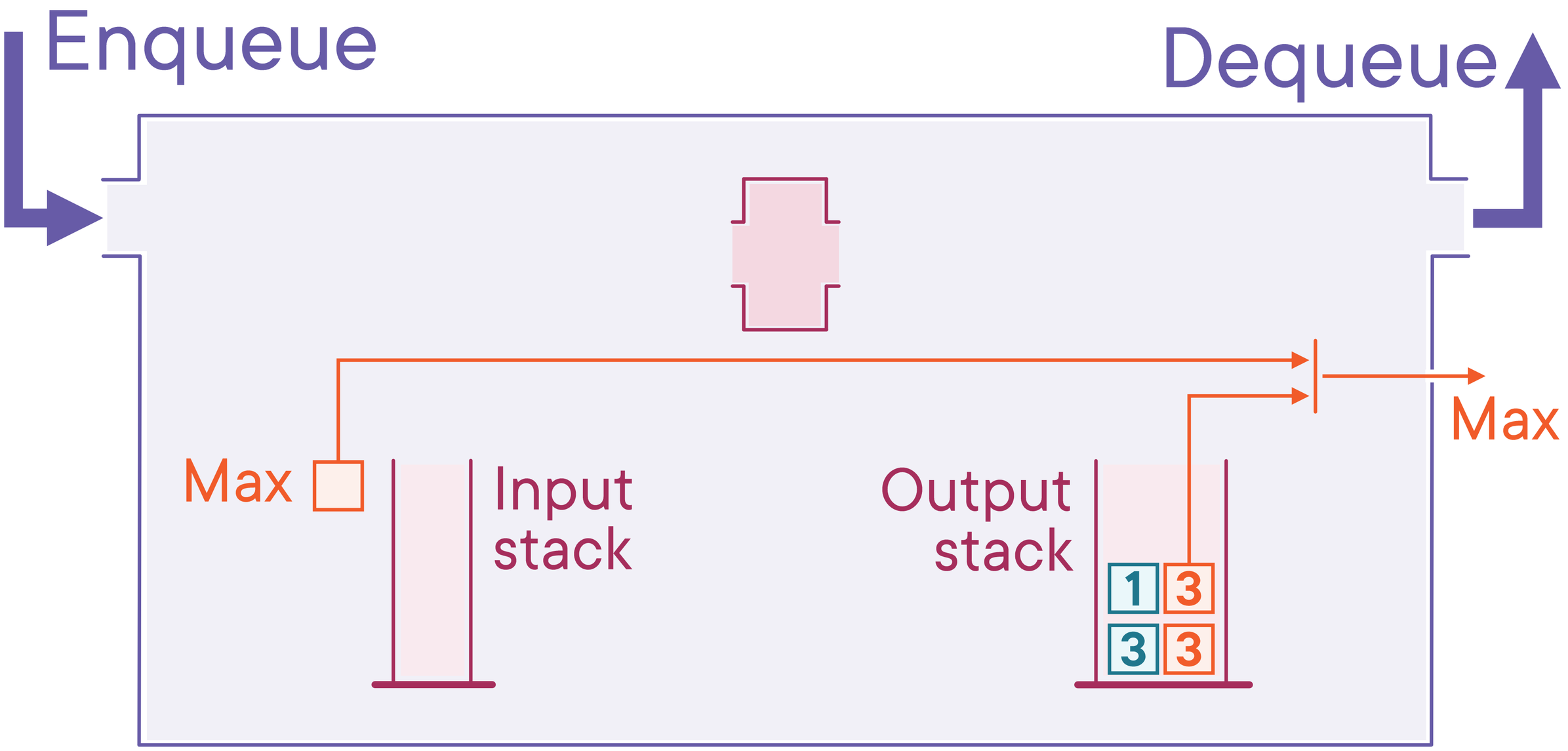

Enqueue
Dequeue
2
4
Max 4
Input stack
2
Output stack
4 4
1 3
3 3


Enqueue
Dequeue
Current maximum
Max
Input stack
Output stack
2 4
4 4
1 3
3 3

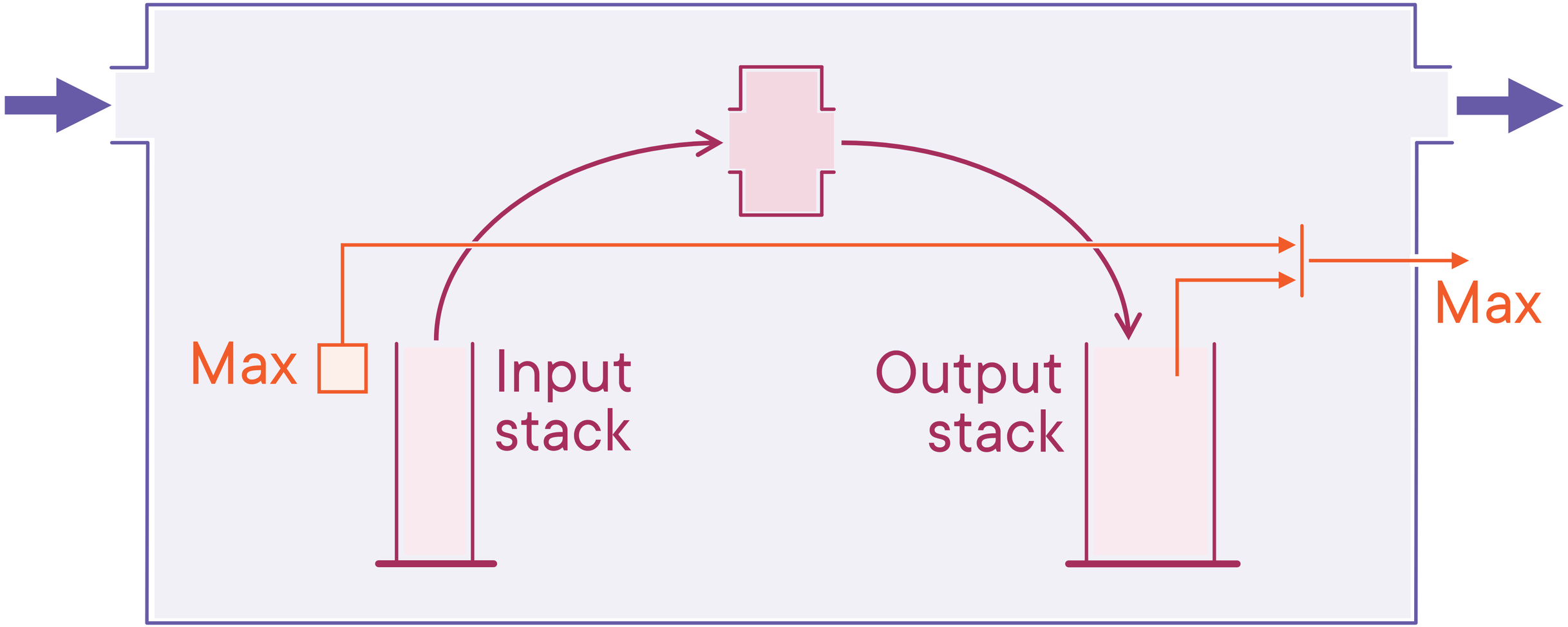

Enqueue
Dequeue
2
Max
Input stack
Output stack
4
4
1
3
3
3


Enqueue
Dequeue
4
Max
Input stack
Output stack
1
3
3
3


Enqueue
Dequeue
New maximum
Max
Input stack
Output stack
1
3
3
3


Enqueue
Dequeue
Max
Input
stack
Output
stack
1
3
3
3
Max

A maximum queue



Summary

Covered the sliding window technique

- Used a queue to aggregate objects
- A complex analysis running in time proportional to input

Designed a maximum queue

- A queue based on two stacks

Solving a specialized problem

- Look for an appropriate collection design
- Implement the new collection

Summary

Collections applied in this course

- Common array
- `List<T>`
- `SortedList<TKey, TValue>`
- `LinkedList<T>`
- `Dictionary<TKey, TValue>`
- `SortedDictionary<TKey, TValue>`
- `ImmutableSortedDictionary<TKey, TValue>`
- `Queue<T>`
- `Stack<T>`
- Custom `MaximumQueue<T>`

Applied .NET generics through all examples