
#GEEKSTONE


ZORAN
HORVAT
#GEEKSTONE

#GEEKSTONE

STRUKTURE PODATAKA

i šta će to, recimo, meni?

Razgovor za posao (istinita priča)

- Kako izdvojiti objekat … iz hijerarhijske strukture …?
- Koristiću funkciju … iz biblioteke …
- Tačno. Drugo pitanje: Sve isto, nema biblioteke.
- Kako nema biblioteke?
- Ponekad, mi pišemo biblioteke.

Koder

Programer

Inženjer

Nije programer

Moderni mitovi

- Mit o brzom i sve bržem hardveru
- Mit o velikoj i sve većoj memoriji
- Mit o brzom i sve bržem internetu

Mit o paralelnom izvršavanju

- Srednje vreme odziva bez paralelnog izvršavanja

$$t_{sr} = \frac{t + 2t + \cdots + nt}{n} = \frac{n(n+1)}{2n}t = \frac{n+1}{2}t \approx \frac{n}{2}t$$

- Srednje vreme odziva sa paralelnim izvršavanjem

$$t_{sr} = nt$$

Performanse algoritama

- Koristi se O-notacija
- Postoje još i o-, Ω- i θ-notacija
- Cilj je opisati gornju granicu cene izvršavanja algoritma
- Razlikujemo srednji slučaj i najgori slučaj

Performanse algoritama

- $O(1)$ – konstantna kompleksnost (dodavanje na kraj liste)
- $O(\log n)$ – logaritamska kompleksnost (binarno pretraživanje)
- $O(n)$ – linearna kompleksnost (dodavanje u sredinu liste)
- $O(n \log n)$ – log-linearna kompleksnost (quicksort, heapsort, …)
- $O(n^2)$ – kvadratna kompleksnost (selection sort)
- $O(n^3)$ – kubna kompleksnost (nagradno pitanje)
- $O(2^n)$ – eksponencijalna kompleksnost (kombinatorni problemi)

Eksponencijalni procesi

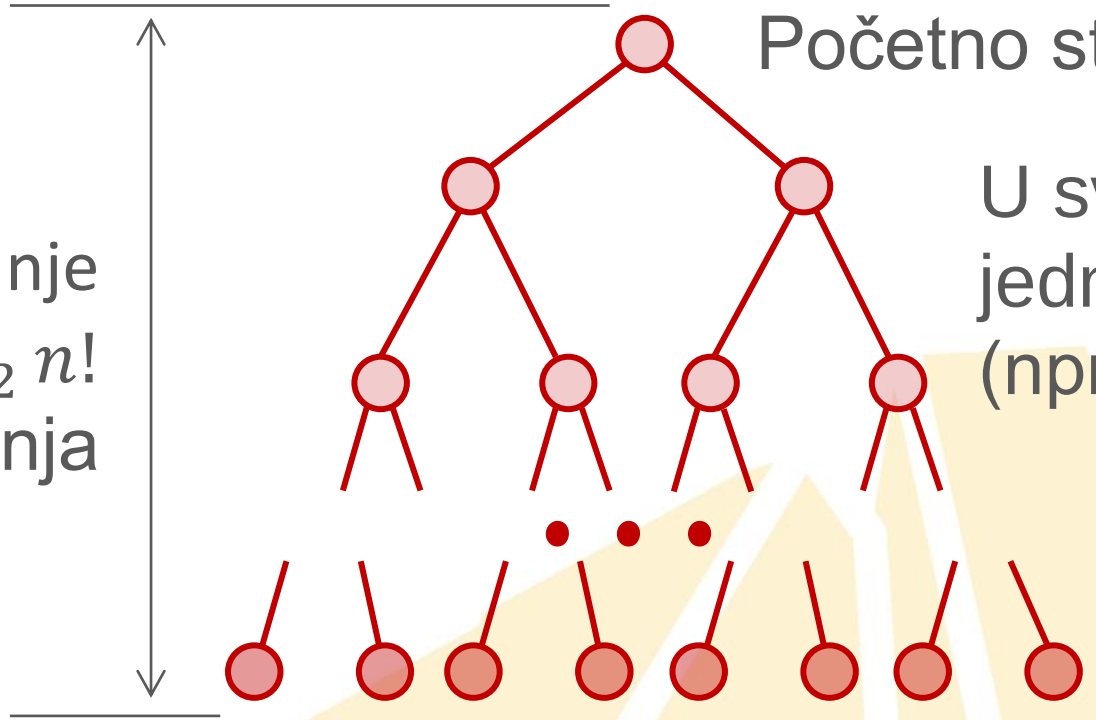
- Eksponencijalna konvergencija
  $f(t) = e^{-t}$ ili $f(t) = 1 - e^{-t}$
- Primer: Polovljenje intervala
  $1/2, 1/4, 1/8 \cdots$
- Vreme za dostizanje vrednosti x:
  $t = -\ln x$
  $x = 0 \Rightarrow t \to +\infty$

Eksponencijalni procesi

– Koja je osnova logaritma u $O(\log n)$?

$$
\begin{aligned}
O(\log n) &= C \log n + o(\log n) \\
O(\log_2 n) &= C \log_2 n + o(\log_2 n) \\
&= C \log_2 k \log_k n + o(\log_2 n) \\
&= D \log_k n + o(\log_2 n) \\
&= O(\log_k n) = O(\log n)
\end{aligned}
$$

Primer: Sortiranje niza



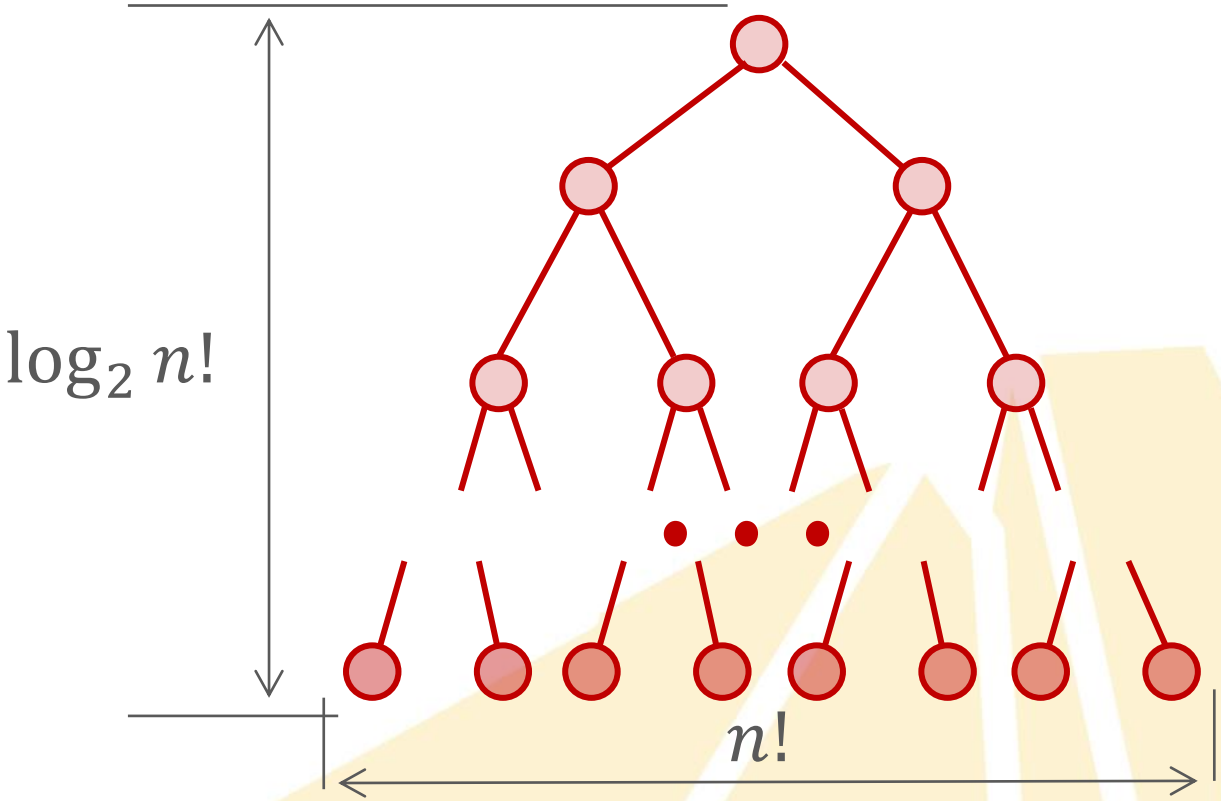
Najmanje $\log_2 n!$ poređenja

Početno stanje niza od n elemenata

U svakom koraku jedno da/ne pitanje (npr. poređenje)

$n!$ Mogućih permutacija

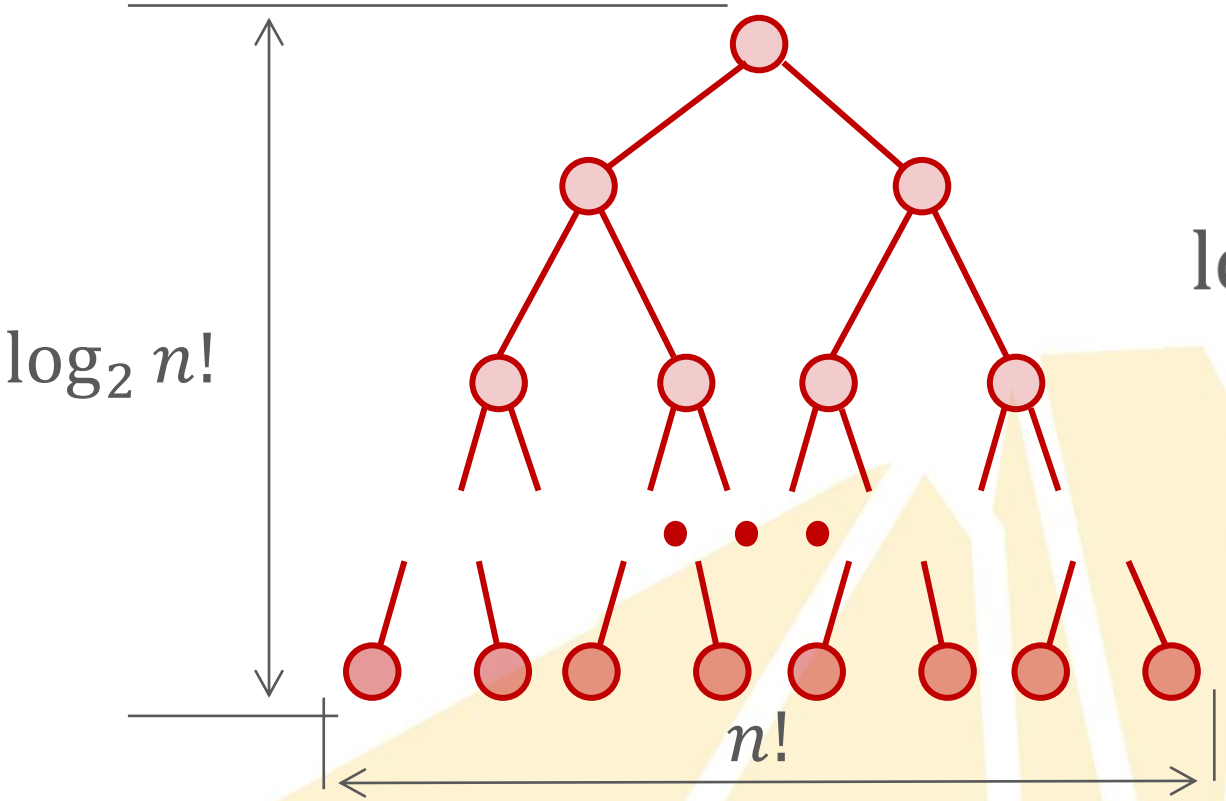
Primer: Sortiranje niza



$$n! \approx \sqrt{2\pi n}\left(\frac{n}{e}\right)^n$$

Stirlingova aproksimacija

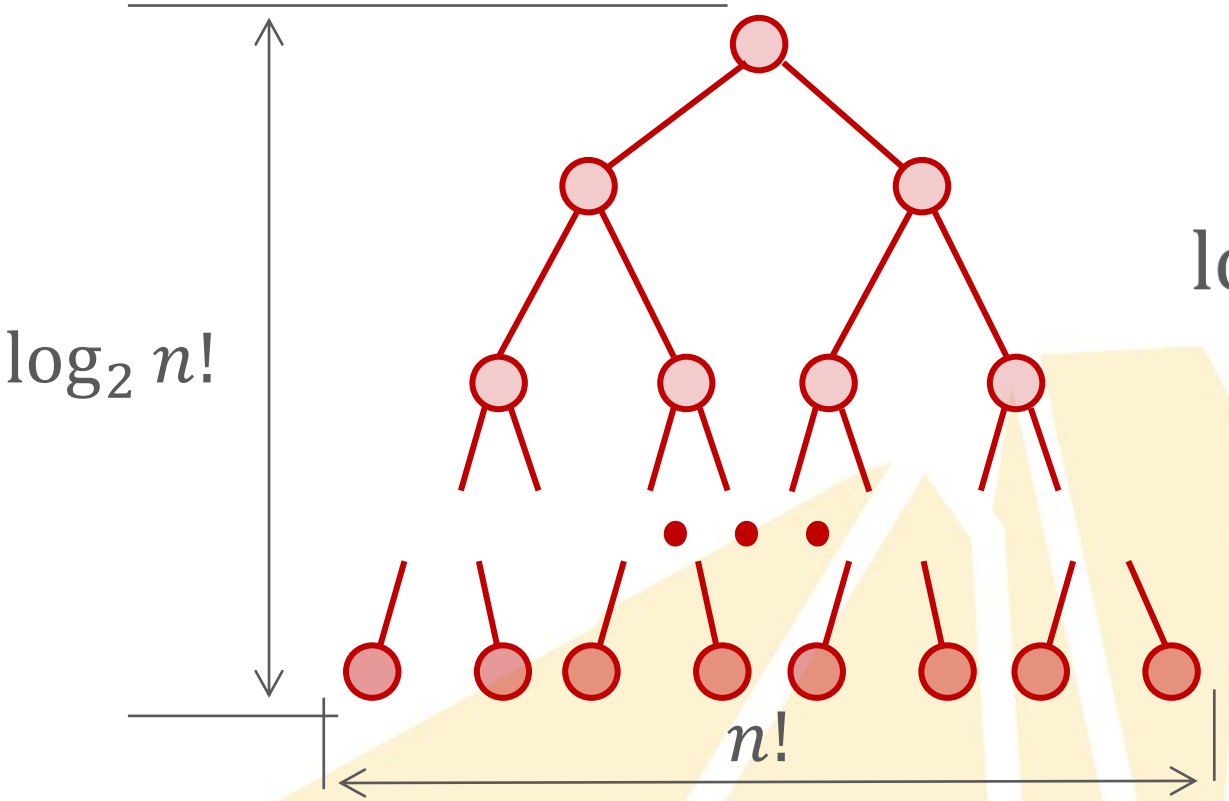
Primer: Sortiranje niza



$$n! \approx \sqrt{2\pi n}\left(\frac{n}{e}\right)^n$$

$$\log_2 n! \approx n \log_2 n - n \log_2 e + \frac{1}{2}\log_2 2\pi n = O(n \log n)$$

Primer: Sortiranje niza



$$n! \approx \sqrt{2\pi n}\left(\frac{n}{e}\right)^n$$

$$\log_2 n! = O(n \log n)$$

Sortiranje poređenjem zahteva najmanje $O(n \log n)$ operacija u srednjem slučaju.

Primer: Sortiranje niza

- Sortiranje poređenjem zahteva najmanje $O(n \log n)$ operacija
- Korisnik:
  Ali… mi bi to brže

Primer: Sortiranje niza

- Sortiranje poređenjem zahteva najmanje $O(n \log n)$ operacija
- Korisnik:
  Ali… mi bi to brže
- Inženjer:
  Primenićemo sortiranje koje ne zavisi od poređenja elemenata
  Npr. Bucket sort i sl.

Primer: Svi najkraći putevi

0	∞	4	∞	∞	2
∞	0	∞	5	2	∞
1	∞	0	1	∞	3
∞	3	2	0	∞	∞
∞	4	∞	∞	0	∞
5	∞	7	∞	∞	0

```
for k from 1 to n
  for i from 1 to n
    for j from 1 to n
      if dst[i][j] > dst[i][k] + dst[k][j]
        dst[i][j] = dst[i][k] + dst[k][j]
```

Floyd-Warshall algoritam $O(n^3)$

Primer: Svi najkraći putevi

0	8	4	5	10	2
8	0	7	5	2	10
1	4	0	1	6	3
3	3	2	0	5	5
12	4	11	9	0	14
5	11	7	8	13	0

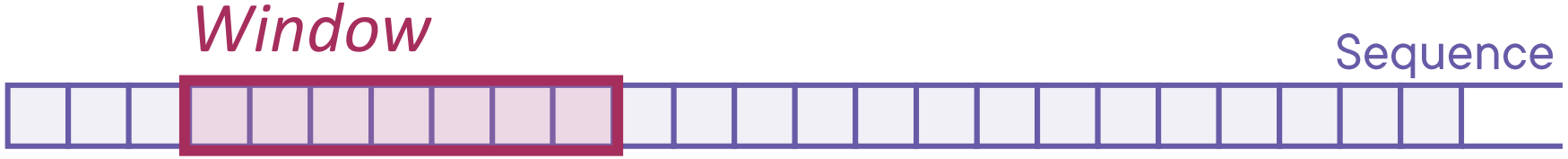
```
for k from 1 to n
  for i from 1 to n
    for j from 1 to n
      if dst[i][j] > dst[i][k] + dst[k][j]
        dst[i][j] = dst[i][k] + dst[k][j]
```

Floyd-Warshall algoritam $O(n^3)$

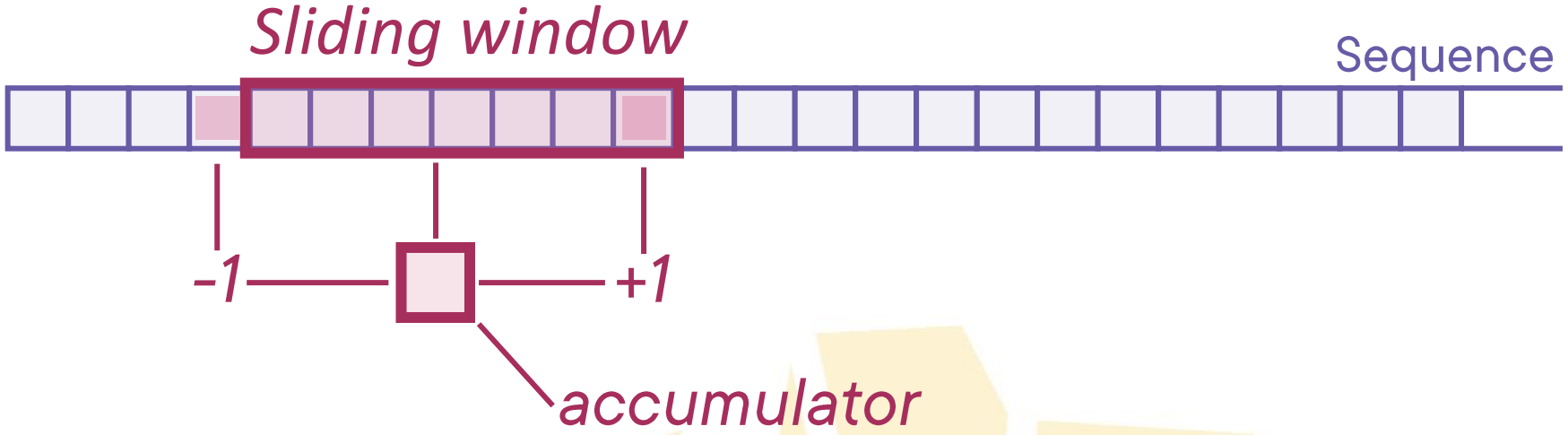
Podržava rekonstrukciju putanja

Alternativno: Dijkstrin algoritam $O(n^2 \log n)$

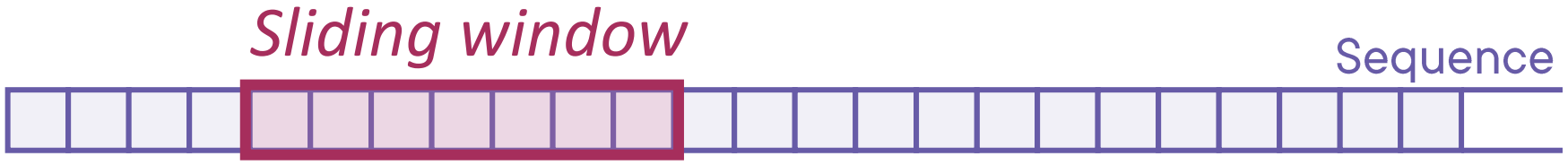
Primer: Sliding window



Primer: Sliding window



Primer: Sliding window

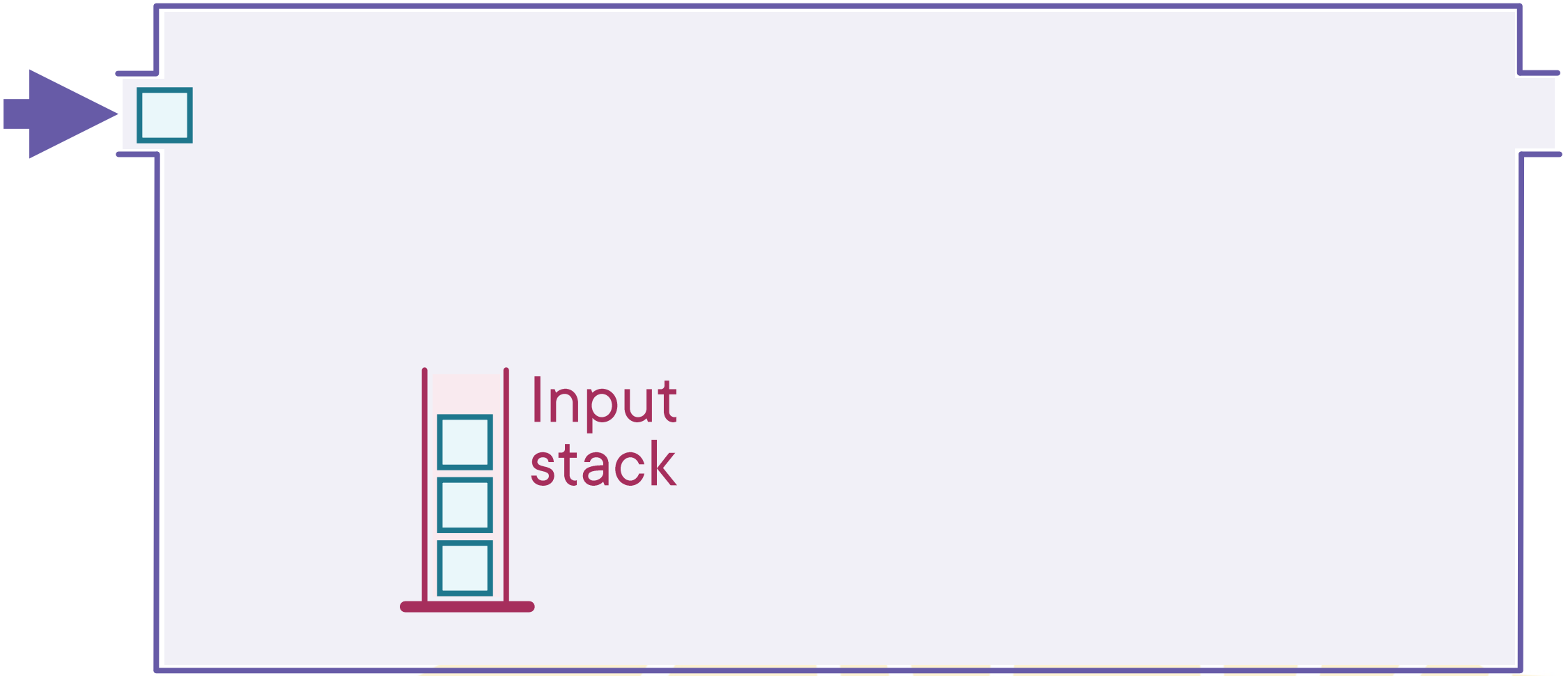


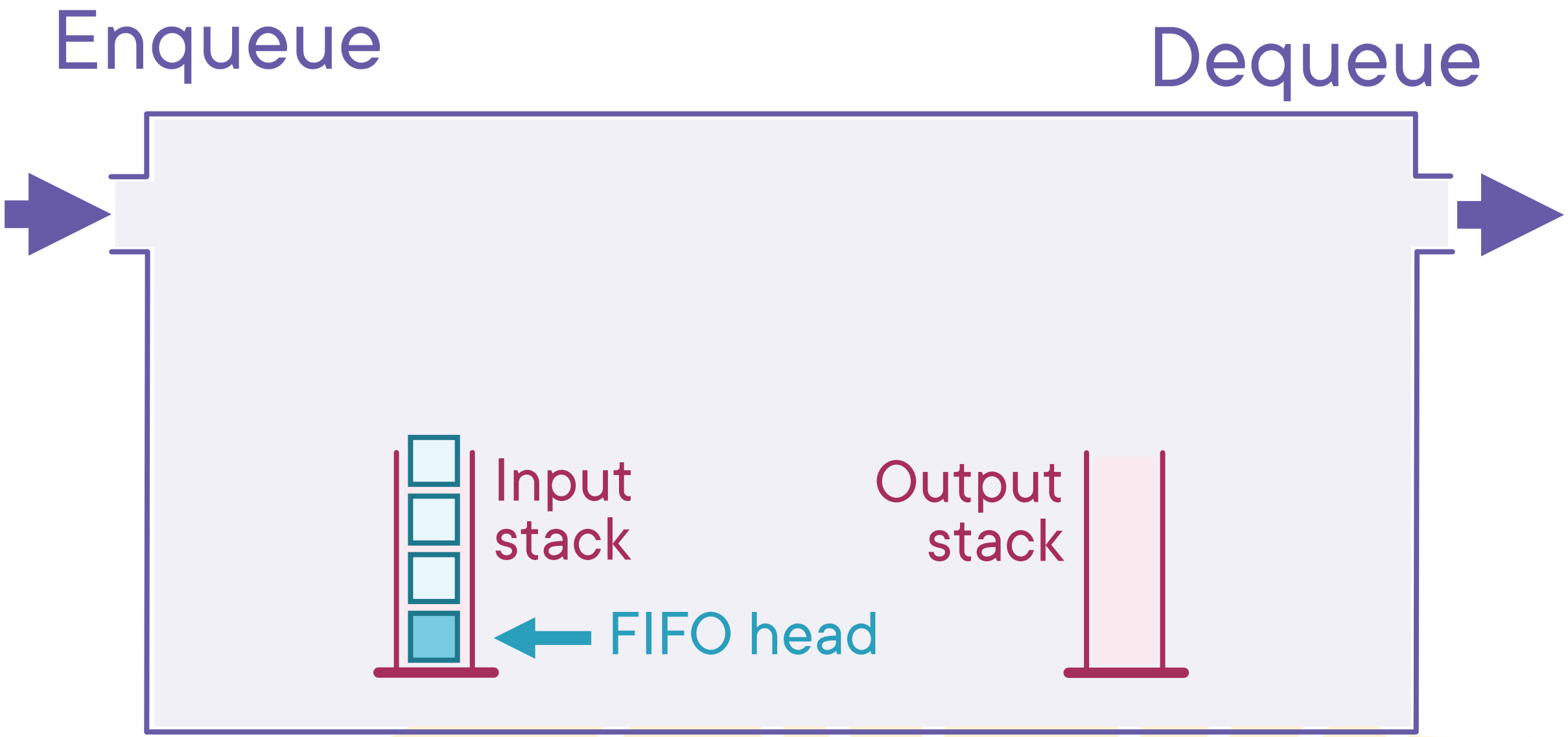

```
queue
sum = 0

add(item)
  queue.enqueue(item)
  sum = sum + item
  while shouldRemove(queue)
    sum = sum - queue.dequeue()
```

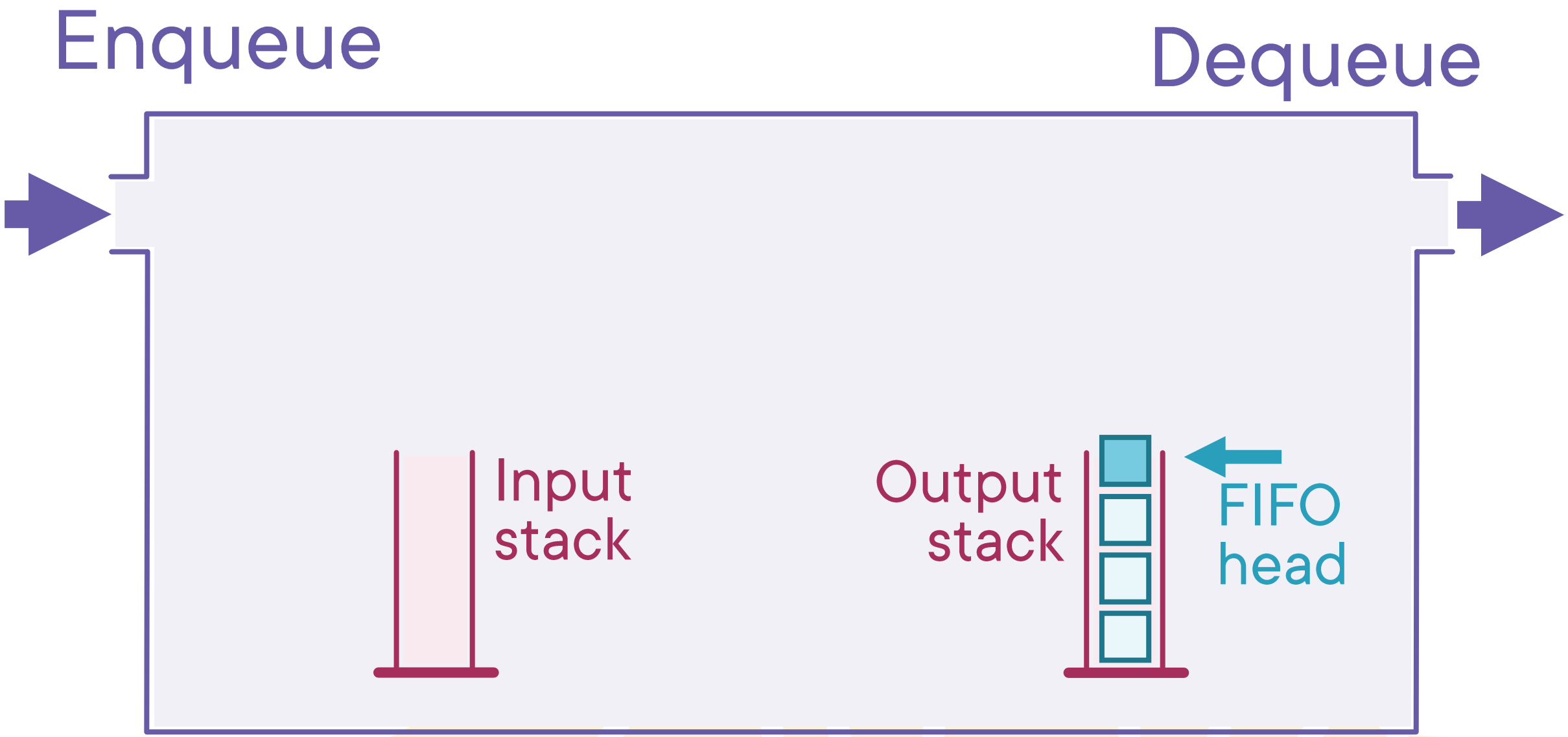
```
movingAverage()
  if queue.isEmpty return 0
  return sum / queue.count
```

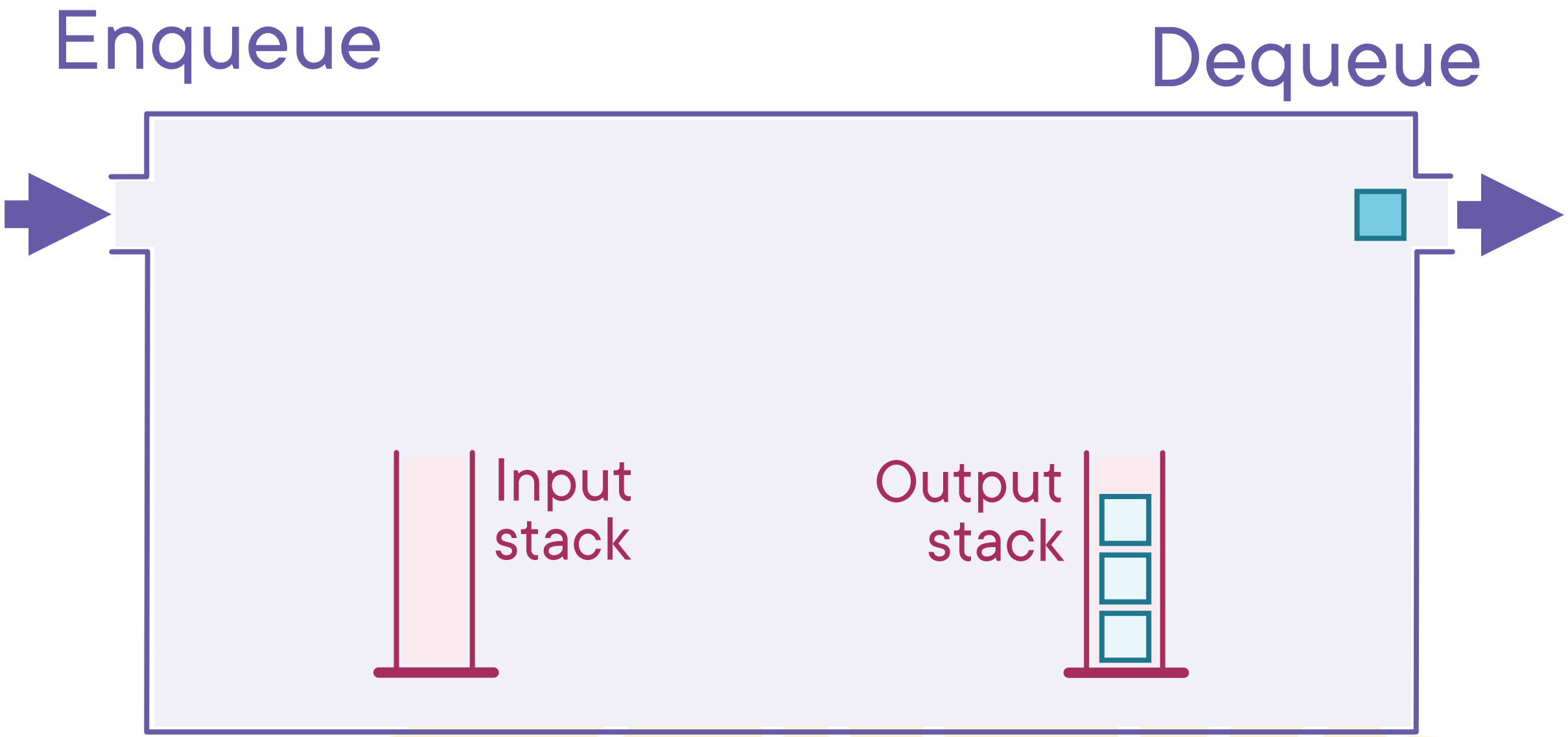
Dodavanje elementa,
čitanje rezultata
u vremenu $O(1)$

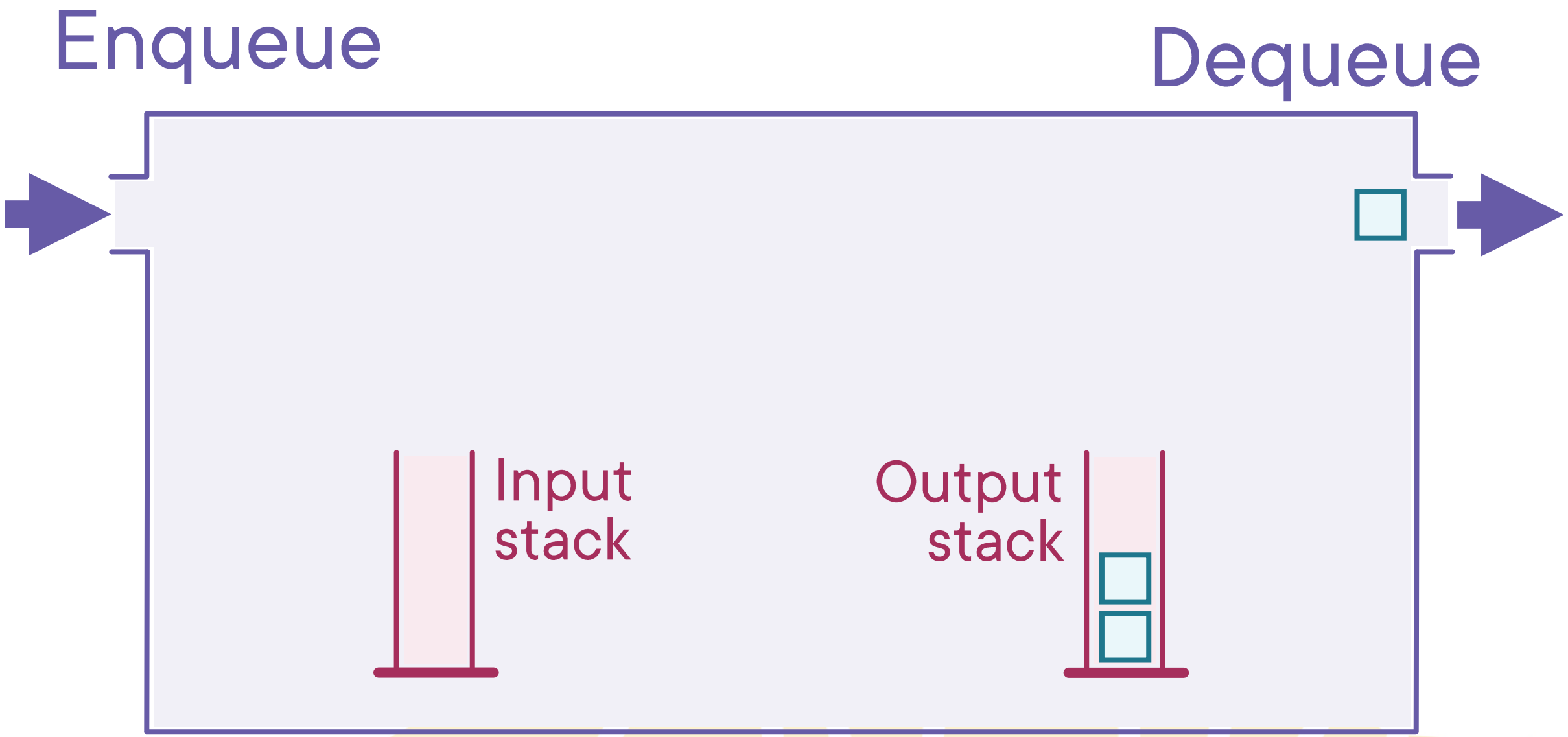
Enqueue

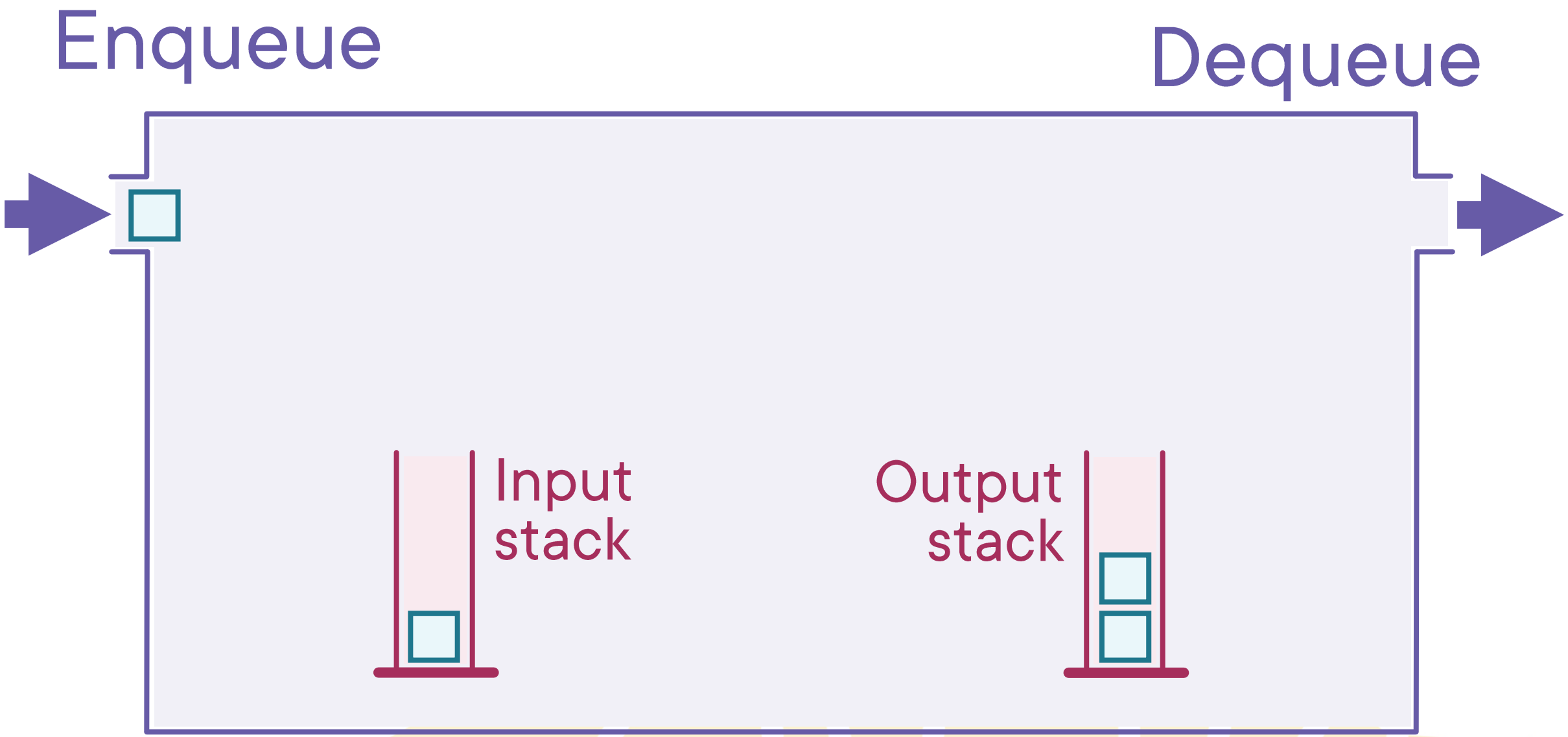


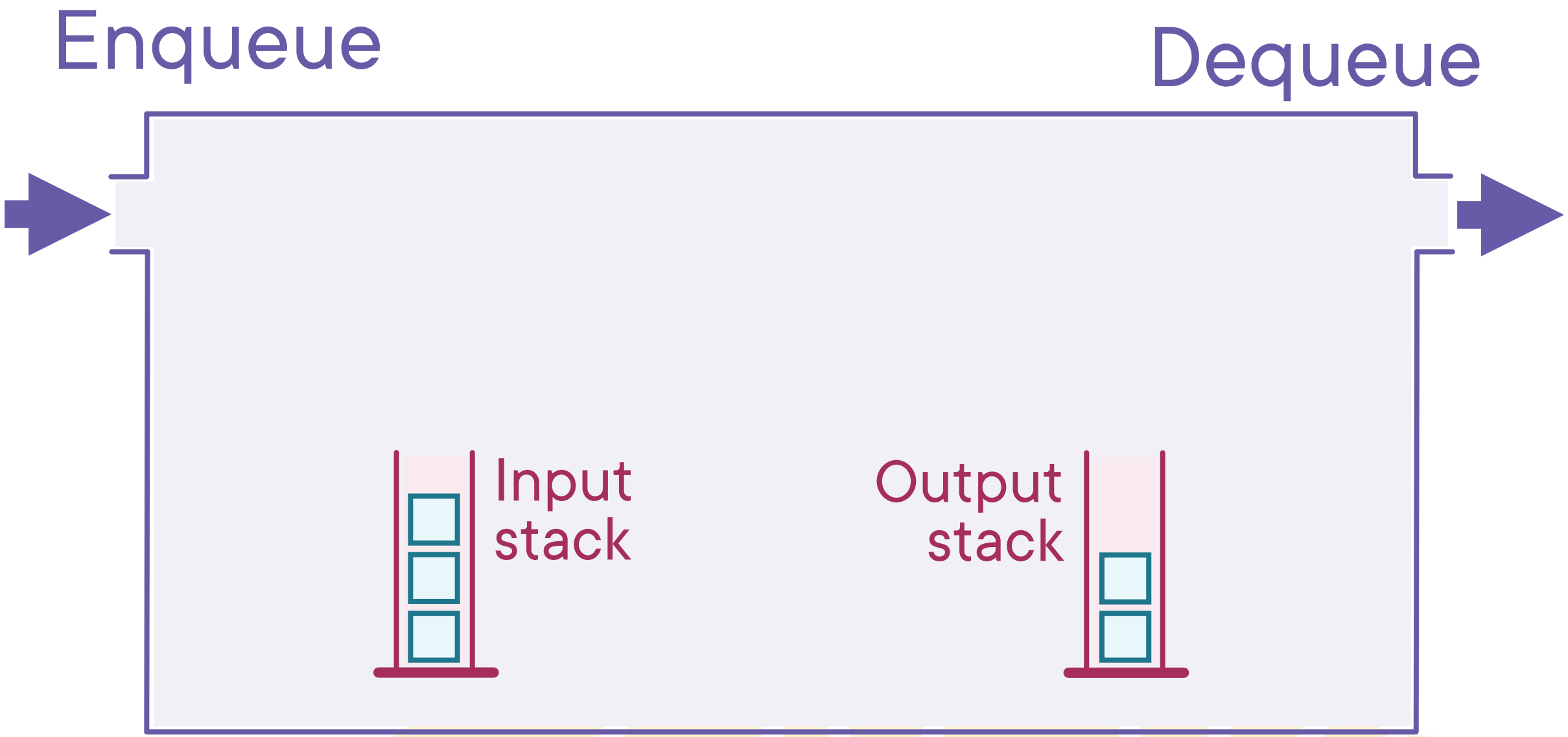

Enqueue
Dequeue
Input stack
FIFO head
Output stack

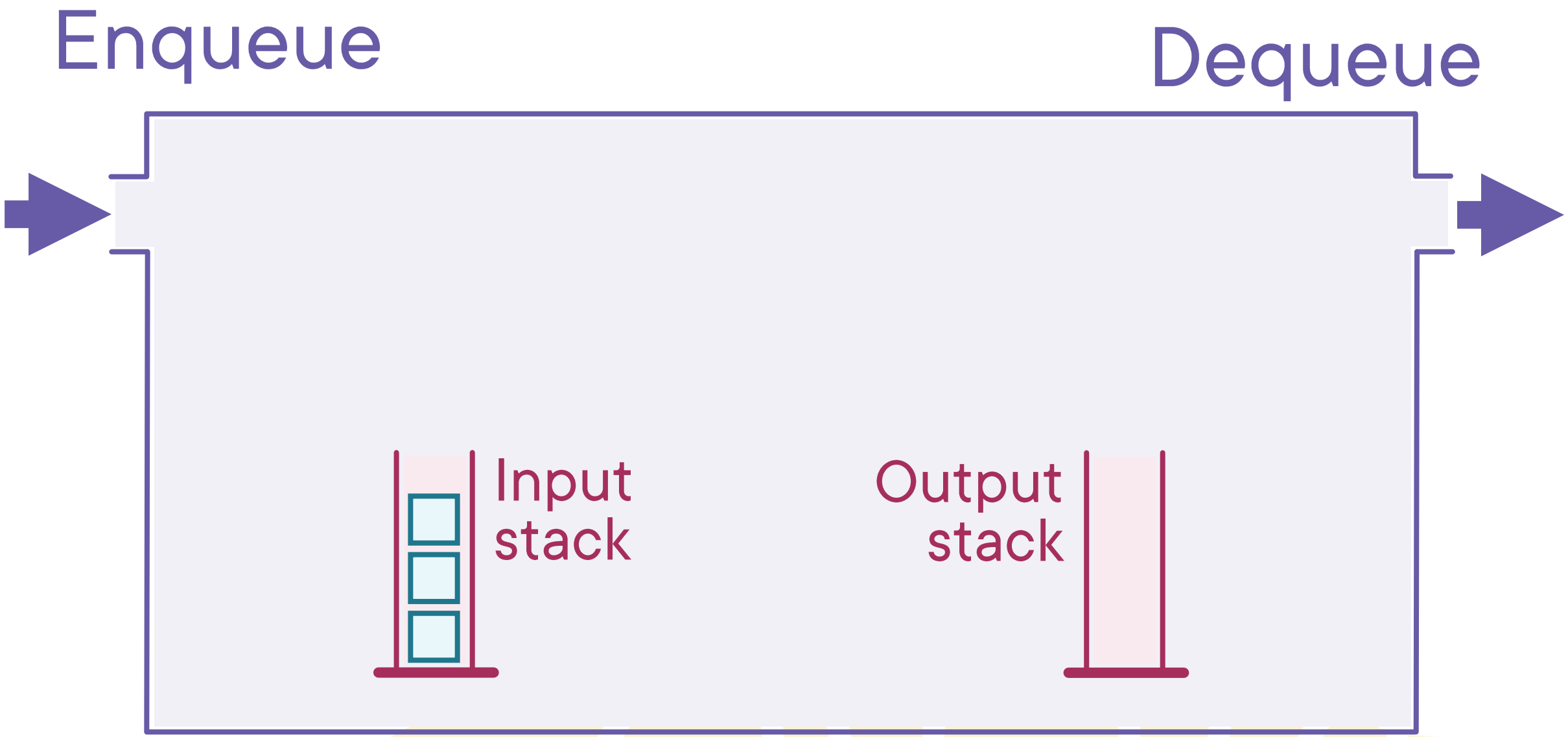

Enqueue
Dequeue
Input stack
Output stack
FIFO head

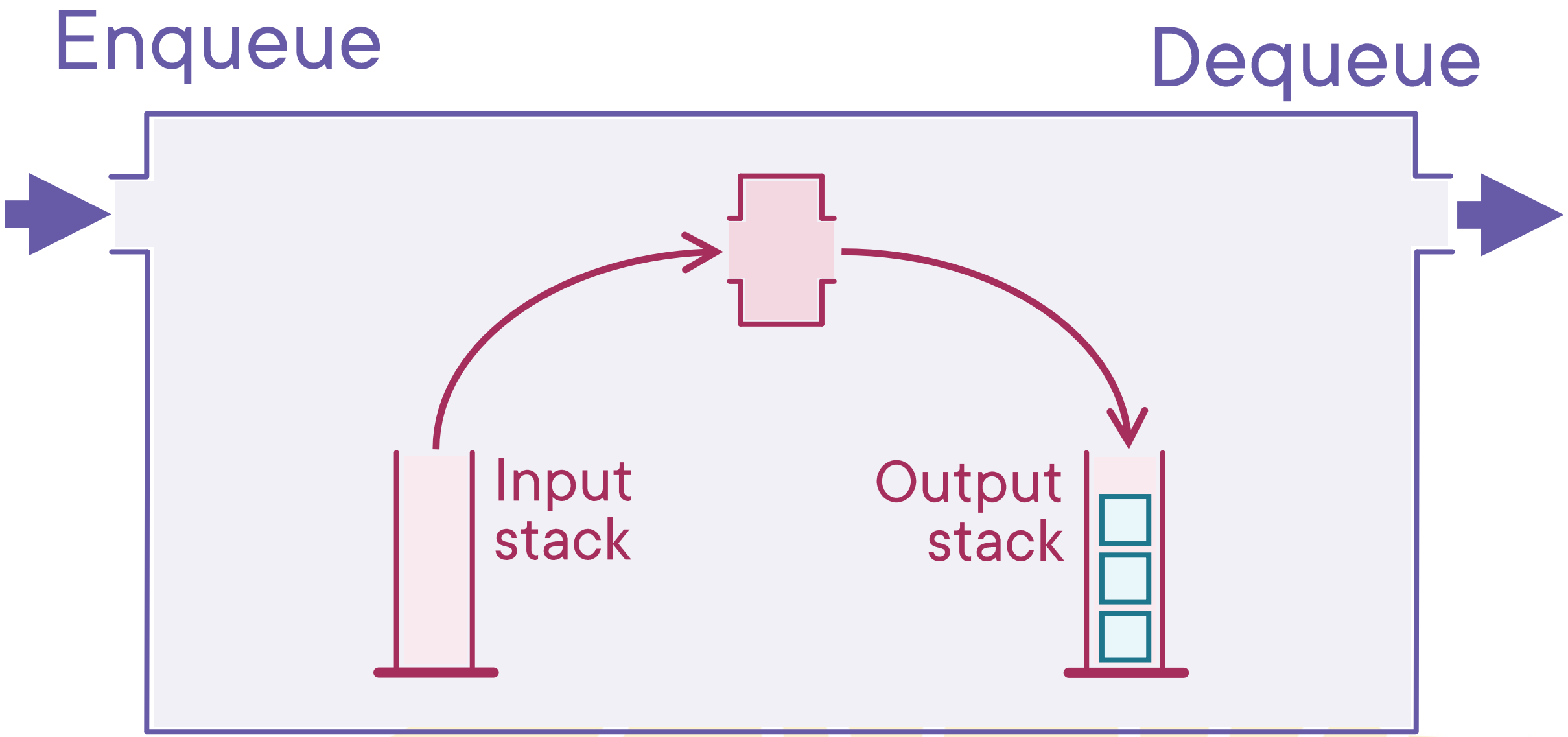

Enqueue
Dequeue
Input stack
Output stack

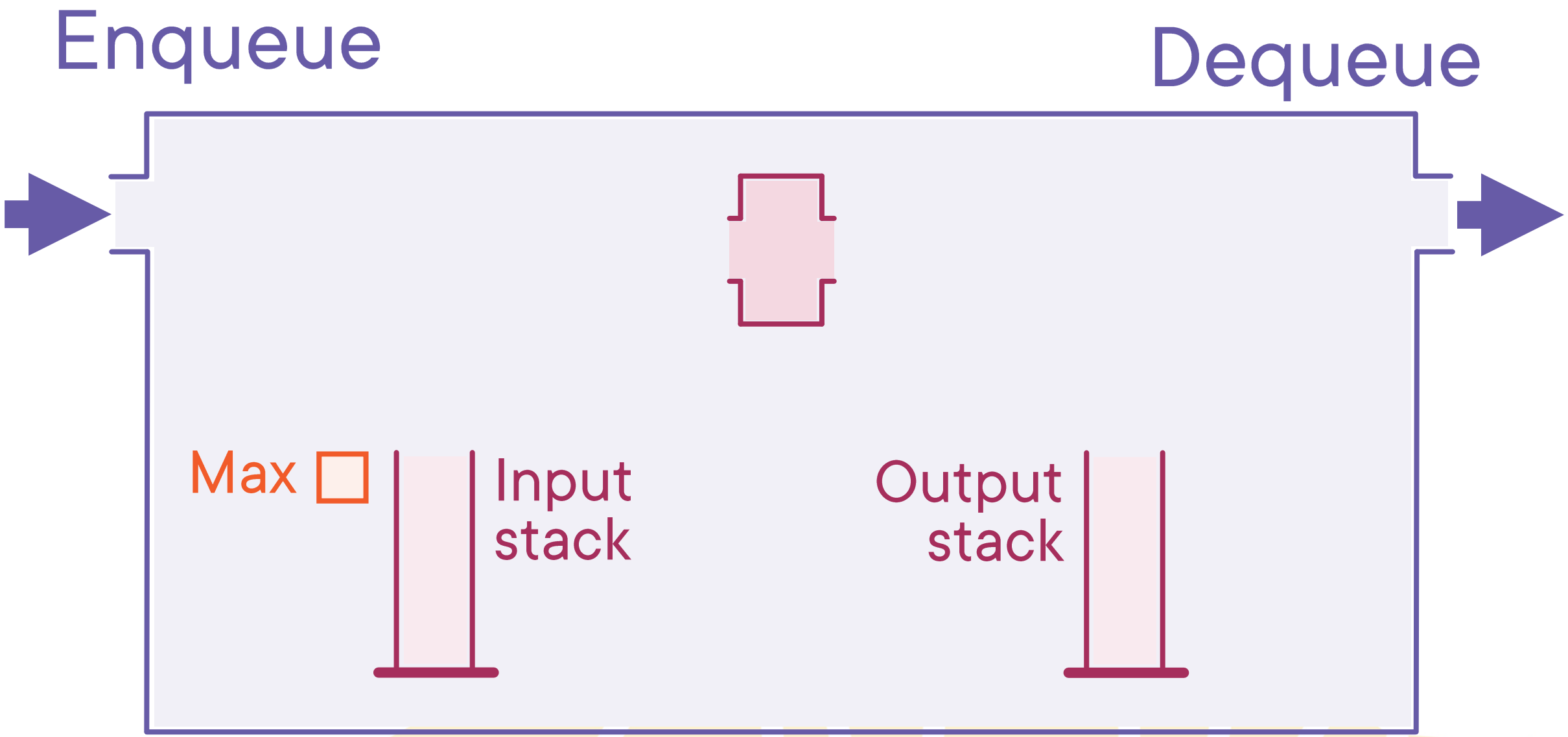

Enqueue
Dequeue
Input stack
Output stack

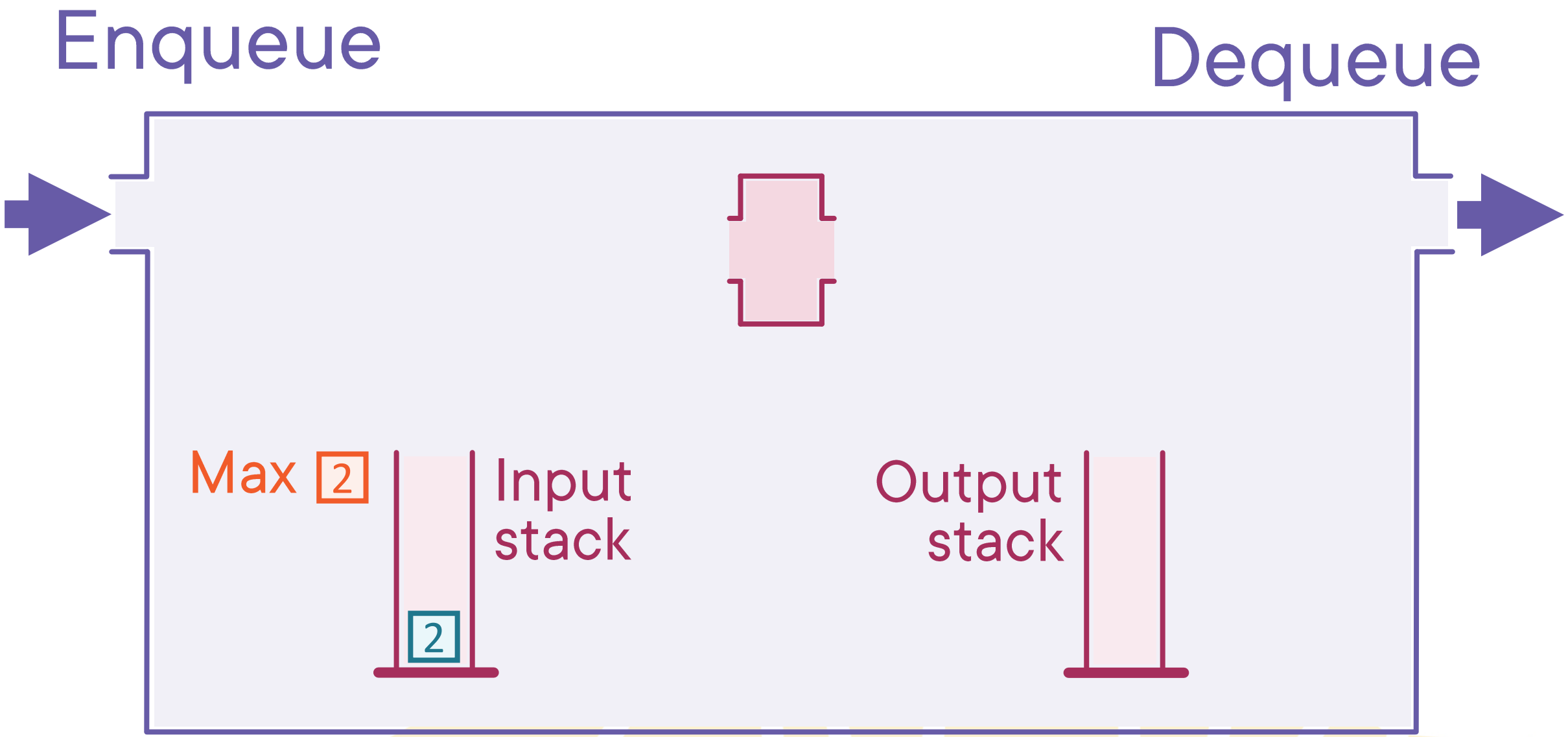

Enqueue
Dequeue
Input stack
Output stack

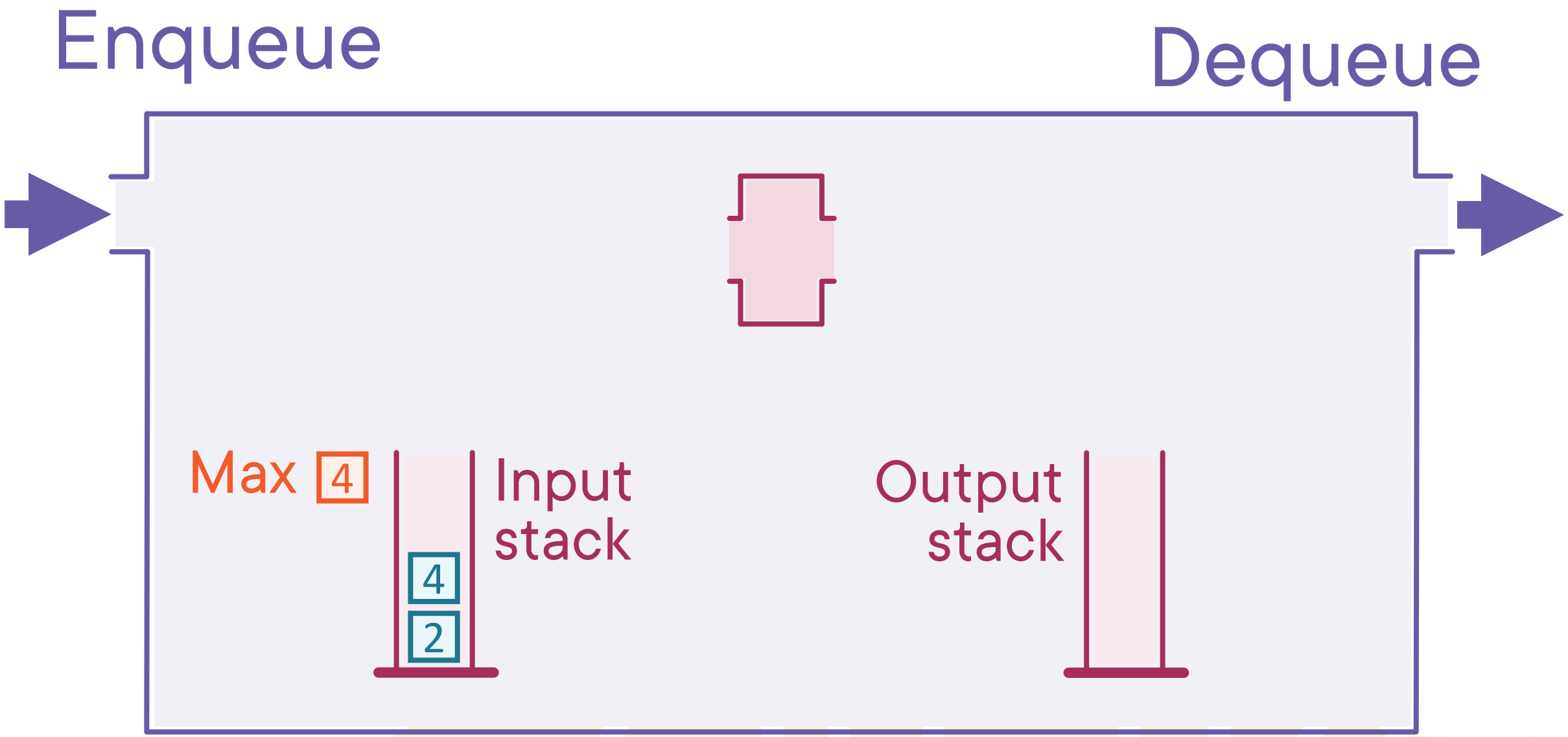

Enqueue
Dequeue
Input stack
Output stack

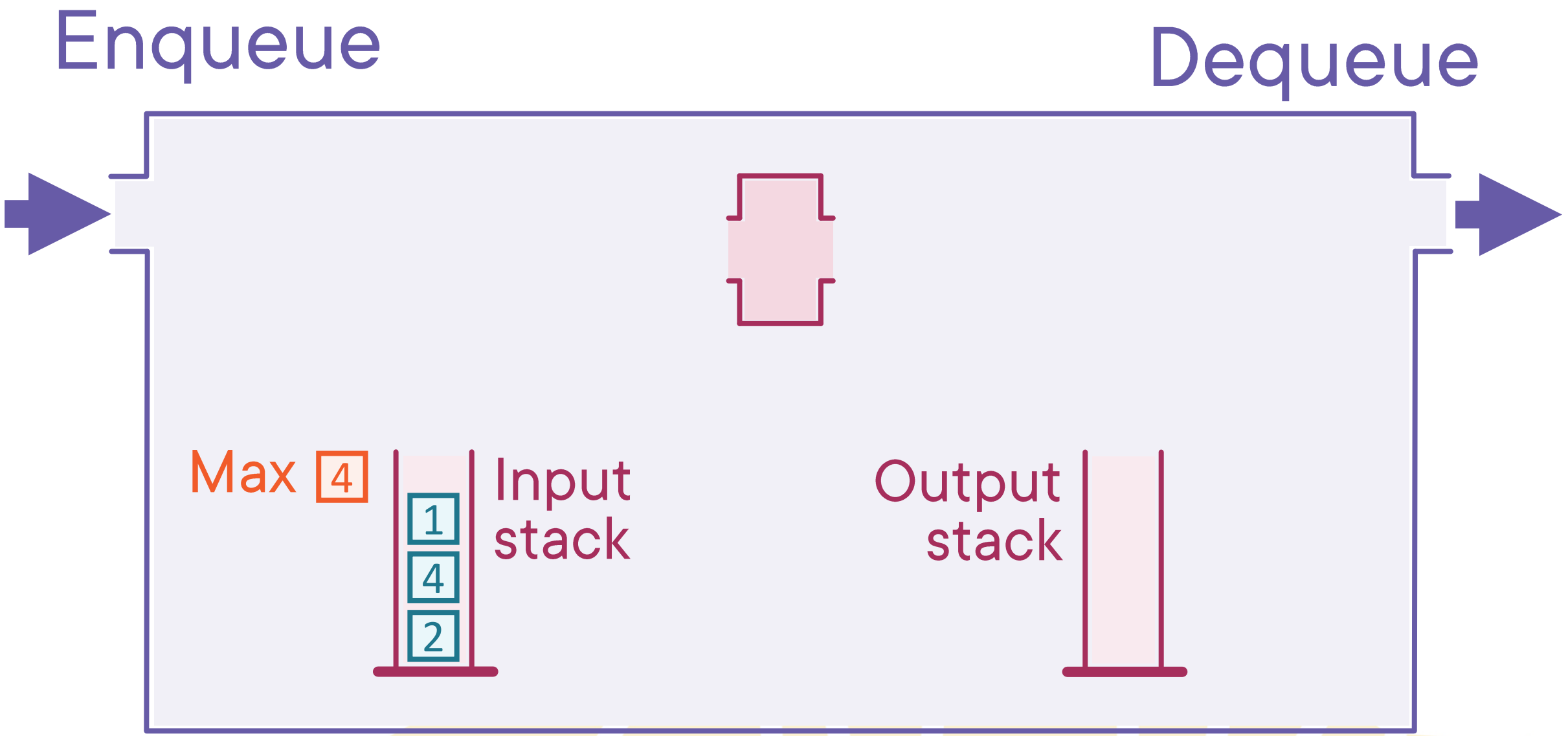

Enqueue
Dequeue
Input stack
Output stack

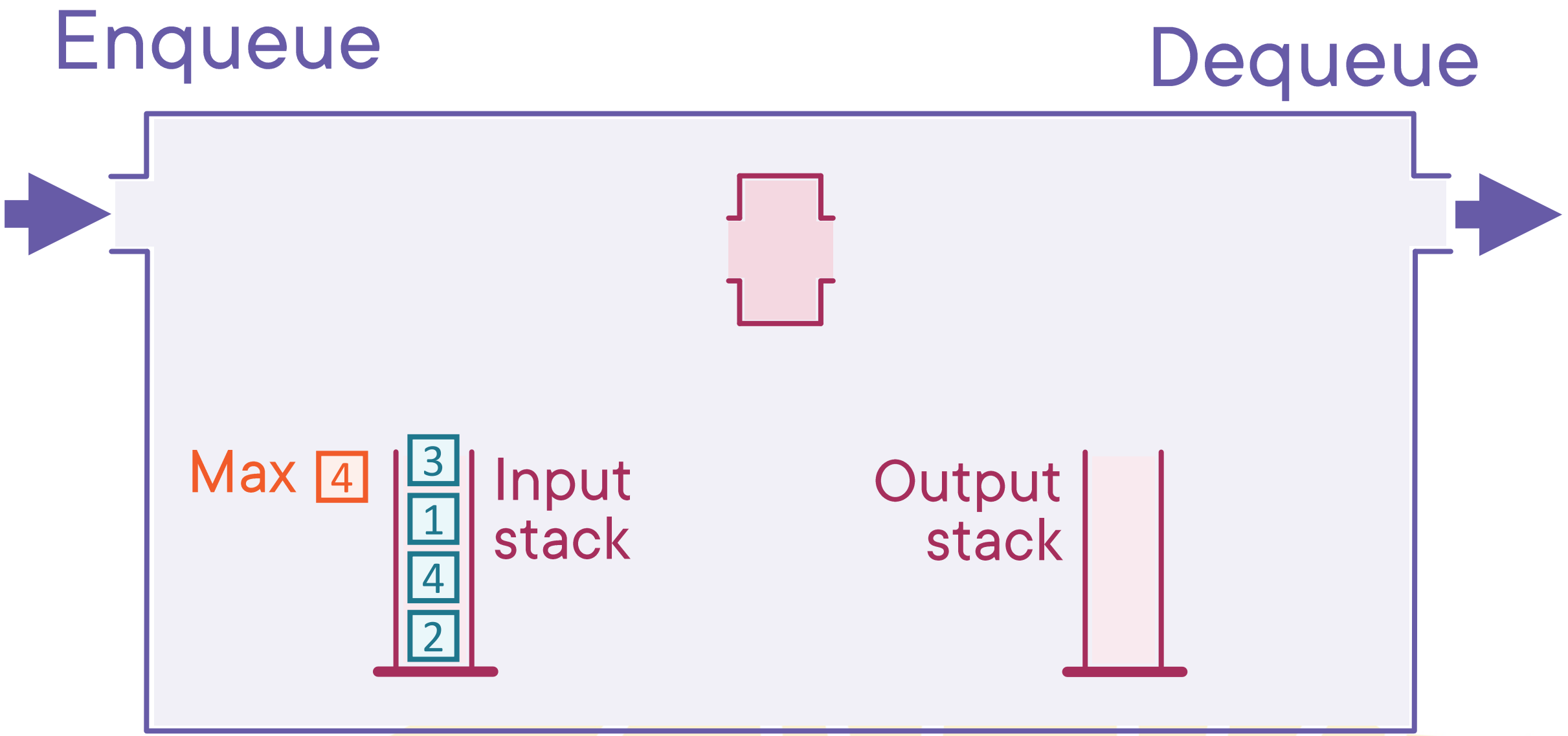

Enqueue
Dequeue
Input stack
Output stack

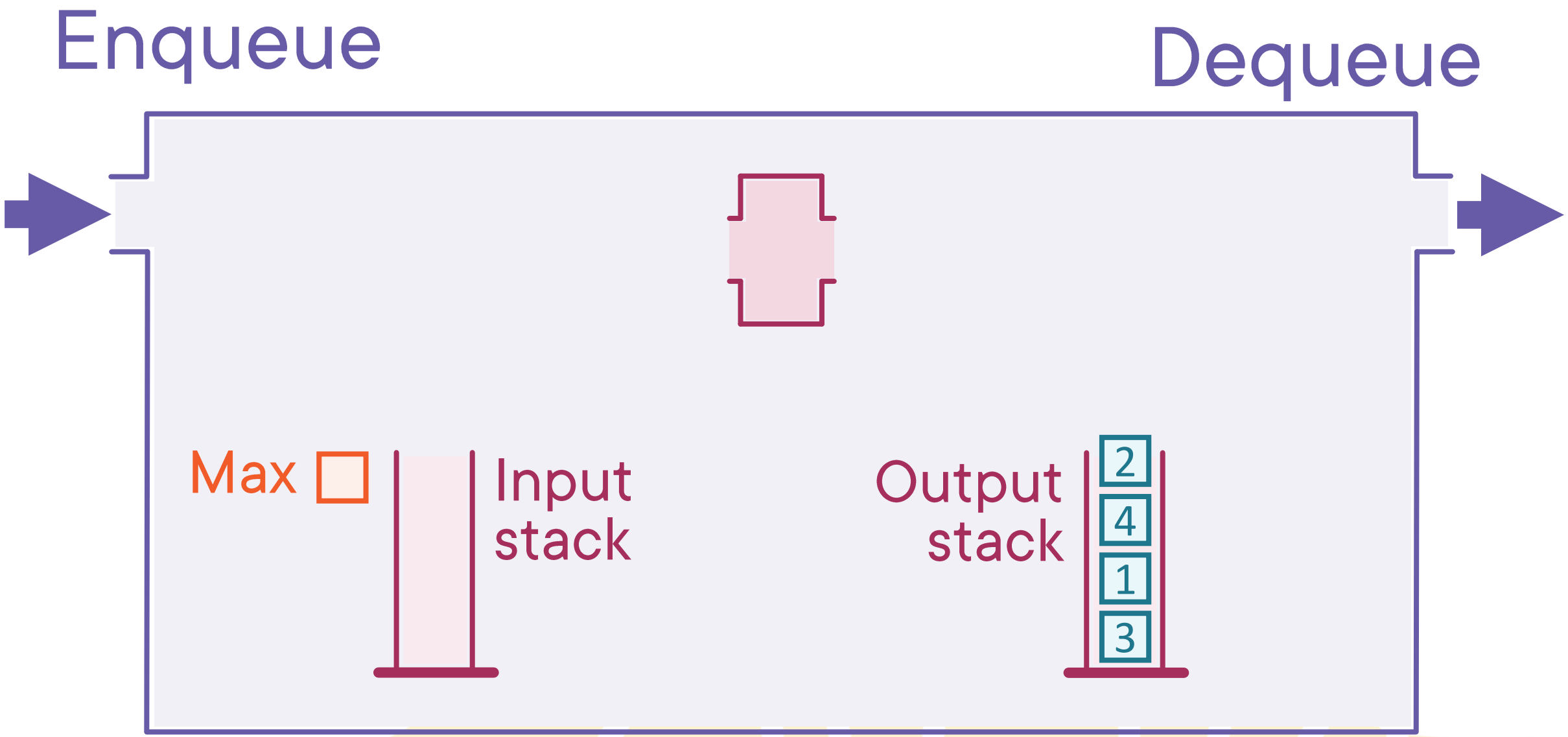

Enqueue
Dequeue
Max
Input stack
Output stack

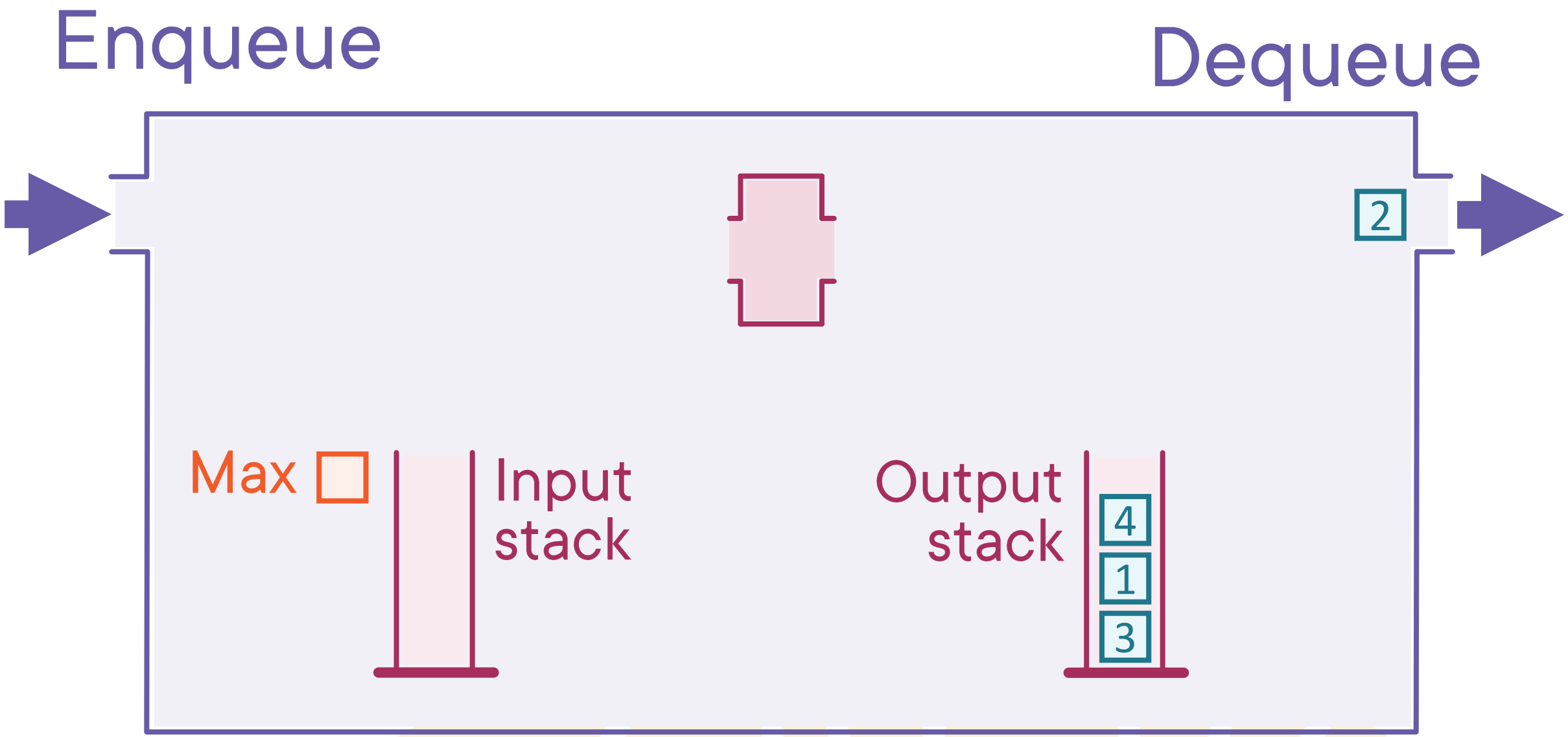

Enqueue
Dequeue
Max 2
Input stack
2
Output stack

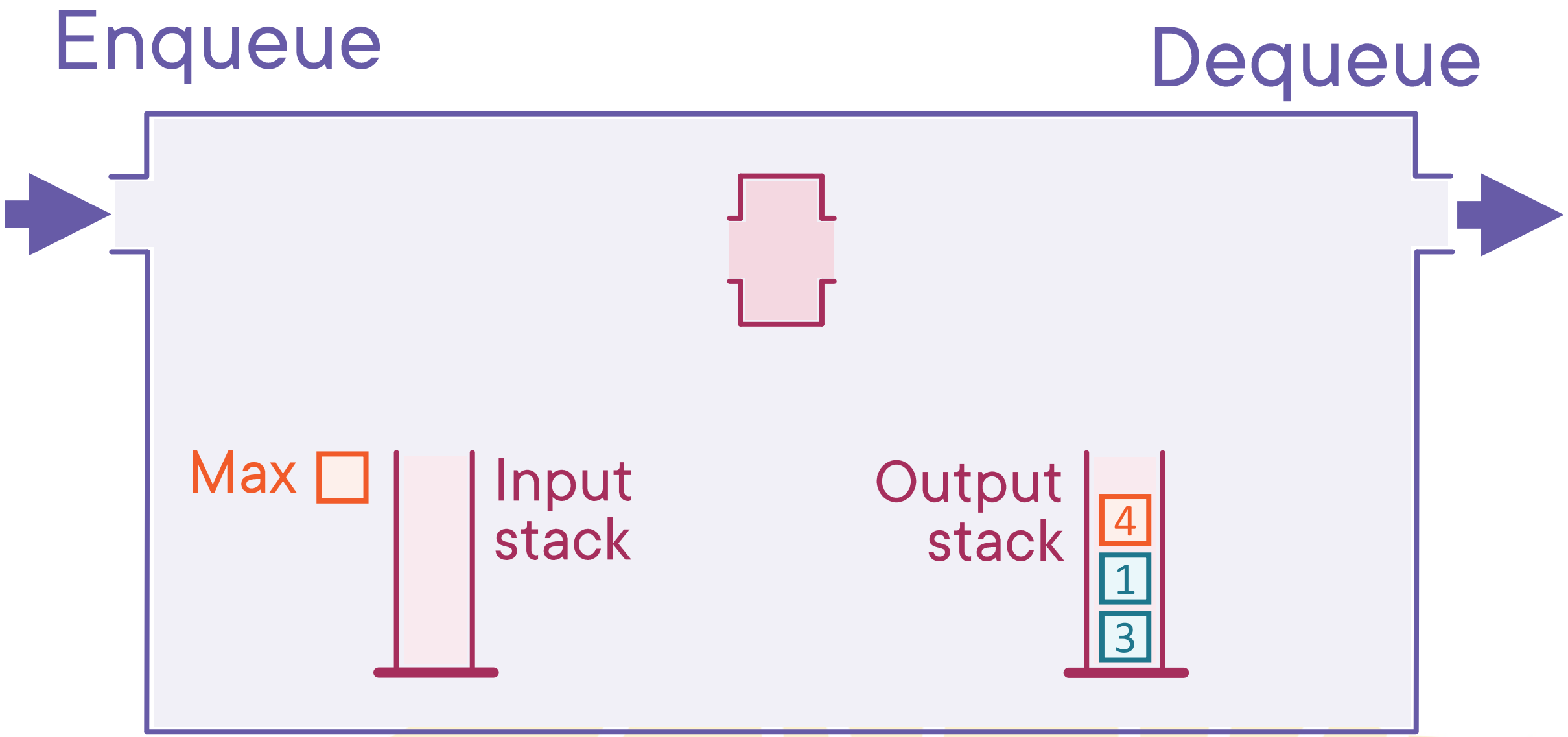

Enqueue
Dequeue
Max 4
Input stack
4
2
Output stack

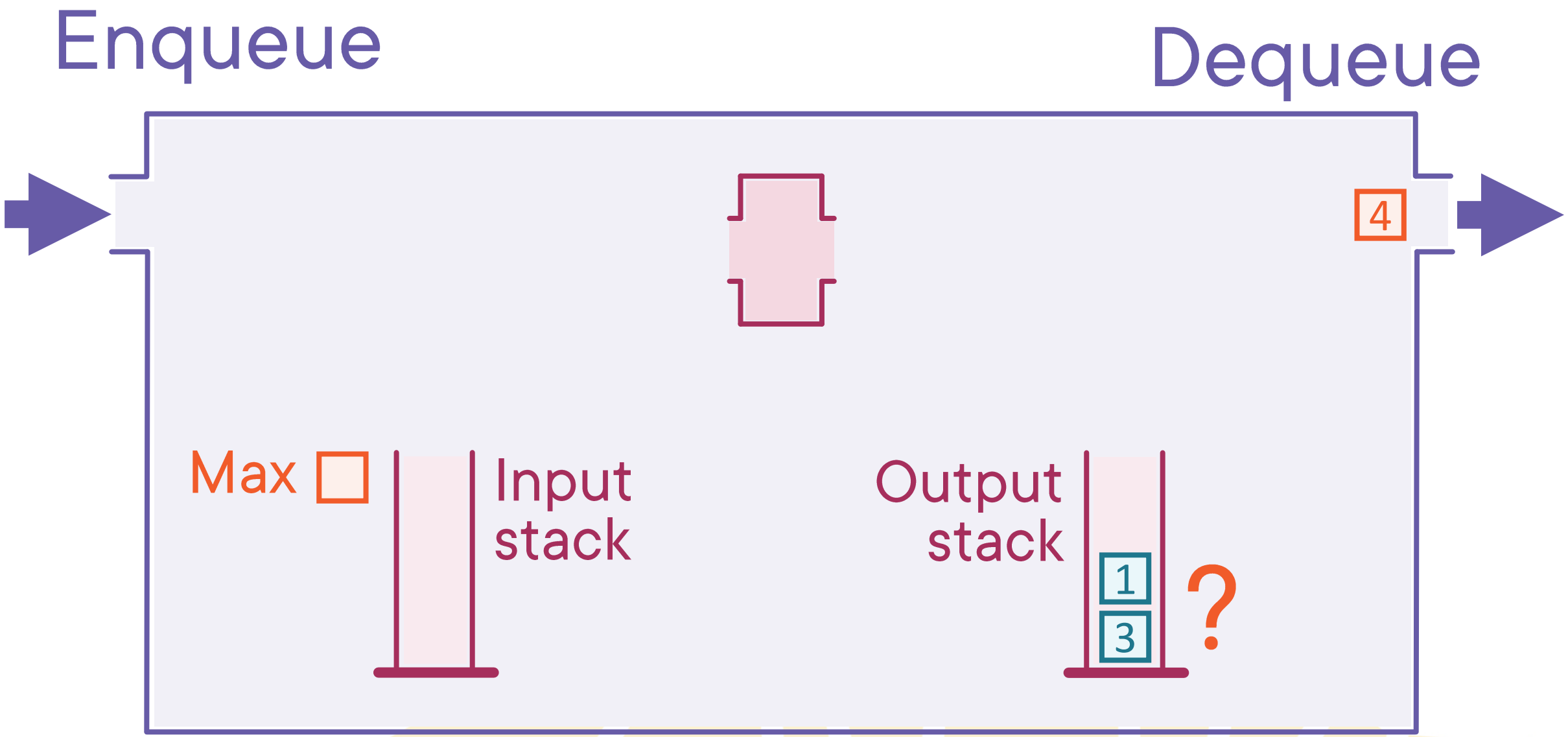

Enqueue
Dequeue
Max 4
1
4
2
Input stack
Output stack

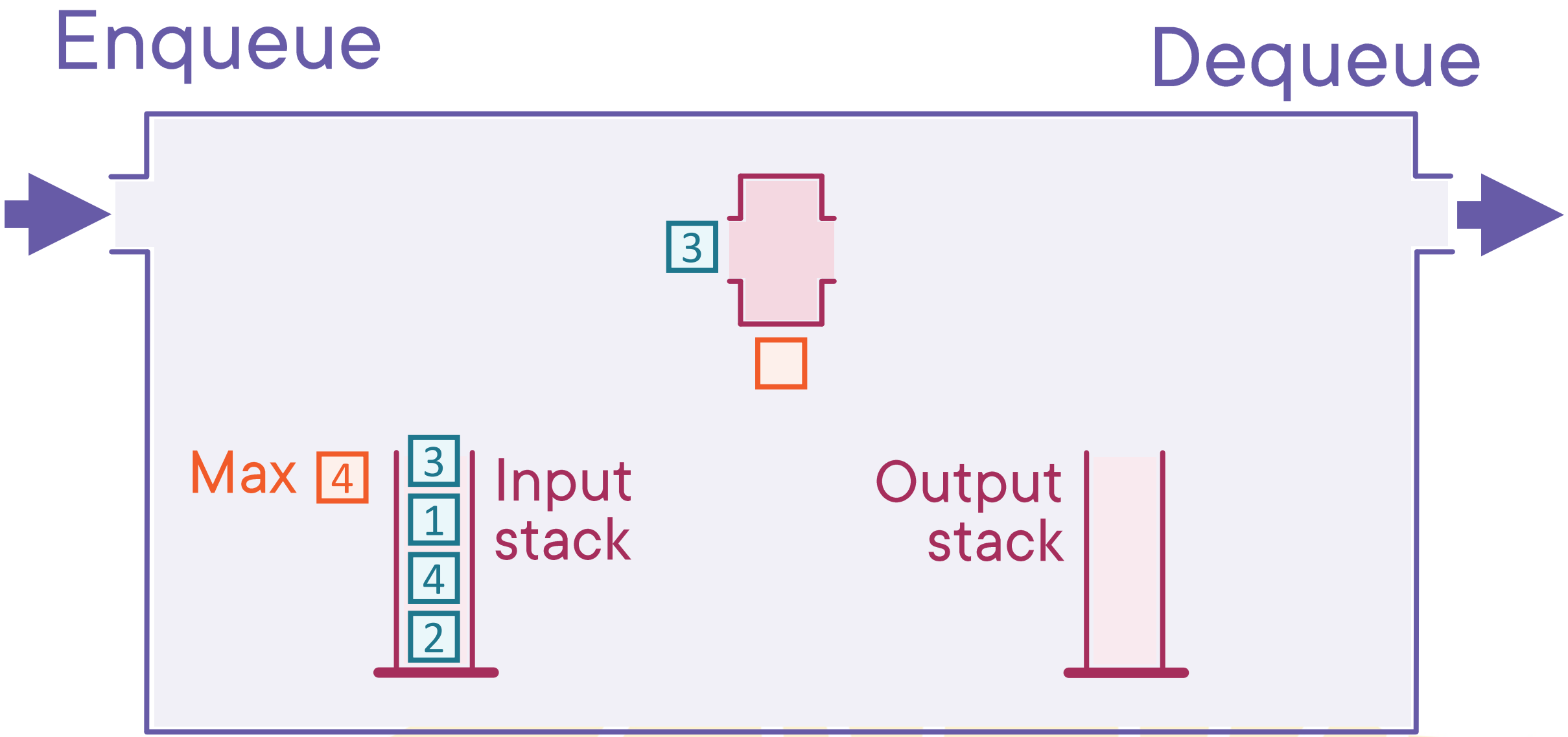

Enqueue
Dequeue
Max 4
3
1
4
2
Input stack
Output stack

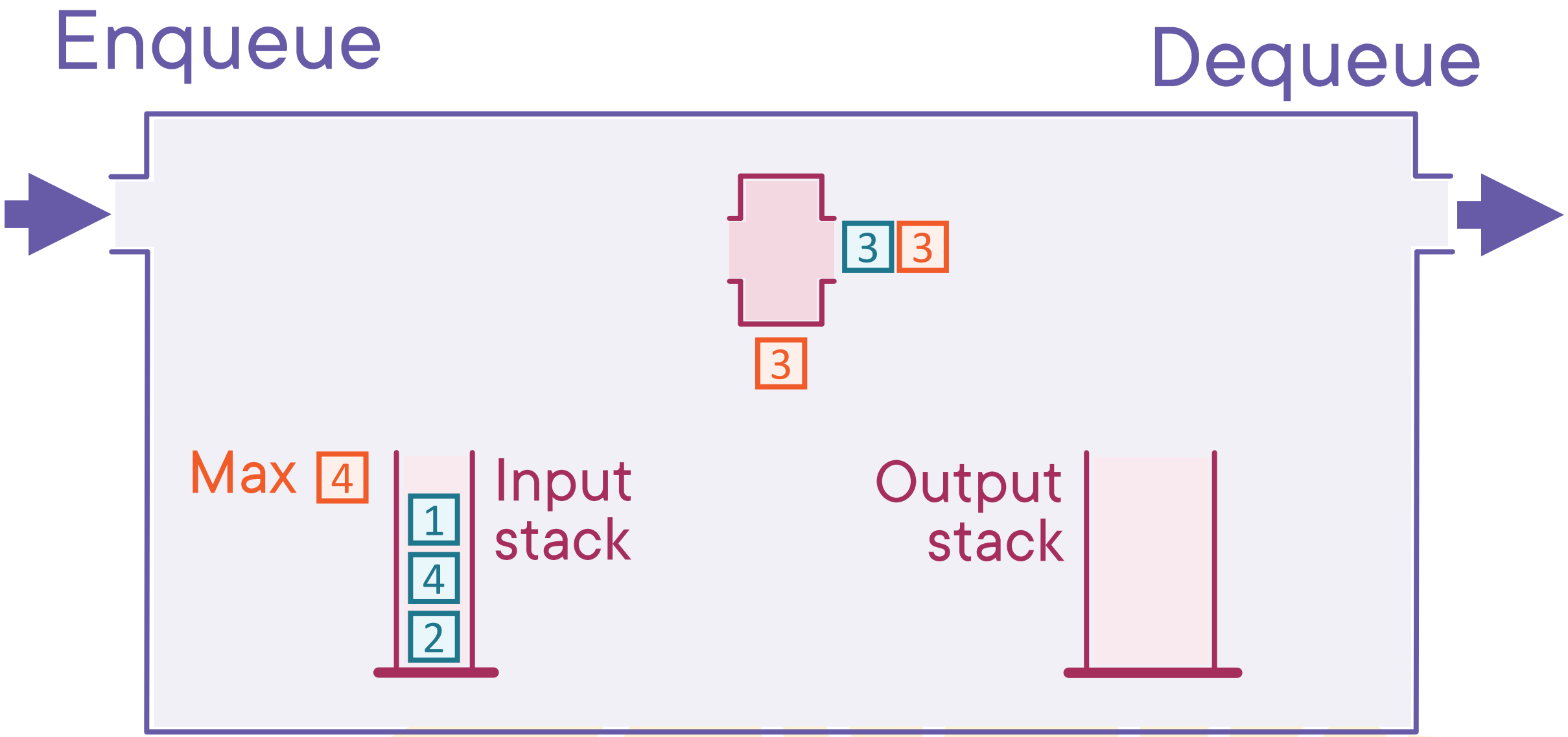

Enqueue
Dequeue
Max
Input stack
Output stack
2
4
1
3

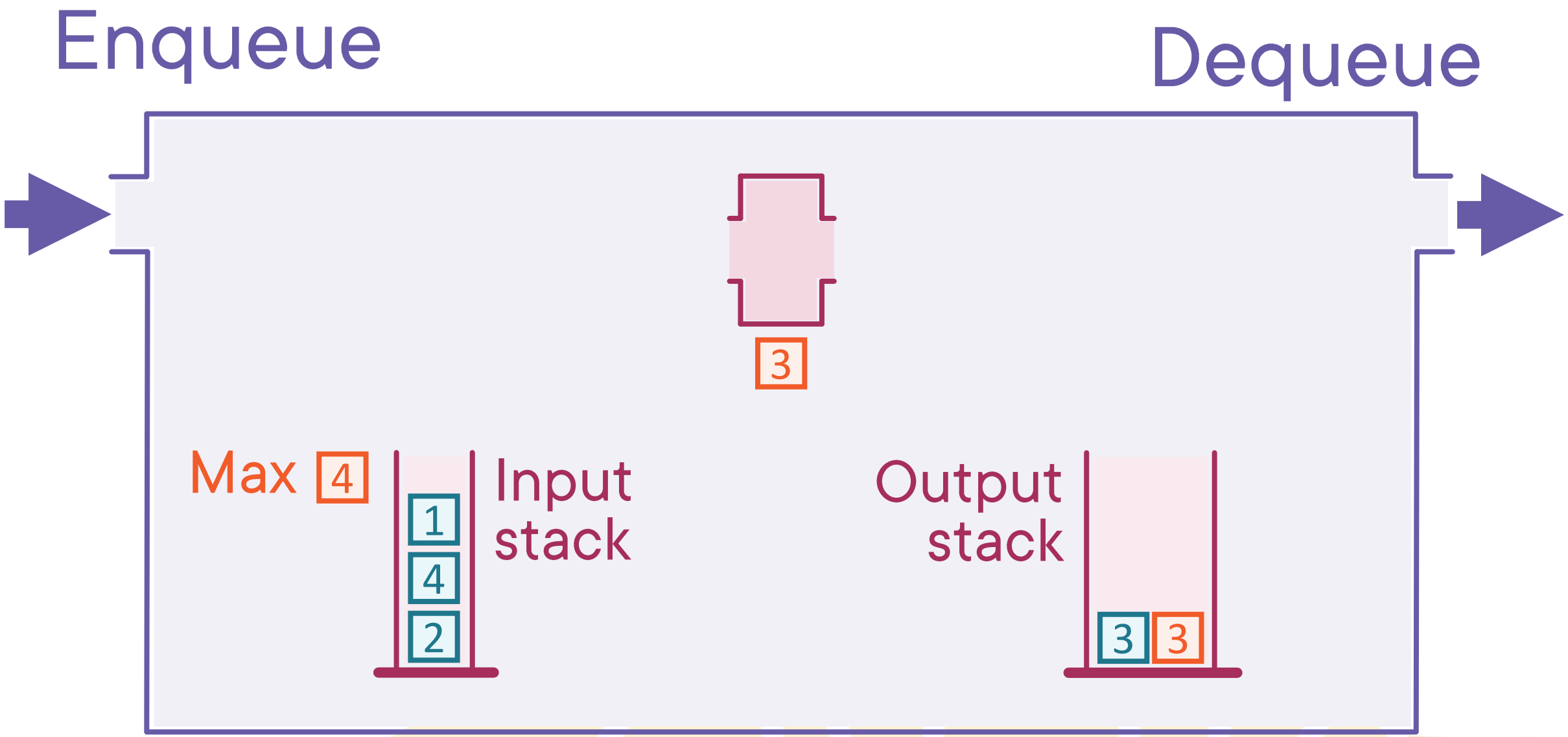

Enqueue
Dequeue
2
Max
Input
stack
Output
stack
4
1
3

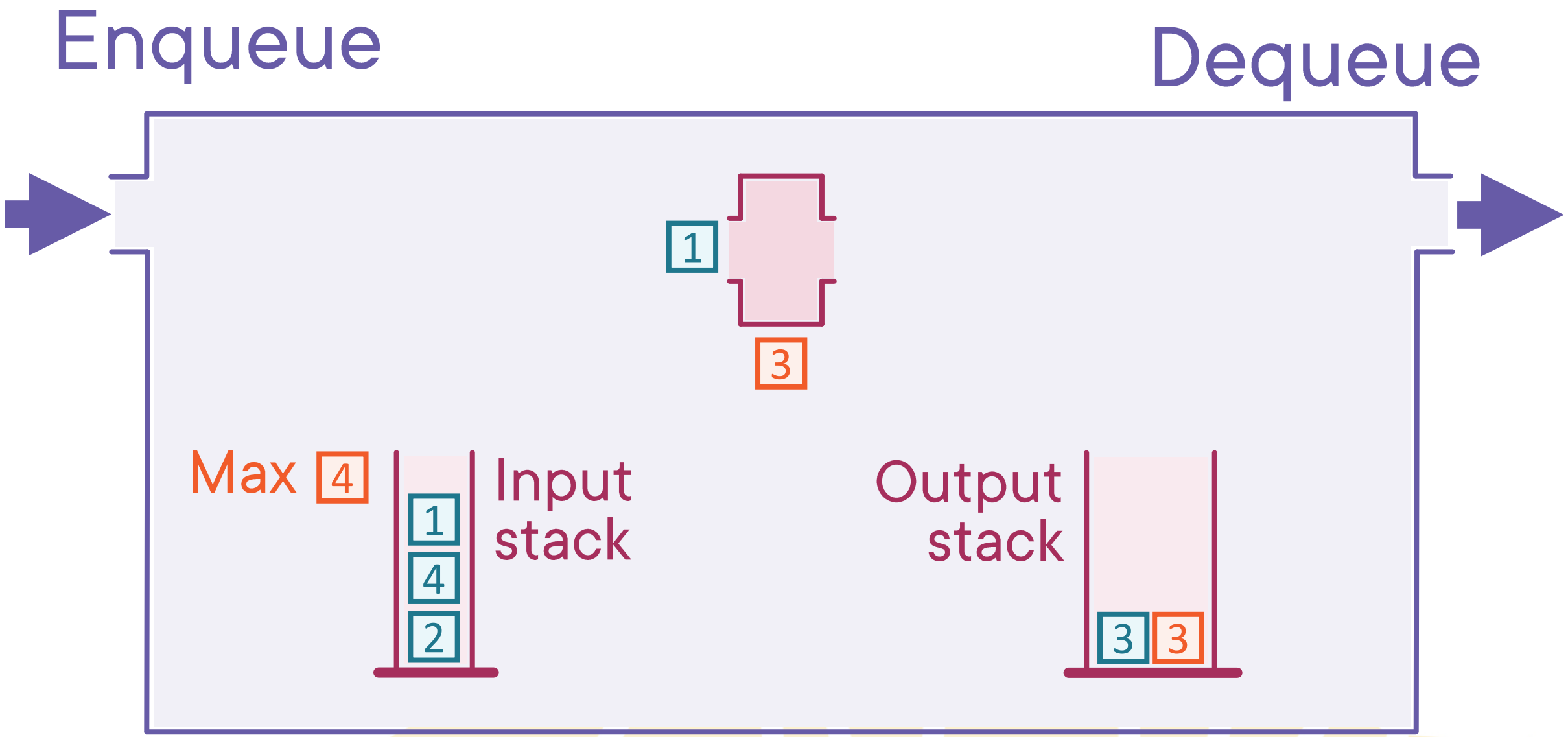

Enqueue
Dequeue
Max
Input stack
Output stack
4
1
3

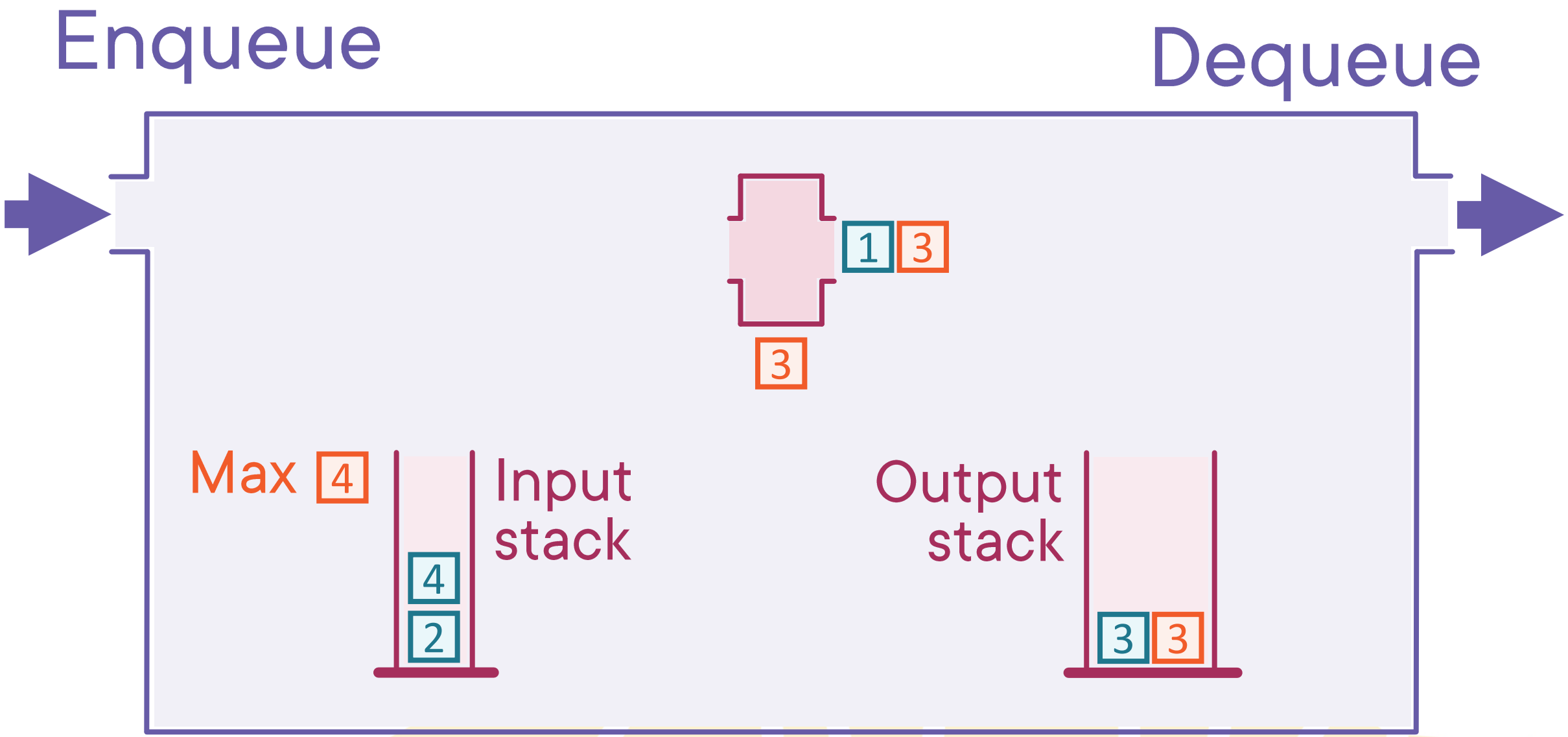

Enqueue
Dequeue
4
Max
Input
stack
Output
stack
1
3
?

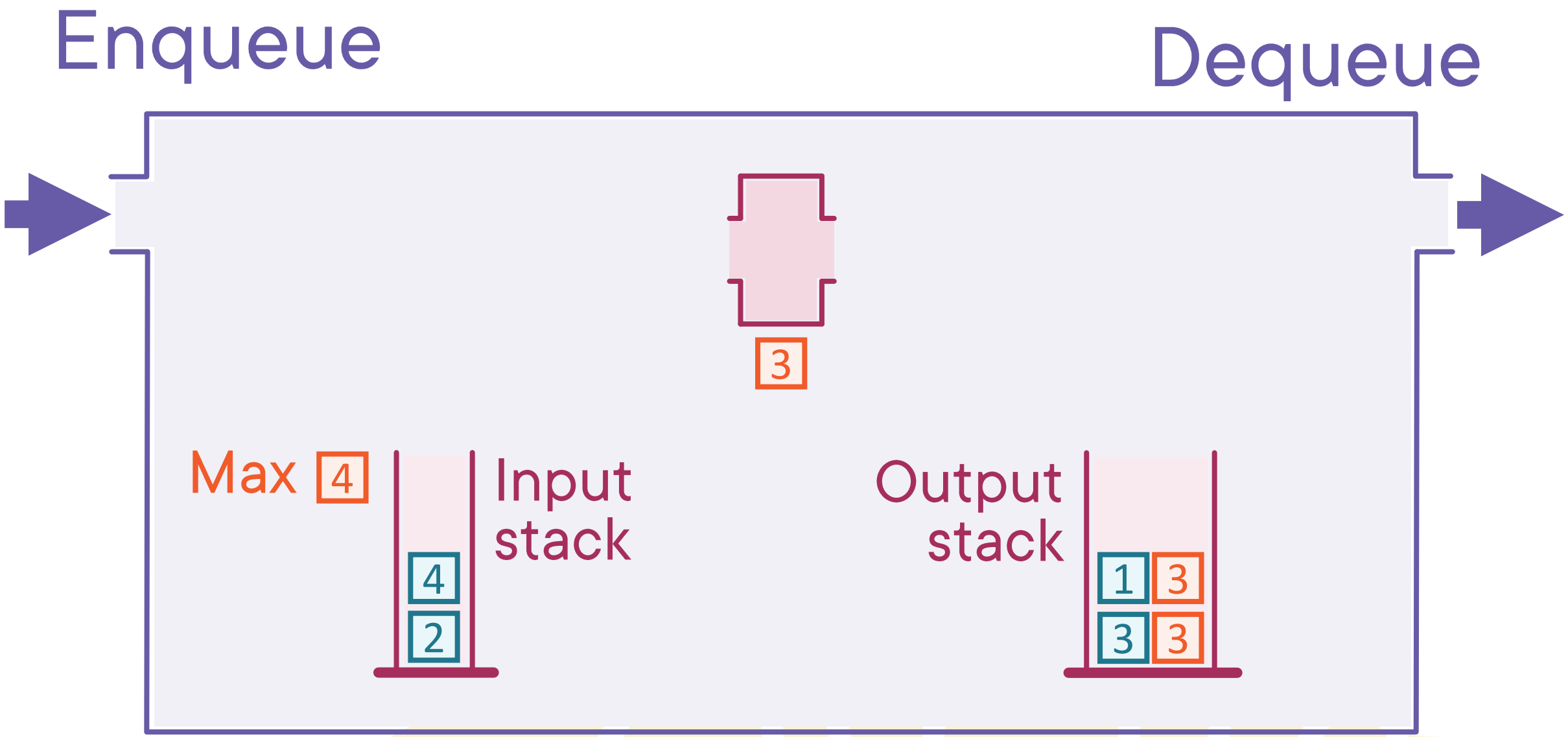

Enqueue
Dequeue
3
Max
4
3
1
4
2
Input
stack
Output
stack

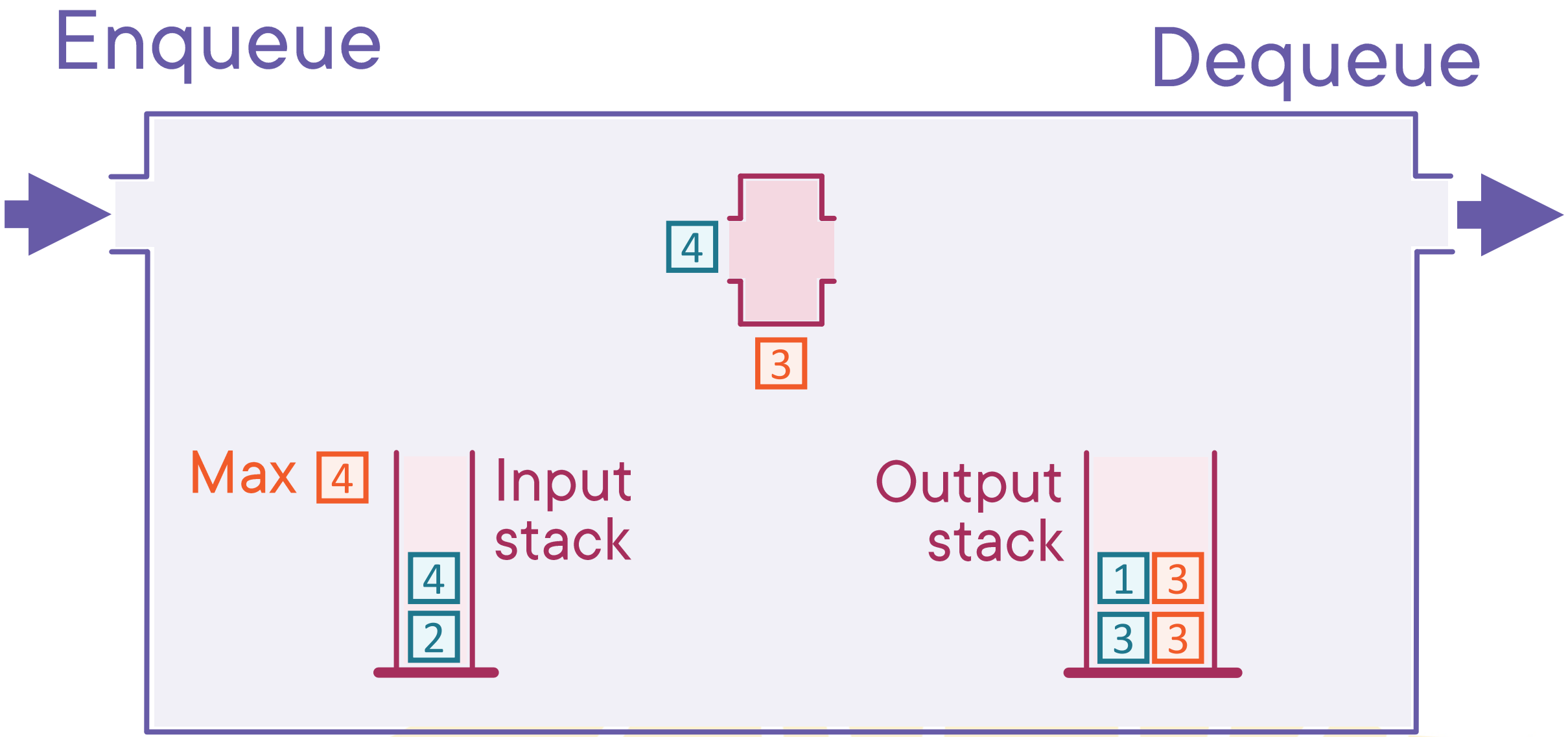

Enqueue
Dequeue
3
3
3
Max
4
Input
stack
1
4
2
Output
stack

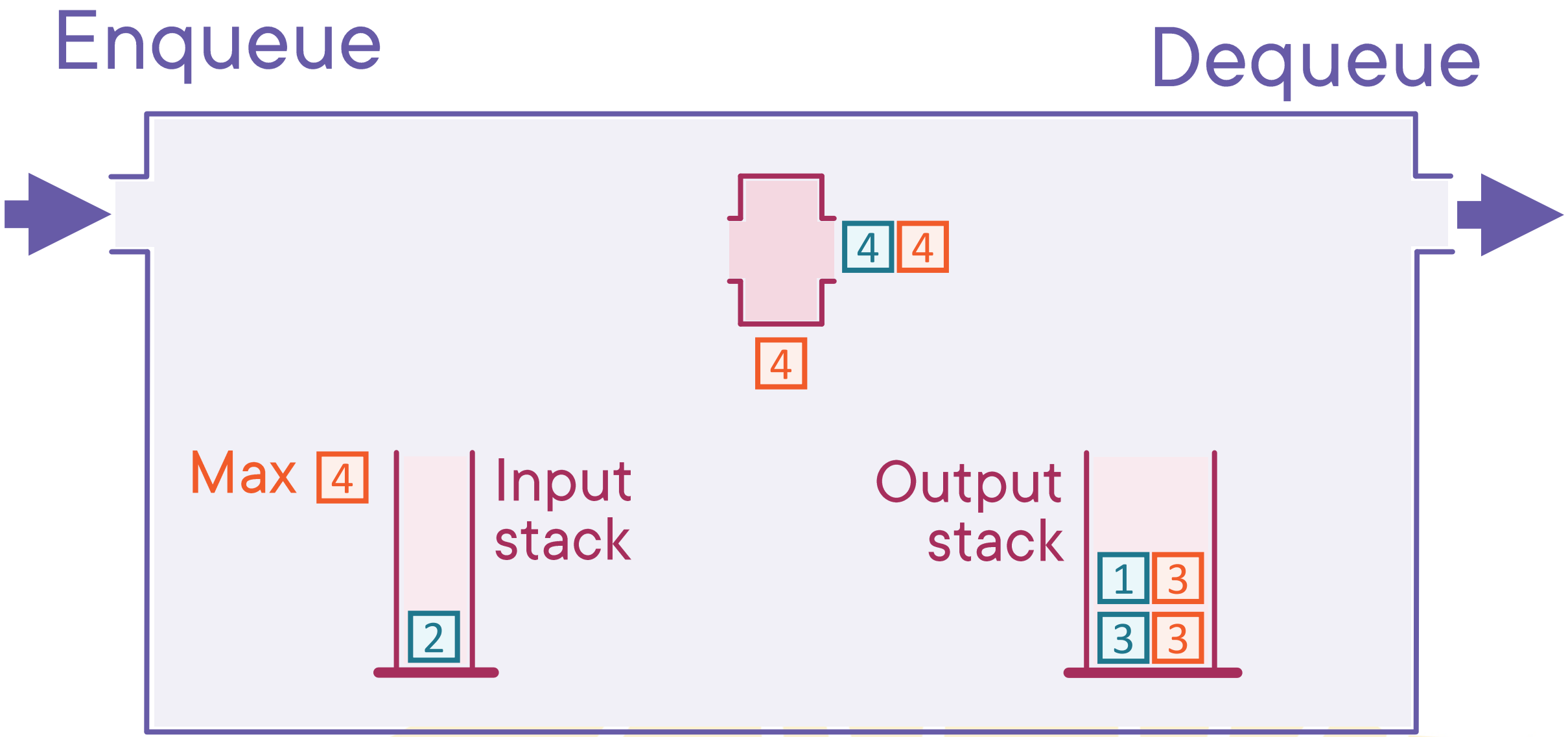

Enqueue
Dequeue
3
Max 4
1
4
2
Input stack
Output stack
3 3

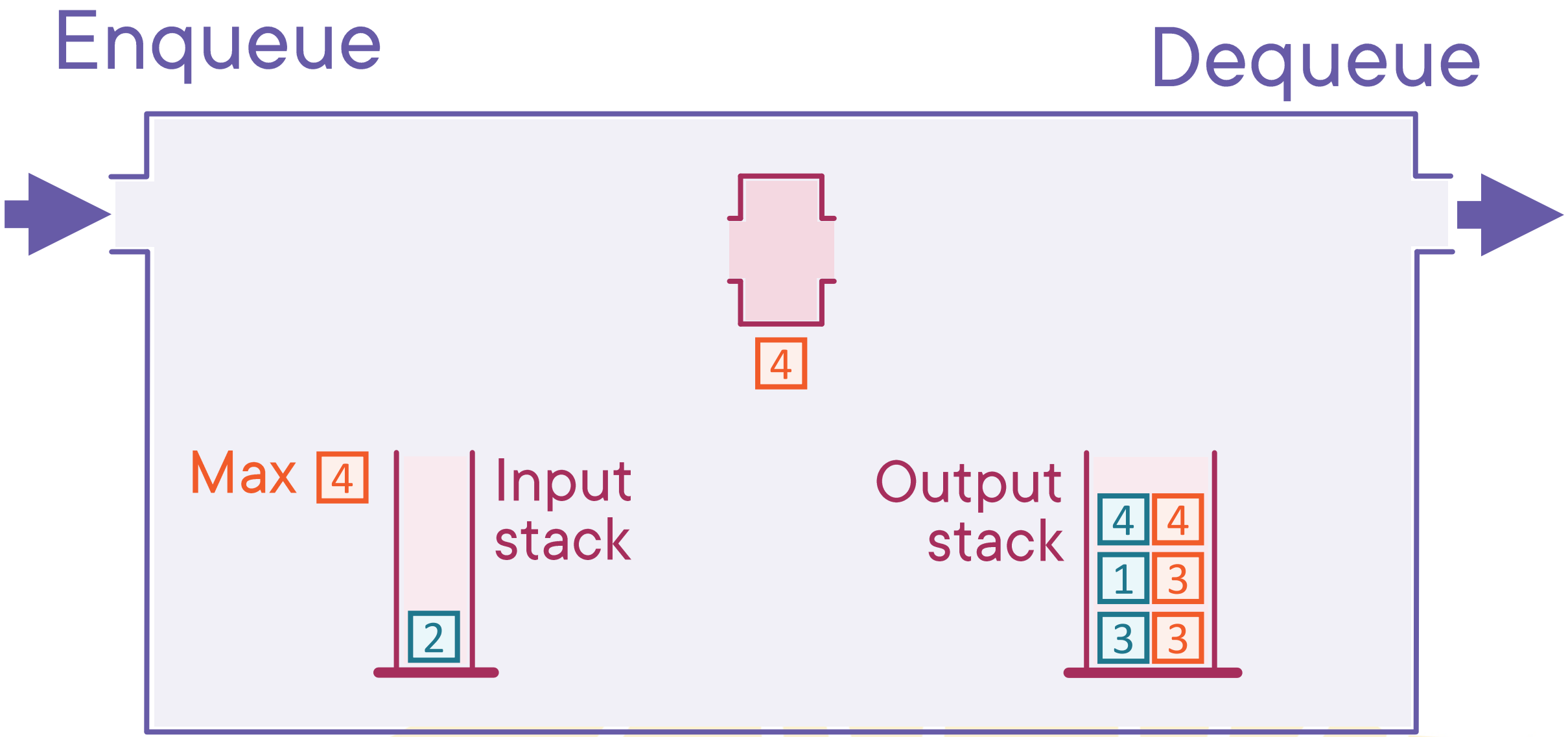

Enqueue
Dequeue
1
3
Max
4
Input
stack
1
4
2
Output
stack
3
3

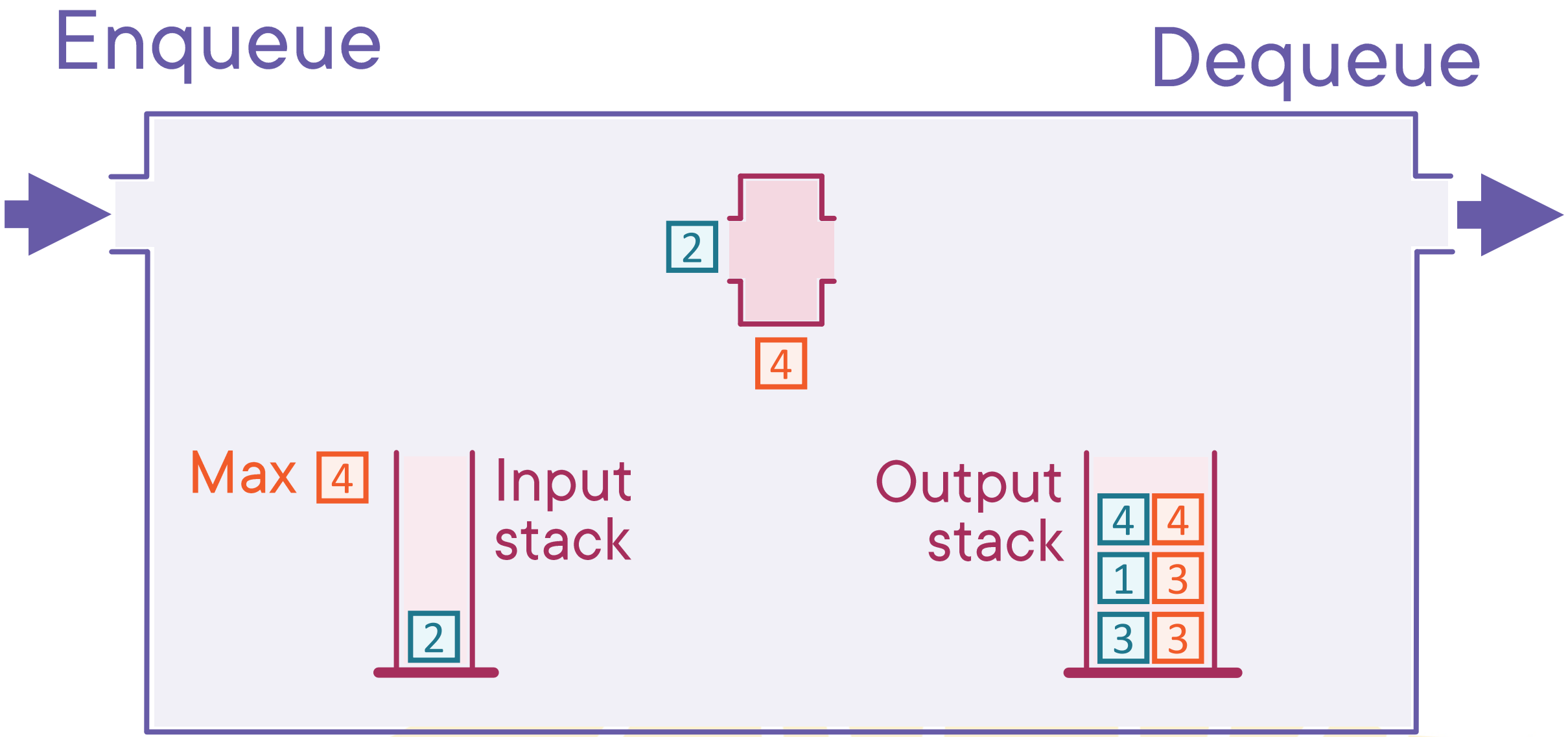

Enqueue
Dequeue
1
3
3
Max
4
Input
stack
4
2
Output
stack
3
3

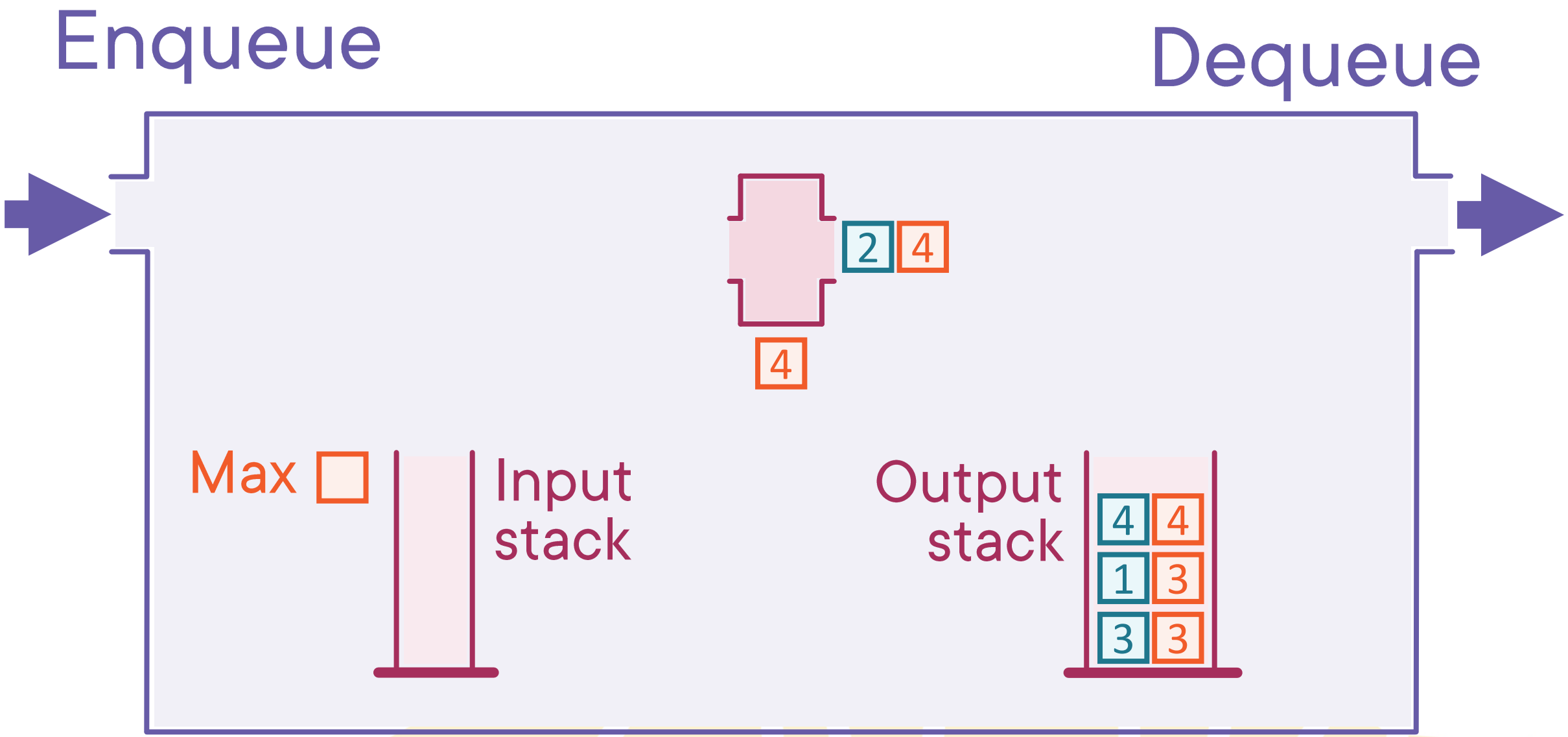

Enqueue
Dequeue
3
Max 4
Input stack
4
2
Output stack
1
3
3
3

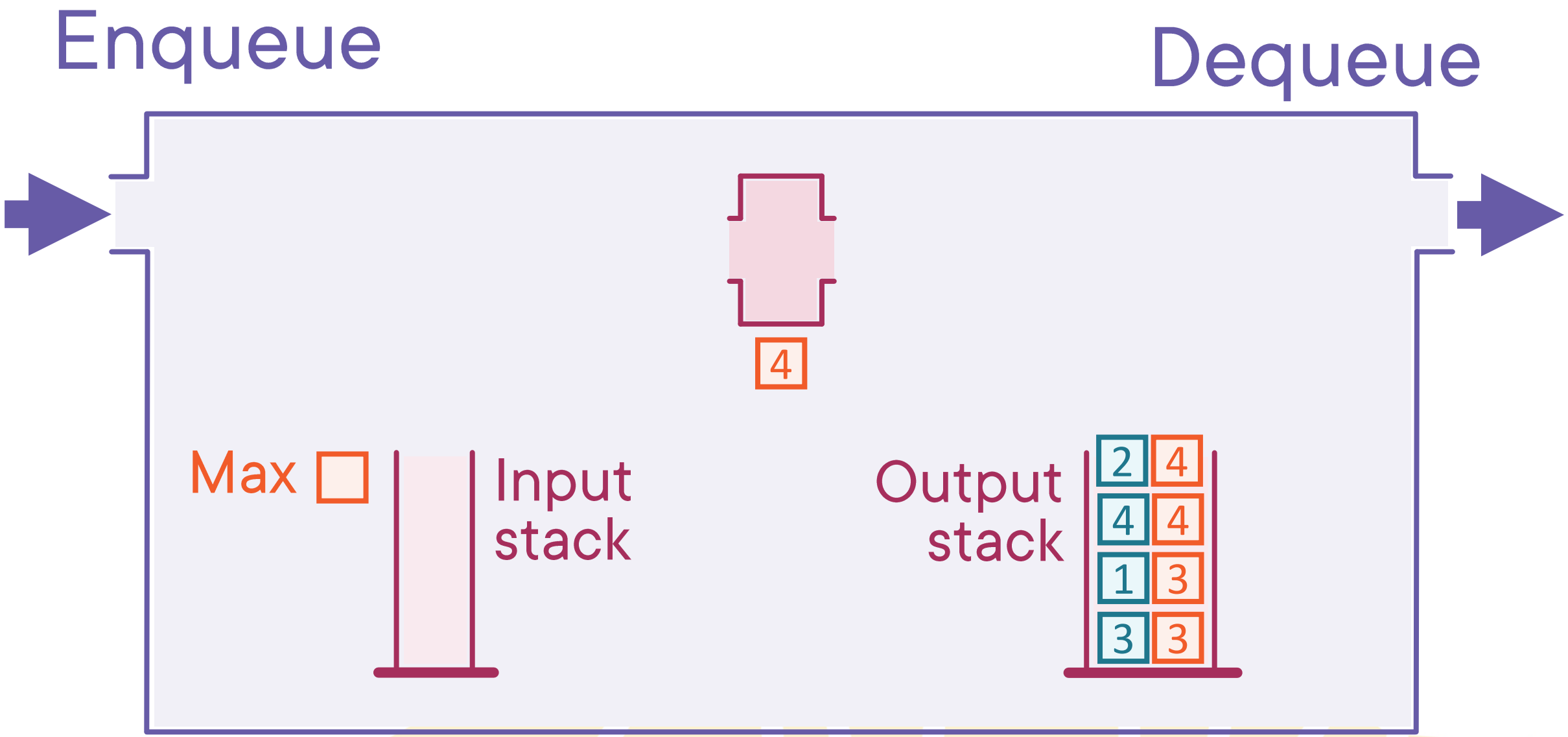

Enqueue
Dequeue
4
3
Max 4
Input stack
4
2
Output stack
1 3
3 3

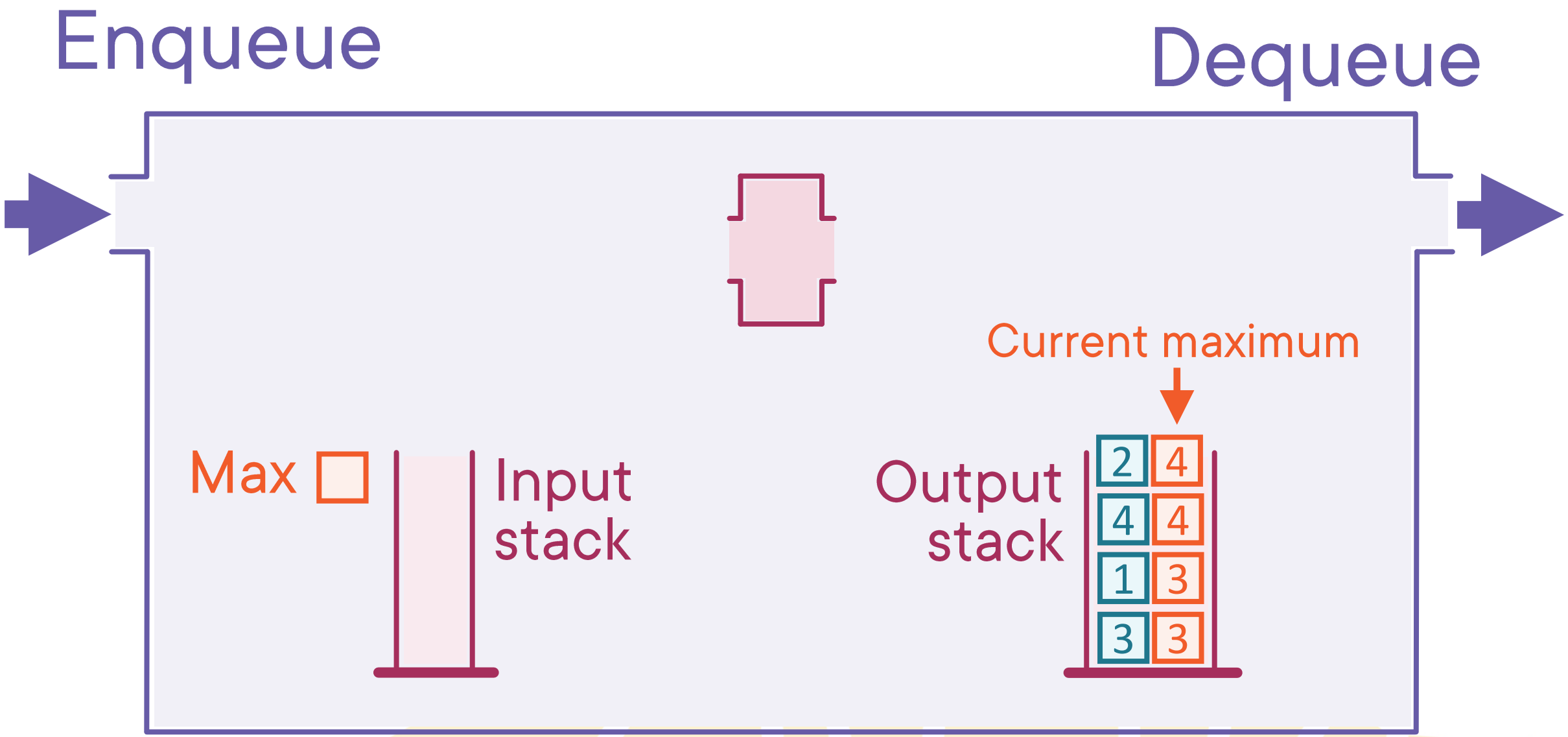

Enqueue
Dequeue
4
4
4
Max
4
Input
stack
2
Output
stack
1
3
3
3

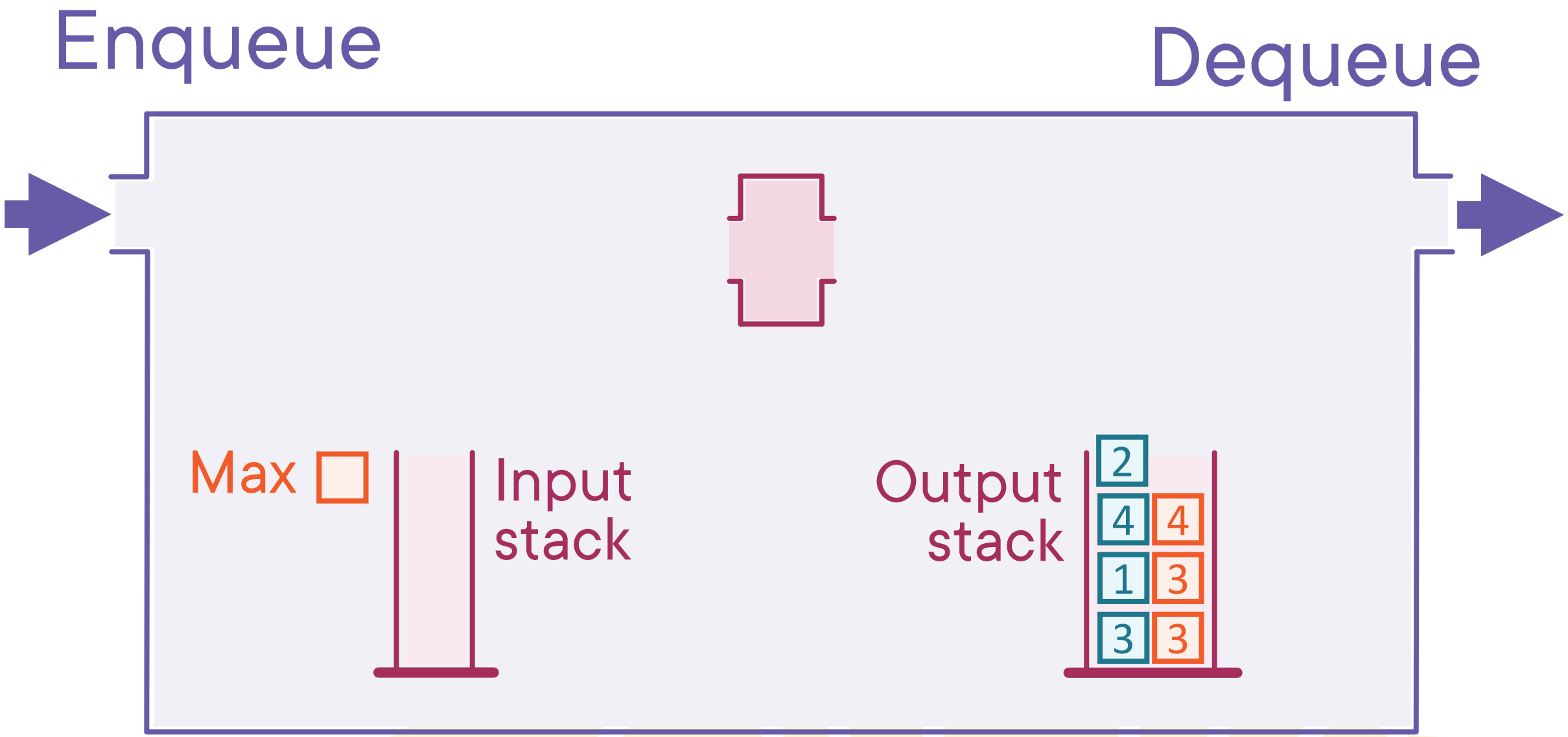

Enqueue
Dequeue
4
Max 4
Input stack
2
Output stack
4 4
1 3
3 3

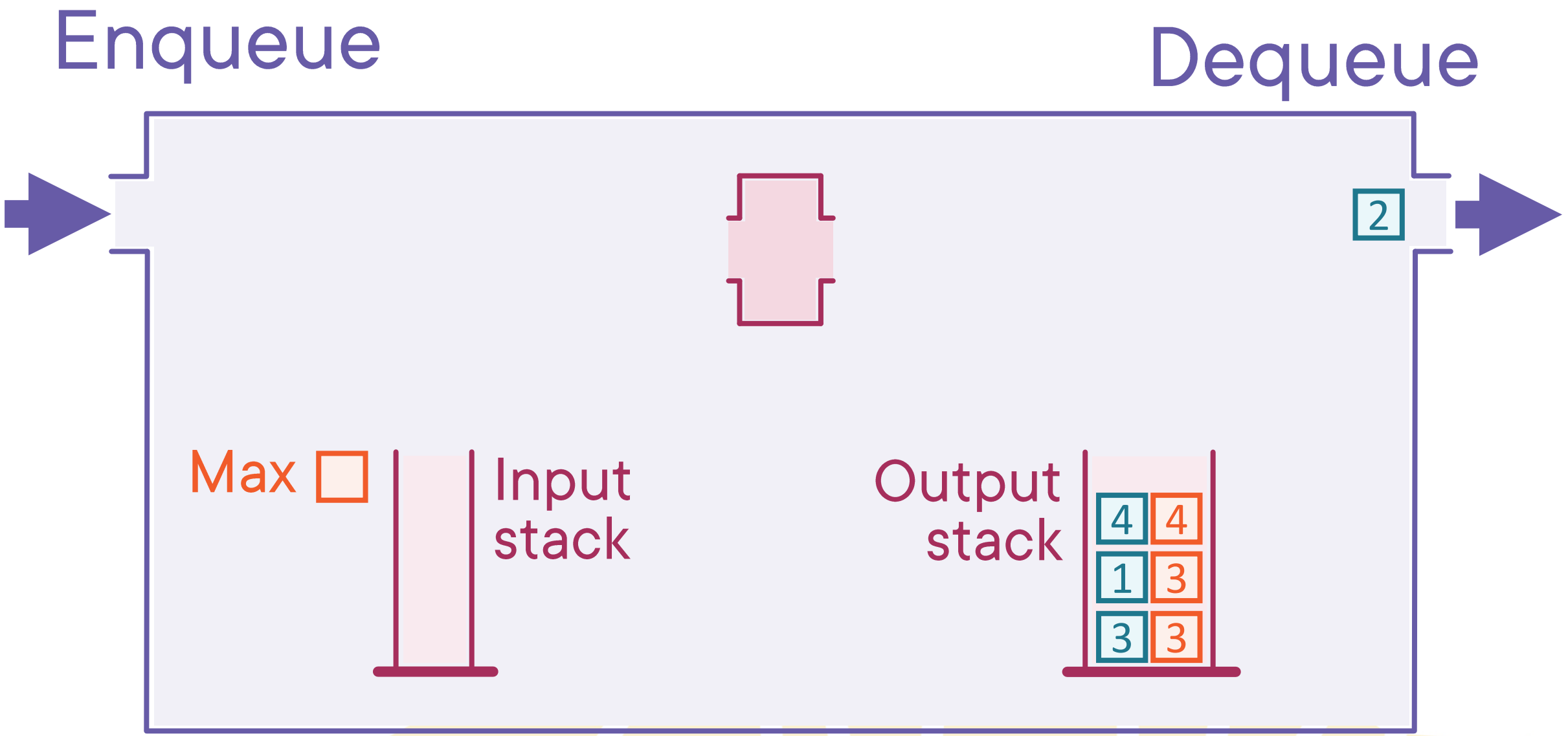

Enqueue
Dequeue
2
4
Max
4
Input
stack
2
Output
stack
4
4
1
3
3
3

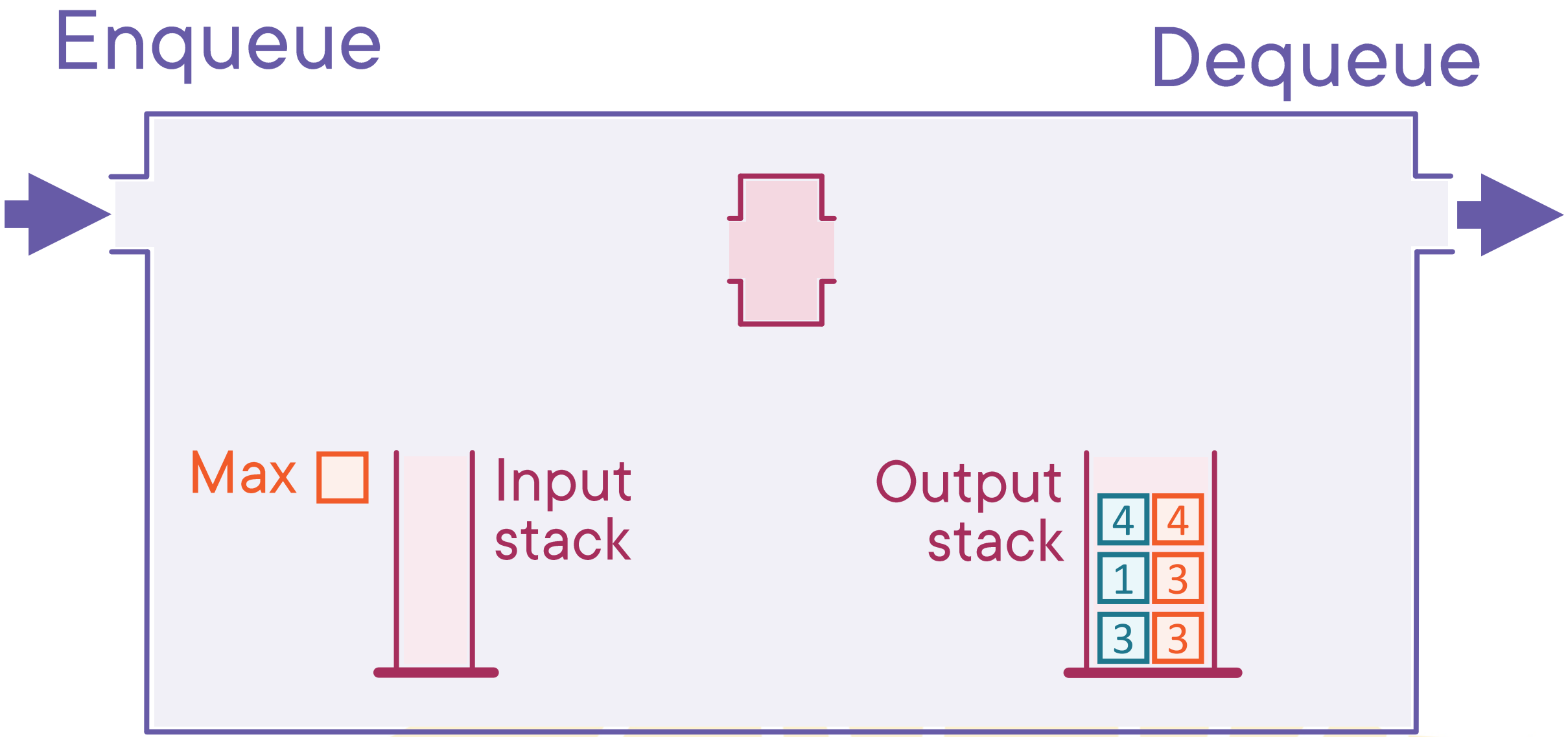

Enqueue
Dequeue
2
4
4
Max
Input
stack
Output
stack
4
4
1
3
3
3

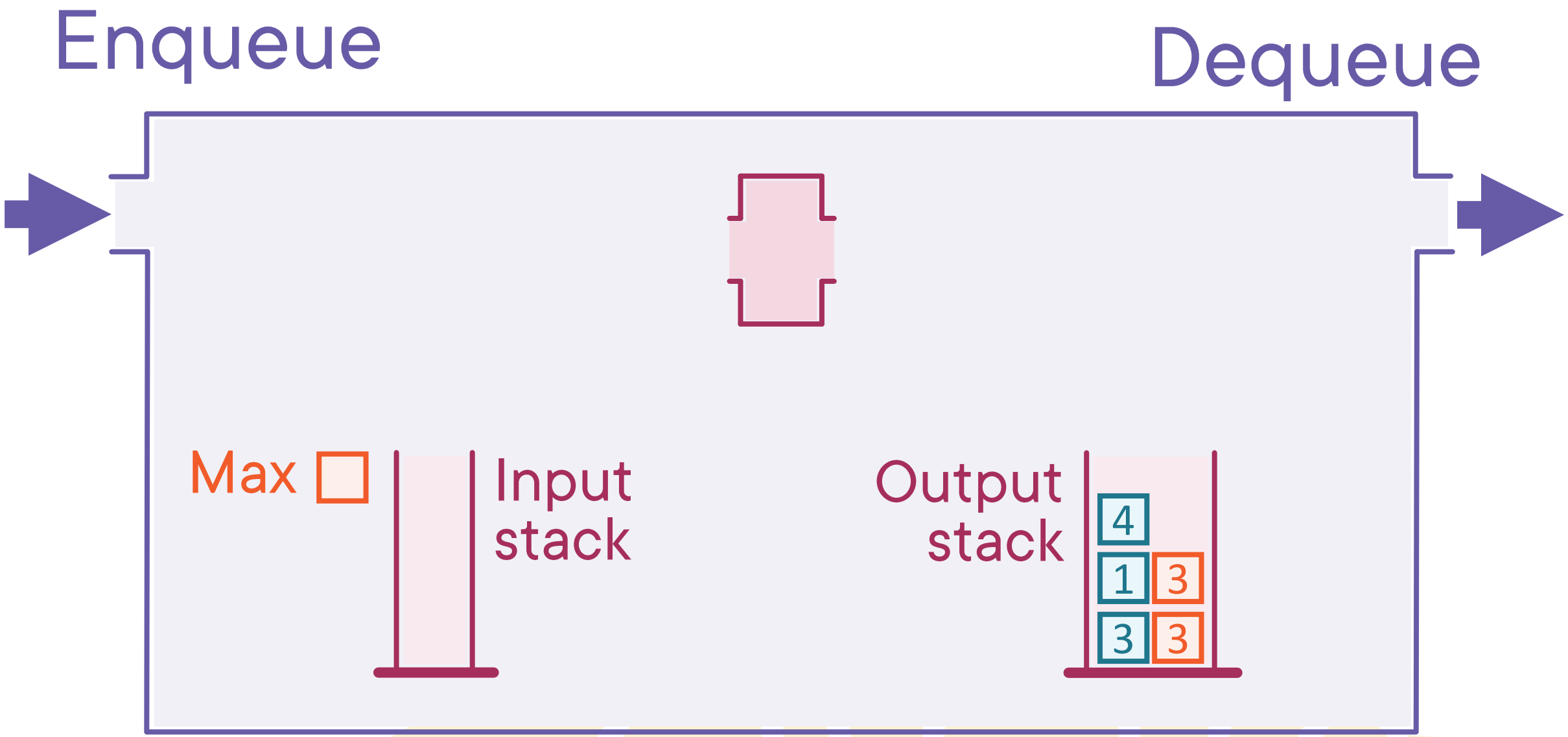

Enqueue
Dequeue
4
Max
Input stack
Output stack
2 4
4 4
1 3
3 3

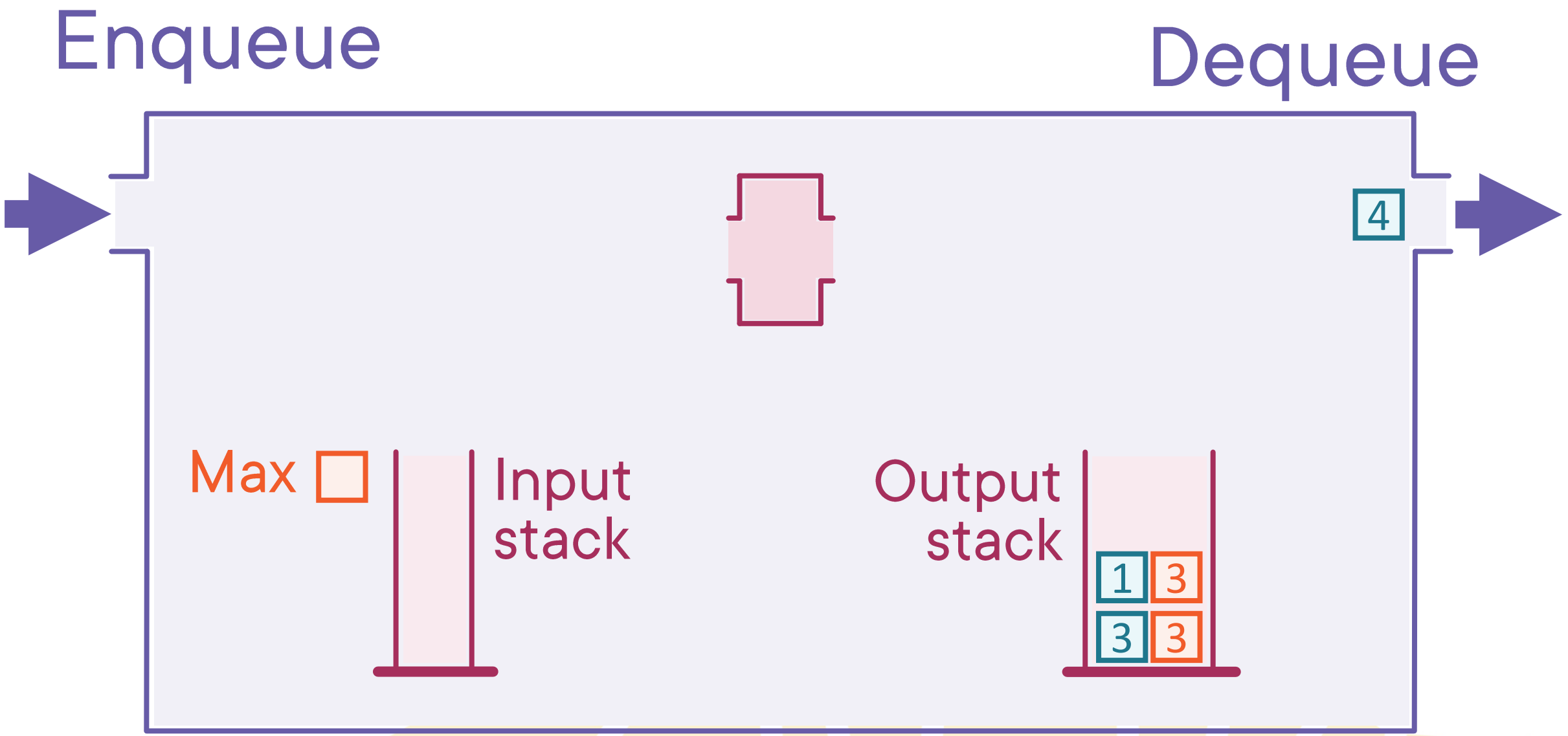

Enqueue
Dequeue
Current maximum
Max
Input stack
Output stack
2 4
4 4
1 3
3 3

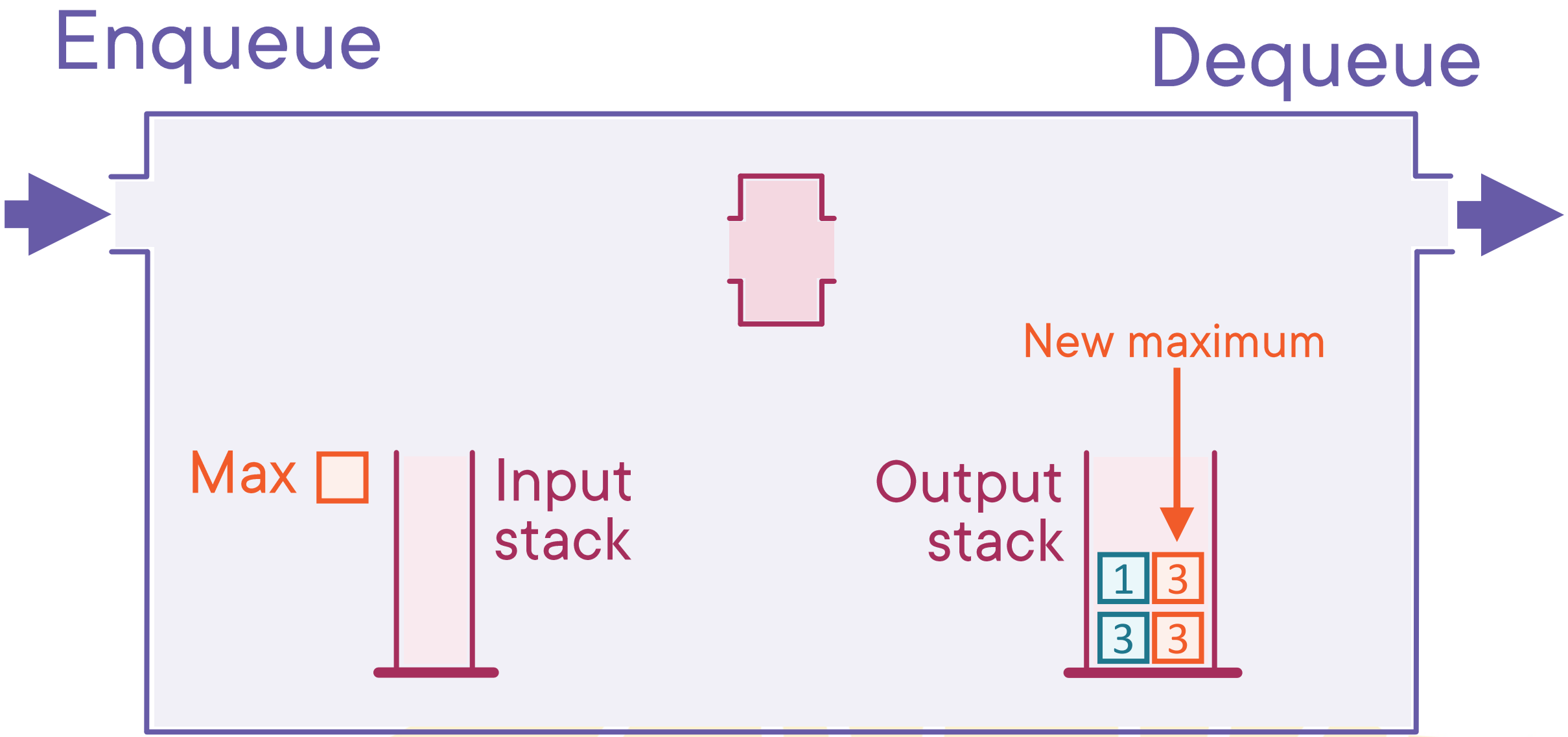

Enqueue
Dequeue
Max
Input stack
Output stack
2
4
4
1
3
3
3

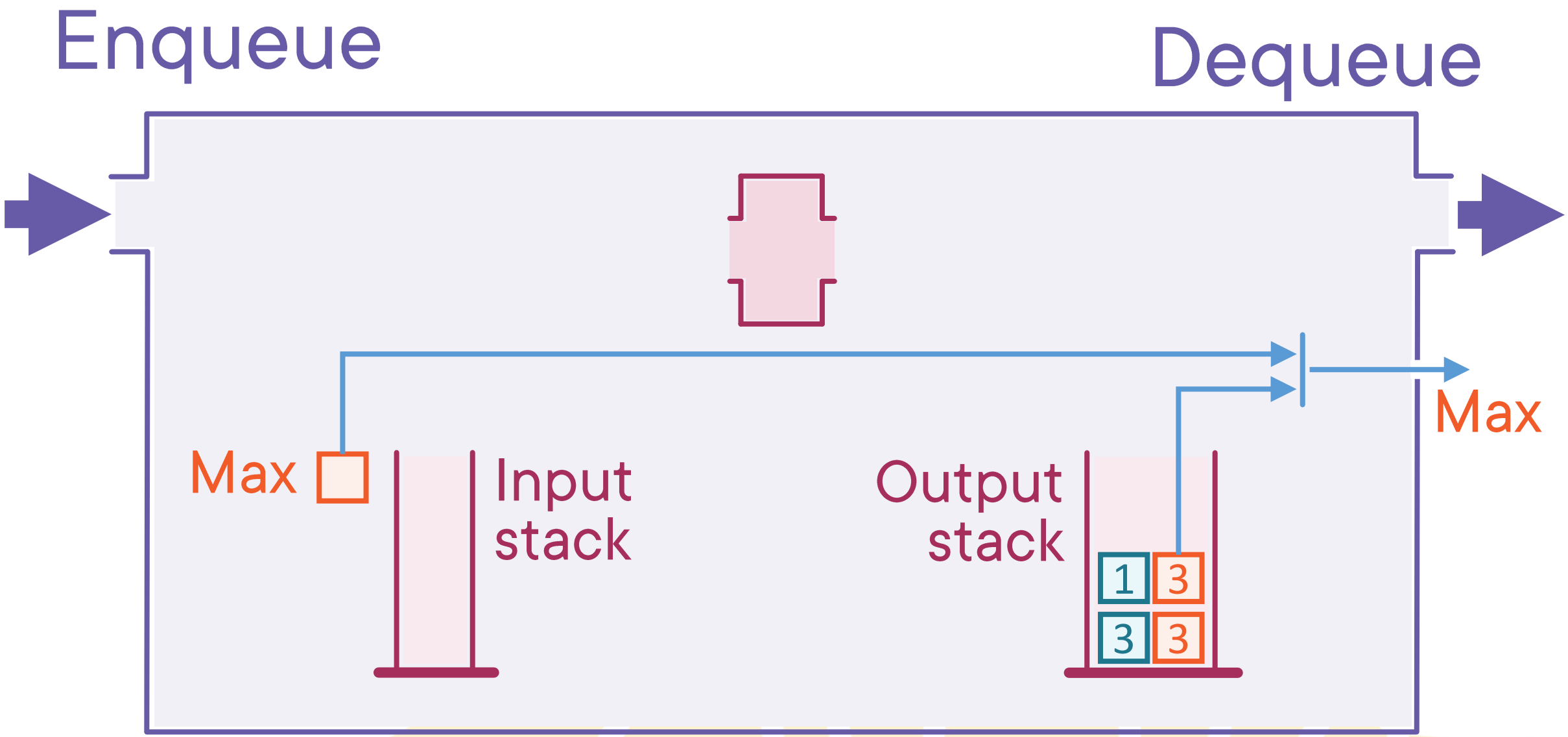

Enqueue
Dequeue
2
Max
Input stack
Output stack
4
4
1
3
3
3


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

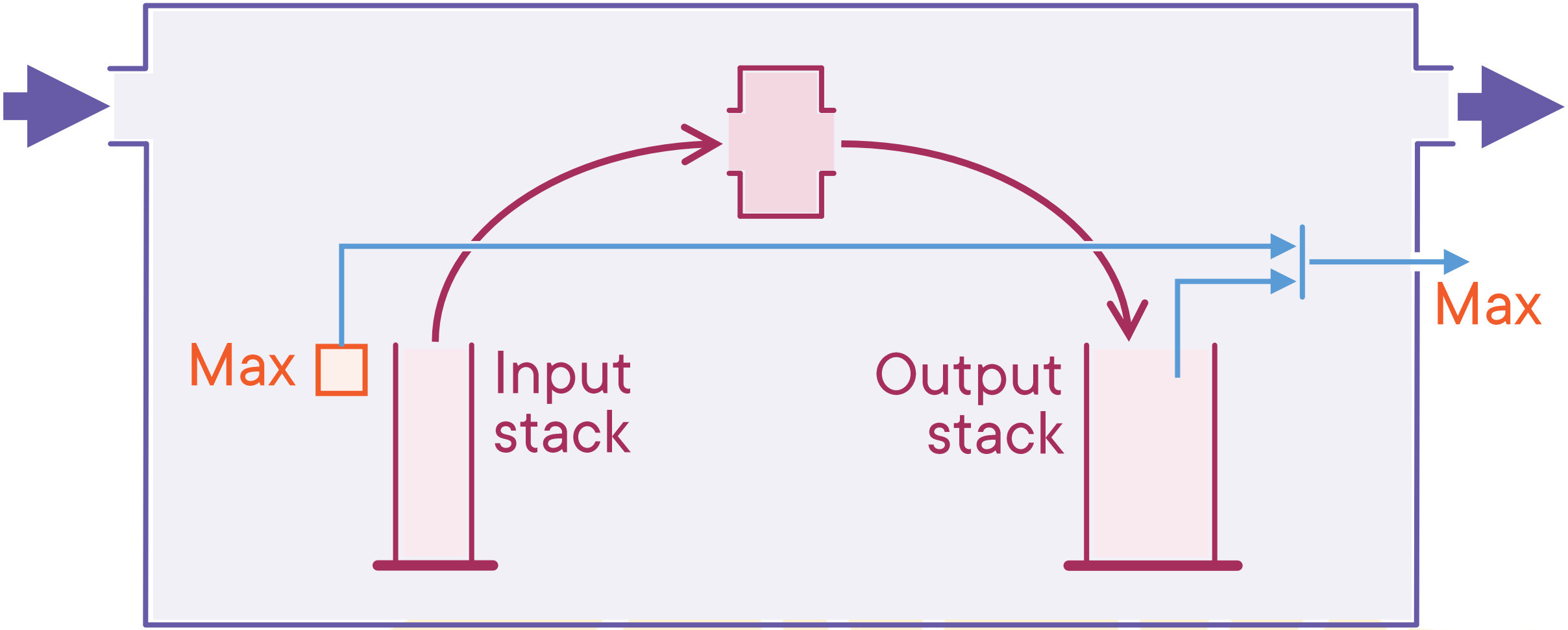

Enqueue
Dequeue
Max
Input stack
Output stack
4
1
3
3
3


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


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

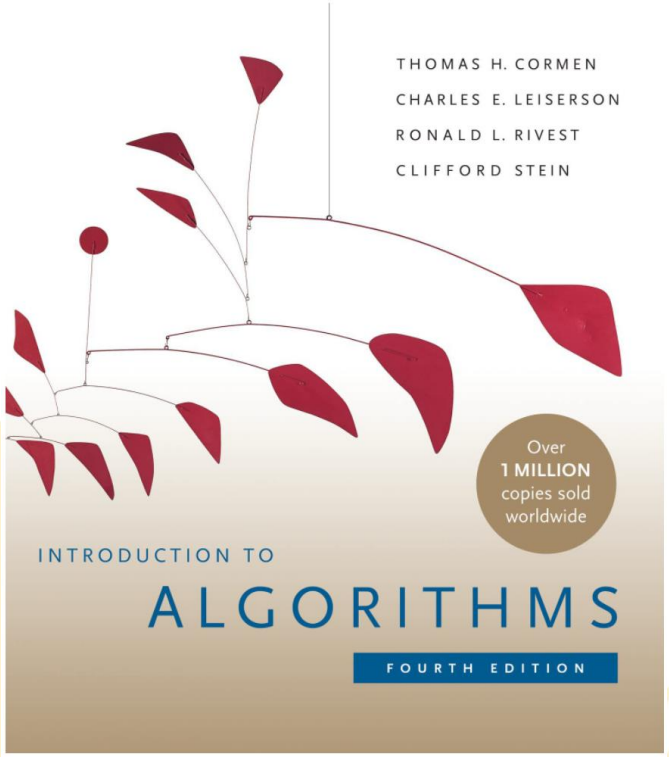

Enqueue
Dequeue
Max
Input
stack
Output
stack
Max
1
3
3
3

A maximum queue



Dalje učenje

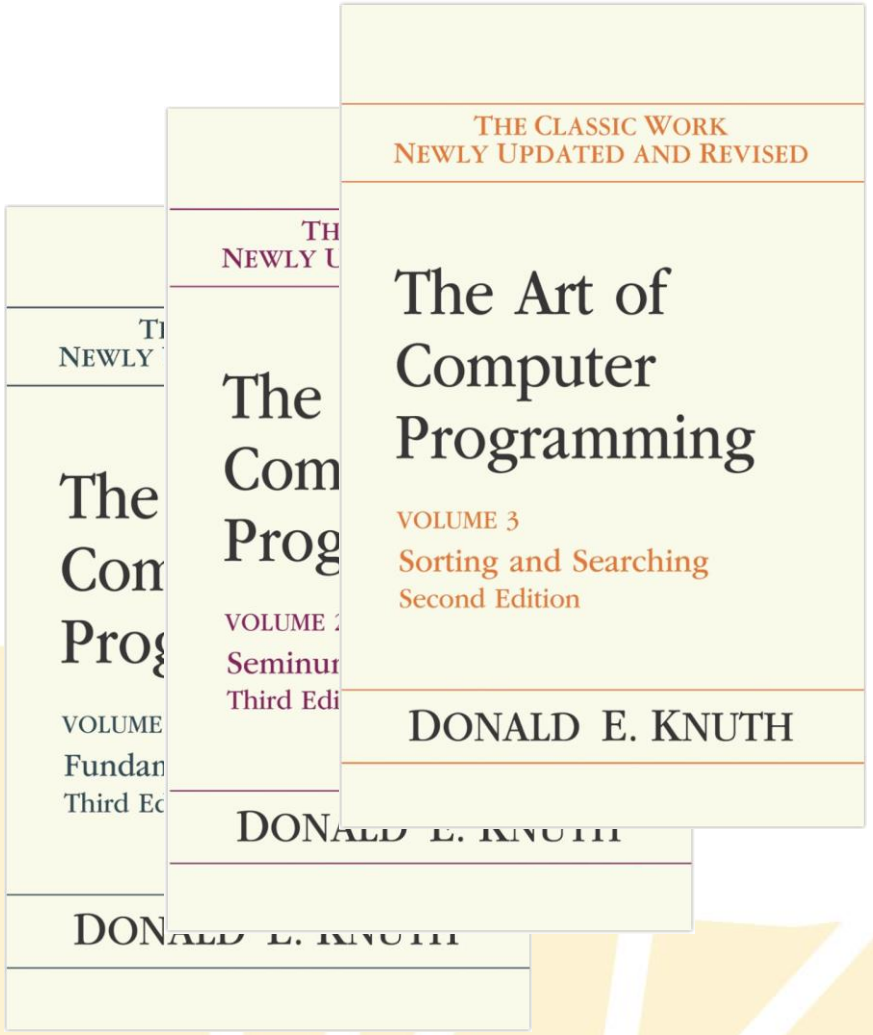
Cormen et al.,
Introduction to Algorithms



Dalje učenje

Cormen et al.,
Introduction to Algorithms

Donald Knuth,
The Art of Computer Programming



Dalje učenje

Coding Interviews and Snake (the game) Have This One Thing in Common

https://blog.pramp.com/coding-interviews-and-the-snake-game-have-this-one-thing-in-common-e0189fba1c9c

Dalje učenje

Pluralsight:
Collections and Generics in C# 10

https://codinghelmet.com/go/collections-and-generics-in-cs


#GEEKSTONE
THANK YOU!

SUPORTED BY:

VEGA IT