



<http://algs4.cs.princeton.edu>

## 1.3 DIJKSTRA'S 2-STACK DEMO

---

# Dijkstra's two-stack algorithm

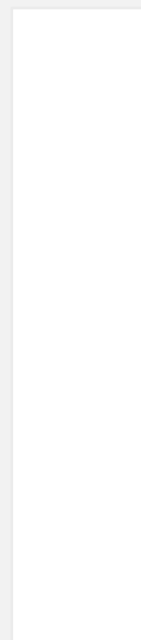
---

**Value:** push onto the value stack.

**Operator:** push onto the operator stack.

**Left parenthesis:** ignore.

**Right parenthesis:** pop operator and two values; push the result of applying that operator to those values onto the value stack.



value stack



operator stack

**infix expression**

**(fully parenthesized)**



operand

operator

# Dijkstra's two-stack algorithm

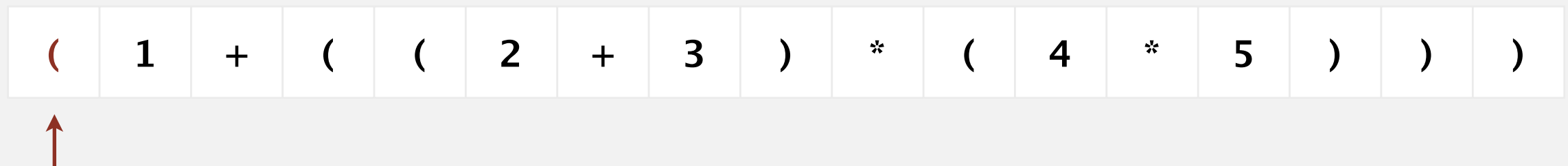
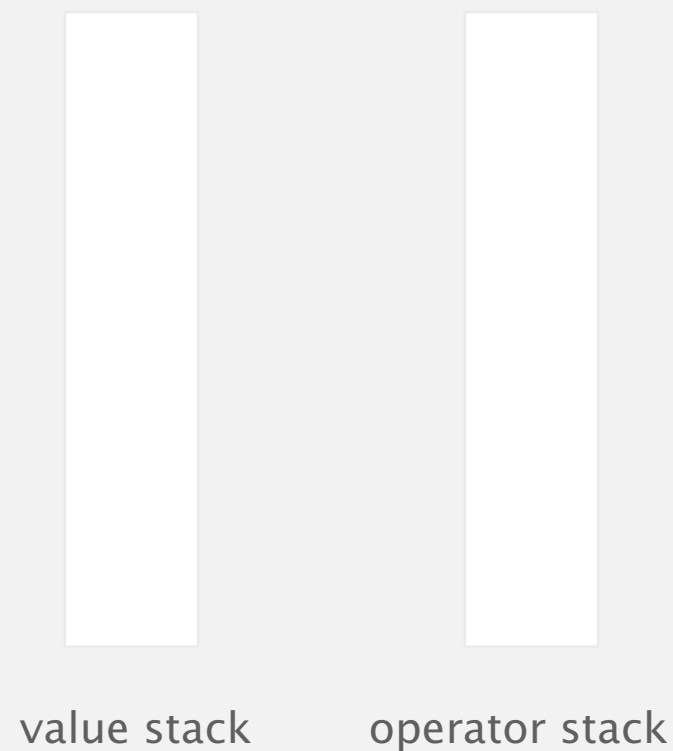
---

**Value:** push onto the value stack.

**Operator:** push onto the operator stack.

**Left parenthesis:** ignore.

**Right parenthesis:** pop operator and two values; push the result of applying that operator to those values onto the value stack.



# Dijkstra's two-stack algorithm

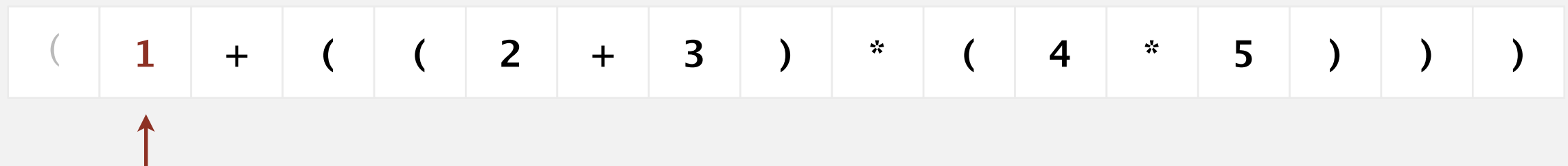
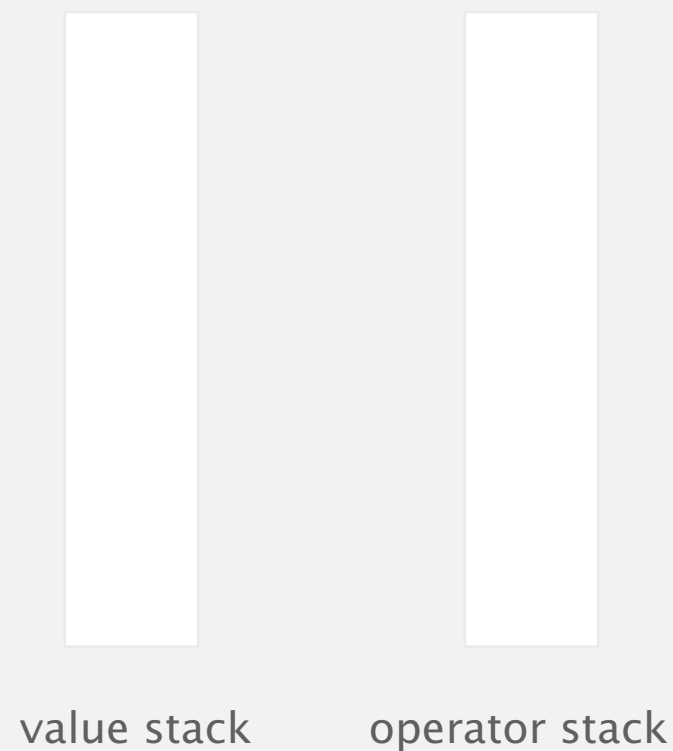
---

**Value:** push onto the value stack.

**Operator:** push onto the operator stack.

**Left parenthesis:** ignore.

**Right parenthesis:** pop operator and two values; push the result of applying that operator to those values onto the value stack.



# Dijkstra's two-stack algorithm

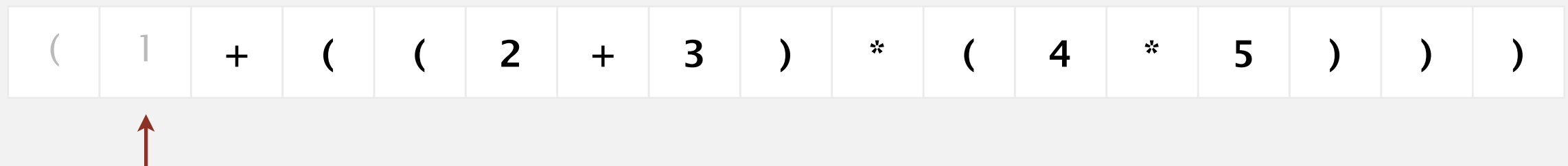
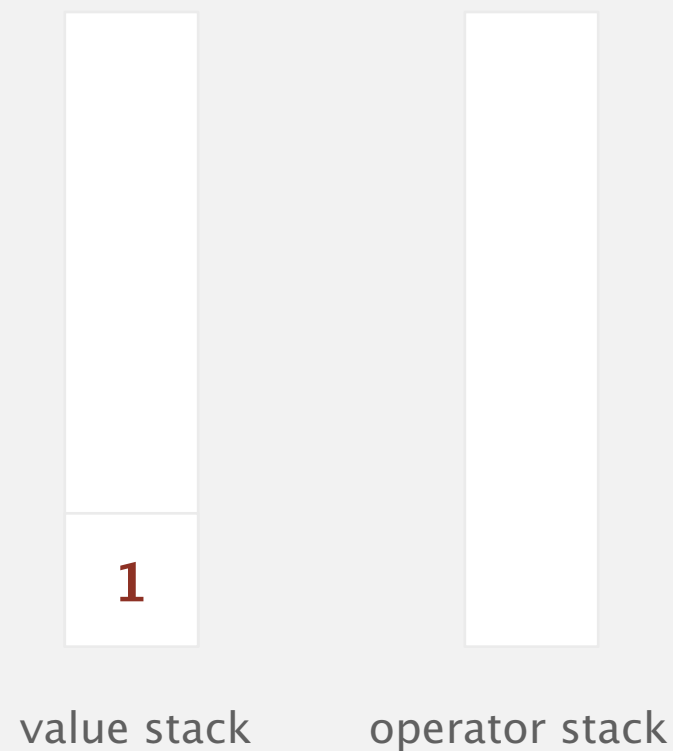
---

**Value:** push onto the value stack.

**Operator:** push onto the operator stack.

**Left parenthesis:** ignore.

**Right parenthesis:** pop operator and two values; push the result of applying that operator to those values onto the value stack.



# Dijkstra's two-stack algorithm

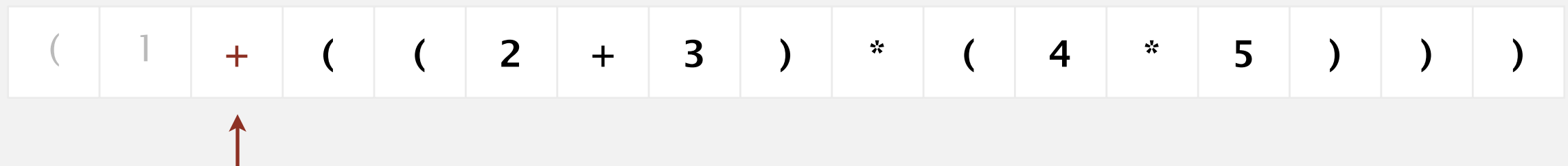
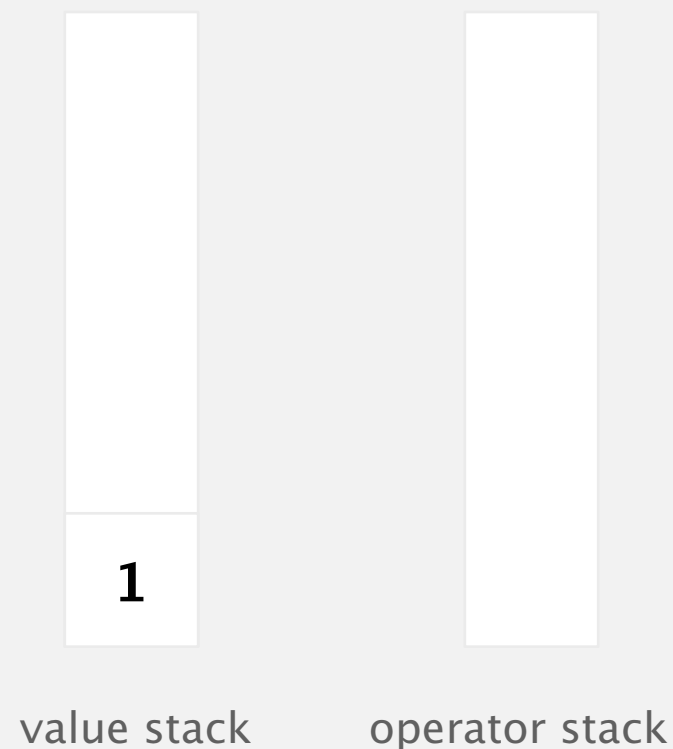
---

**Value:** push onto the value stack.

**Operator:** push onto the operator stack.

**Left parenthesis:** ignore.

**Right parenthesis:** pop operator and two values; push the result of applying that operator to those values onto the value stack.



# Dijkstra's two-stack algorithm

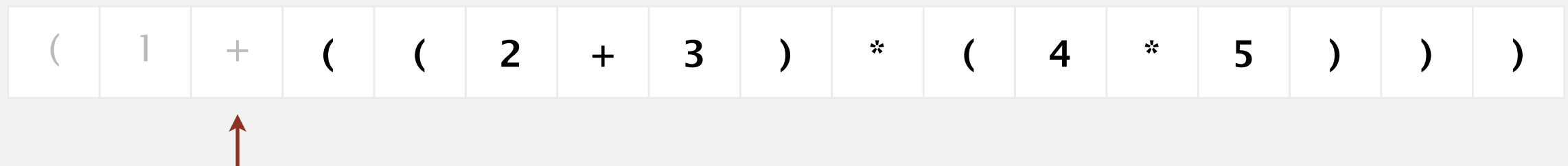
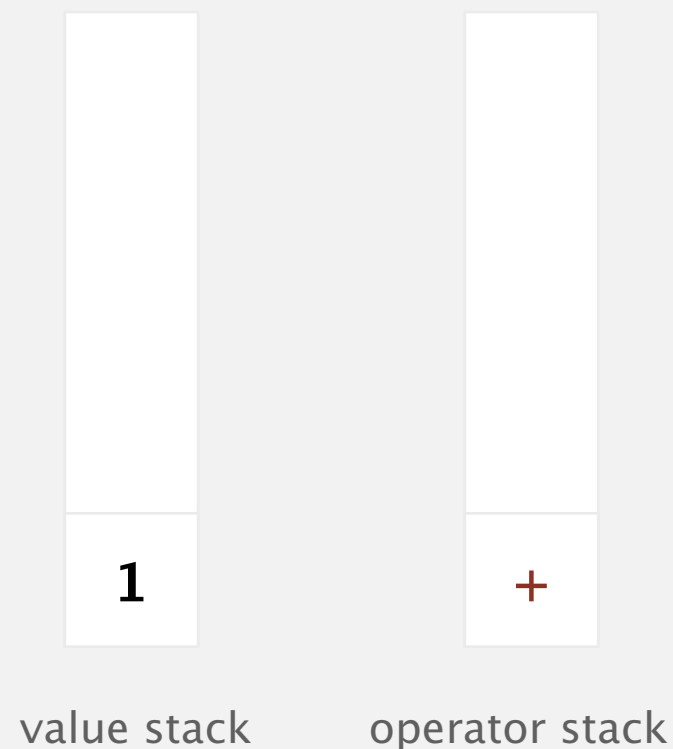
---

**Value:** push onto the value stack.

**Operator:** push onto the operator stack.

**Left parenthesis:** ignore.

**Right parenthesis:** pop operator and two values; push the result of applying that operator to those values onto the value stack.



# Dijkstra's two-stack algorithm

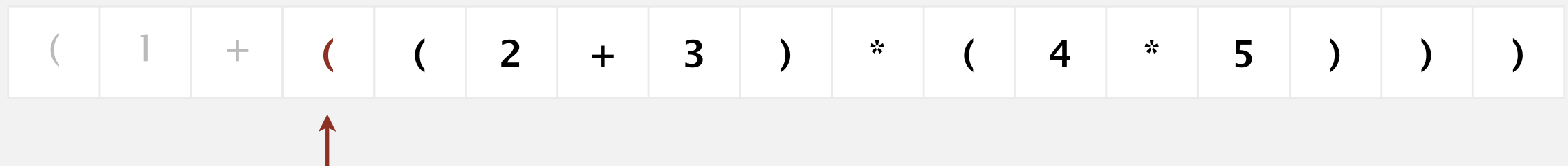
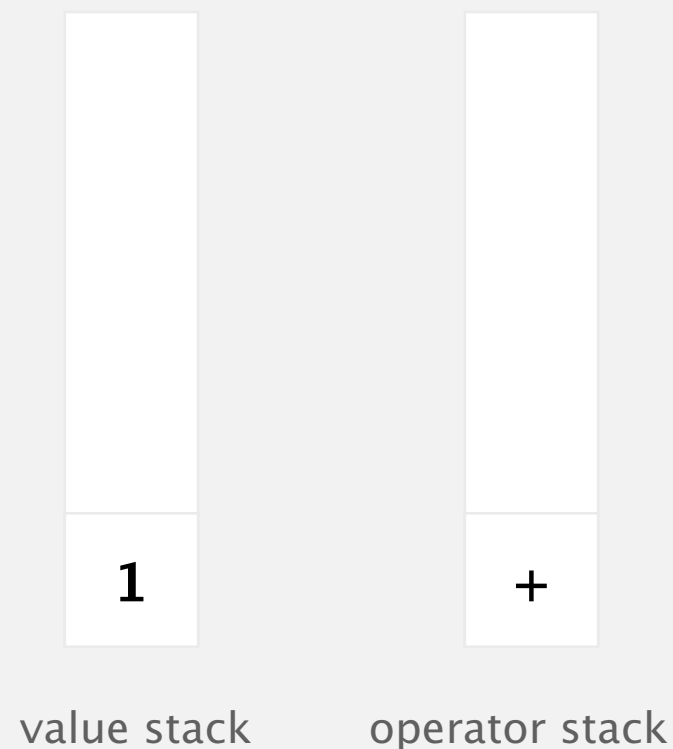
---

**Value:** push onto the value stack.

**Operator:** push onto the operator stack.

**Left parenthesis:** ignore.

**Right parenthesis:** pop operator and two values; push the result of applying that operator to those values onto the value stack.





# Dijkstra's two-stack algorithm

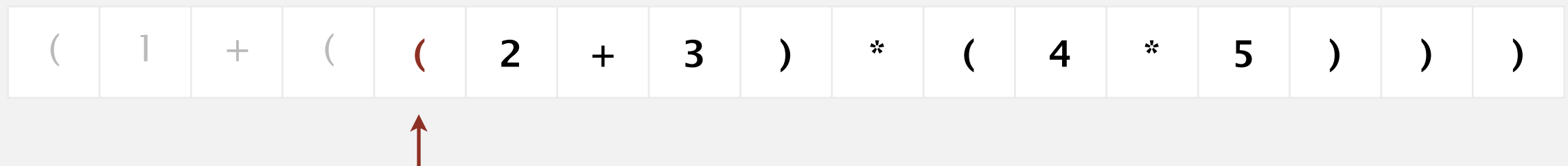
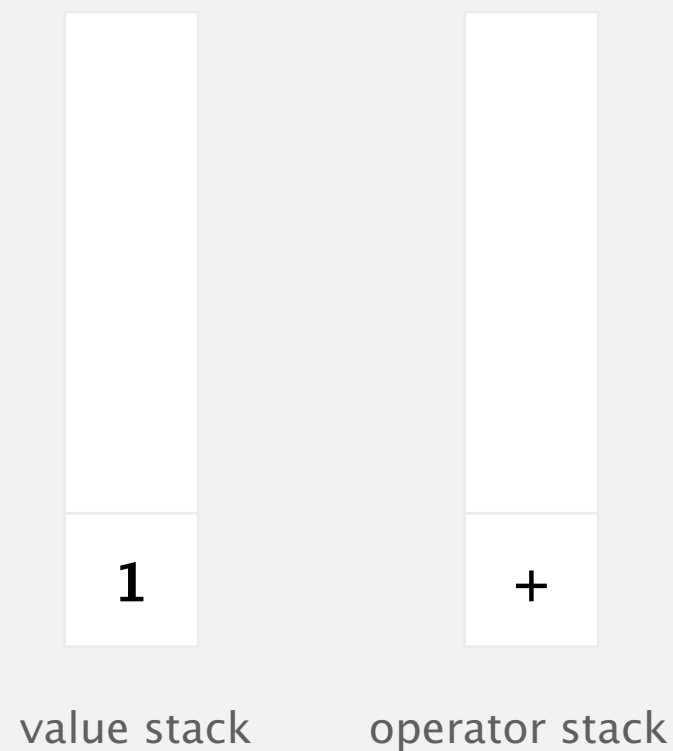
---

**Value:** push onto the value stack.

**Operator:** push onto the operator stack.

**Left parenthesis:** ignore.

**Right parenthesis:** pop operator and two values; push the result of applying that operator to those values onto the value stack.



# Dijkstra's two-stack algorithm

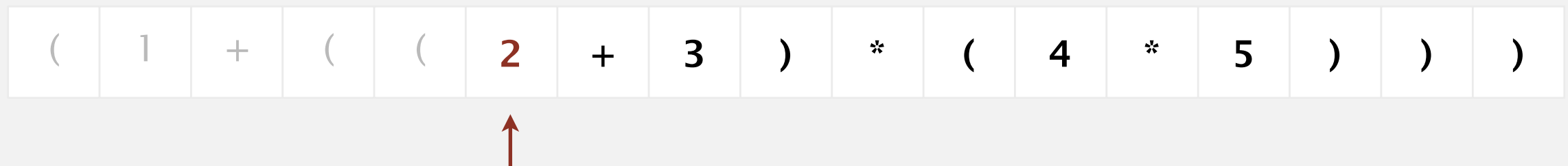
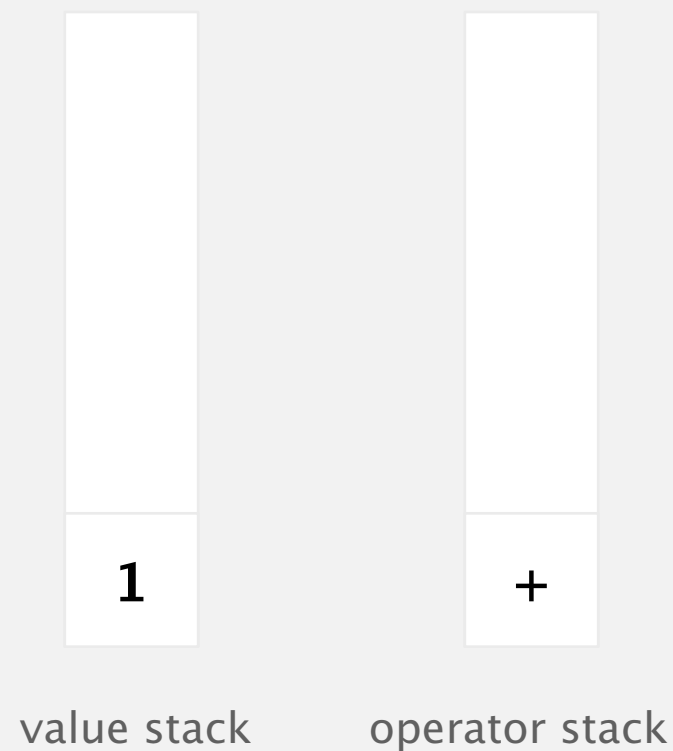
---

**Value:** push onto the value stack.

**Operator:** push onto the operator stack.

**Left parenthesis:** ignore.

**Right parenthesis:** pop operator and two values; push the result of applying that operator to those values onto the value stack.



# Dijkstra's two-stack algorithm

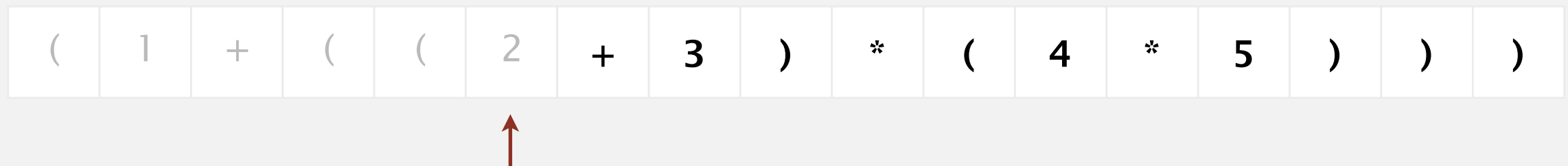
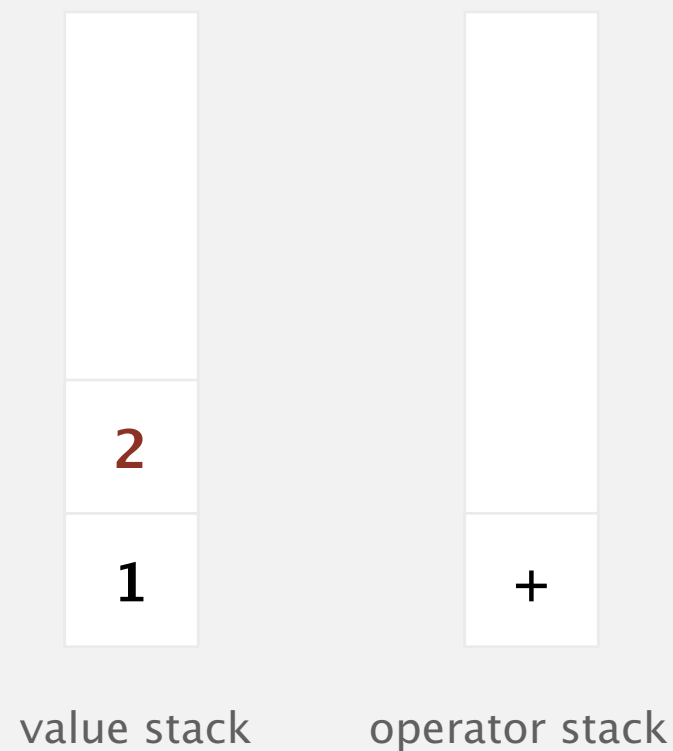
---

**Value:** push onto the value stack.

**Operator:** push onto the operator stack.

**Left parenthesis:** ignore.

**Right parenthesis:** pop operator and two values; push the result of applying that operator to those values onto the value stack.



# Dijkstra's two-stack algorithm

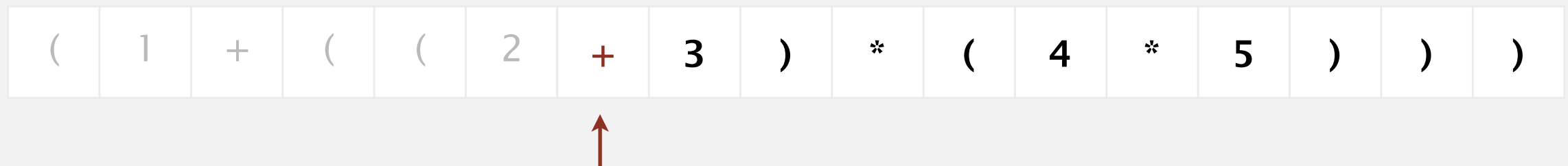
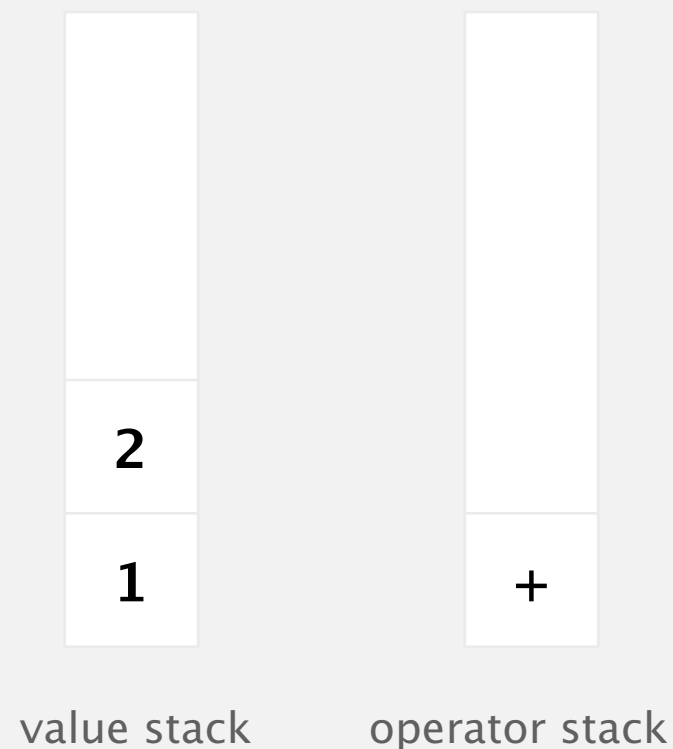
---

**Value:** push onto the value stack.

**Operator:** push onto the operator stack.

**Left parenthesis:** ignore.

**Right parenthesis:** pop operator and two values; push the result of applying that operator to those values onto the value stack.



# Dijkstra's two-stack algorithm

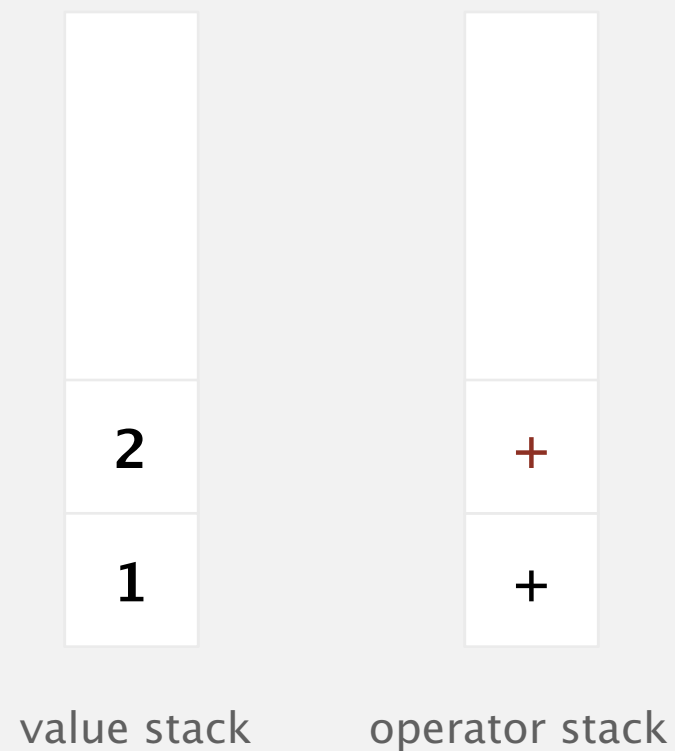
---

**Value:** push onto the value stack.

**Operator:** push onto the operator stack.

**Left parenthesis:** ignore.

**Right parenthesis:** pop operator and two values; push the result of applying that operator to those values onto the value stack.



# Dijkstra's two-stack algorithm

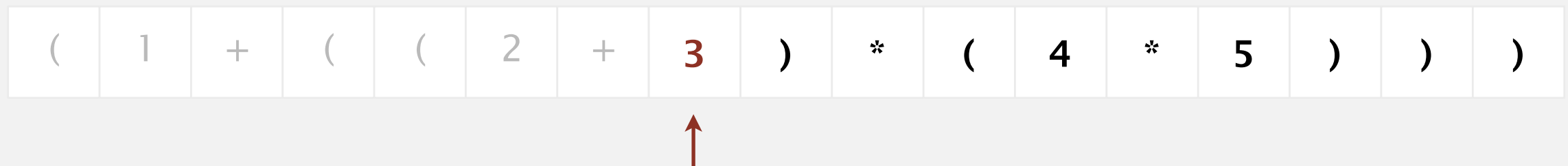
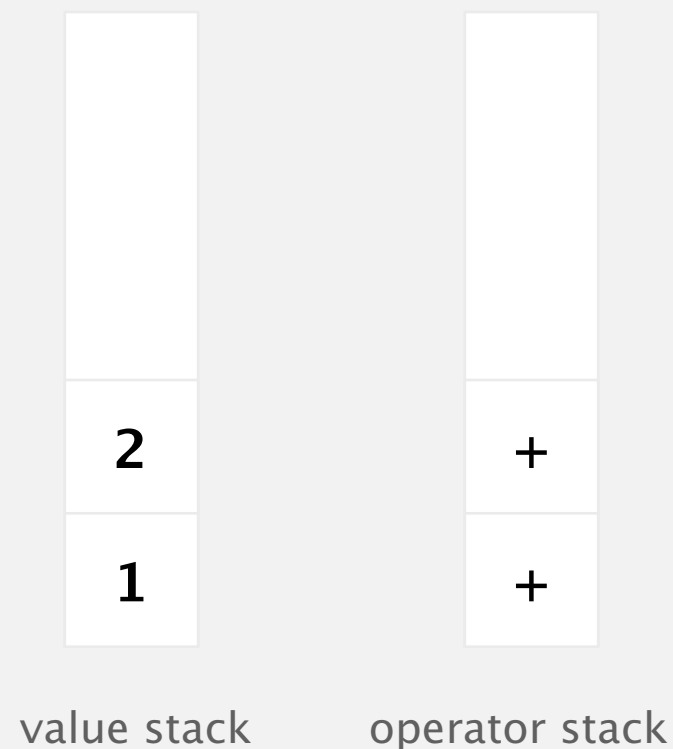
---

**Value:** push onto the value stack.

**Operator:** push onto the operator stack.

**Left parenthesis:** ignore.

**Right parenthesis:** pop operator and two values; push the result of applying that operator to those values onto the value stack.



# Dijkstra's two-stack algorithm

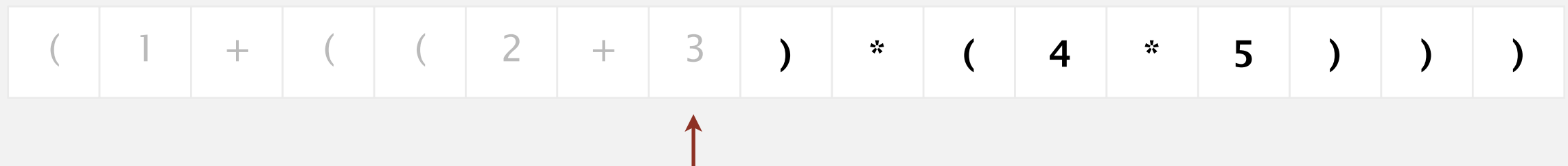
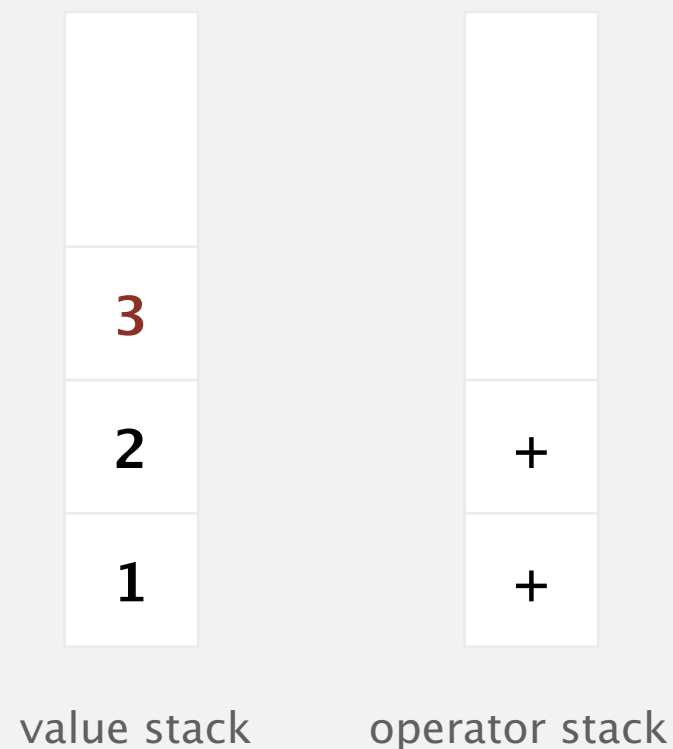
---

**Value:** push onto the value stack.

**Operator:** push onto the operator stack.

**Left parenthesis:** ignore.

**Right parenthesis:** pop operator and two values; push the result of applying that operator to those values onto the value stack.



# Dijkstra's two-stack algorithm

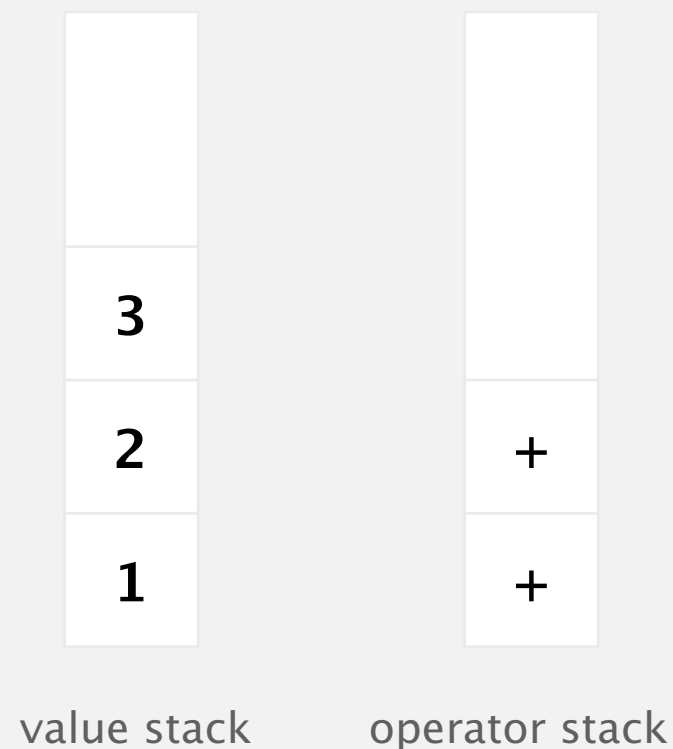
---

**Value:** push onto the value stack.

**Operator:** push onto the operator stack.

**Left parenthesis:** ignore.

**Right parenthesis:** pop operator and two values; push the result of applying that operator to those values onto the value stack.





# Dijkstra's two-stack algorithm

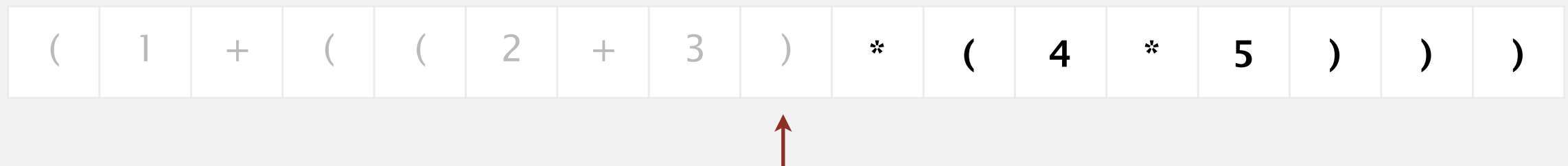
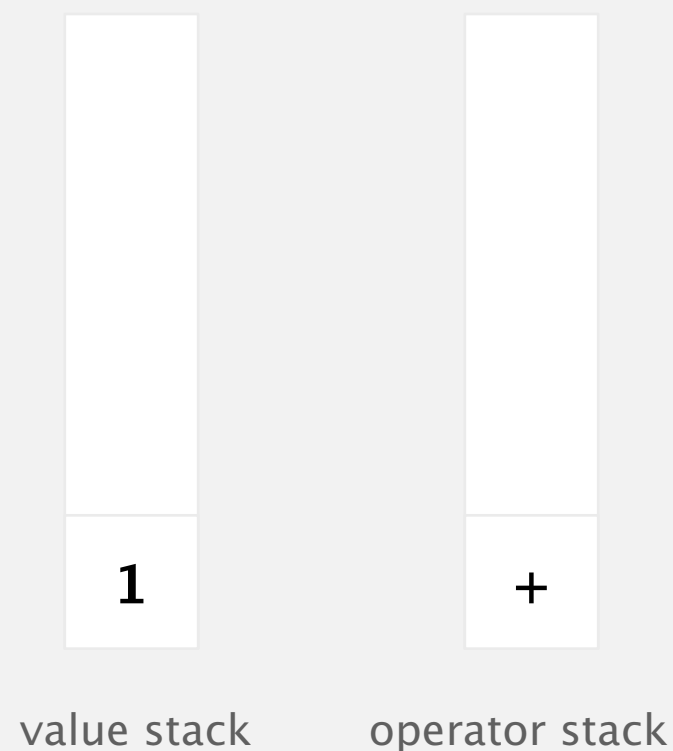
---

**Value:** push onto the value stack.

**Operator:** push onto the operator stack.

**Left parenthesis:** ignore.

**Right parenthesis:** pop operator and two values; push the result of applying that operator to those values onto the value stack.



# Dijkstra's two-stack algorithm

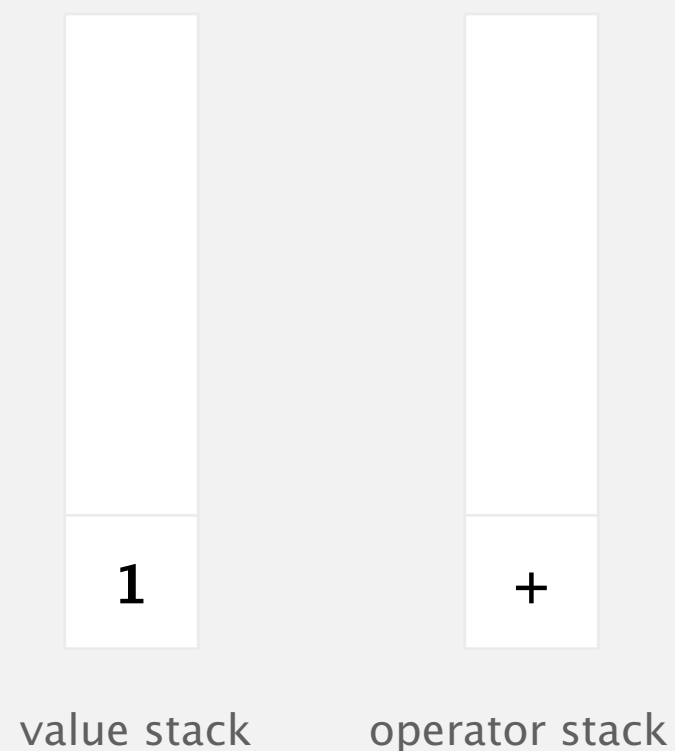
---

**Value:** push onto the value stack.

**Operator:** push onto the operator stack.

**Left parenthesis:** ignore.

**Right parenthesis:** pop operator and two values; push the result of applying that operator to those values onto the value stack.



3 + 2 = 5

( 1 + ( ( 2 + 3 ) \* ( 4 \* 5 ) ) )

↑

# Dijkstra's two-stack algorithm

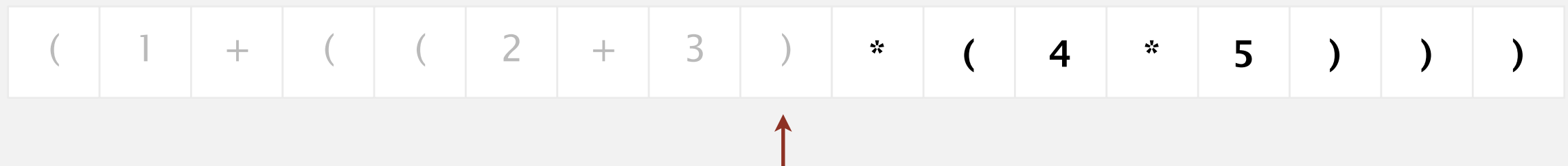
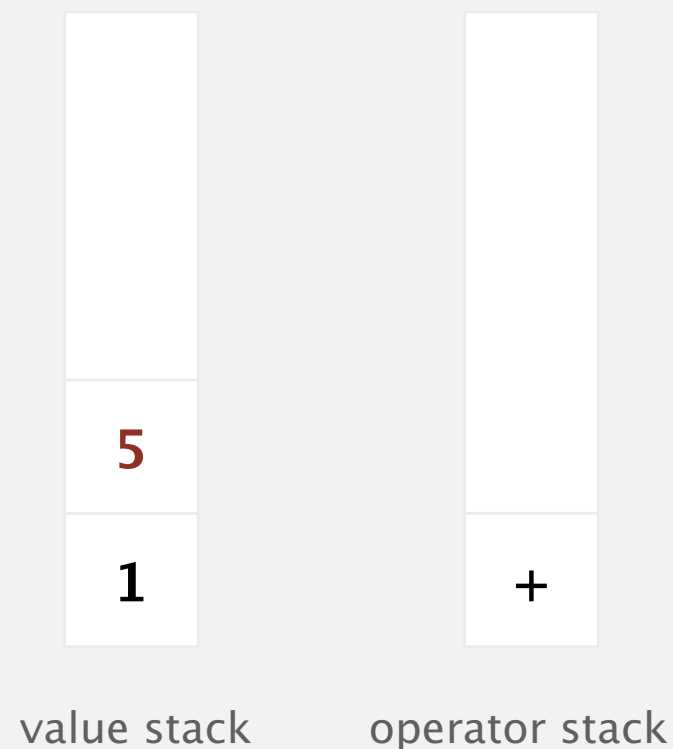
---

**Value:** push onto the value stack.

**Operator:** push onto the operator stack.

**Left parenthesis:** ignore.

**Right parenthesis:** pop operator and two values; push the result of applying that operator to those values onto the value stack.



# Dijkstra's two-stack algorithm

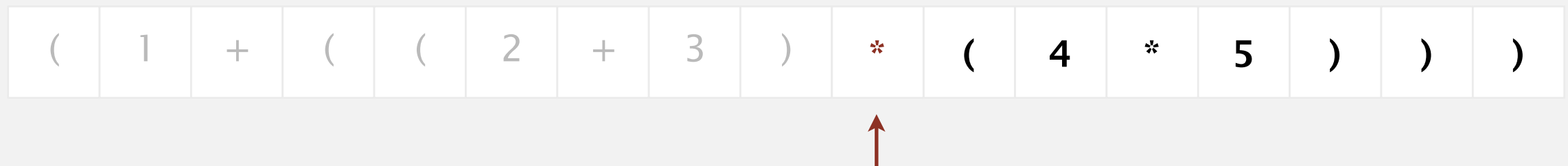
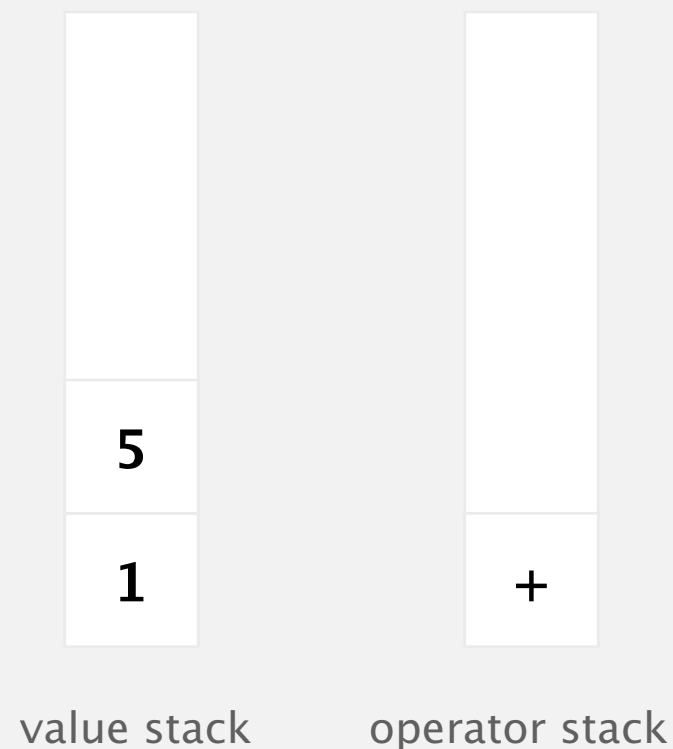
---

**Value:** push onto the value stack.

**Operator:** push onto the operator stack.

**Left parenthesis:** ignore.

**Right parenthesis:** pop operator and two values; push the result of applying that operator to those values onto the value stack.



# Dijkstra's two-stack algorithm

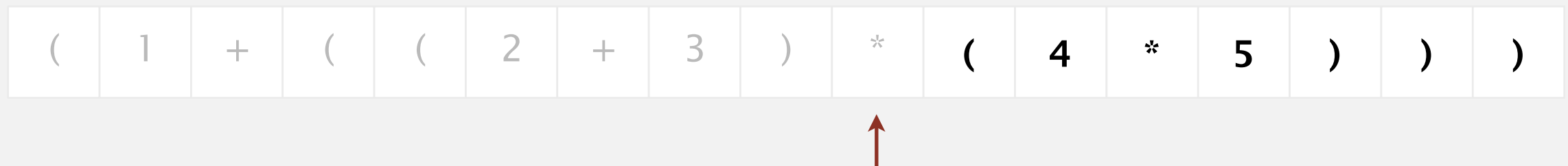
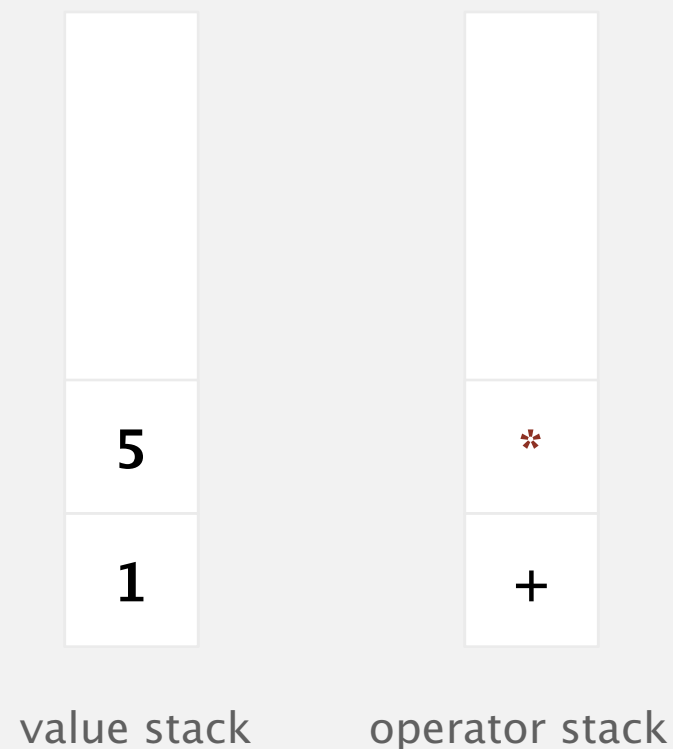
---

**Value:** push onto the value stack.

**Operator:** push onto the operator stack.

**Left parenthesis:** ignore.

**Right parenthesis:** pop operator and two values; push the result of applying that operator to those values onto the value stack.



# Dijkstra's two-stack algorithm

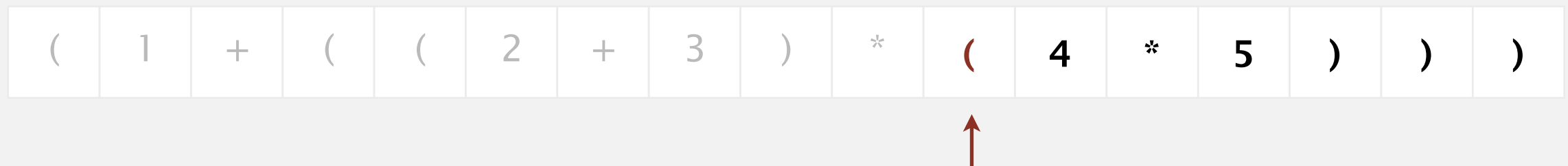
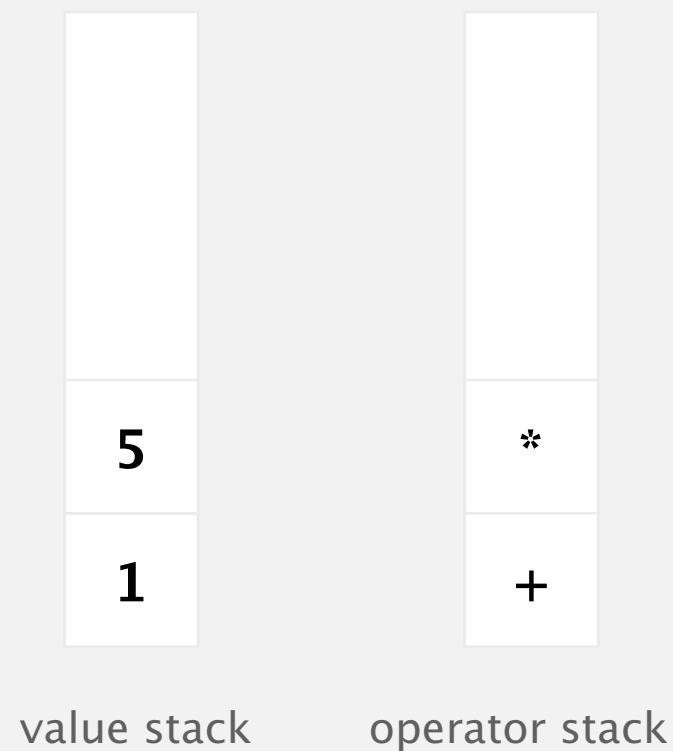
---

**Value:** push onto the value stack.

**Operator:** push onto the operator stack.

**Left parenthesis:** ignore.

**Right parenthesis:** pop operator and two values; push the result of applying that operator to those values onto the value stack.



# Dijkstra's two-stack algorithm

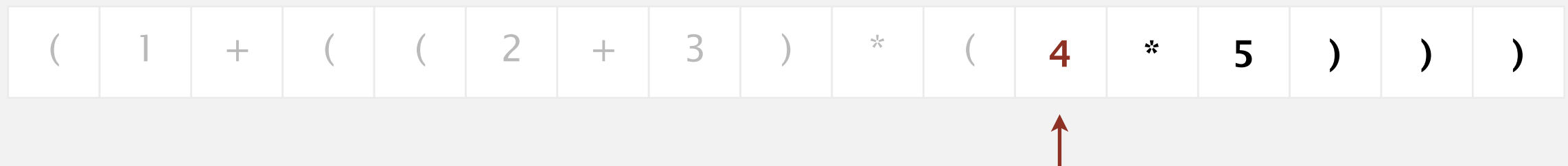
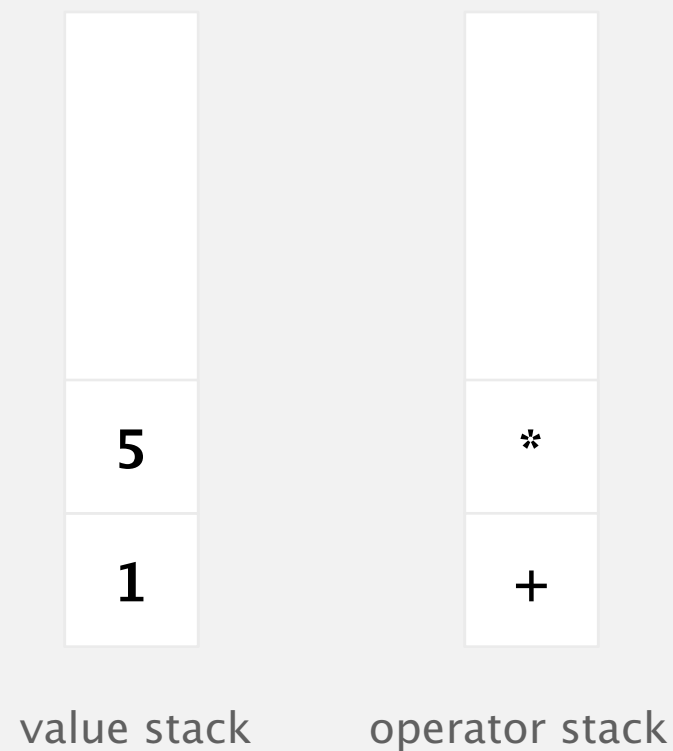
---

**Value:** push onto the value stack.

**Operator:** push onto the operator stack.

**Left parenthesis:** ignore.

**Right parenthesis:** pop operator and two values; push the result of applying that operator to those values onto the value stack.



# Dijkstra's two-stack algorithm

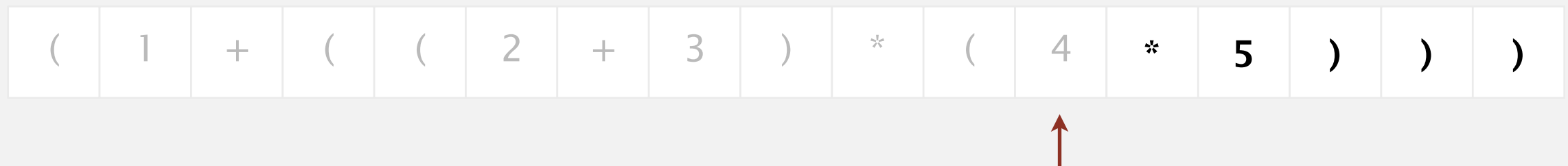
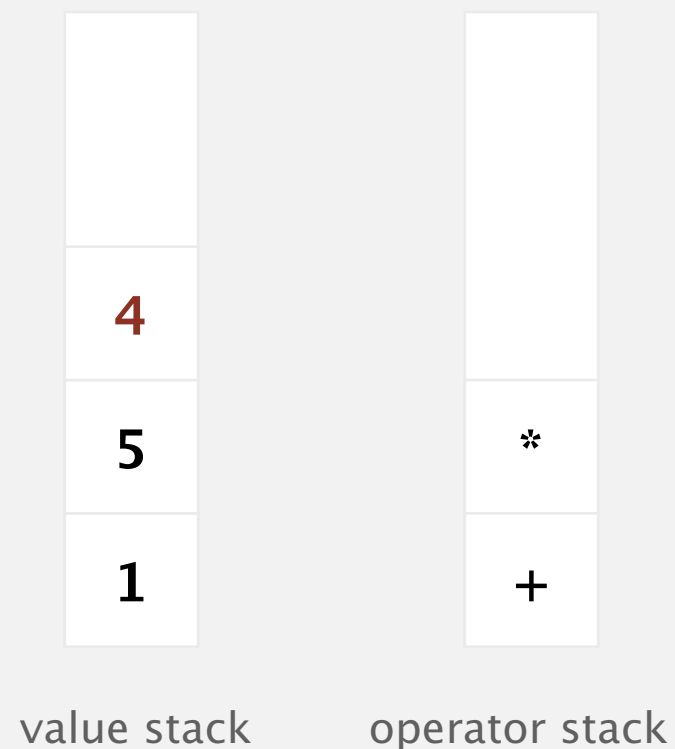
---

**Value:** push onto the value stack.

**Operator:** push onto the operator stack.

**Left parenthesis:** ignore.

**Right parenthesis:** pop operator and two values; push the result of applying that operator to those values onto the value stack.





# Dijkstra's two-stack algorithm

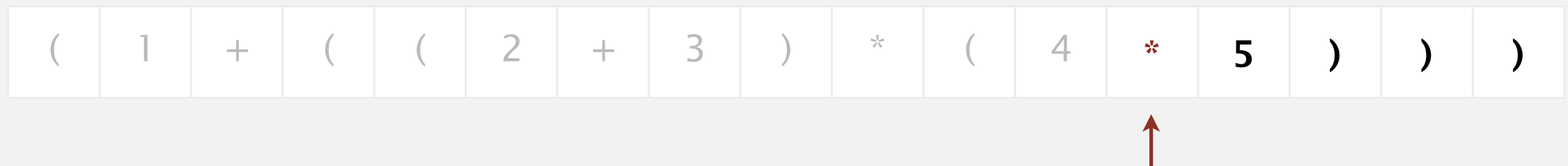
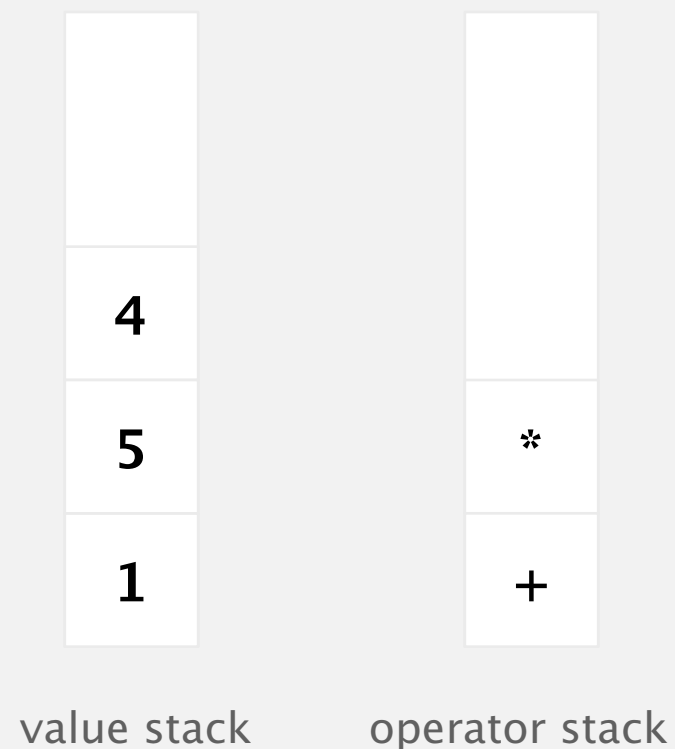
---

**Value:** push onto the value stack.

**Operator:** push onto the operator stack.

**Left parenthesis:** ignore.

**Right parenthesis:** pop operator and two values; push the result of applying that operator to those values onto the value stack.



# Dijkstra's two-stack algorithm

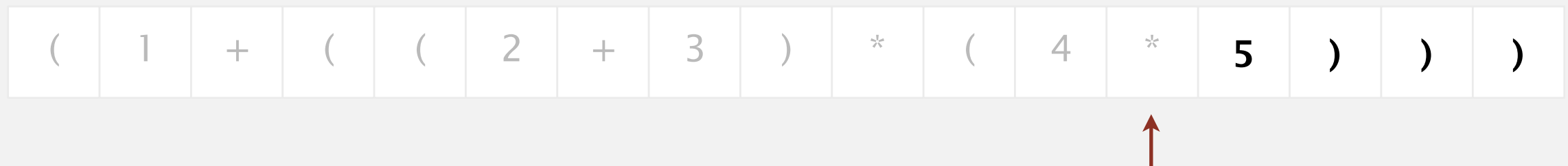
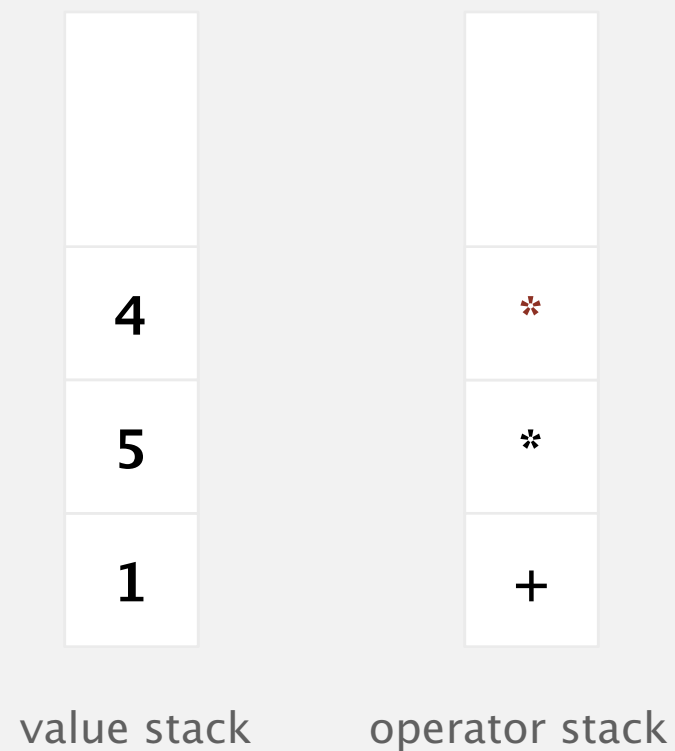
---

**Value:** push onto the value stack.

**Operator:** push onto the operator stack.

**Left parenthesis:** ignore.

**Right parenthesis:** pop operator and two values; push the result of applying that operator to those values onto the value stack.



# Dijkstra's two-stack algorithm

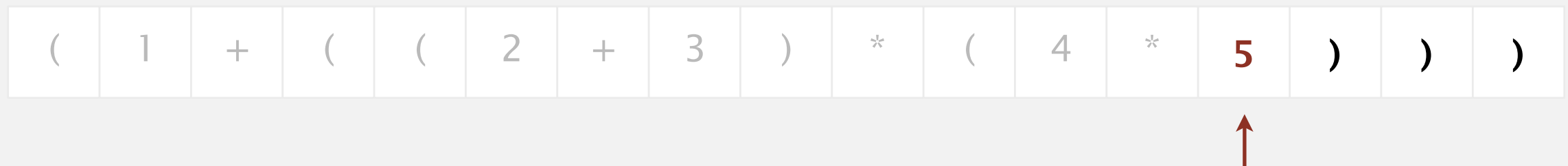
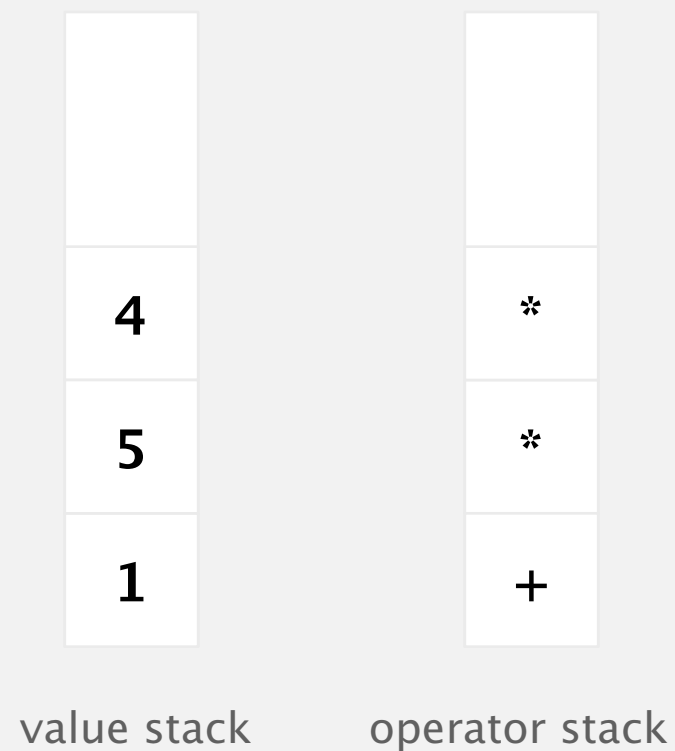
---

**Value:** push onto the value stack.

**Operator:** push onto the operator stack.

**Left parenthesis:** ignore.

**Right parenthesis:** pop operator and two values; push the result of applying that operator to those values onto the value stack.



# Dijkstra's two-stack algorithm

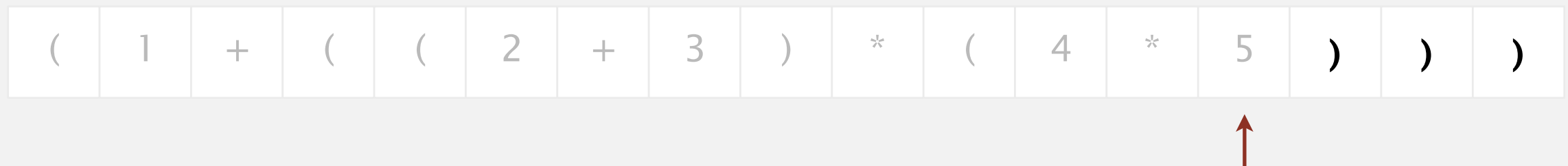
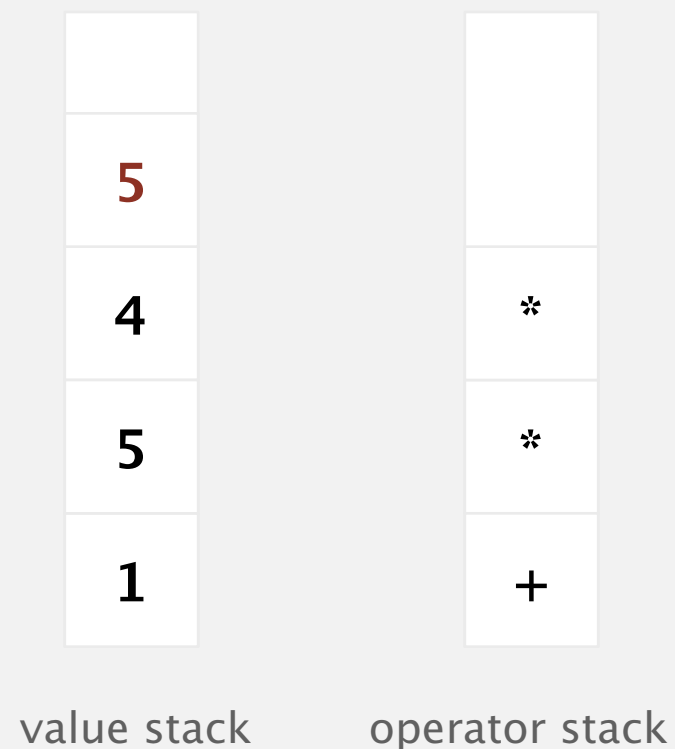
---

**Value:** push onto the value stack.

**Operator:** push onto the operator stack.

**Left parenthesis:** ignore.

**Right parenthesis:** pop operator and two values; push the result of applying that operator to those values onto the value stack.



# Dijkstra's two-stack algorithm

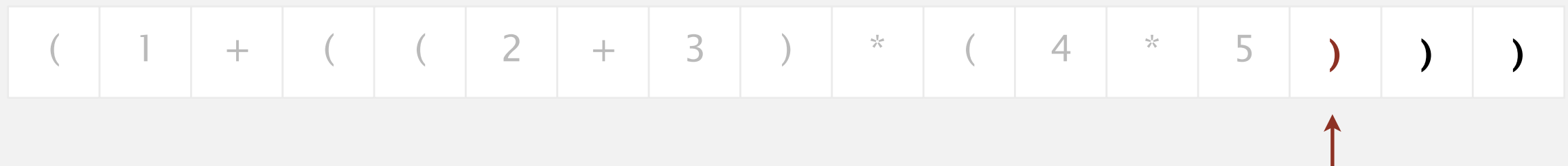
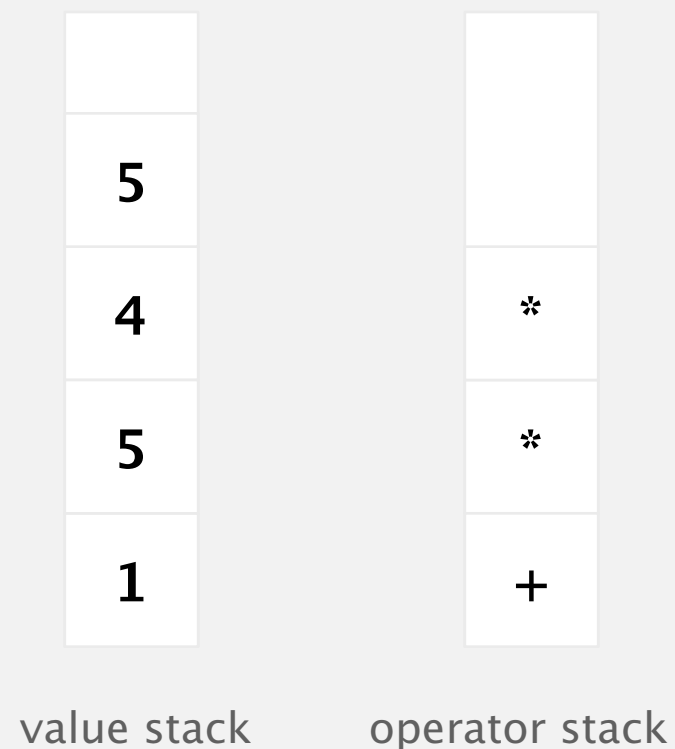
---

**Value:** push onto the value stack.

**Operator:** push onto the operator stack.

**Left parenthesis:** ignore.

**Right parenthesis:** pop operator and two values; push the result of applying that operator to those values onto the value stack.



# Dijkstra's two-stack algorithm

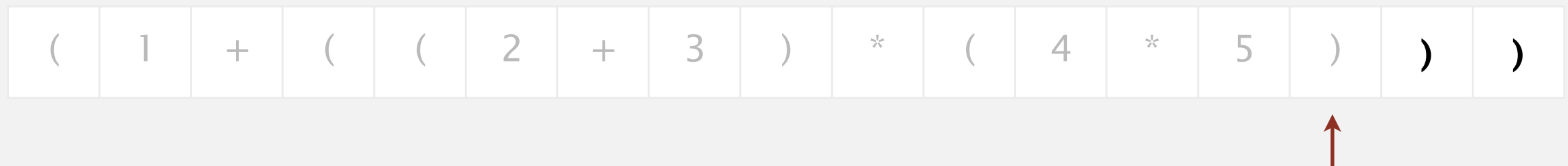
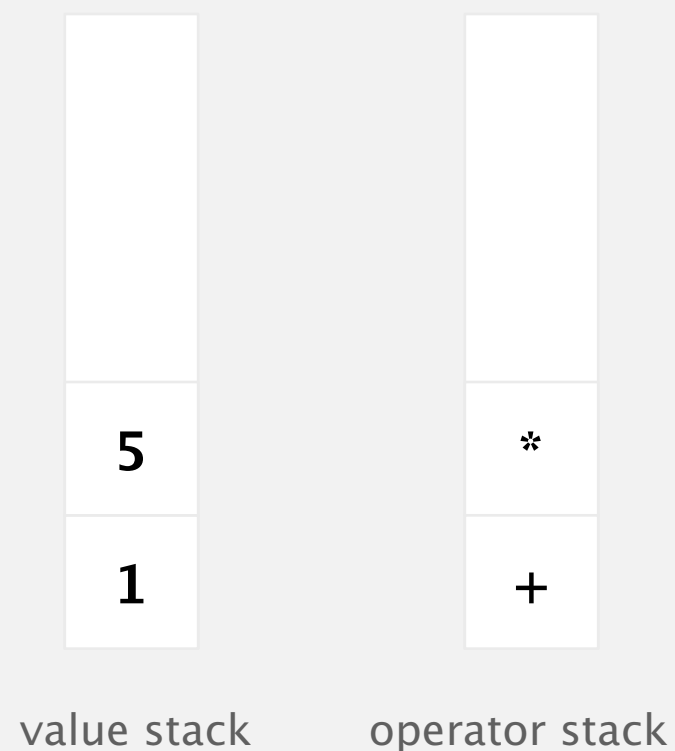
---

**Value:** push onto the value stack.

**Operator:** push onto the operator stack.

**Left parenthesis:** ignore.

**Right parenthesis:** pop operator and two values; push the result of applying that operator to those values onto the value stack.



# Dijkstra's two-stack algorithm

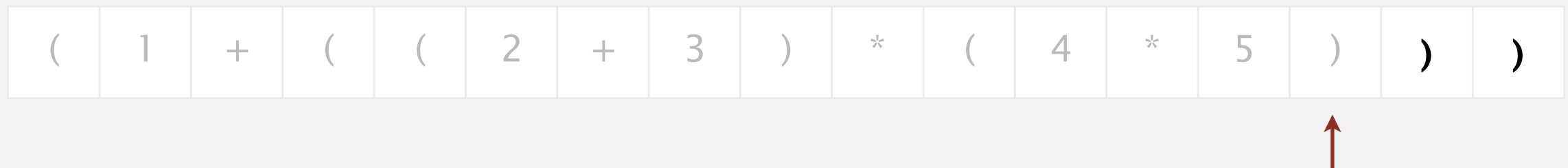
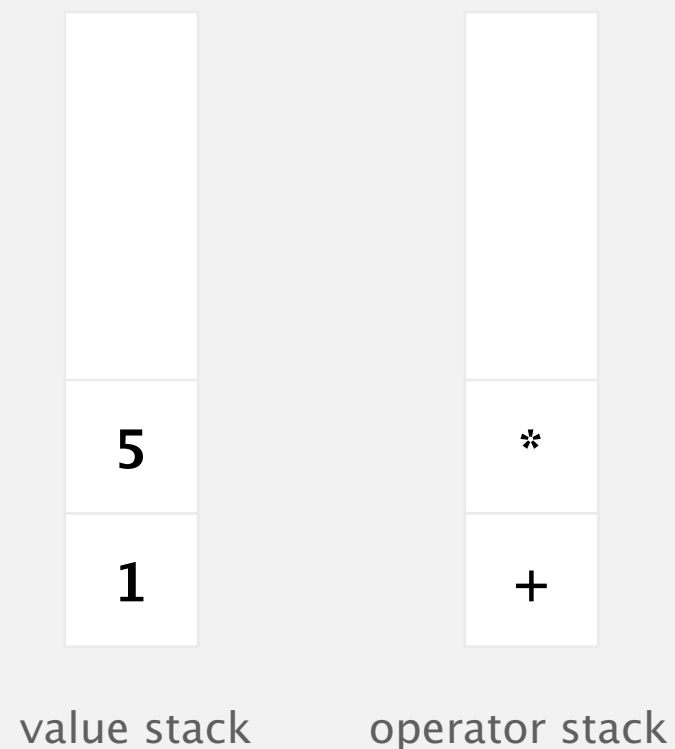
---

**Value:** push onto the value stack.

**Operator:** push onto the operator stack.

**Left parenthesis:** ignore.

**Right parenthesis:** pop operator and two values; push the result of applying that operator to those values onto the value stack.



# Dijkstra's two-stack algorithm

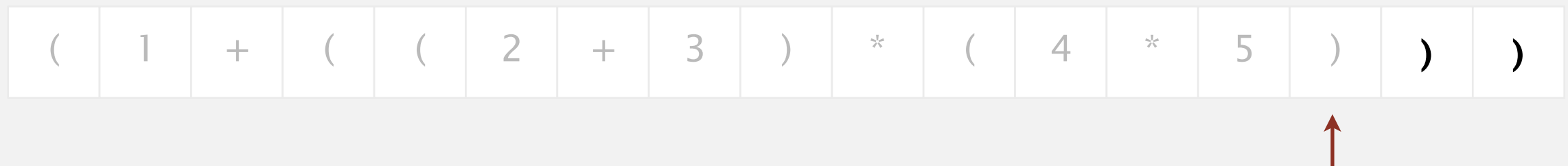
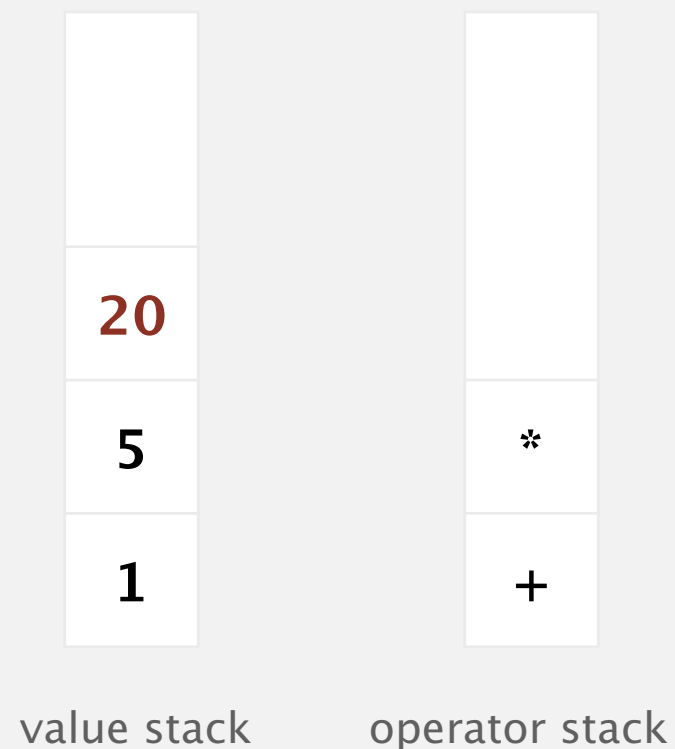
---

**Value:** push onto the value stack.

**Operator:** push onto the operator stack.

**Left parenthesis:** ignore.

**Right parenthesis:** pop operator and two values; push the result of applying that operator to those values onto the value stack.





# Dijkstra's two-stack algorithm

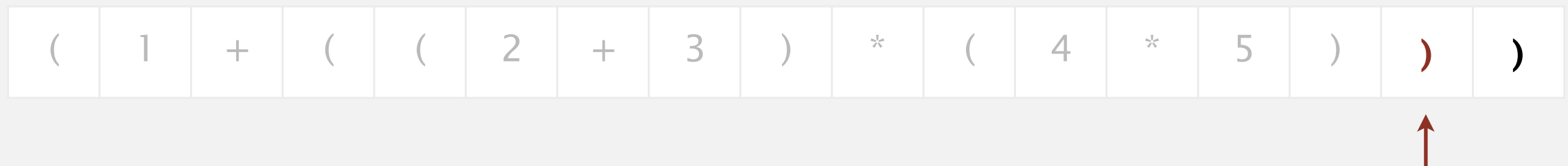
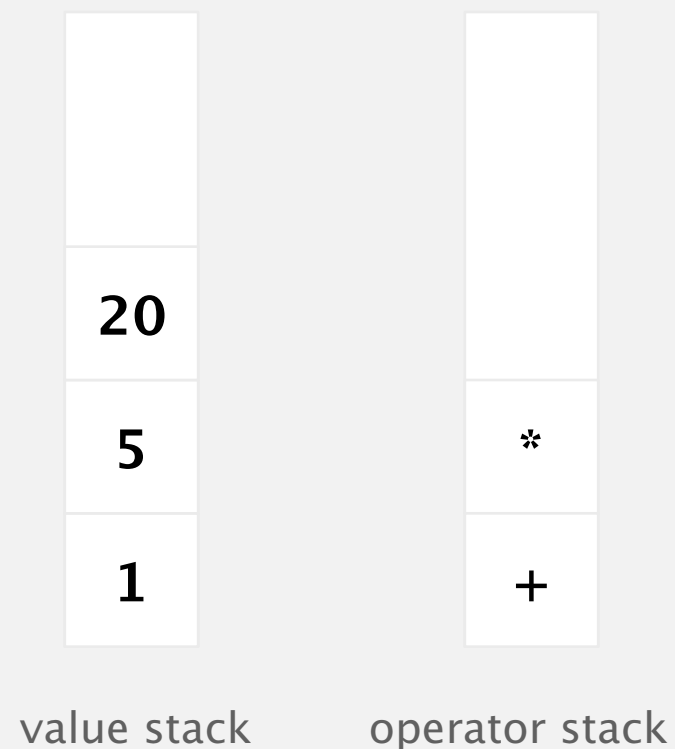
---

**Value:** push onto the value stack.

**Operator:** push onto the operator stack.

**Left parenthesis:** ignore.

**Right parenthesis:** pop operator and two values; push the result of applying that operator to those values onto the value stack.



# Dijkstra's two-stack algorithm

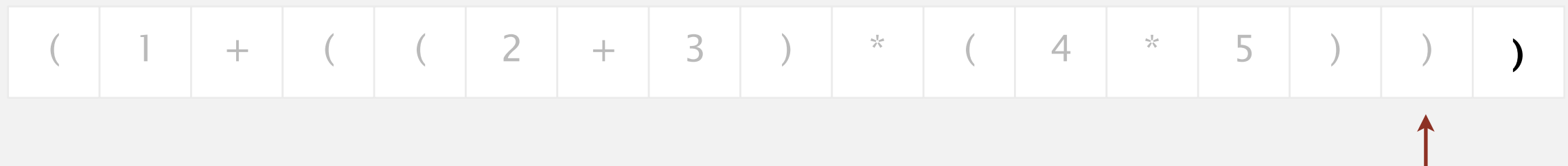
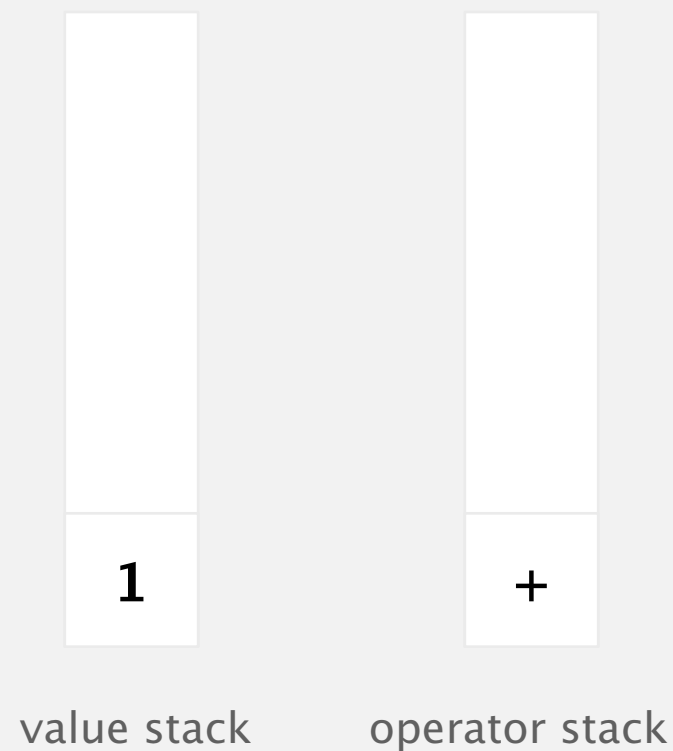
---

**Value:** push onto the value stack.

**Operator:** push onto the operator stack.

**Left parenthesis:** ignore.

**Right parenthesis:** pop operator and two values; push the result of applying that operator to those values onto the value stack.



# Dijkstra's two-stack algorithm

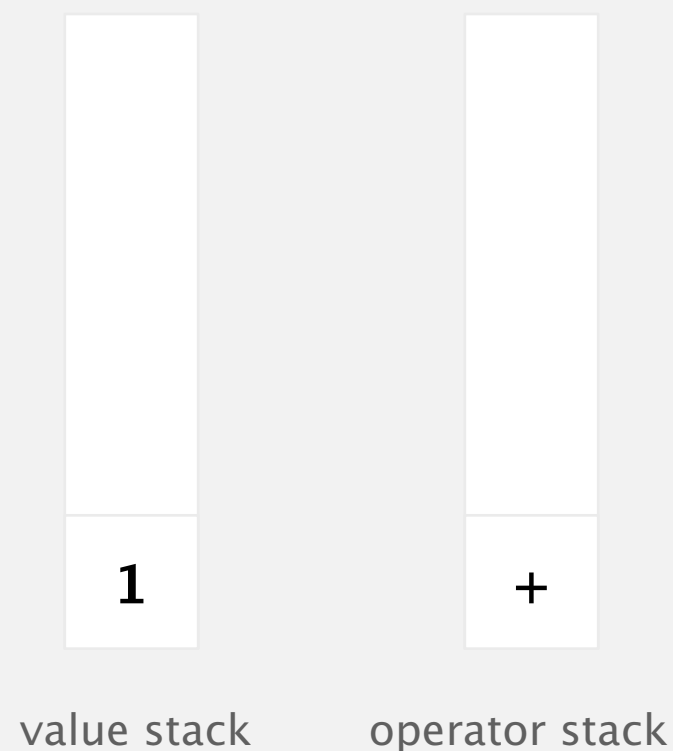
---

**Value:** push onto the value stack.

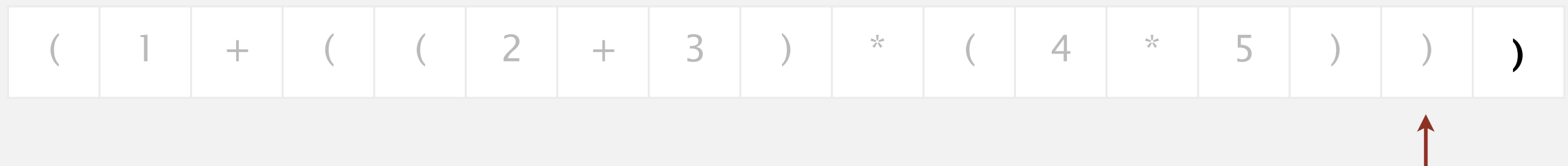
**Operator:** push onto the operator stack.

**Left parenthesis:** ignore.

**Right parenthesis:** pop operator and two values; push the result of applying that operator to those values onto the value stack.



20	*	5	=	100
----	---	---	---	-----



# Dijkstra's two-stack algorithm

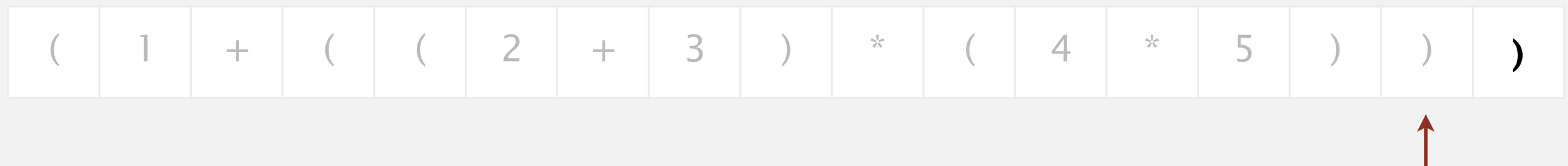
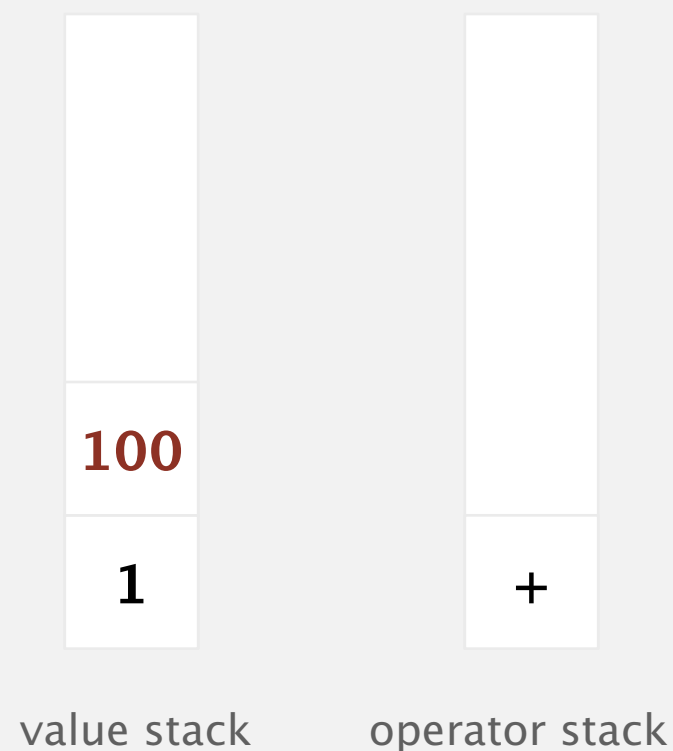
---

**Value:** push onto the value stack.

**Operator:** push onto the operator stack.

**Left parenthesis:** ignore.

**Right parenthesis:** pop operator and two values; push the result of applying that operator to those values onto the value stack.



# Dijkstra's two-stack algorithm

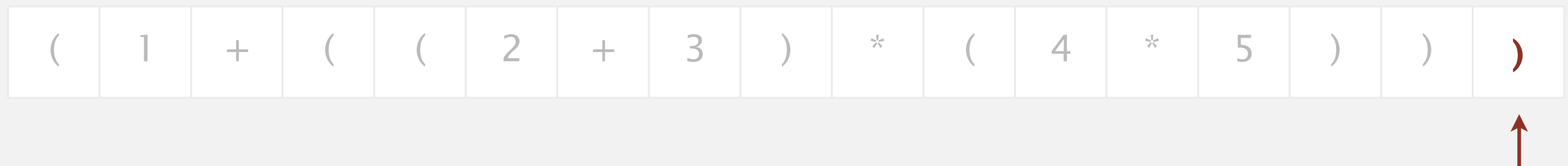
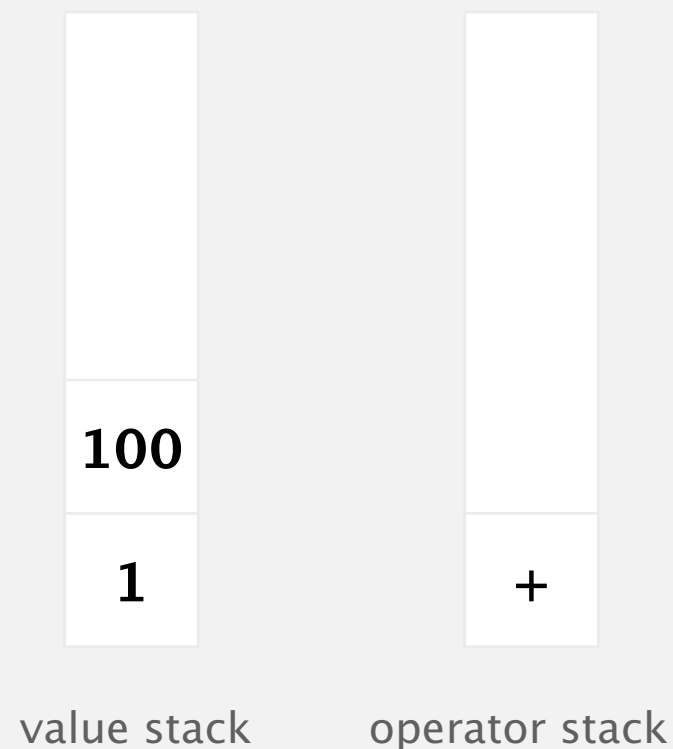
---

**Value:** push onto the value stack.

**Operator:** push onto the operator stack.

**Left parenthesis:** ignore.

**Right parenthesis:** pop operator and two values; push the result of applying that operator to those values onto the value stack.



# Dijkstra's two-stack algorithm

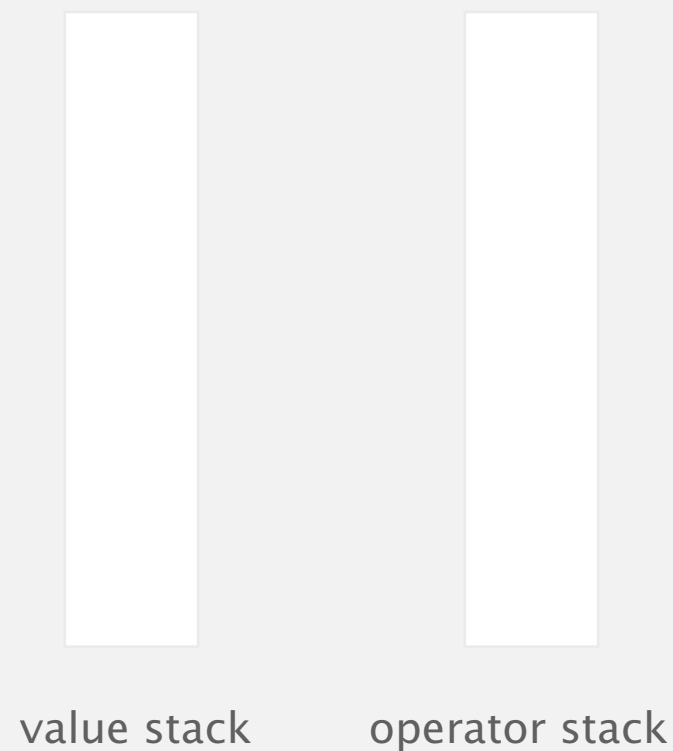
---

**Value:** push onto the value stack.

**Operator:** push onto the operator stack.

**Left parenthesis:** ignore.

**Right parenthesis:** pop operator and two values; push the result of applying that operator to those values onto the value stack.



100	+	1
-----	---	---

(	1	+	(	(	2	+	3	)	*	(	4	*	5	)	)	)
---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---



# Dijkstra's two-stack algorithm

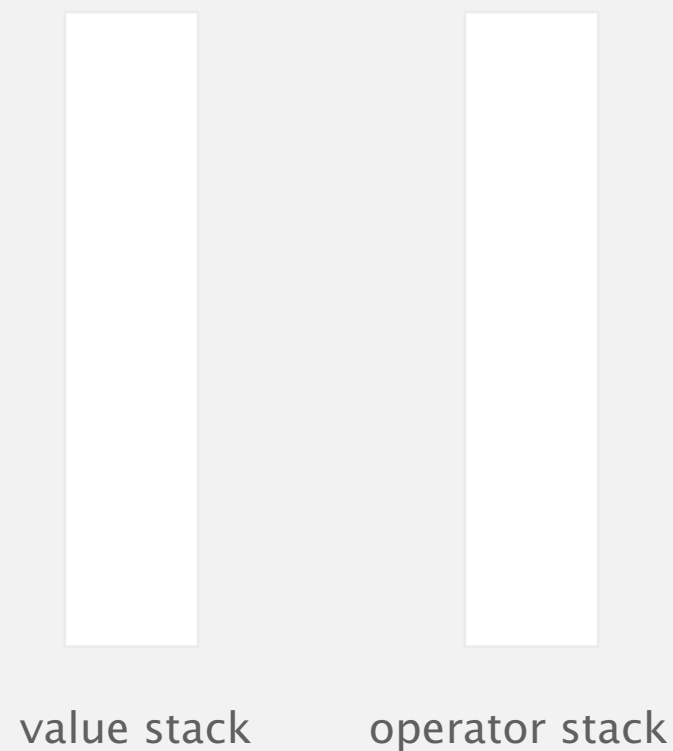
---

**Value:** push onto the value stack.

**Operator:** push onto the operator stack.

**Left parenthesis:** ignore.

**Right parenthesis:** pop operator and two values; push the result of applying that operator to those values onto the value stack.



100	+	1	=	101
-----	---	---	---	-----

(	1	+	(	(	2	+	3	)	*	(	4	*	5	)	)	)
---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---



# Dijkstra's two-stack algorithm

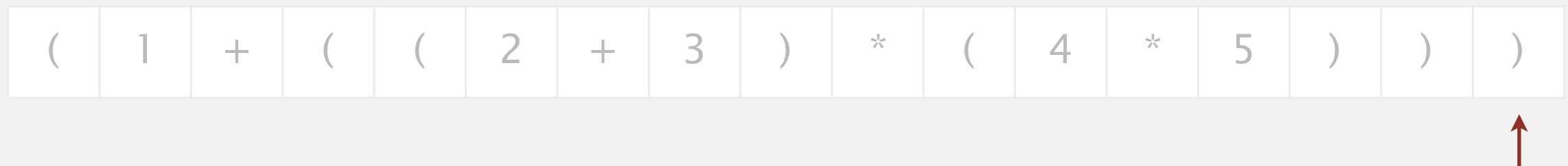
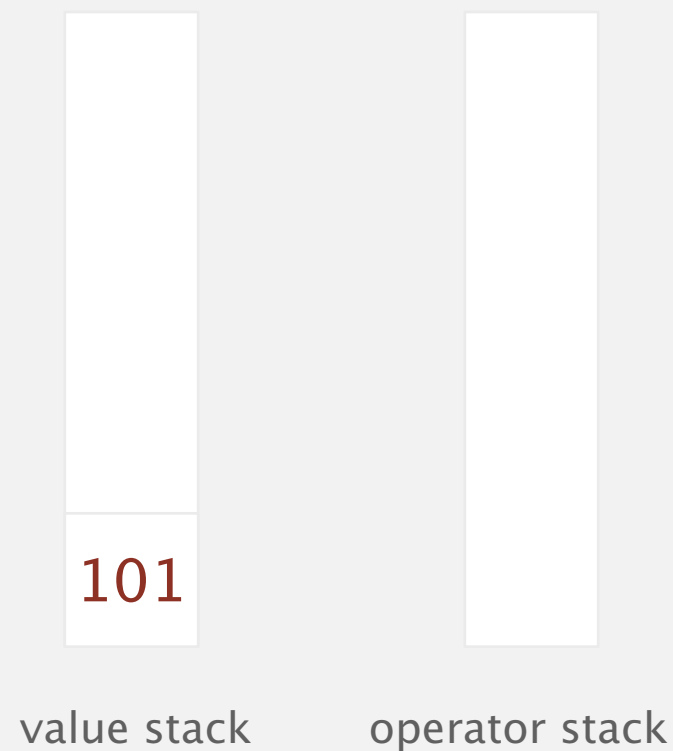
---

**Value:** push onto the value stack.

**Operator:** push onto the operator stack.

**Left parenthesis:** ignore.

**Right parenthesis:** pop operator and two values; push the result of applying that operator to those values onto the value stack.





# Dijkstra's two-stack algorithm

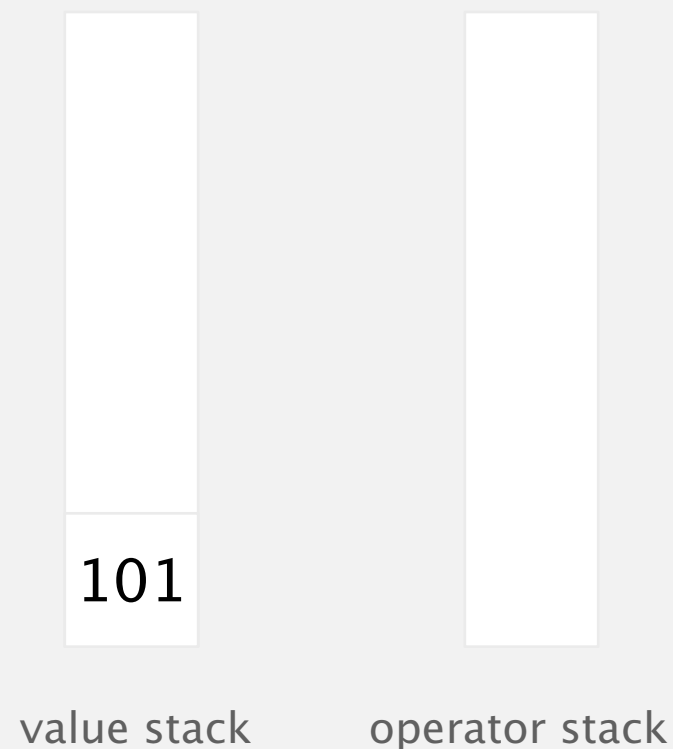
---

**Value:** push onto the value stack.

**Operator:** push onto the operator stack.

**Left parenthesis:** ignore.

**Right parenthesis:** pop operator and two values; push the result of applying that operator to those values onto the value stack.



(	1	+	(	(	2	+	3	)	*	(	4	*	5	)	)	)
---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---



# Dijkstra's two-stack algorithm

---

**Value:** push onto the value stack.

**Operator:** push onto the operator stack.

**Left parenthesis:** ignore.

**Right parenthesis:** pop operator and two values; push the result of applying that operator to those values onto the value stack.

101

result

(	1	+	(	(	2	+	3	)	*	(	4	*	5	)	)	)
---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---