Bubble sort

Programovacie techniky | Pavol Marák

OBSAH

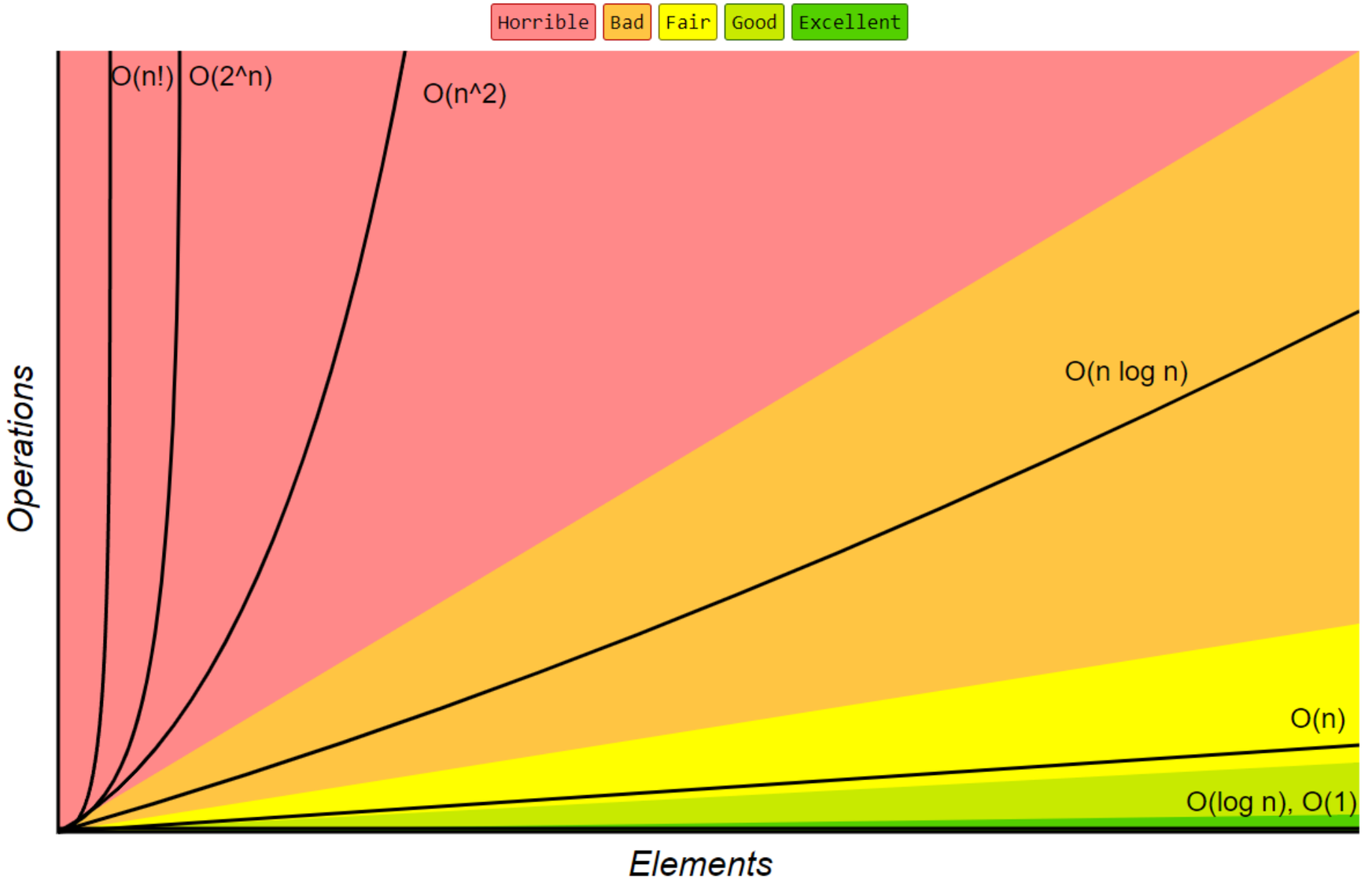
Bubble sort

- Princíp
- Vizualizácia
- Implementácia v C/C++

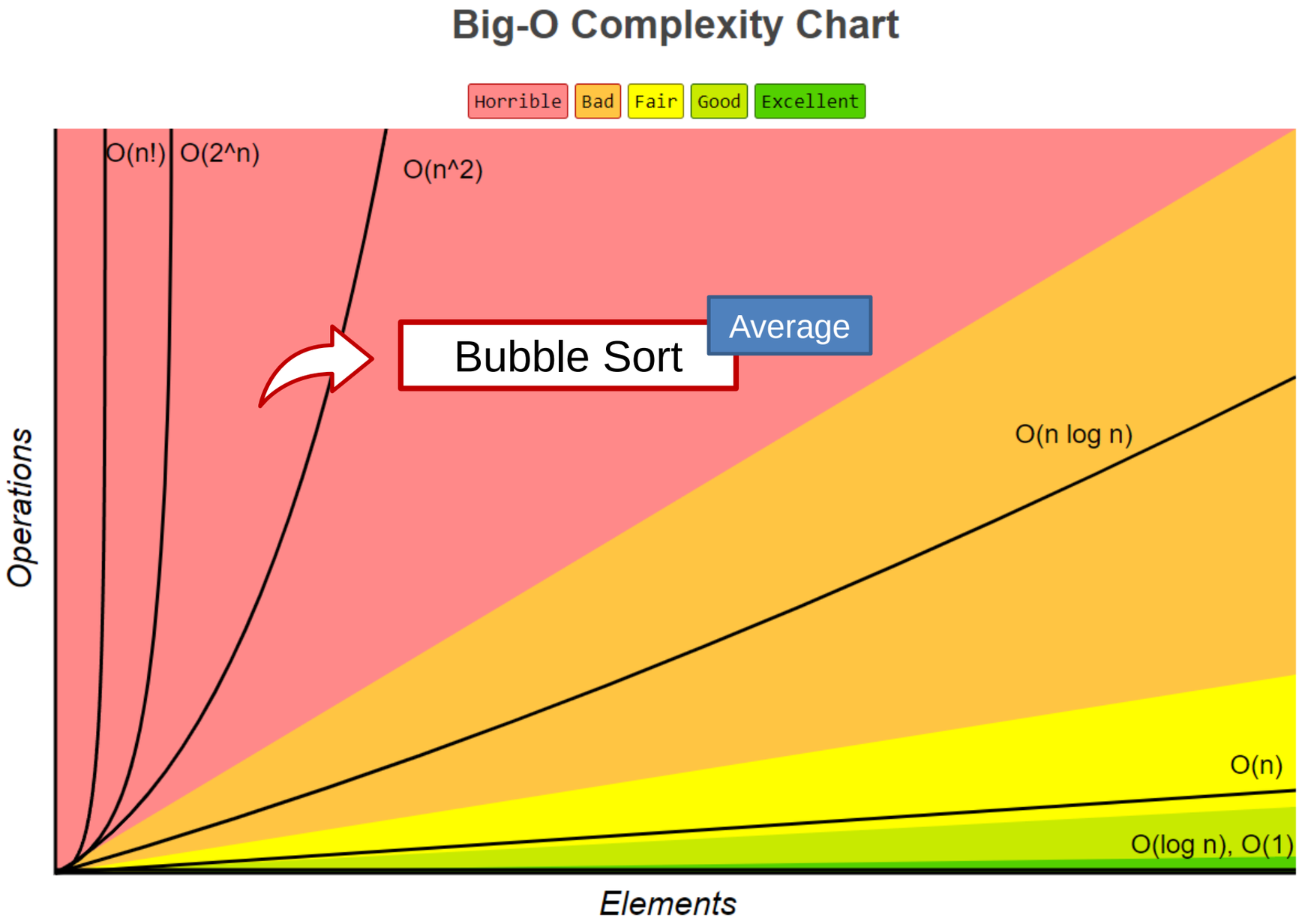
Bubble sort

- **Motivácia:** pedagogický význam a jednoduchosť.
- Pomalý algoritmus, nepraktický, zložitosť $O(n^2)$.
- Stabilný algoritmus.

Big-O Complexity Chart



Zdroj: https://www.bigocheatsheet.com/



Zdroj: https://www.bigocheatsheet.com/

Array Sorting Algorithms

Algorithm	Time Complexity			Space Complexity
	Best	Average	Worst	Worst
Quicksort	Ω(n log(n))	Θ(n log(n))	O(n^2)	O(log(n))
Mergesort	Ω(n log(n))	Θ(n log(n))	O(n log(n))	O(n)
Timsort	Ω(n)	Θ(n log(n))	O(n log(n))	O(n)
Heapsort	Ω(n log(n))	Θ(n log(n))	O(n log(n))	O(1)
Bubble Sort	Ω(n)	Θ(n^2)	O(n^2)	O(1)
Insertion Sort	Ω(n)	Θ(n^2)	O(n^2)	O(1)
Selection Sort	Ω(n^2)	Θ(n^2)	O(n^2)	O(1)
Tree Sort	Ω(n log(n))	Θ(n log(n))	O(n^2)	O(n)
Shell Sort	Ω(n log(n))	Θ(n(log(n))^2)	O(n(log(n))^2)	O(1)
Bucket Sort	Ω(n+k)	Θ(n+k)	O(n^2)	O(n)
Radix Sort	Ω(nk)	Θ(nk)	O(nk)	O(n+k)
Counting Sort	Ω(n+k)	Θ(n+k)	O(n+k)	O(k)
Cubesort	Ω(n)	Θ(n log(n))	O(n log(n))	O(n)

Zdroj: https://www.bigocheatsheet.com/

Bubble sort

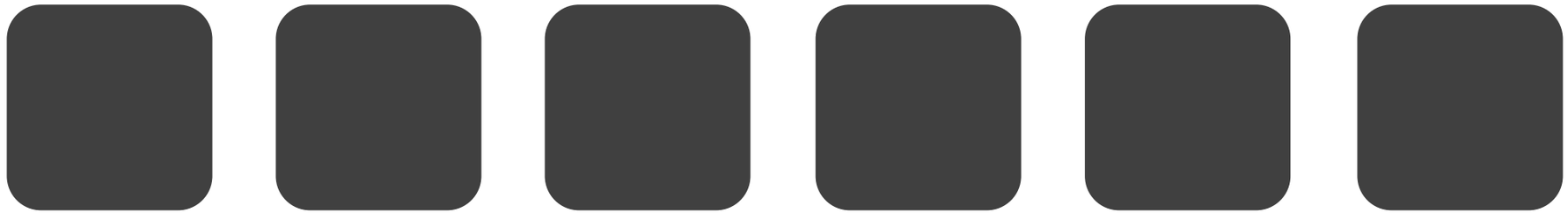
Princíp

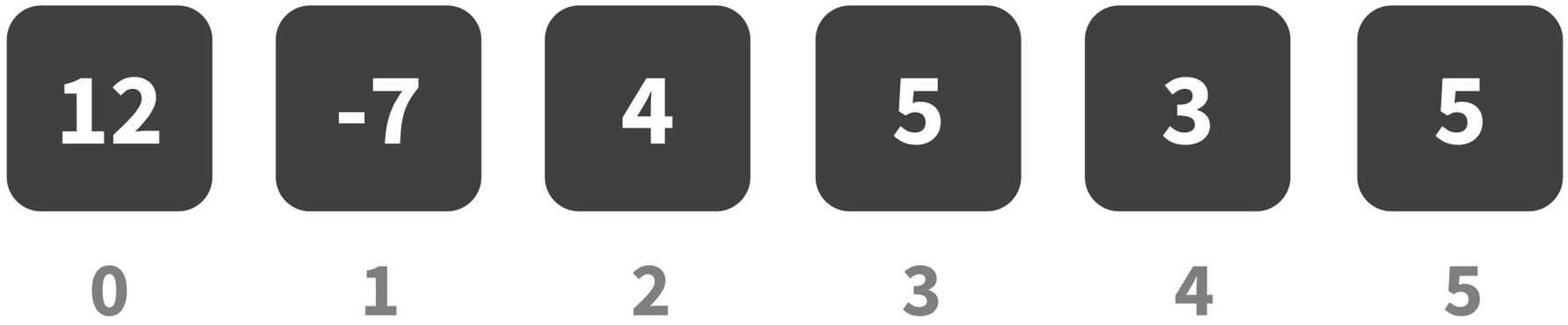
- Opakovaný prechod poľom.
- V každom prechode sa vymieňajú (z angl. swap) prvky, ktoré sú v nesprávnom poradí.
- Po každom prechode sa usporiada jeden prvok.

Bubble sort

Vizualizácia

Pole




12
-7
4
5
3
5
0
1
2
3
4
5

Úloha: zotriediť pole vzostupne

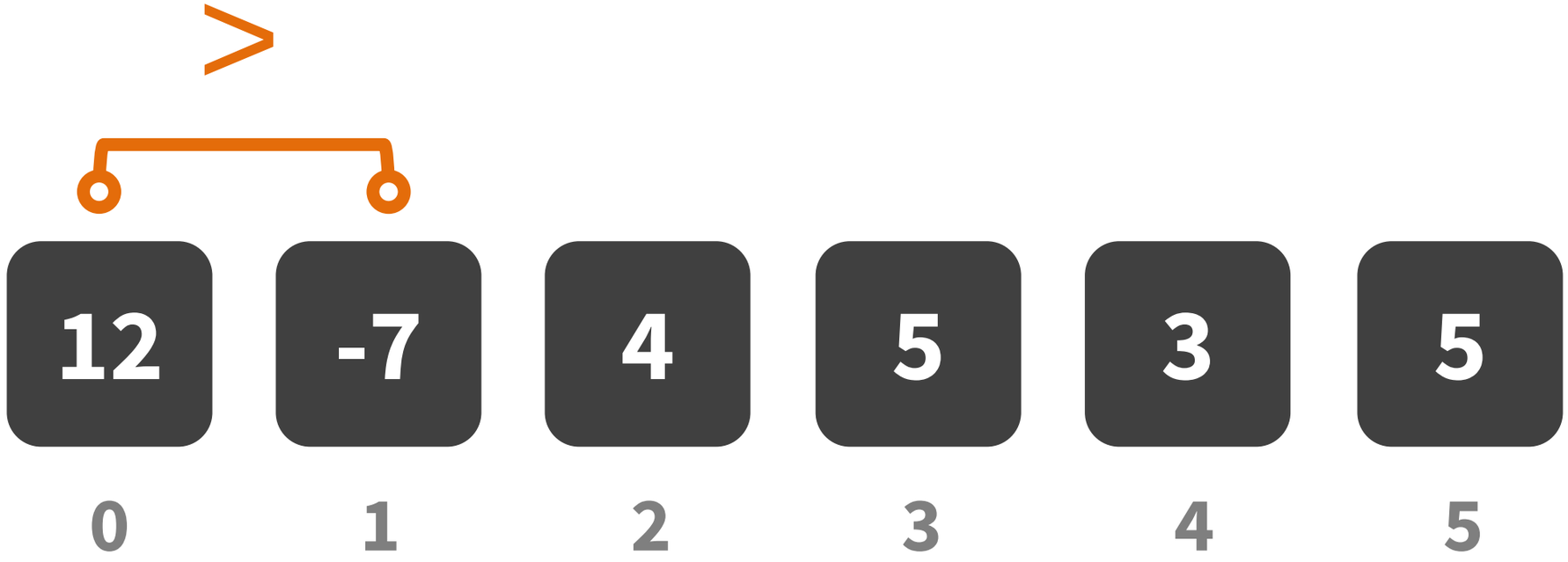
1. prechod poľom

1. prechod

Prechádzame poľom zľava doprava a porovnávame susediace prvky

... ak platí, vykoná sa swap

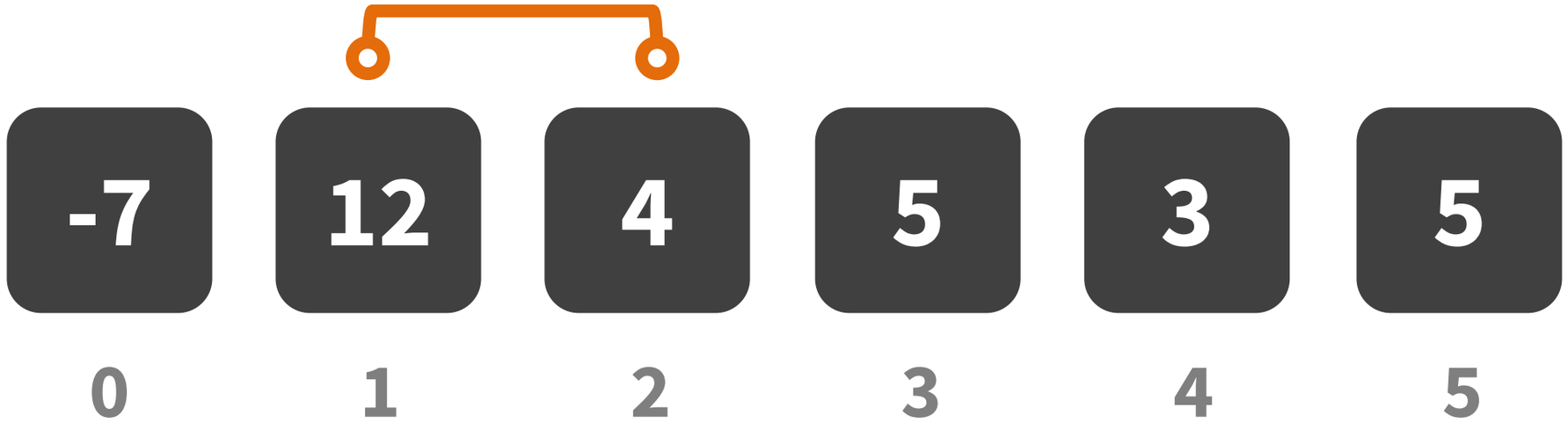
1. prechod



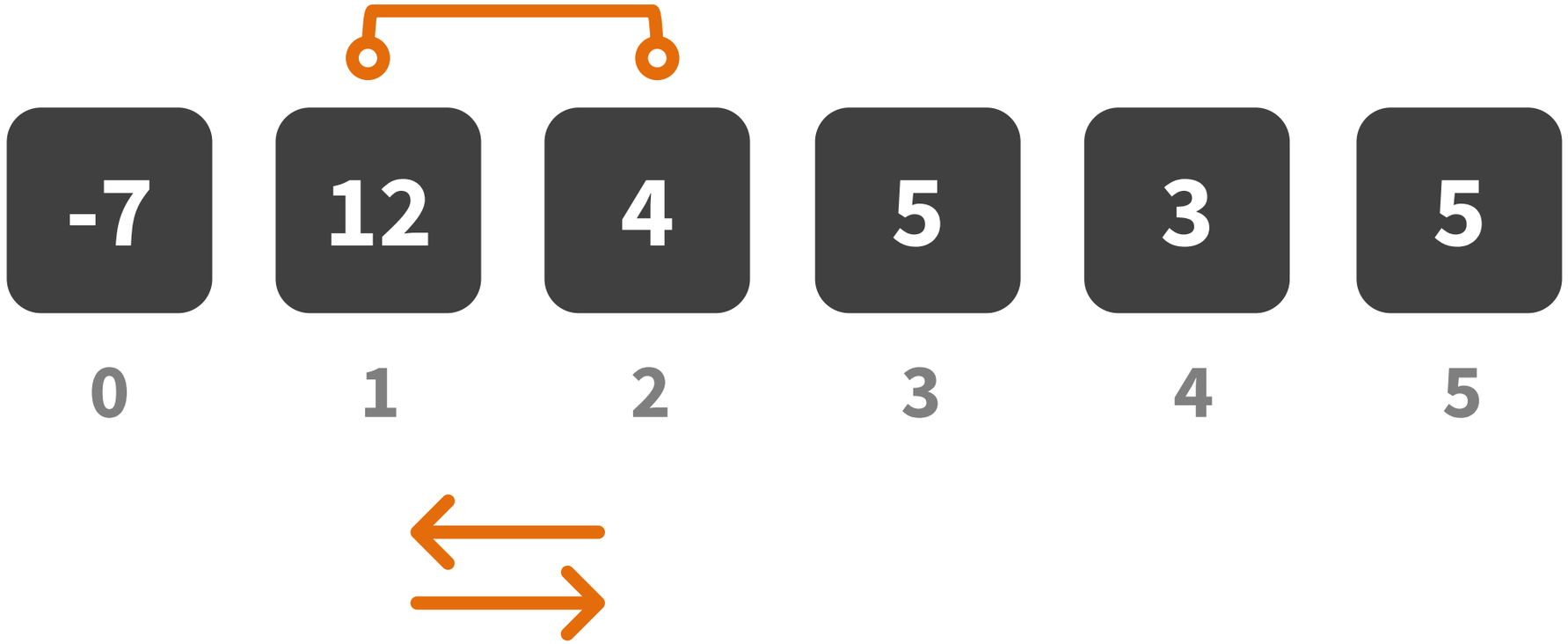
1. prechod

1. prechod

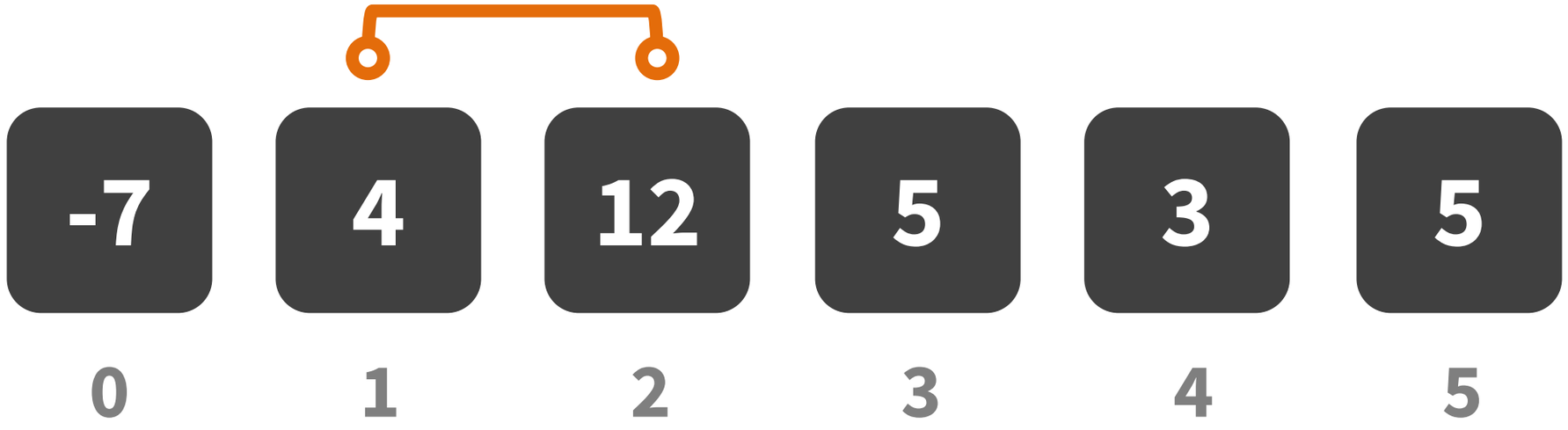
1. prechod



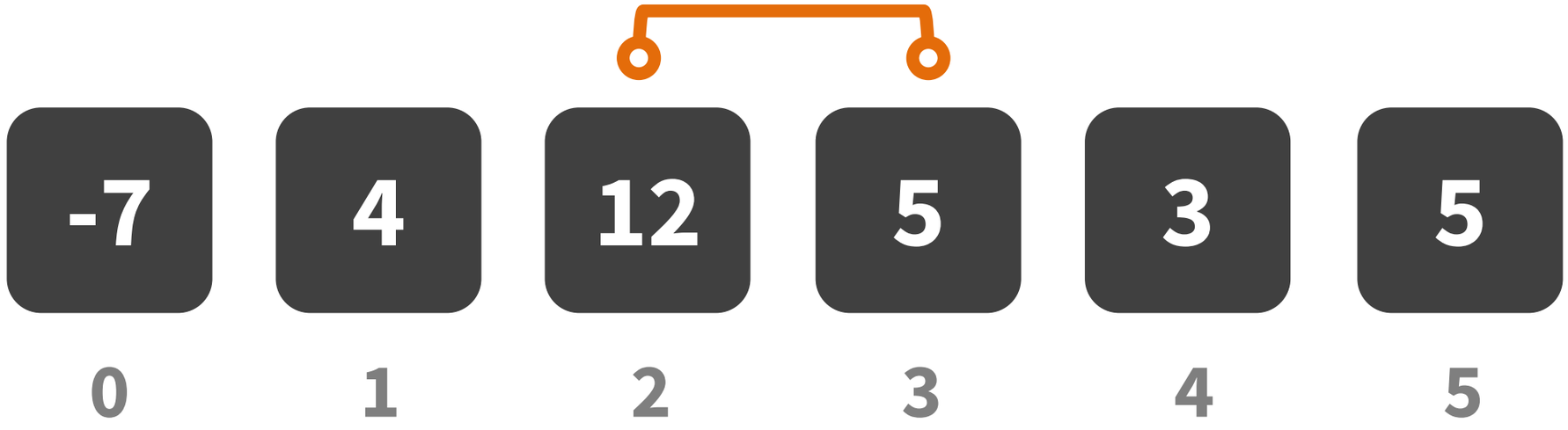
1. prechod

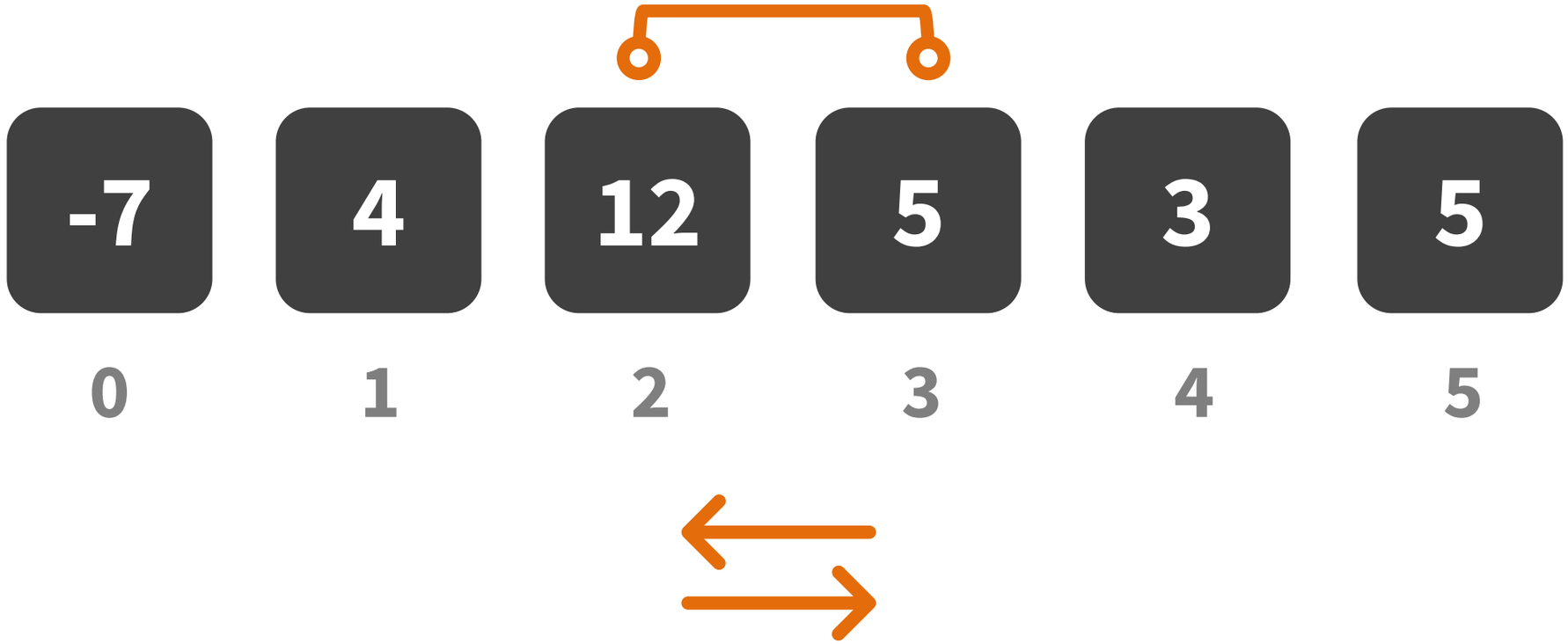


1. prechod

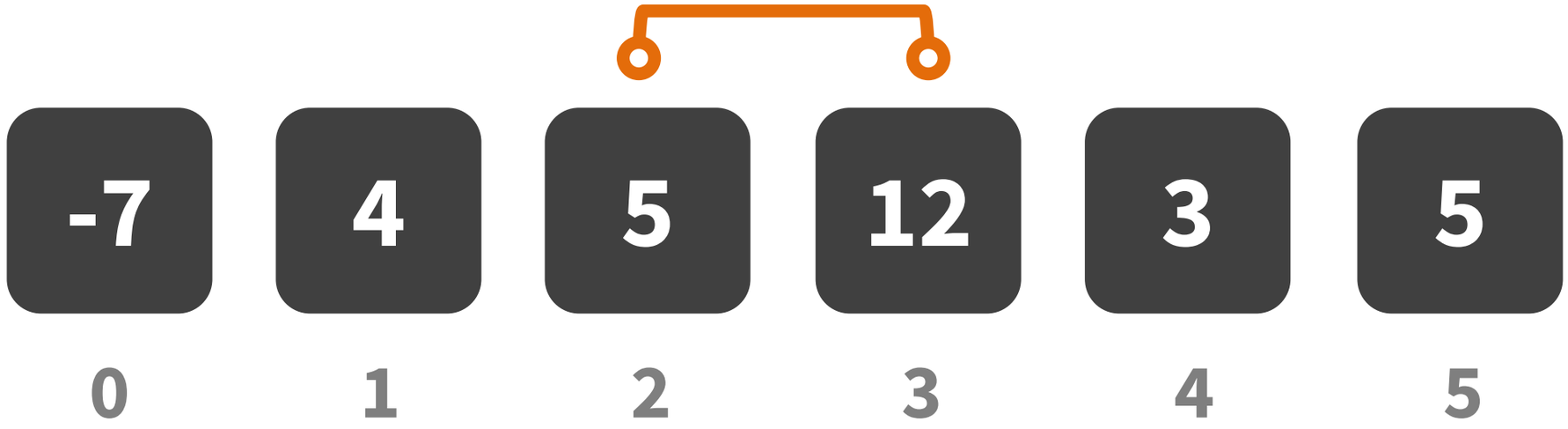


1. prechod

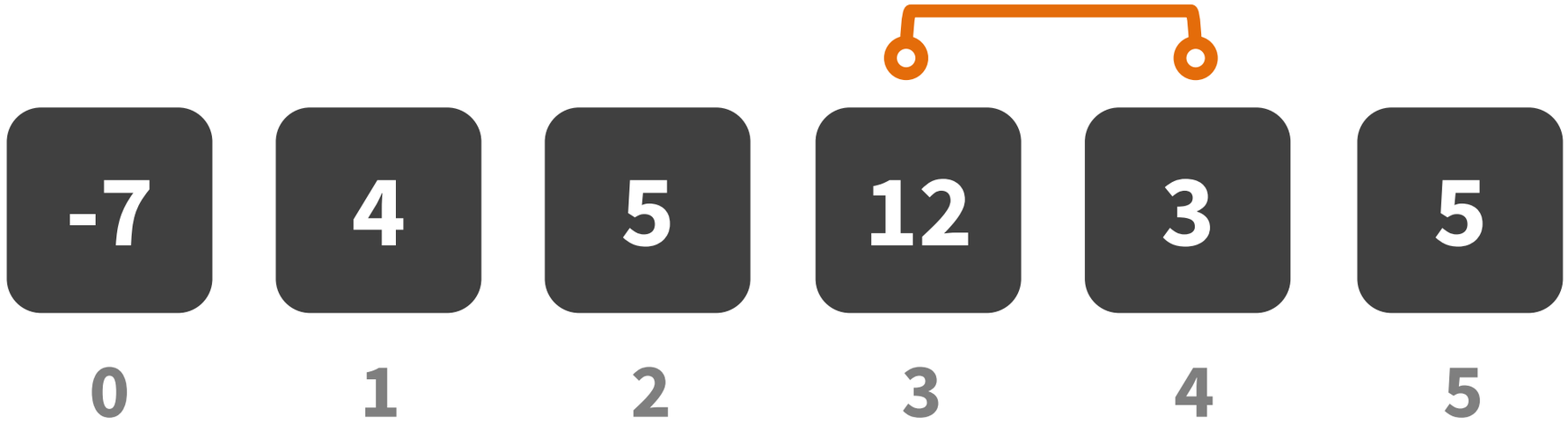



-7
4
12
5
3
5
0
1
2
3
4
5

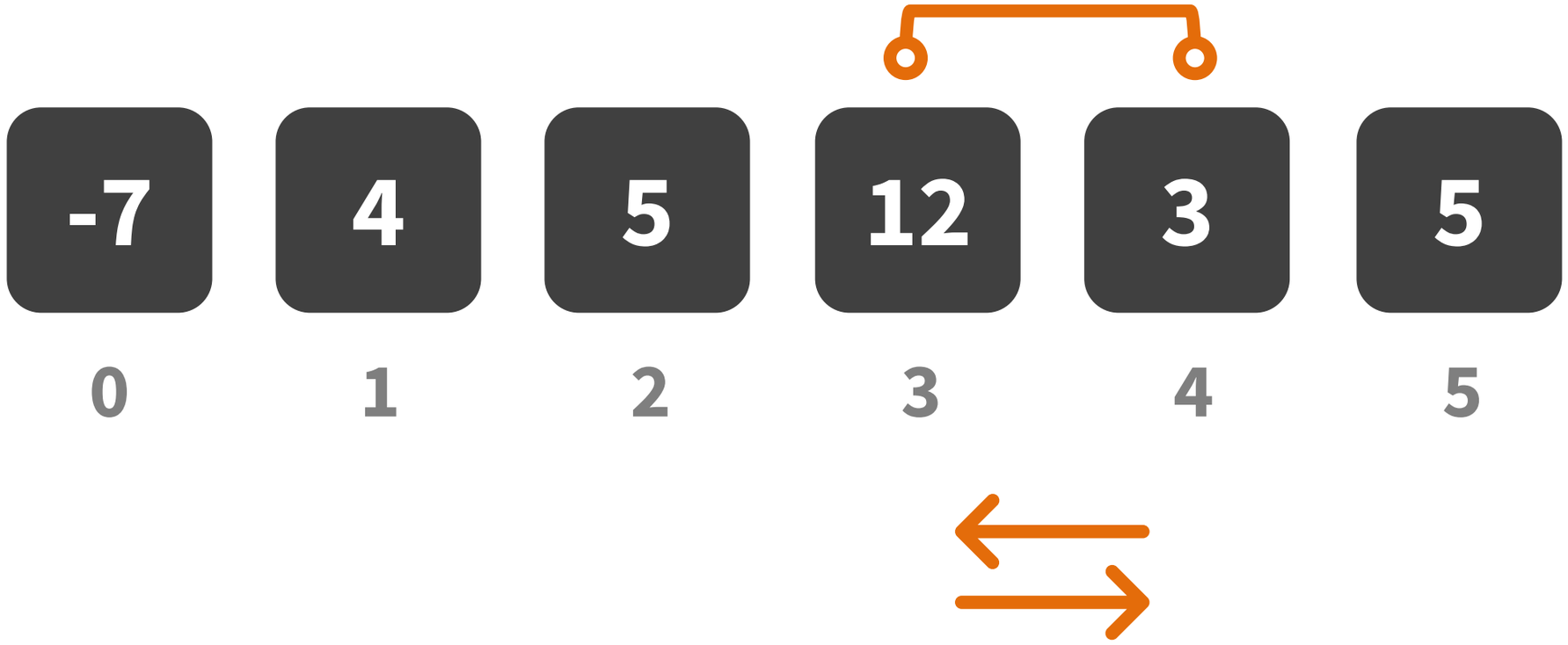
1. prechod



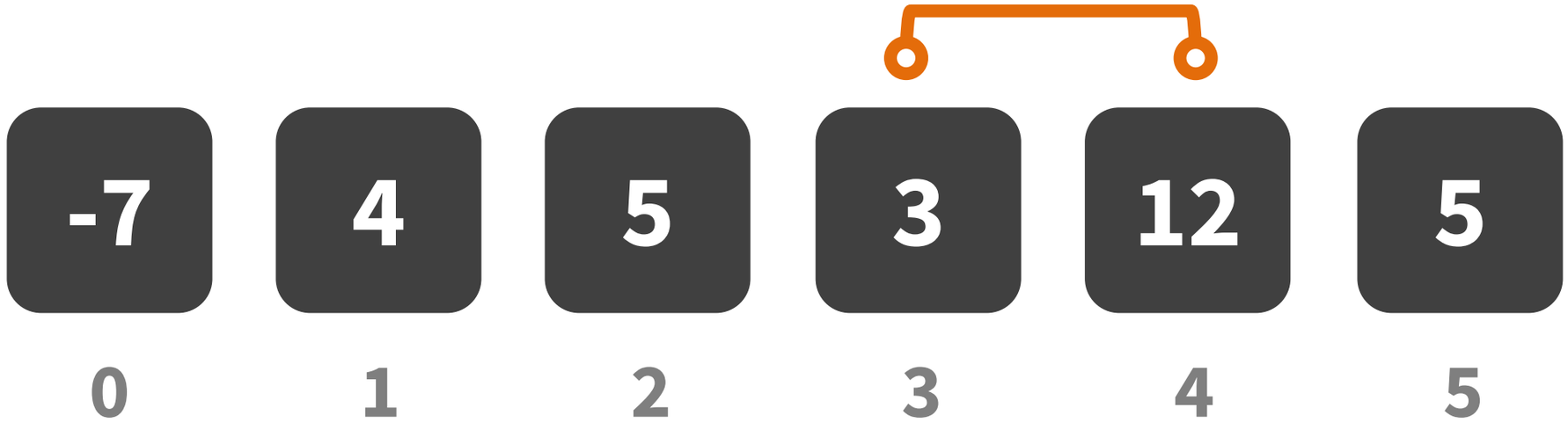
1. prechod



1. prechod



1. prechod



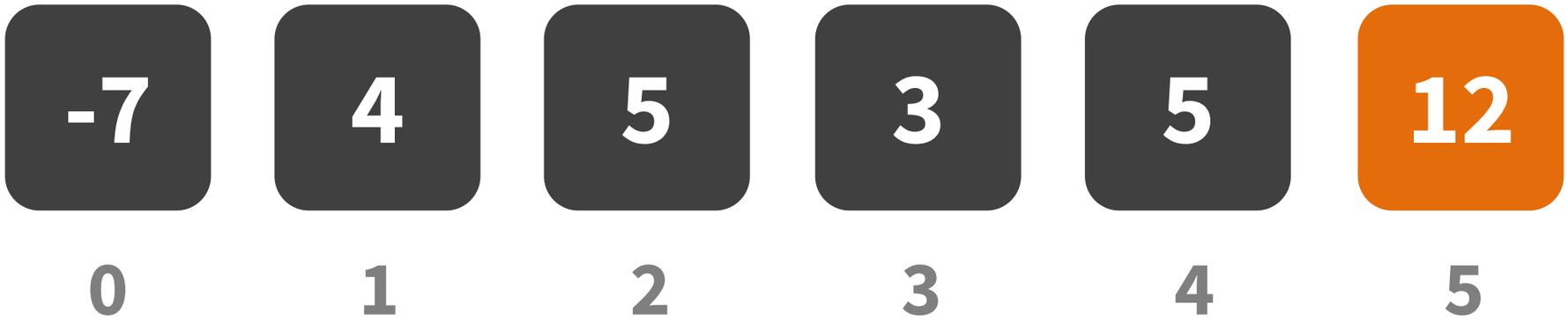
1. prechod


-7
4
5
3
12
5
0
1
2
3
4
5

1. prechod

1. prechod ✓

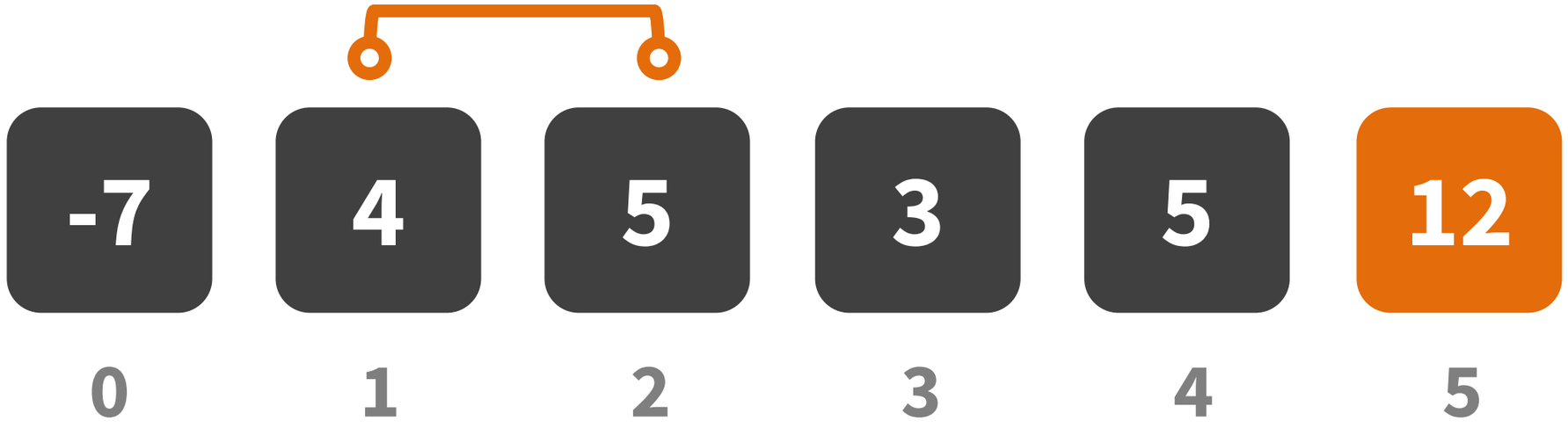
2. prechod poľom



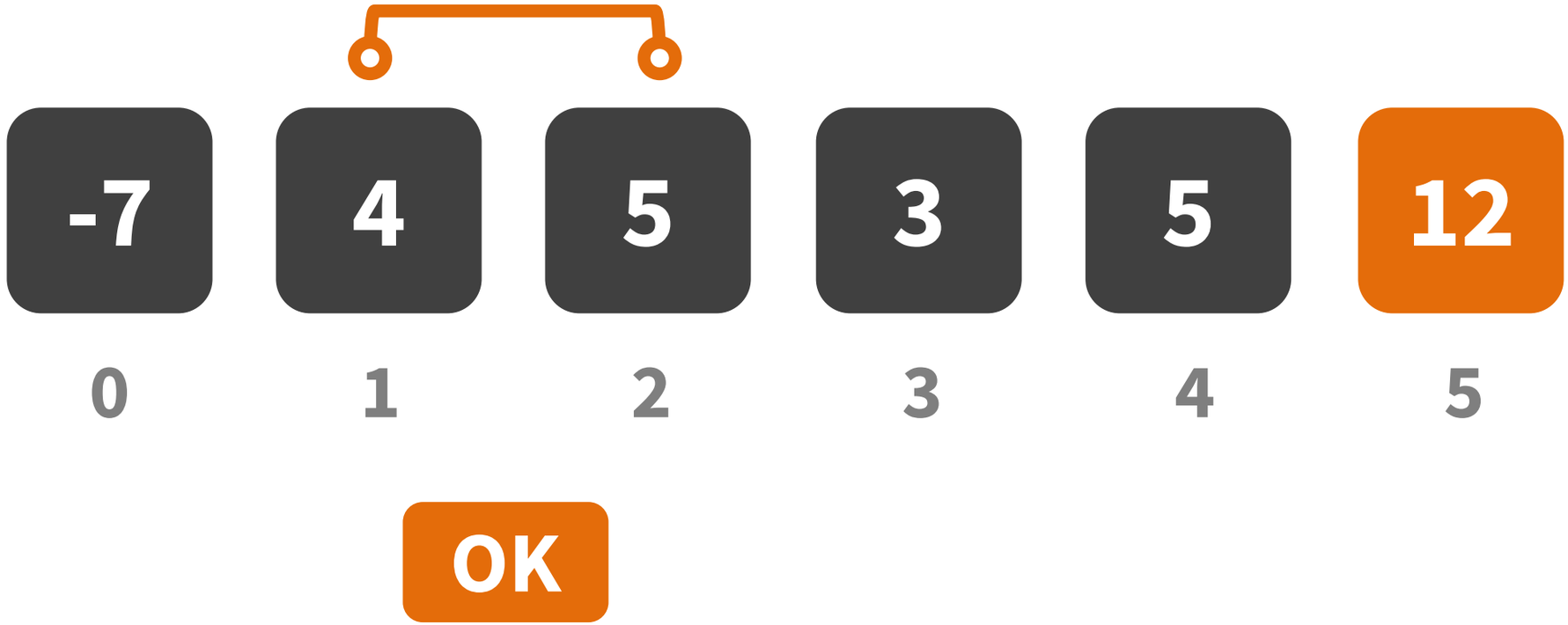
1. prechod ✓
2. prechod

1. prechod ✓
2. prechod

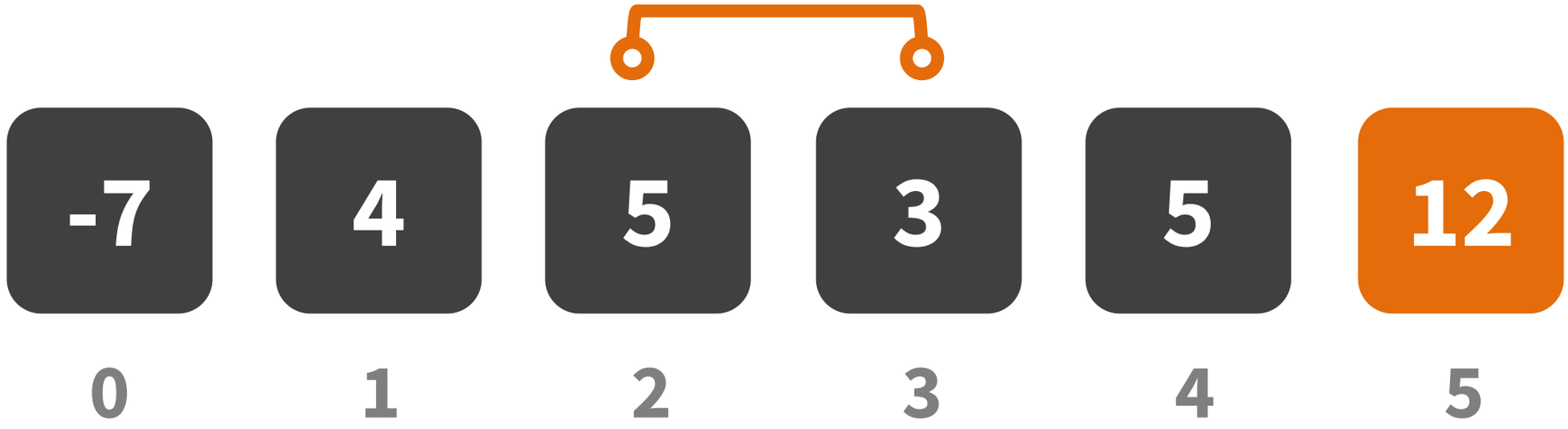
1. prechod ✓
2. prechod



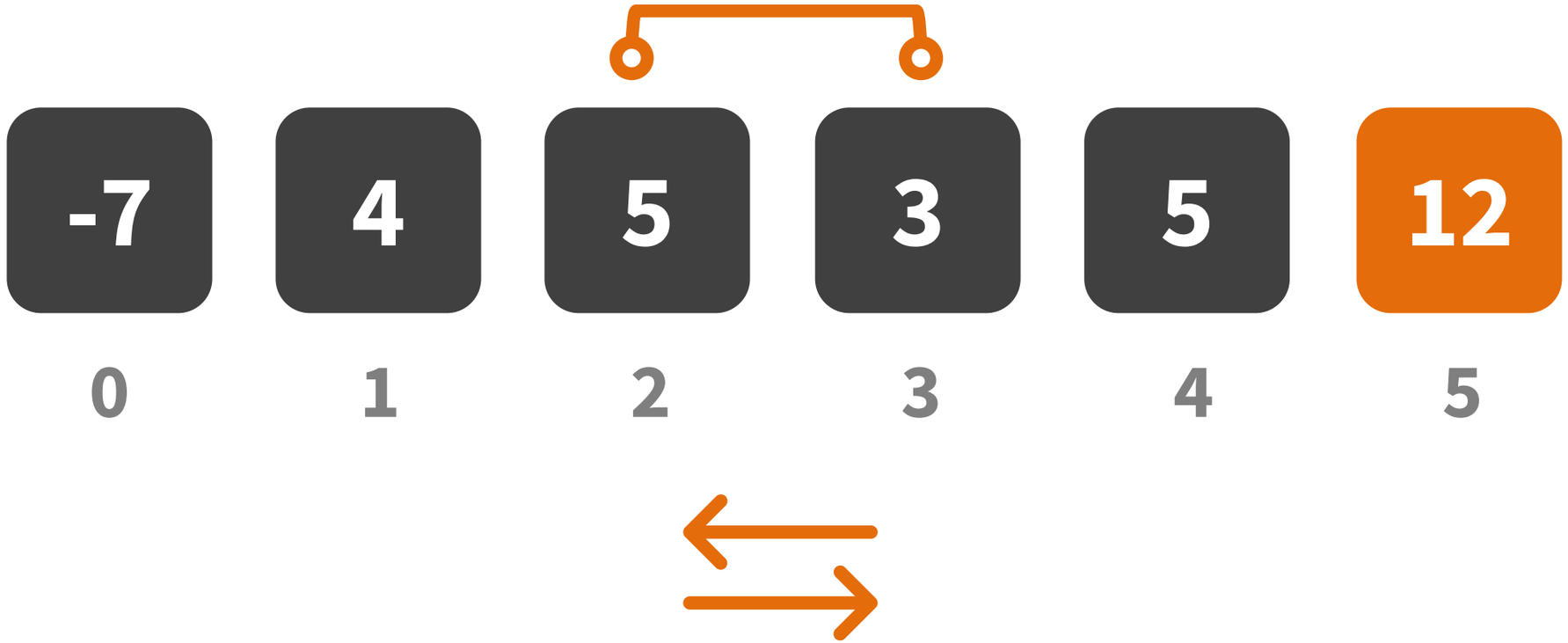
1. prechod ✓
2. prechod

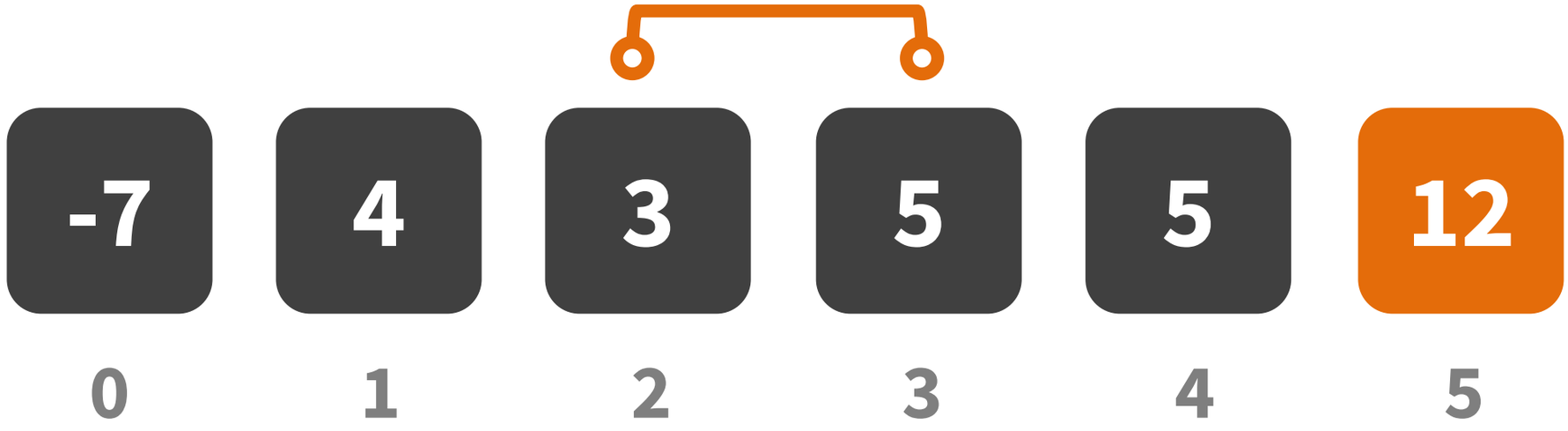


1. prechod ✓
2. prechod

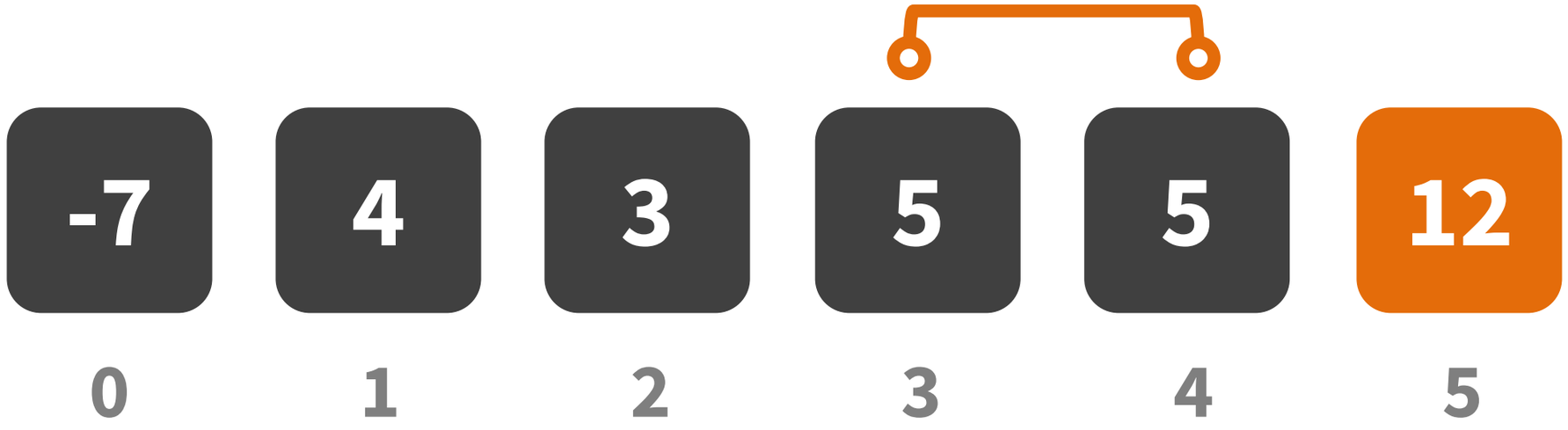


1. prechod ✓
2. prechod

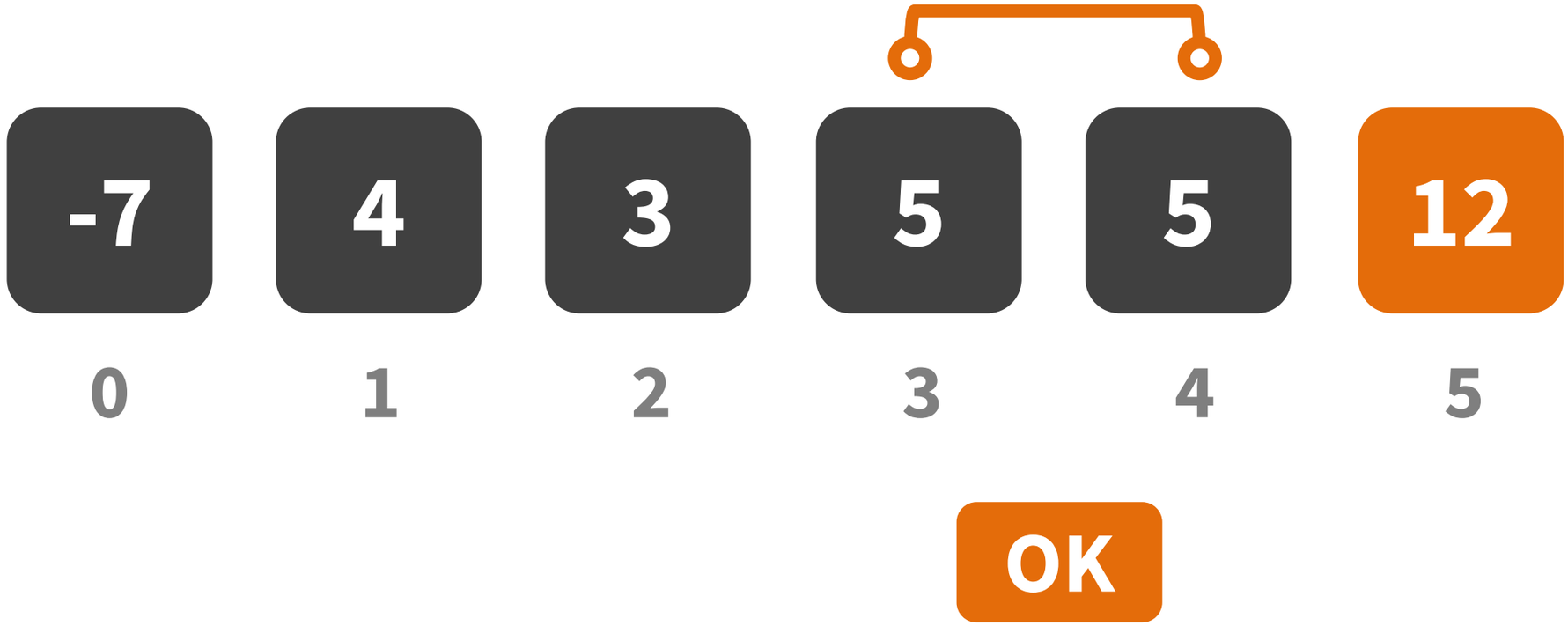



-7
4
3
5
5
12
0
1
2
3
4
5

1. prechod ✓
2. prechod

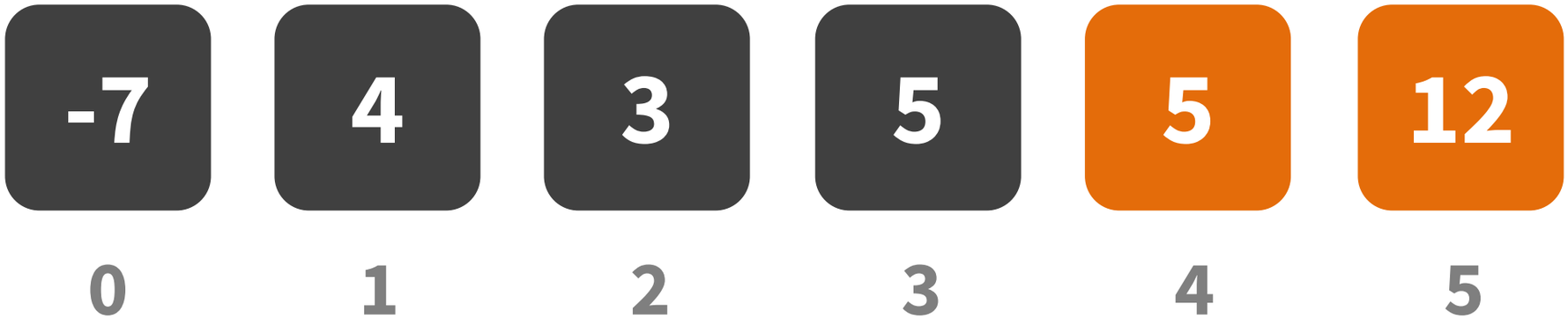


1. prechod ✓
2. prechod



1. prechod ✓
2. prechod ✓

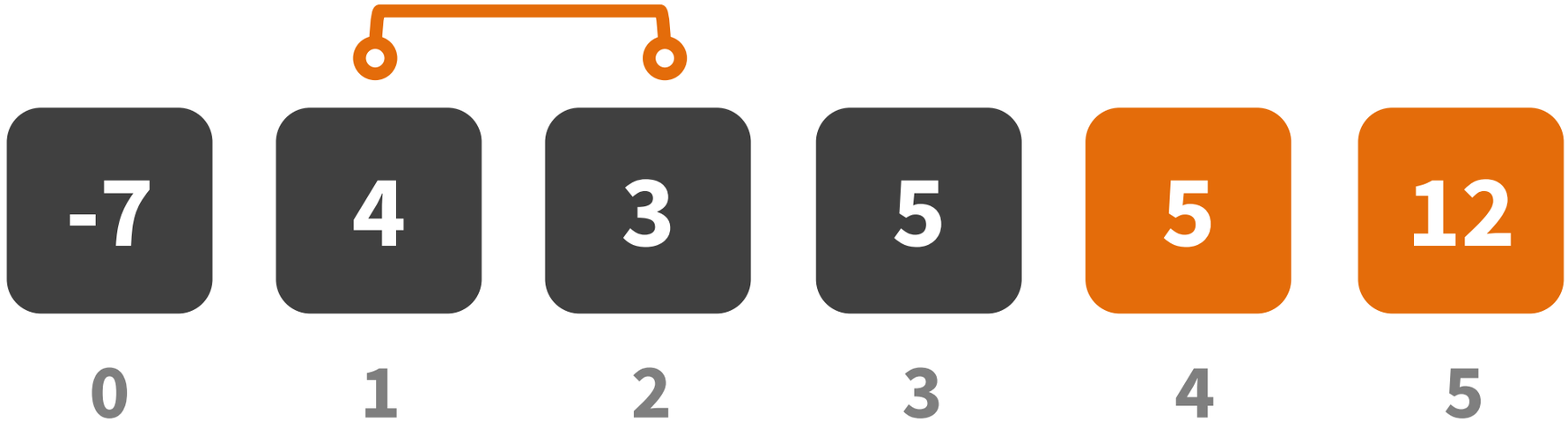
3. prechod poľom



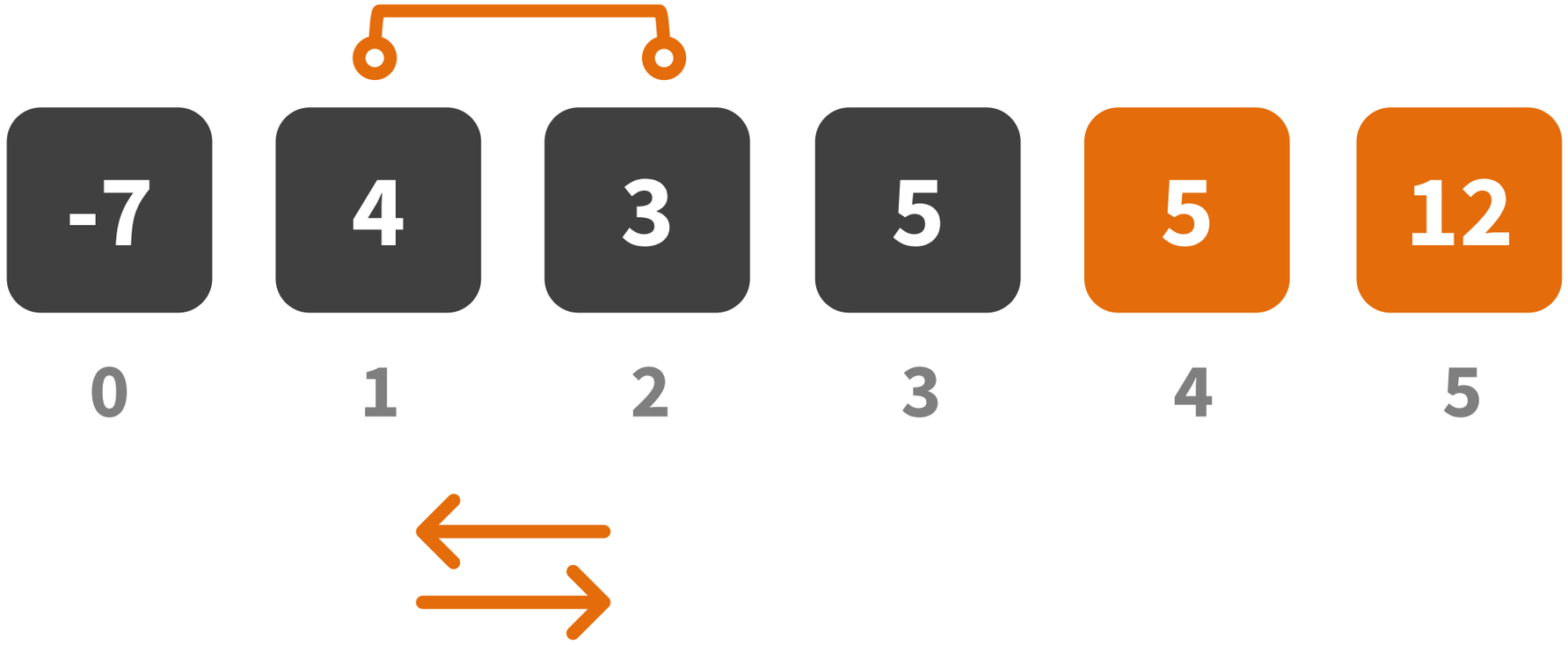
1. prechod ✓
2. prechod ✓
3. prechod

1. prechod ✓
2. prechod ✓
3. prechod

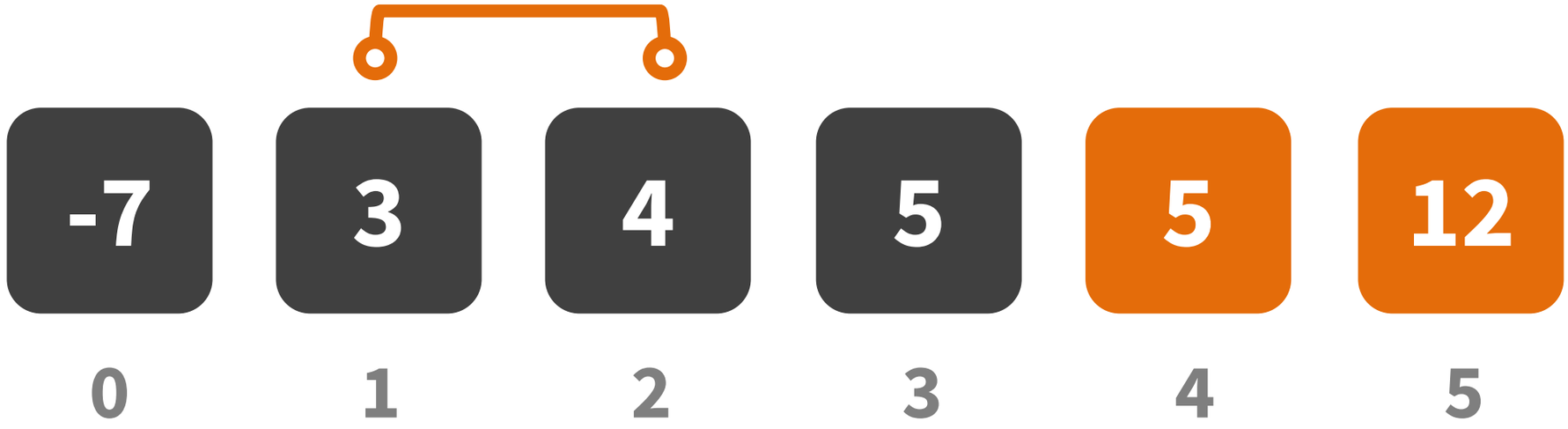
1. prechod ✓
2. prechod ✓
3. prechod



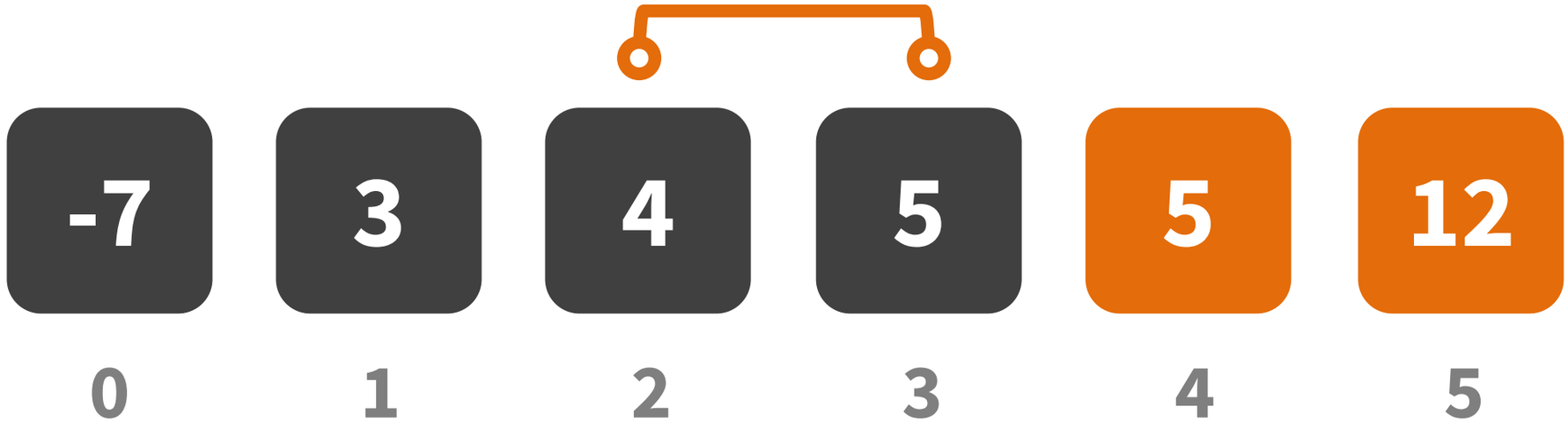
1. prechod ✓
2. prechod ✓
3. prechod



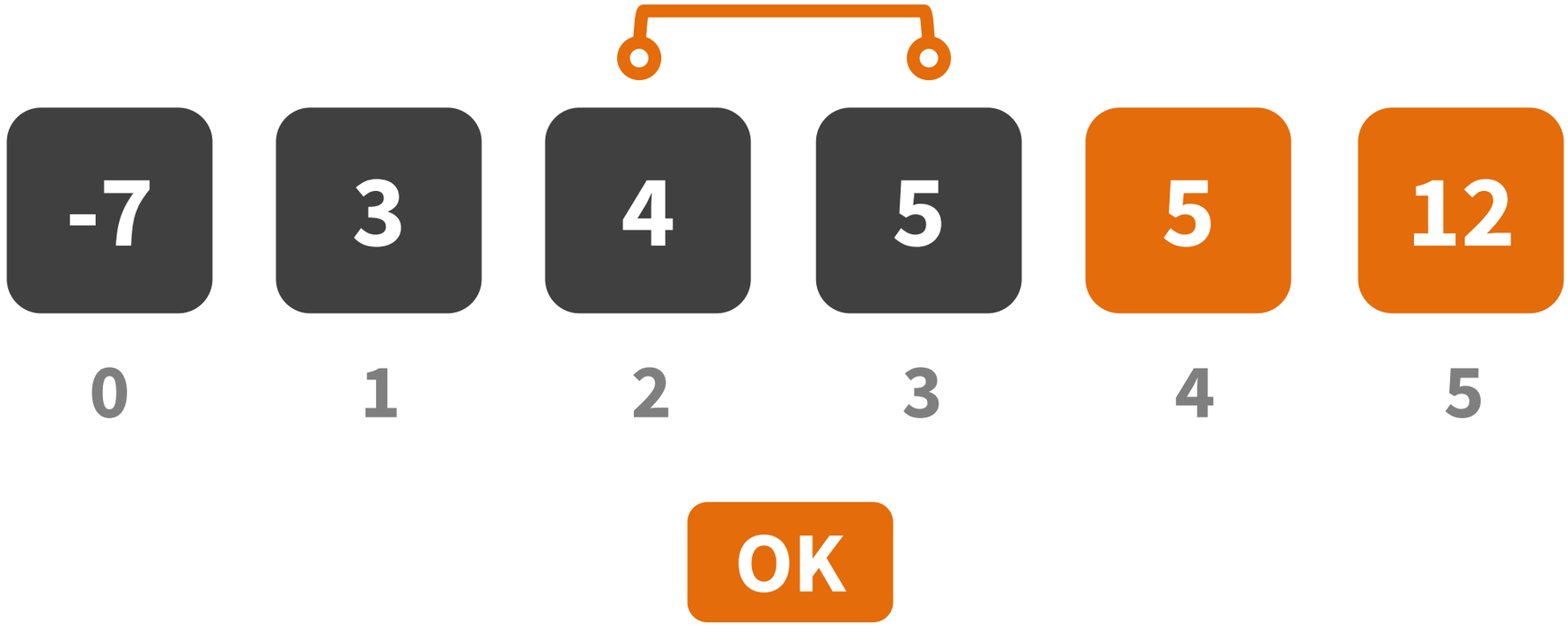
1. prechod ✓
2. prechod ✓
3. prechod



1. prechod ✓
2. prechod ✓
3. prechod

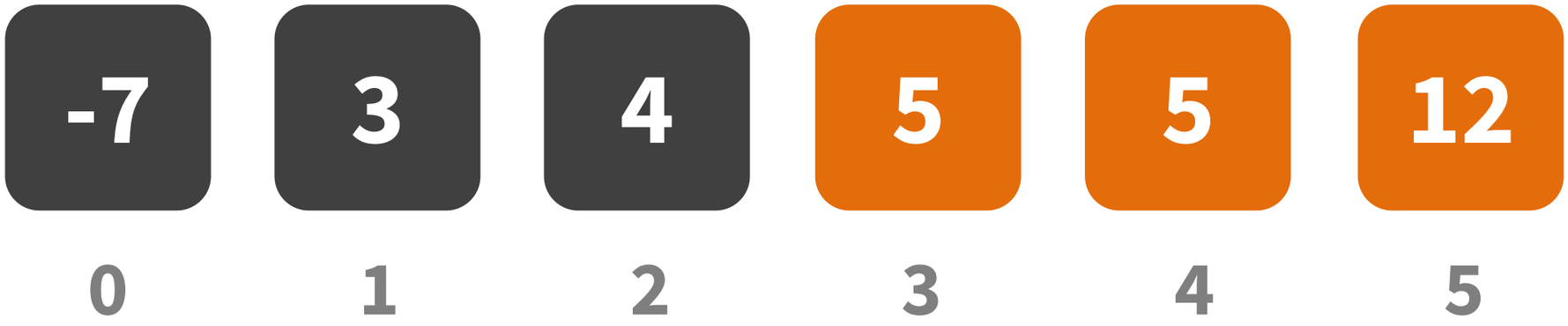


1. prechod ✓
2. prechod ✓
3. prechod



1. prechod ✓
2. prechod ✓
3. prechod ✓

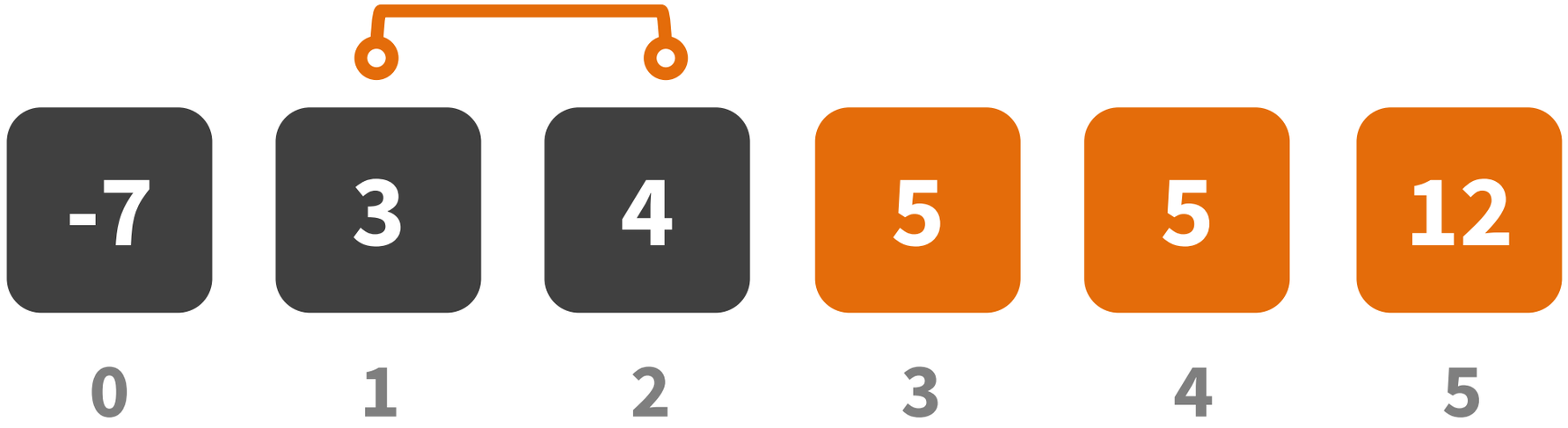
4. prechod poľom



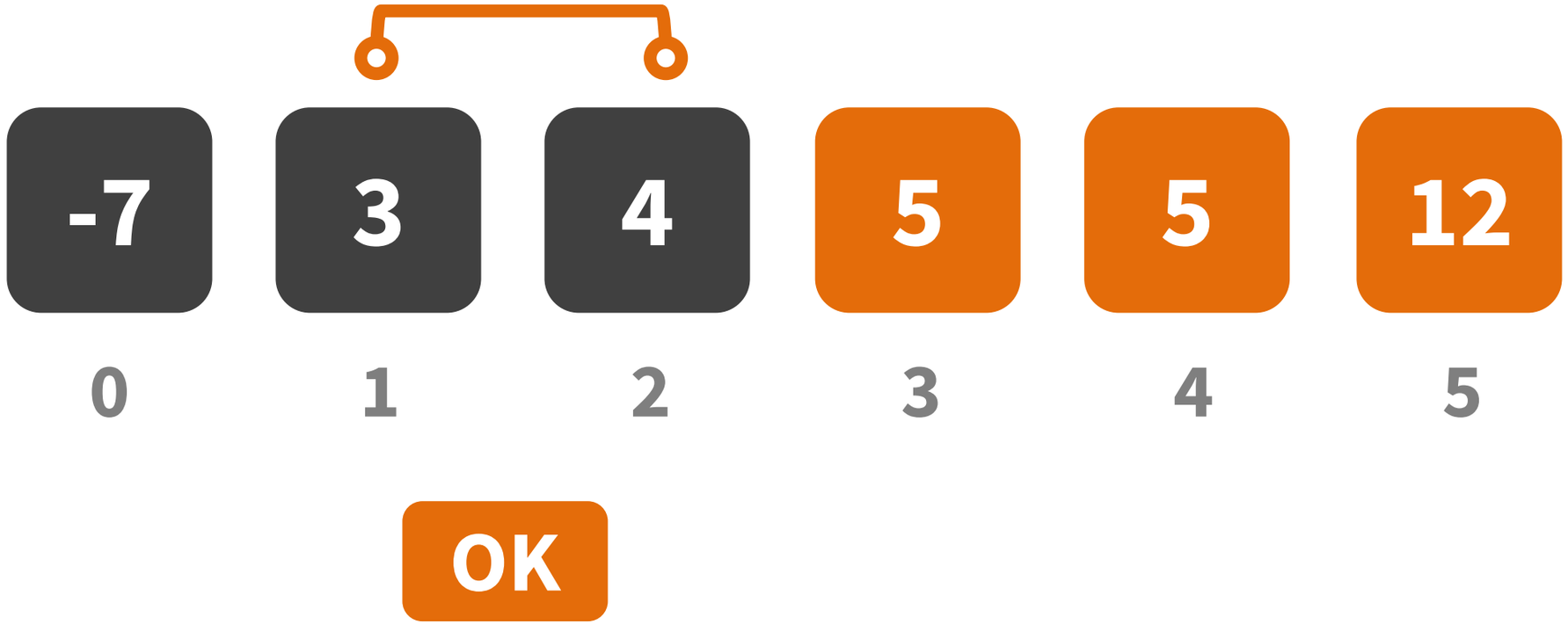
1. prechod ✓
2. prechod ✓
3. prechod ✓
4. prechod

1. prechod ✓
2. prechod ✓
3. prechod ✓
4. prechod

1. prechod ✓
2. prechod ✓
3. prechod ✓
4. prechod



1. prechod ✓
2. prechod ✓
3. prechod ✓
4. prechod



1. prechod ✓
2. prechod ✓
3. prechod ✓
4. prechod ✓

5. prechod poľom

1. prechod ✓
2. prechod ✓
3. prechod ✓
4. prechod ✓
5. prechod

1. prechod ✓
2. prechod ✓
3. prechod ✓
4. prechod ✓
5. prechod

1. prechod ✓
2. prechod ✓
3. prechod ✓
4. prechod ✓
5. prechod ✓

Posledný prvok je na správnej pozícii.

1. prechod ✓
2. prechod ✓
3. prechod ✓
4. prechod ✓
5. prechod ✓

Pole je usporiadané

Analýza zložitosti

Počet prvkov poľa **n=6**

Počet prechodov poľom: **5** = (n-1)

Prechod	**1**	**2**	**3**	**4**	**5**
Počet porovnaní	5	4	3	2	1

Vzorová implementácia v C/C++