CS50 Test Review

Algorithms and Programming Languages

Representation

Decimal

0 1 2 3 4 5 6 7 8 9

Decimal

123

100s 10s 1s

Decimal

123

100s 10s 1s

1 × 100

Decimal

123

100s 10s 1s

$1 \times 100 + 2 \times 10$

Decimal

123

100s 10s 1s

$1 \times 100 + 2 \times 10 + 3 \times 1$

Decimal

123

100s 10s 1s

$1 \times 100 + 2 \times 10 + 3 \times 1$

Decimal

123

100s 10s 1s

100 + 20 + 3

Decimal

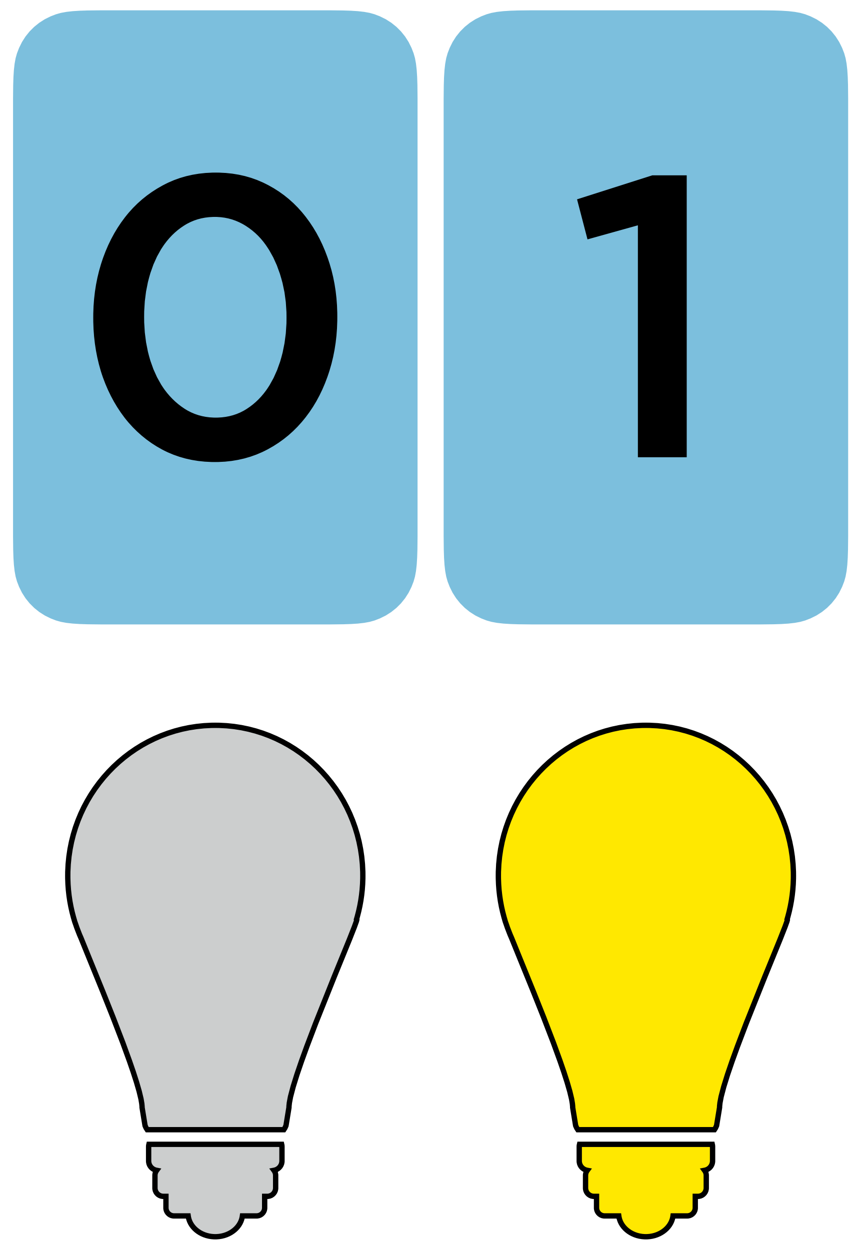
123

10^2 10^1 10^0

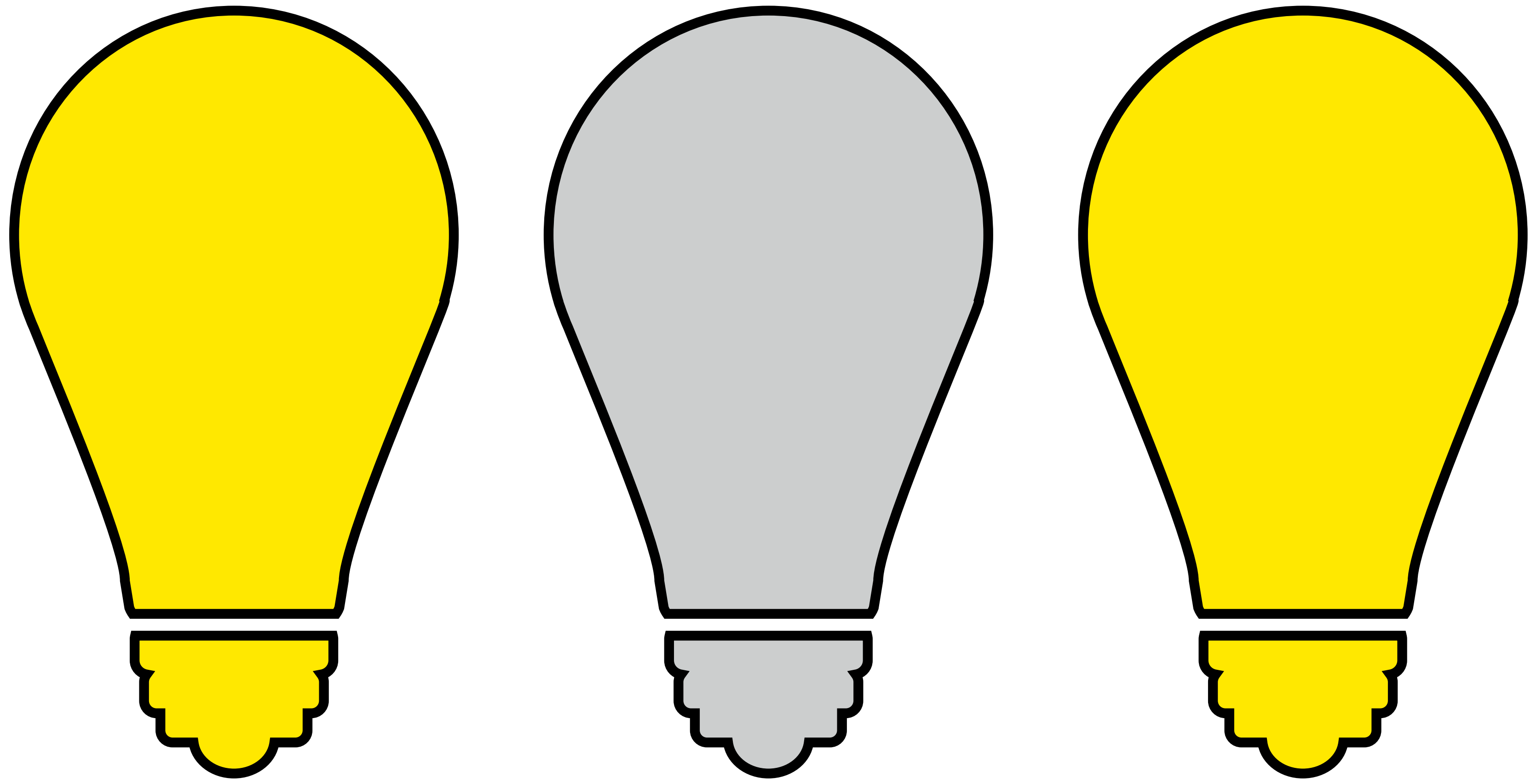
Decimal

0	1	2	3	4	5	6	7	8	9

Binary



Binary



Binary

101

2^2 2^1 2^0

Binary

101

4s 2s 1s

$1 \times 4 + 0 \times 2 + 1 \times 1$

Binary

101

4s 2s 1s

4 + 0 + 1

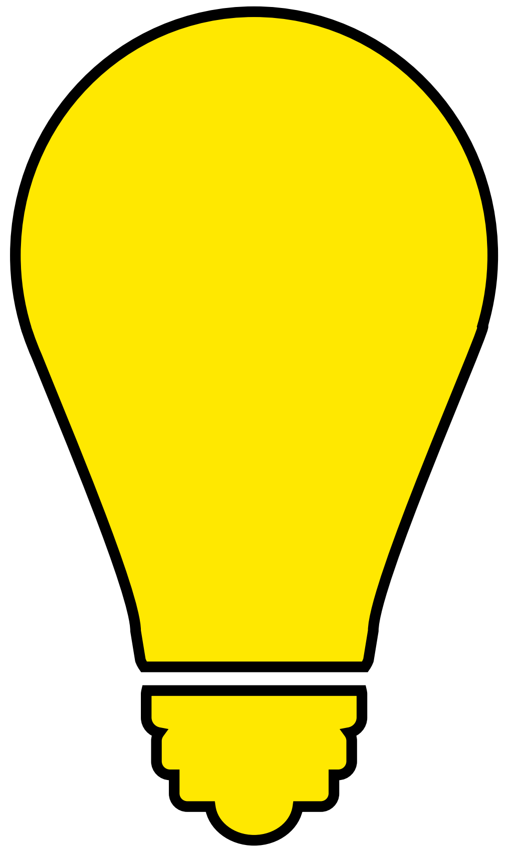
Binary

101

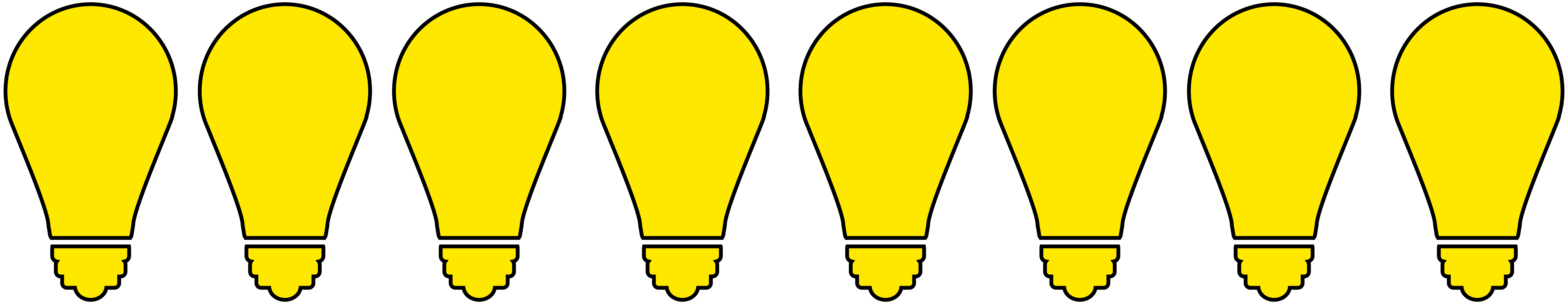
4s 2s 1s

5

Bit



Byte



Numbers

0	00000000
255	11111111

ASCII

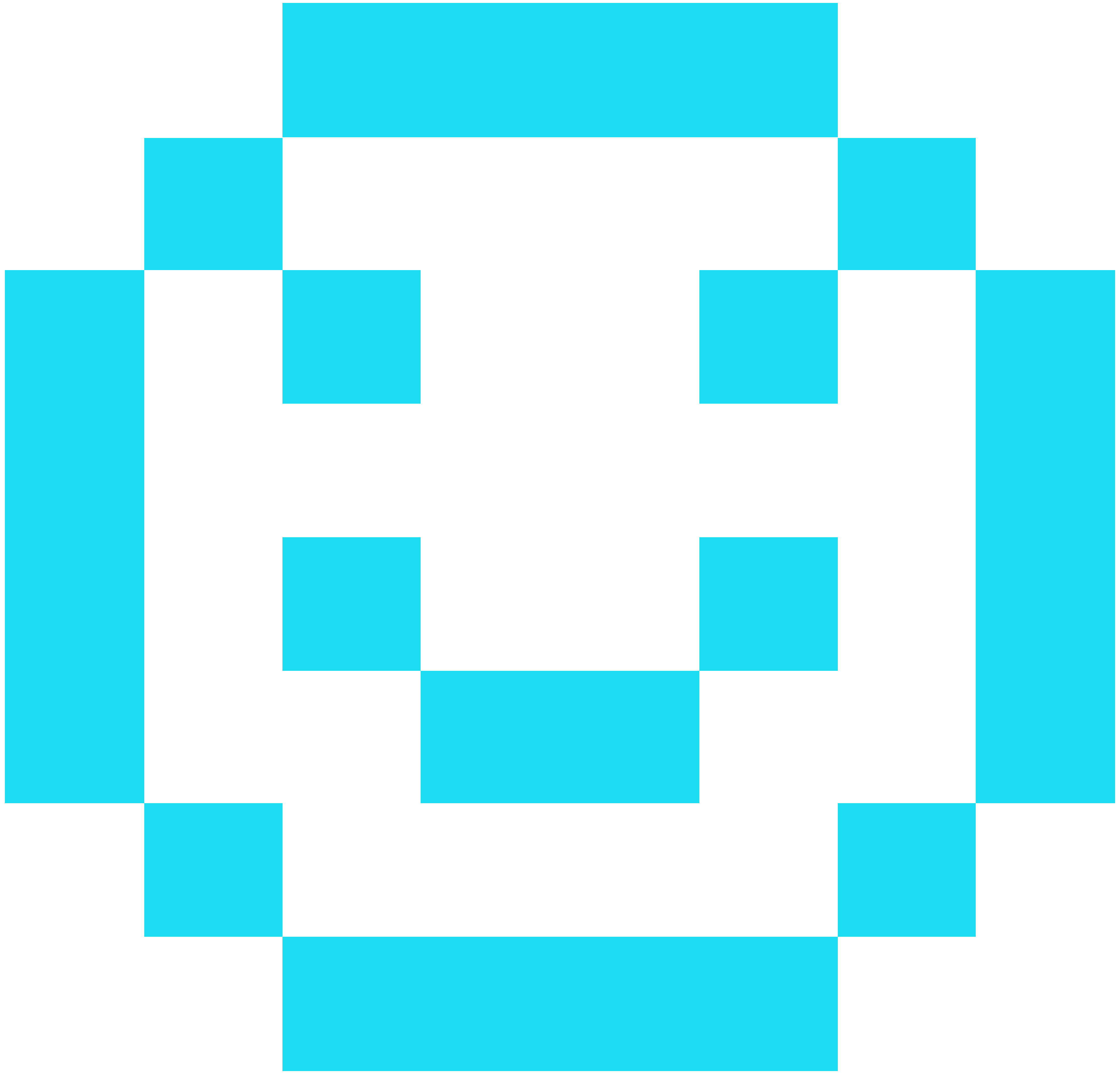
A	65	01000001
B	66	01000010
C	67	01000011

ASCII

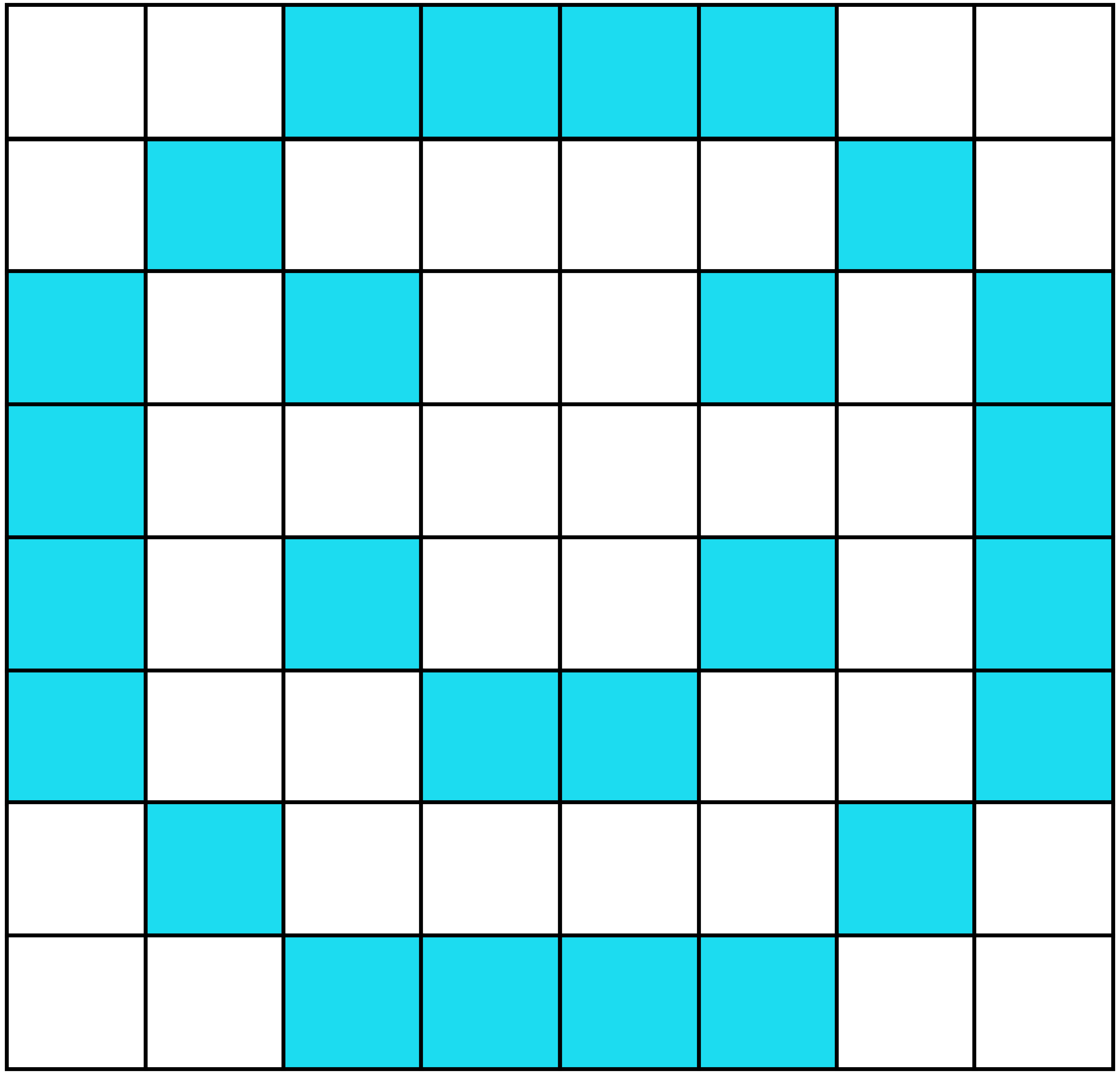
Hi!

H	i	!
72	105	33
01001000	01101001	00100001

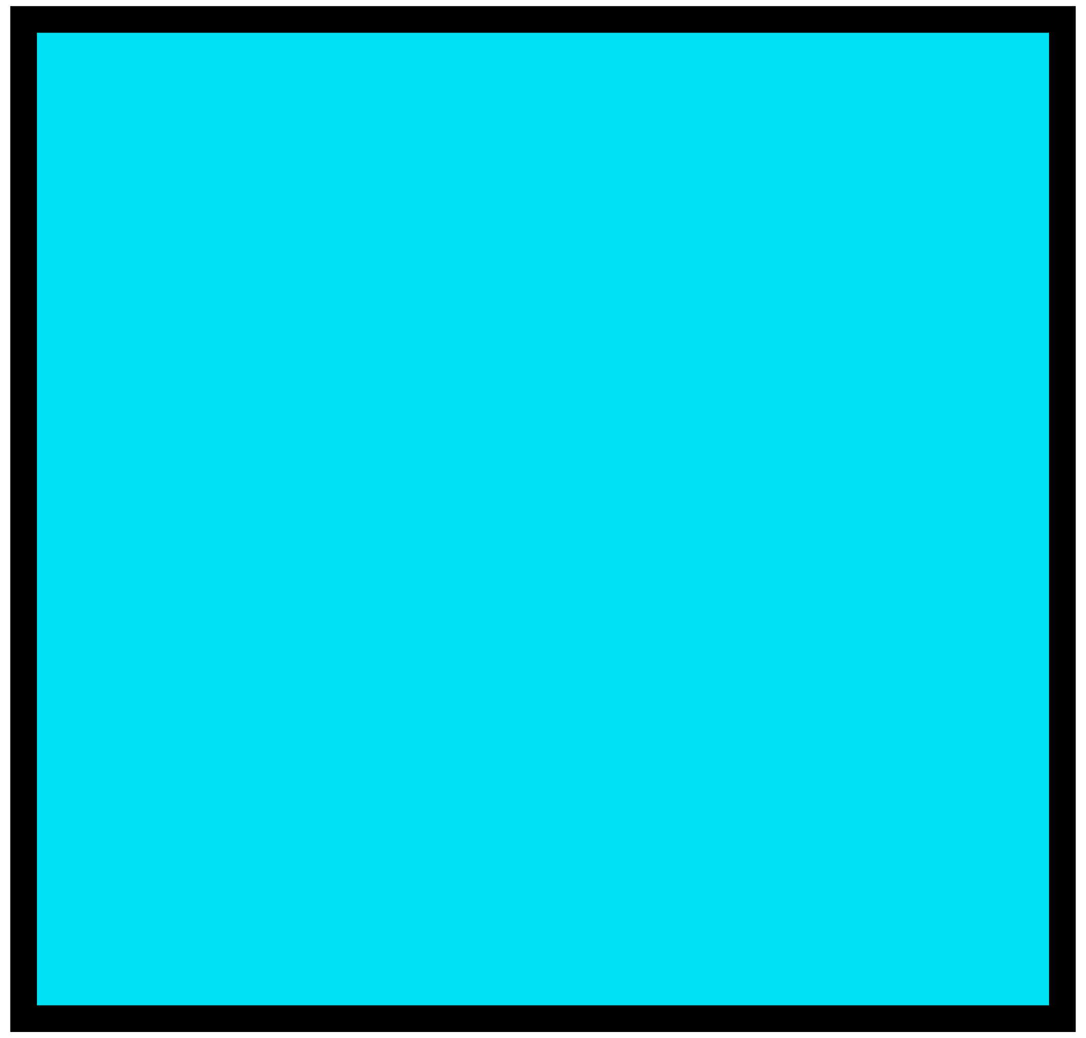
Images



Images



Images



28 220 240

00011100 11011100 11110000

Integer Overflow

`00000000000000000000000000000000`

0

Integer Overflow

`00000000000000000000000000000001`

1

Integer Overflow

00000000000000000000000000000010

2

Integer Overflow

00000000000000000000000000000011

3

Integer Overflow

`00000000000000000000000000000100`

4

Integer Overflow

`11111111111111111111111111111110`

4,294,967,294

Integer Overflow

11111111111111111111111111111111

4,294,967,295

Integer Overflow

`00000000000000000000000000000000`

0

Programming

Compiling

Source Code

```
#include <stdio.h>

int main(void)
{
    printf("hello, world!\n");
}
```

→

Machine Code

```
01111111 01000101 01001100
01000110 00000010 00000001
00000001 00000000 00000000
00000000 00000000 00000000
00000000 00000000 00000000
00000000 00000010 00000000
00111110 00000000 00000001 ...
```

Programming Languages

C

```c
#include <stdio.h>

int main(void)
{
    printf("hello, world!\n");
}
```

Python

```python
print("hello, world")
```

Variables

C

```c
int x = 10;
```

Python

```python
x = 10
```

Common Types in C

`bool`

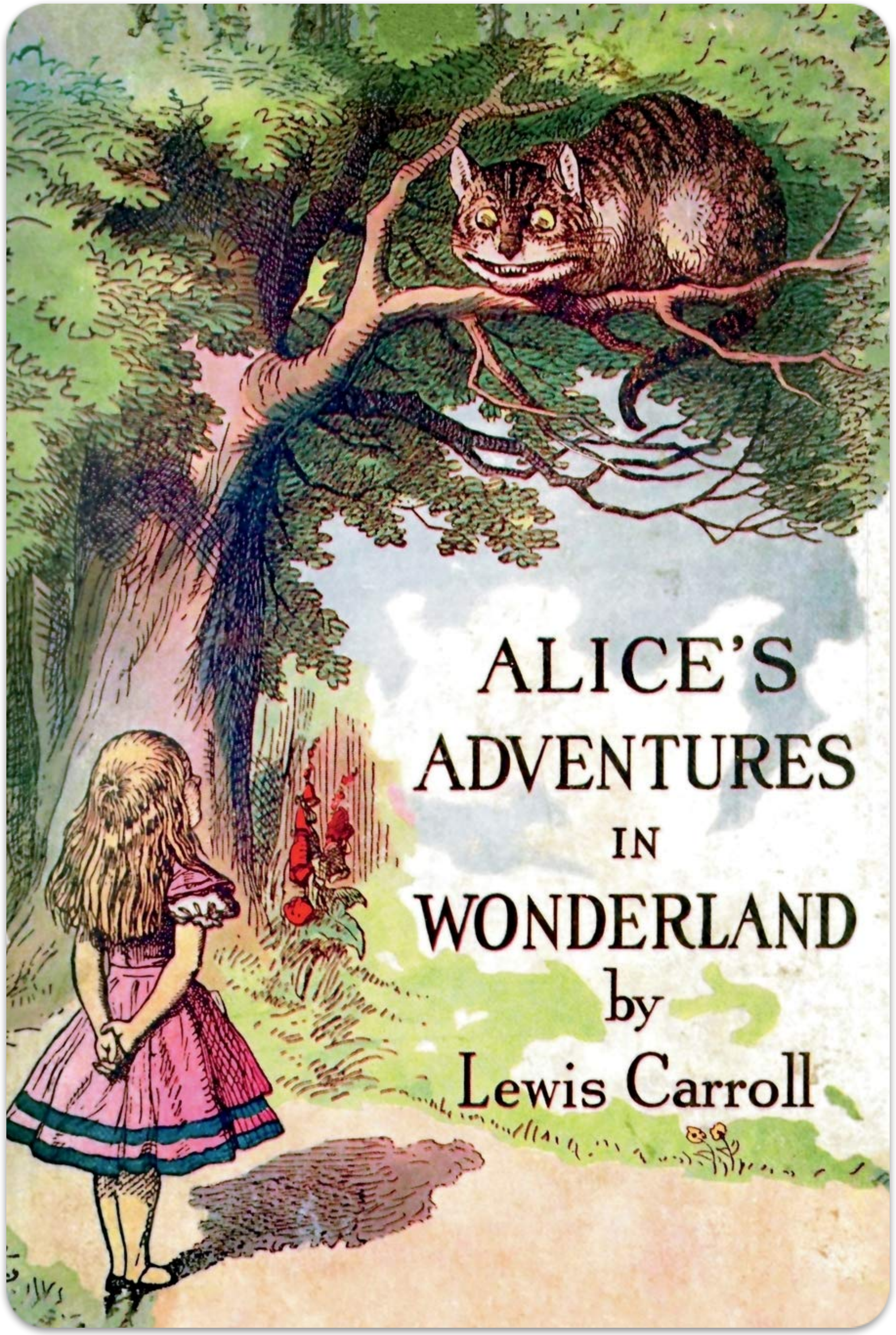
`char`

`double`

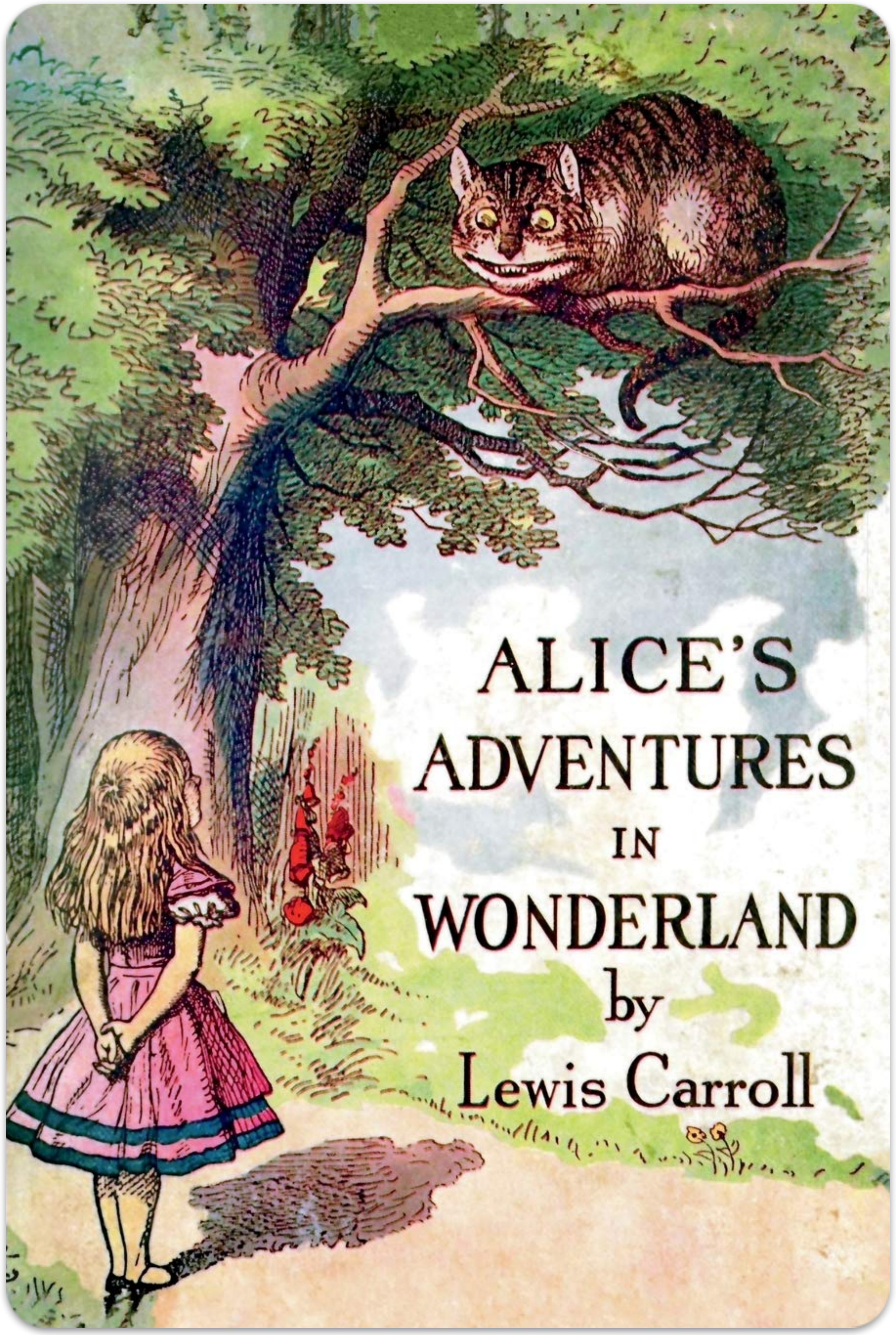
`float`

`int`

`long`



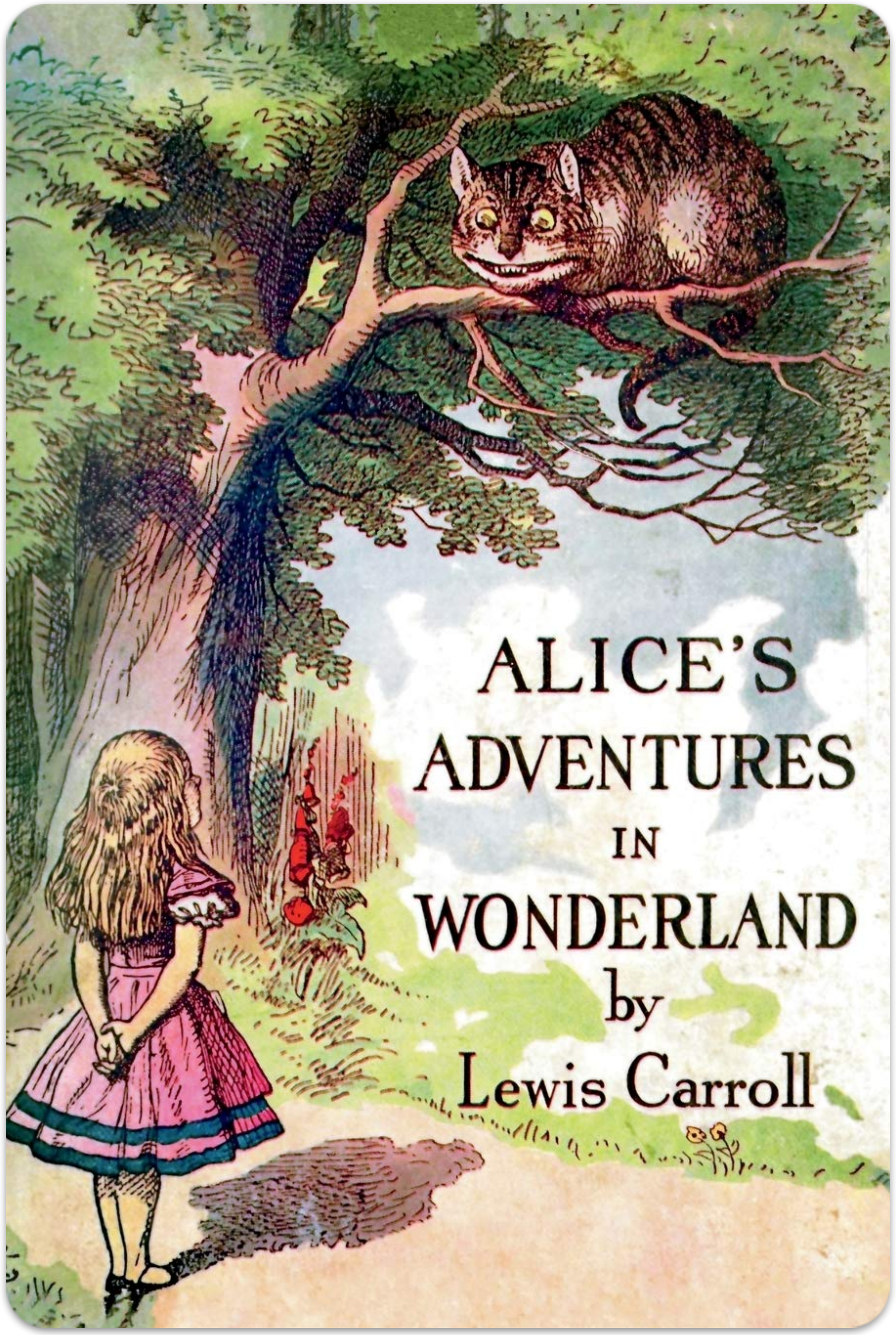
Title	Alice's Adventures in Wonderland
Author	Lewis Carroll
Year	1865
Fiction	Yes




```
string title = "Alice's ...";

string author = "Lewis Carroll";

int year = 1865;

bool fiction = true;
```

Struct

```
typedef struct
{
    string title;
    string author;
    int year;
    bool fiction;
}
book;
```




```
book alice;
alice.title = "Alice's ...";
alice.author = "Lewis Carroll";
alice.year = 1865;
alice.fiction = true;
```

Common Types in Python

`bool`

`float`

`int`

`str`

`range`

`list`

`tuple`

`dict`

`set`

Conditionals

C

```c
if (x > 0)
{
    printf("positive\n");
}
```

Python

```python
if x > 0:
    print("positive")
```

Loops

C

```c
for (int i = 0; i < 5; i++)
{
    print("%i\n", i);
}
```

Python

```python
for i in range(5):
    print(i)
```

Loops

C

```c
while (x >= 1)
{
    printf("%i\n", x);
    x = x / 2;
}
```

Python

```python
while x >= 1:
    print(x)
    x = x / 2
```

Arrays in C

```
int values[3];
values[0] = 10;
values[1] = 20;
values[2] = 30;

for (int i = 0; i < 3; i++)
{
    printf("%i\n", values[i]);
}
```

Lists in Python

```
values = []
values.append(10)
values.append(20)
values.append(30)

for value in values:
    print(value)
```

Functions

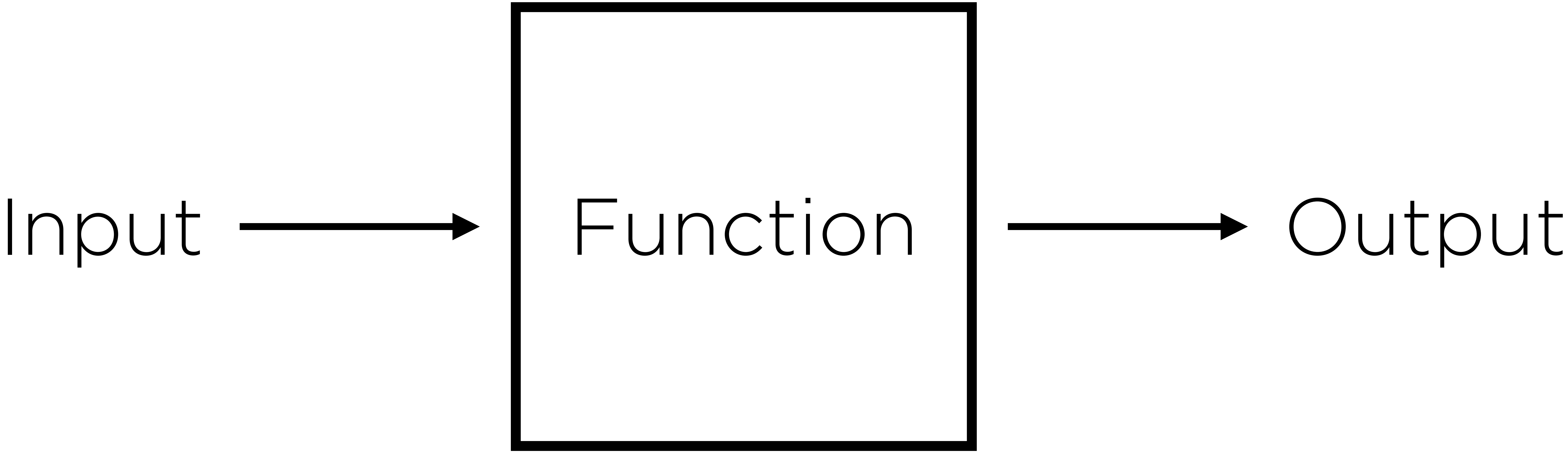
C

```c
int square(int n)
{
    return n * n;
}
```

Python

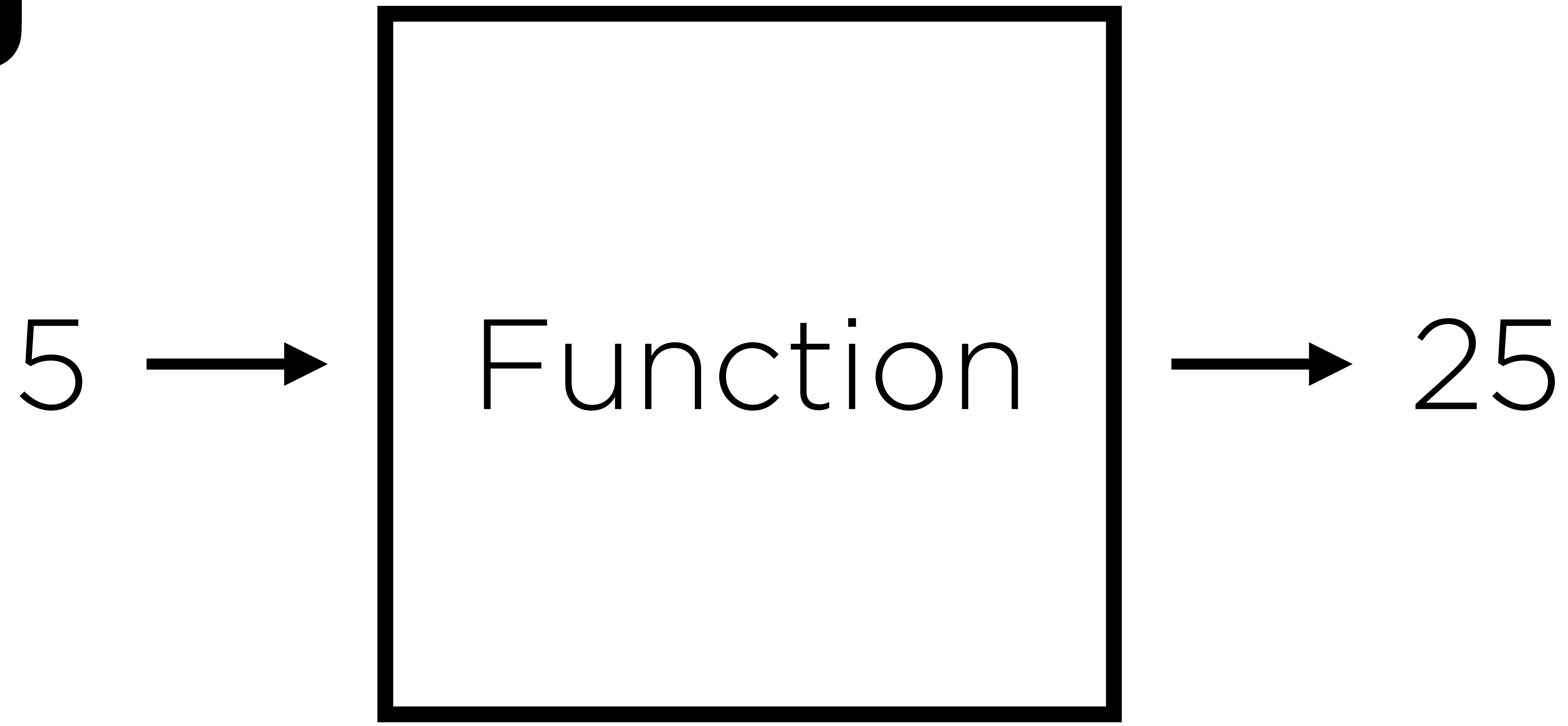
```python
def square(n):
    return n * n
```

Functions



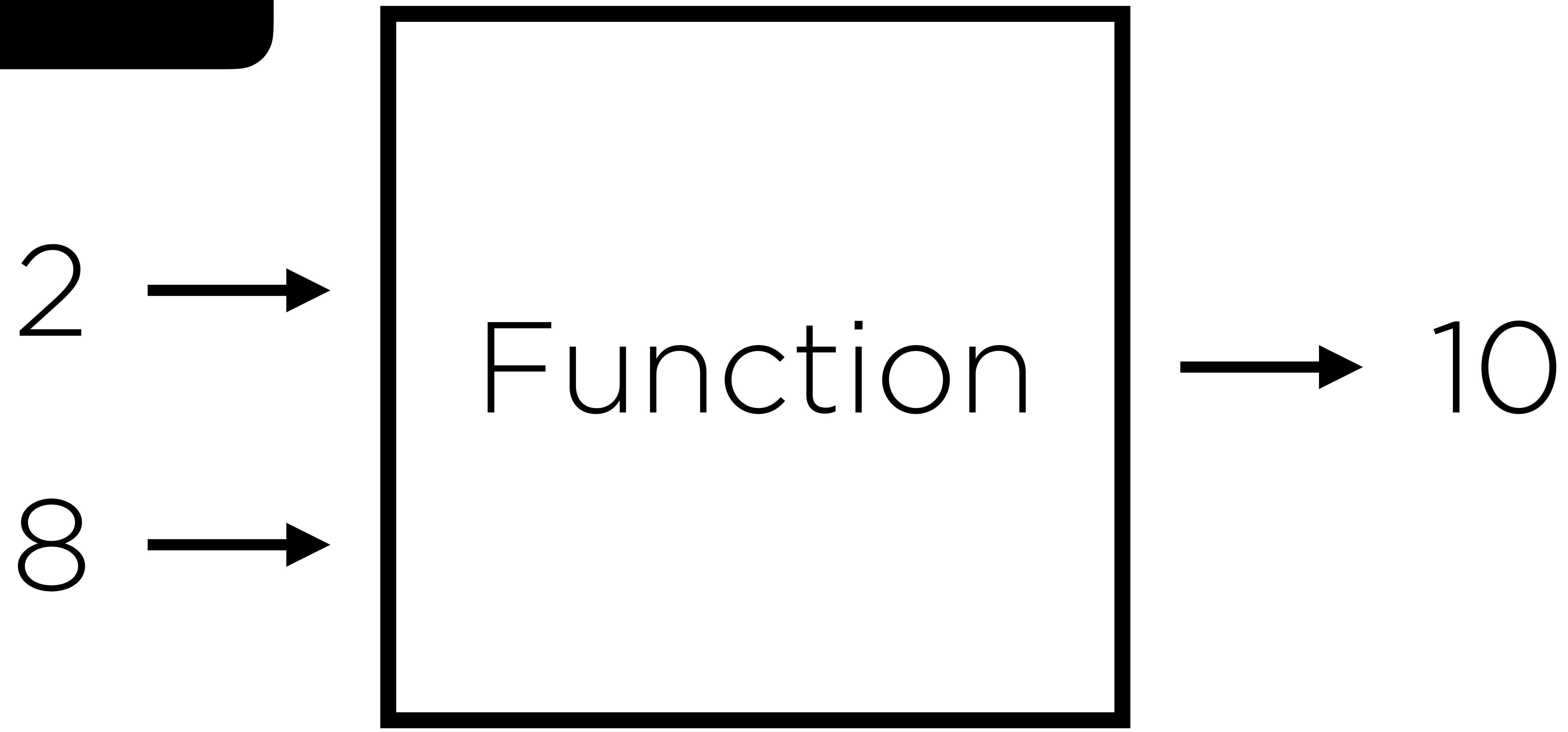
Functions

```
int square(int n)
{
    return n * n;
}
```



Functions

```
int sum(int a, int b)
{
    return a + b;
}
```



Recursive Functions

```
int factorial(int n)
{
    if (n == 0)
    {
        return 1;
    }
    return n * factorial(n - 1);
}
```

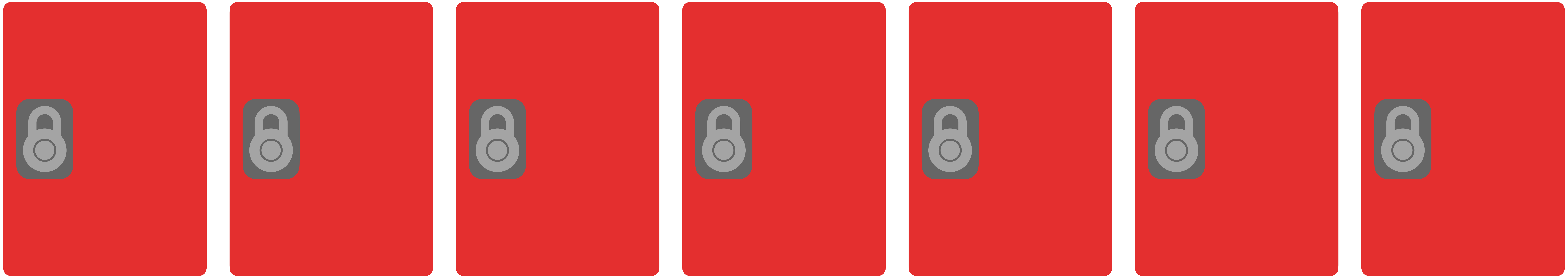
factorial(0) → 1
factorial(1) → 1
factorial(2) → 2
factorial(3) → 6
factorial(4) → 24

Algorithms

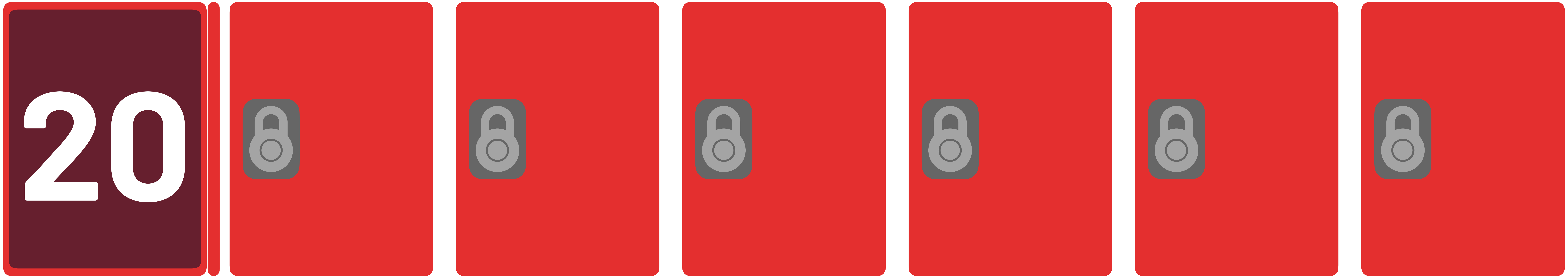
Search

Linear Search

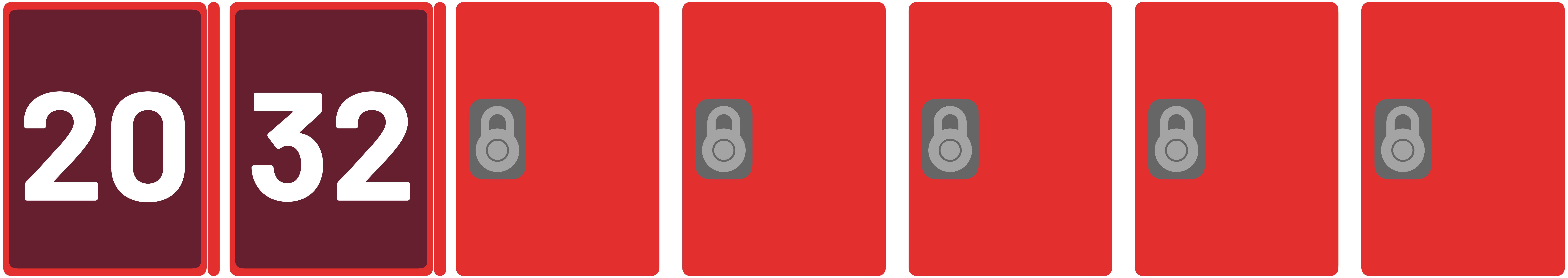
Linear Search



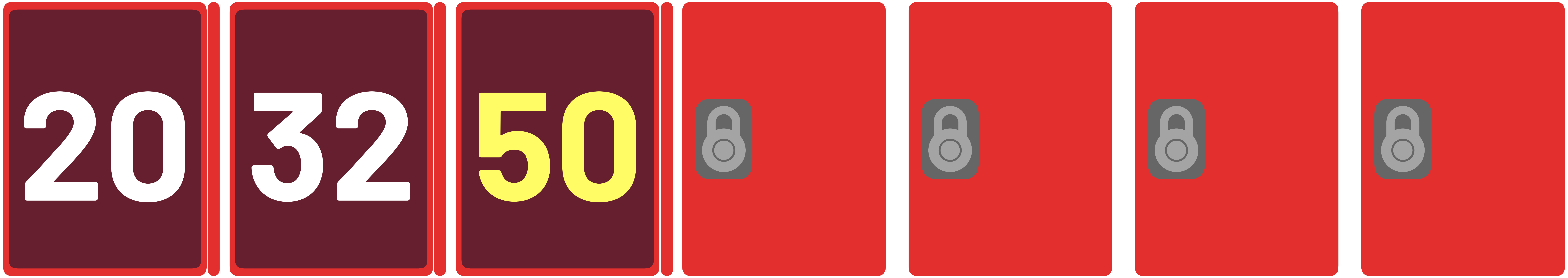
Linear Search



Linear Search



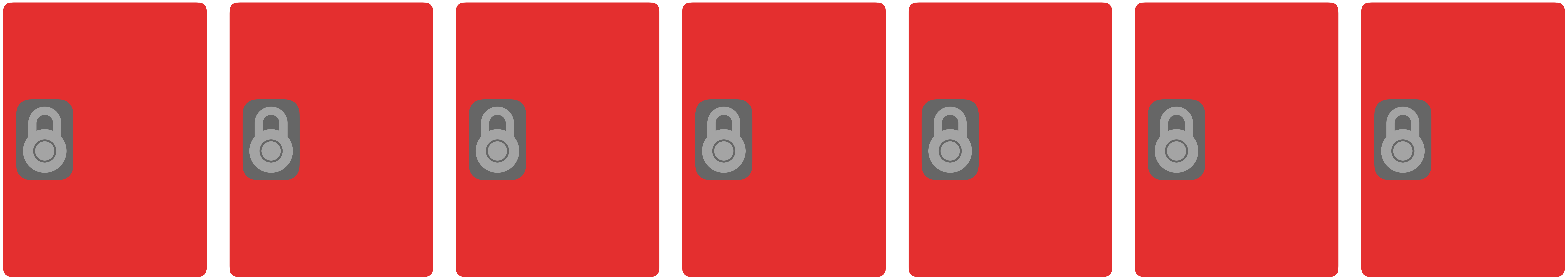
Linear Search



Search

Binary Search

Binary Search



Binary Search

Binary Search

Binary Search

Running Time

Algorithm	Upper Bound	Lower Bound
Linear Search	$O(n)$	$\Omega(1)$
Binary Search	$O(\log n)$	$\Omega(1)$

Sorting

Selection Sort

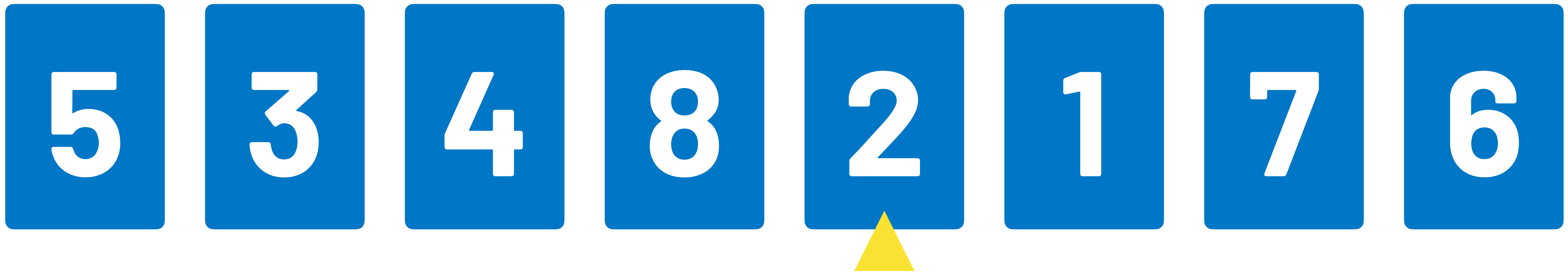
Selection Sort

5	3	4	8	2	1	7	6

Selection Sort

Selection Sort

Selection Sort

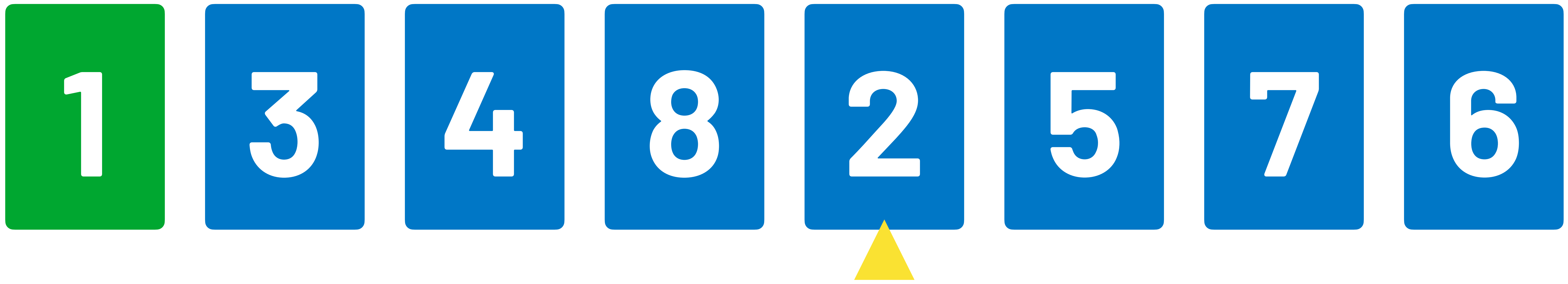


Selection Sort

Selection Sort

Selection Sort

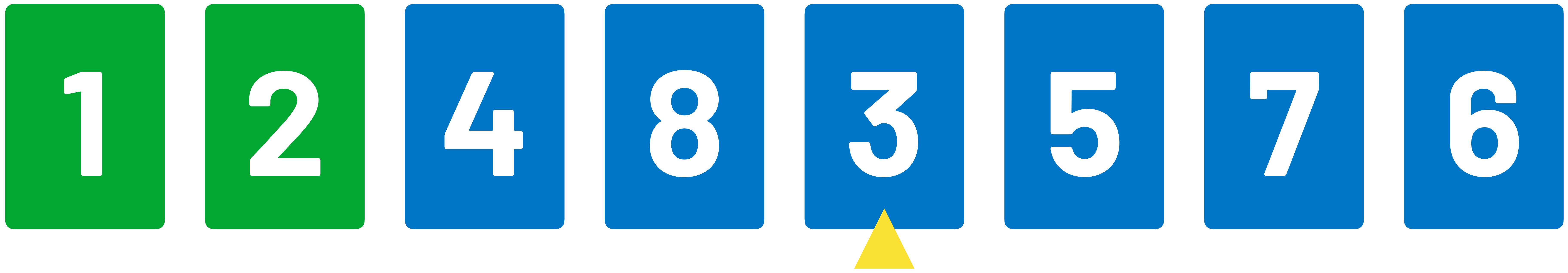
Selection Sort



Selection Sort

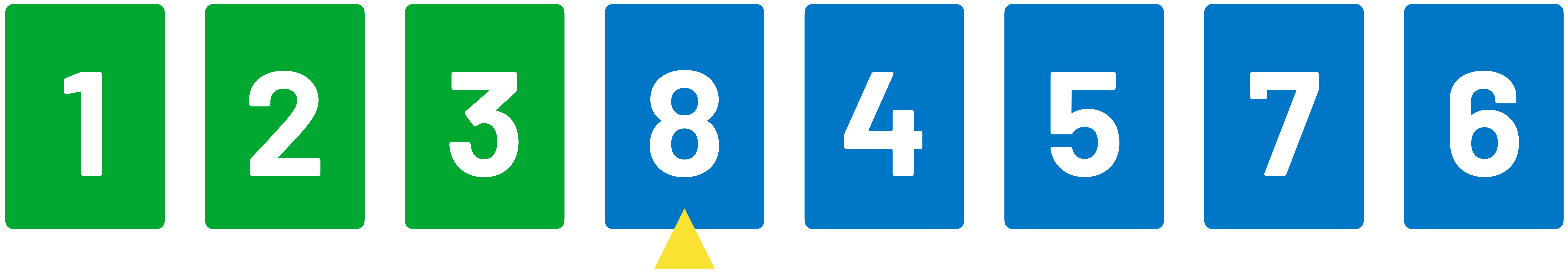
Selection Sort

Selection Sort



Selection Sort

Selection Sort



Selection Sort

Selection Sort

Selection Sort

Selection Sort

Selection Sort

Selection Sort

Selection Sort

Selection Sort

Selection Sort

Selection Sort

Selection Sort

Selection Sort

Algorithm	Upper Bound	Lower Bound
Linear Search	$O(n)$	$\Omega(1)$
Binary Search	$O(\log n)$	$\Omega(1)$
Selection Sort	$O(n^2)$	$\Omega(n^2)$

Sorting

Bubble Sort

Bubble Sort

5 3 4 8 2 1 7 6

Bubble Sort

Bubble Sort

3 4 5 8 2 1 7 6

Bubble Sort

3 4 5 2 8 1 7 6

Bubble Sort

Bubble Sort

Bubble Sort

3 4 5 2 1 7 6 8

Bubble Sort

3 4 5 2 1 7 6 8

Bubble Sort

3 4 2 5 1 7 6 8

Bubble Sort

Bubble Sort

3 4 2 1 5 6 7 8

Bubble Sort

Bubble Sort

Bubble Sort

Bubble Sort

Bubble Sort

Bubble Sort

Bubble Sort

Bubble Sort

Bubble Sort

Bubble Sort

Algorithm	Upper Bound	Lower Bound
Linear Search	$O(n)$	$\Omega(1)$
Binary Search	$O(\log n)$	$\Omega(1)$
Selection Sort	$O(n^2)$	$\Omega(n^2)$
Bubble Sort	$O(n^2)$	$\Omega(n)$

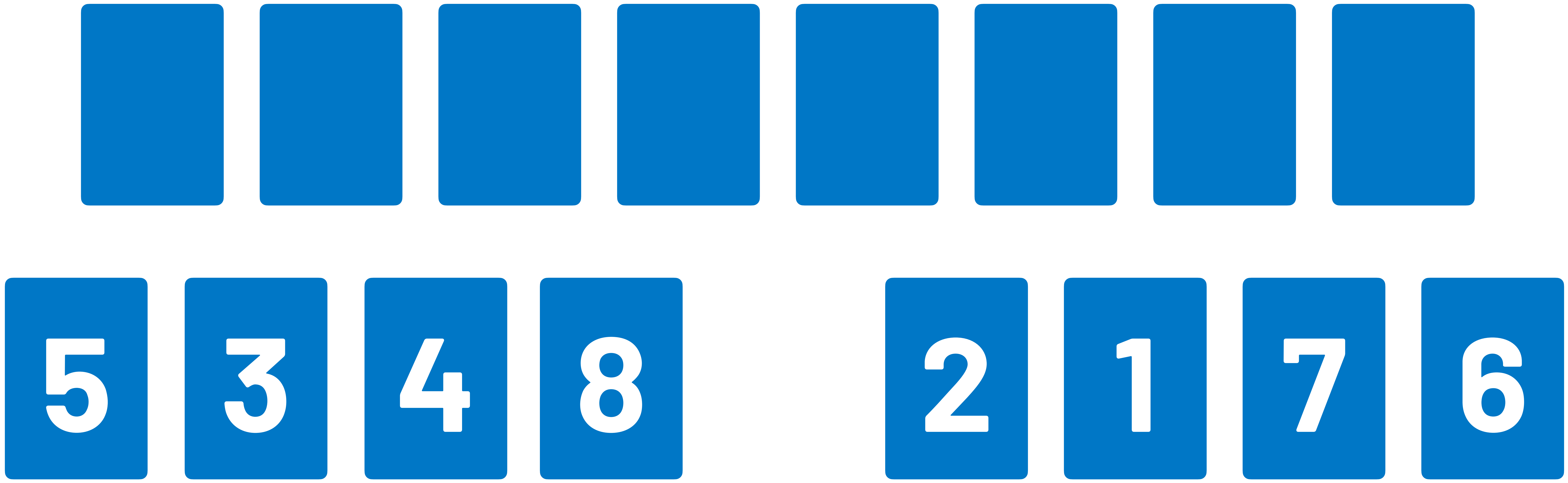
Sorting

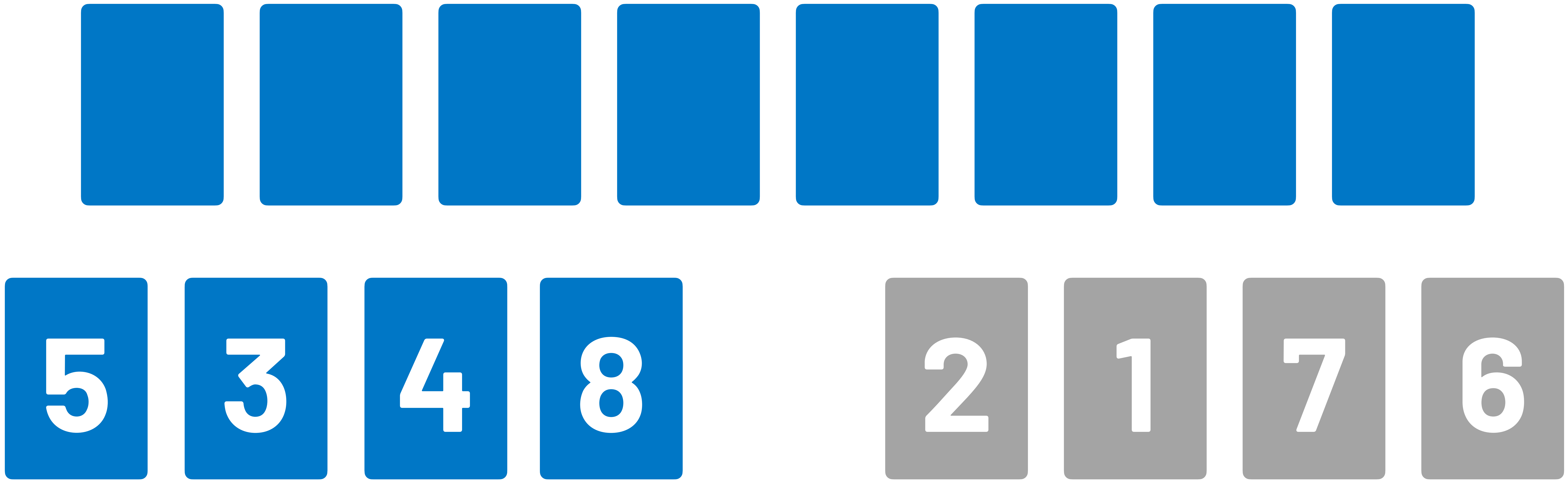
Mergesort

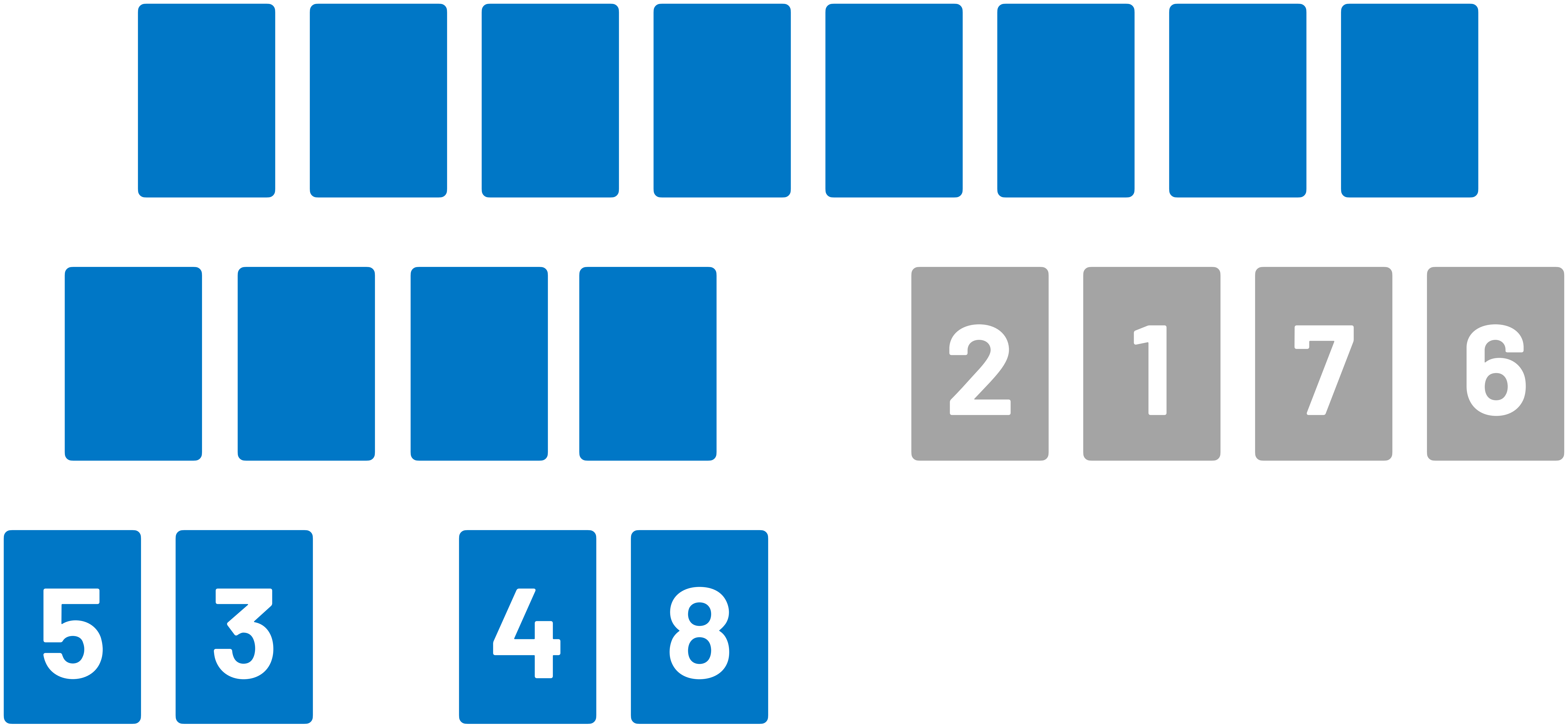
Mergesort

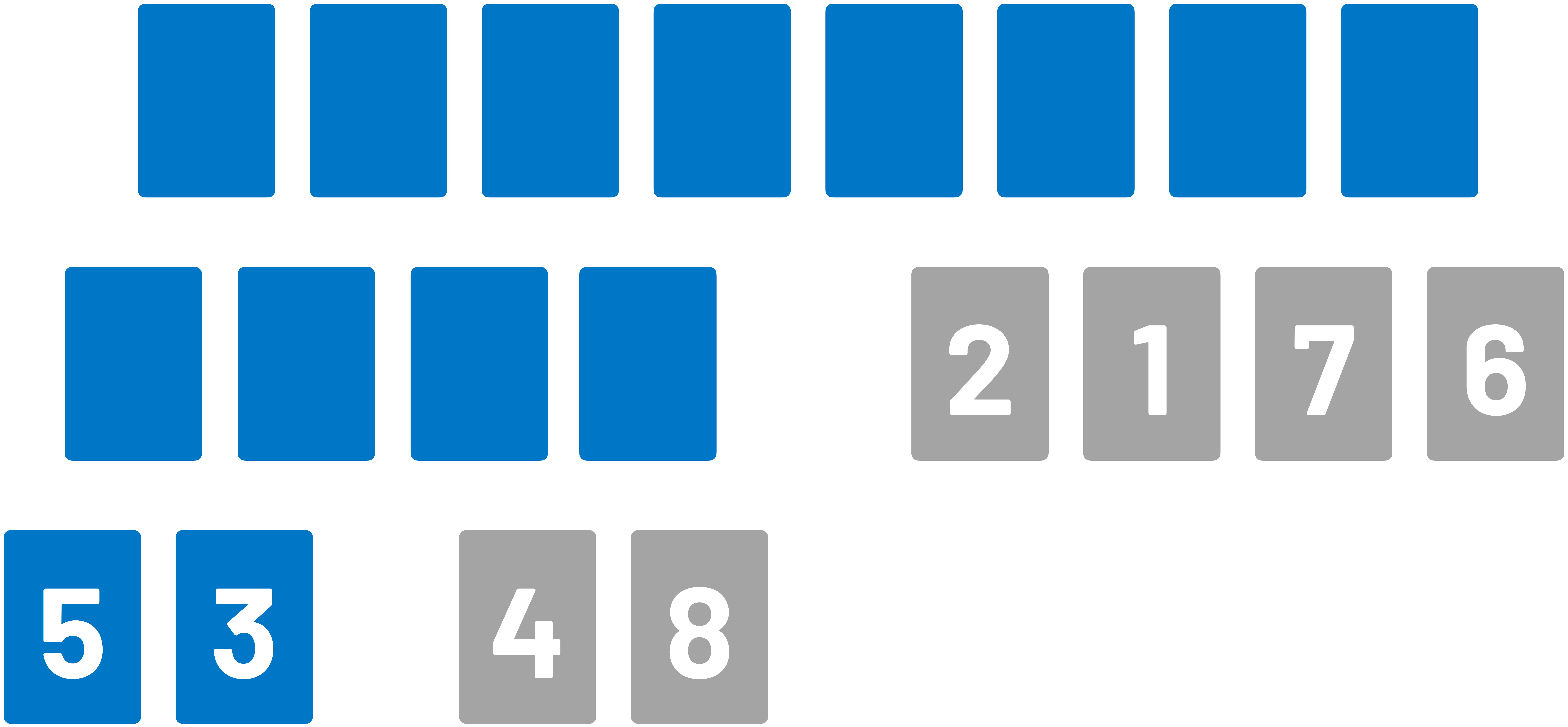
5	3	4	8	2	1	7	6

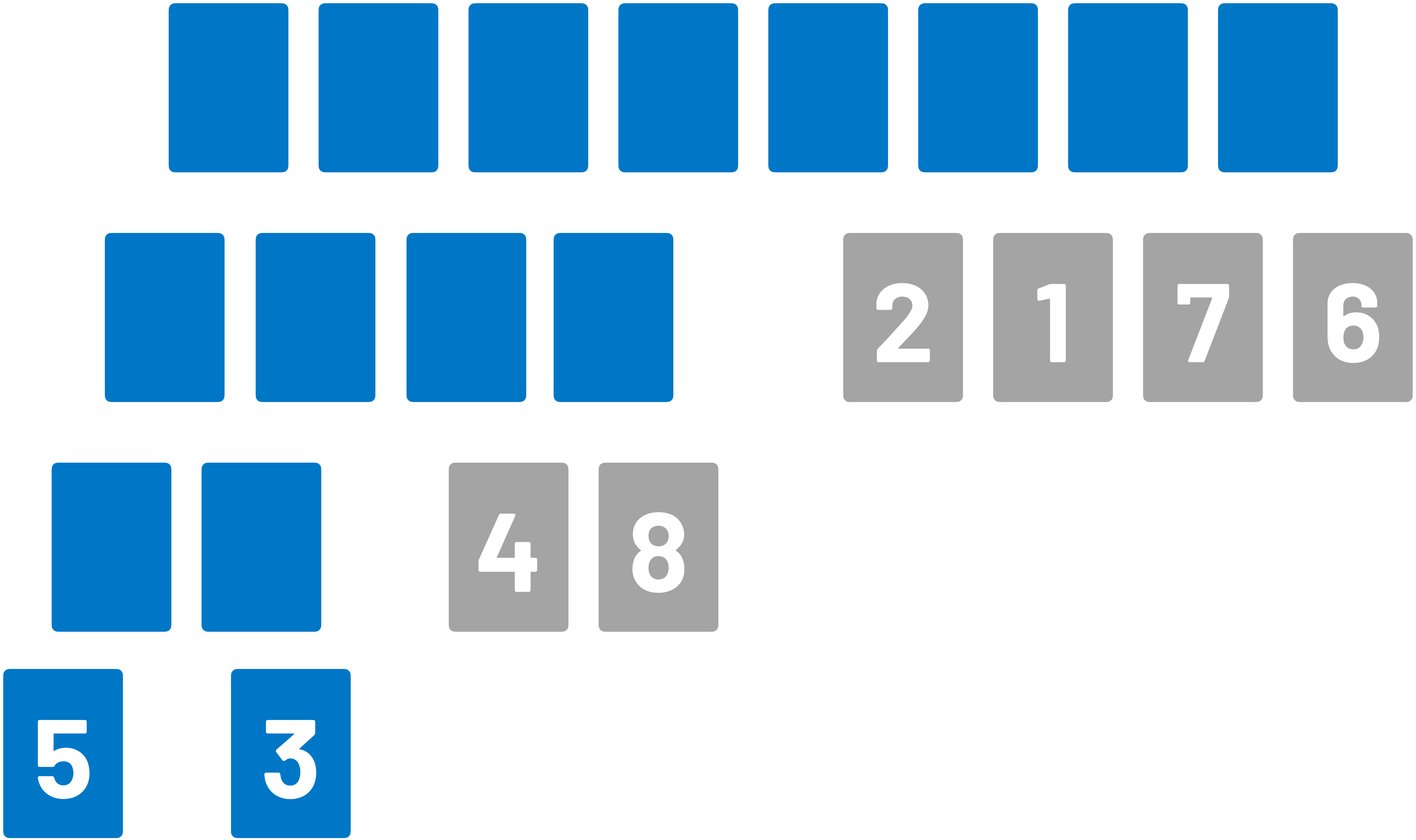
5	3	4	8	2	1	7	6

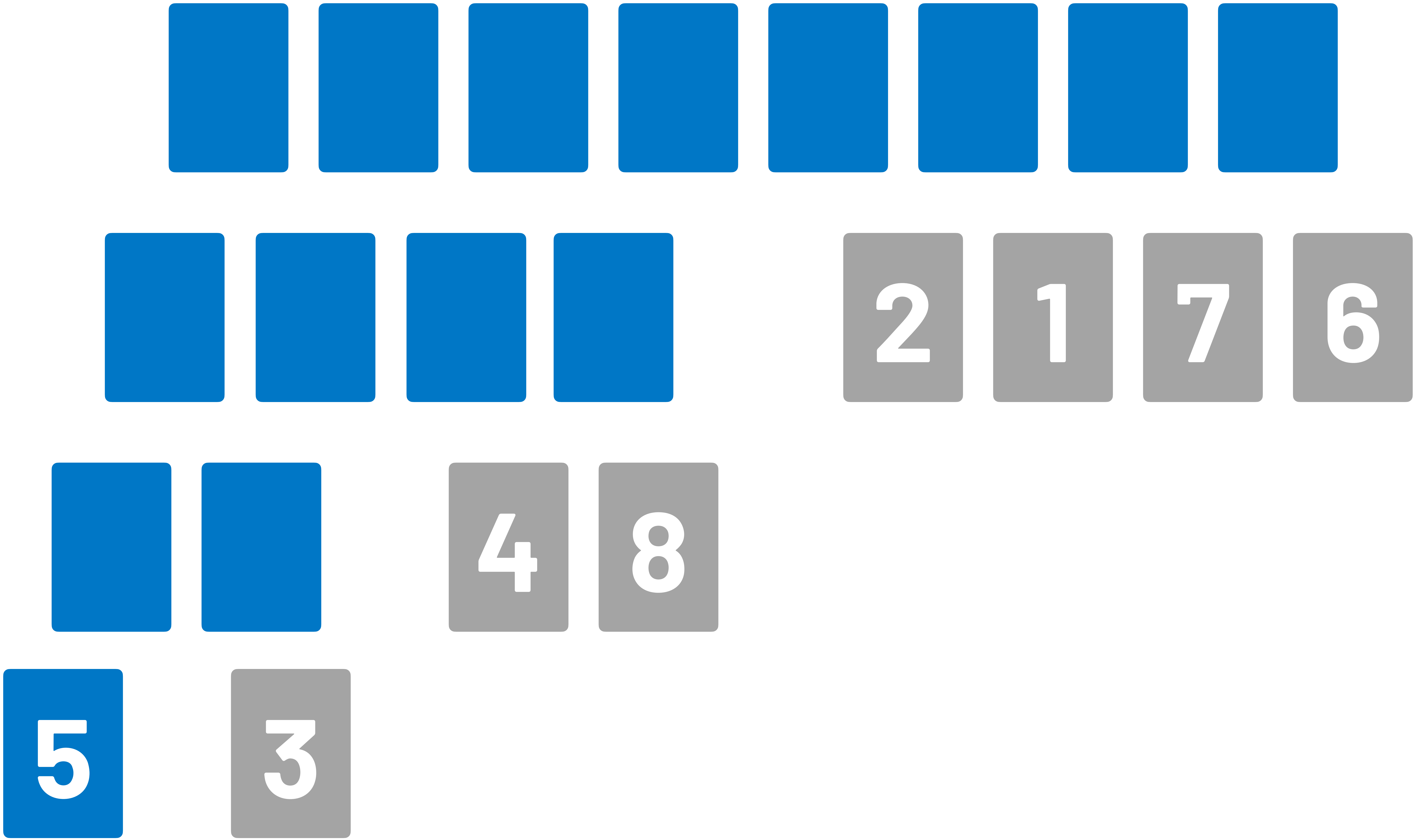

5
3
4
8
2
1
7
6

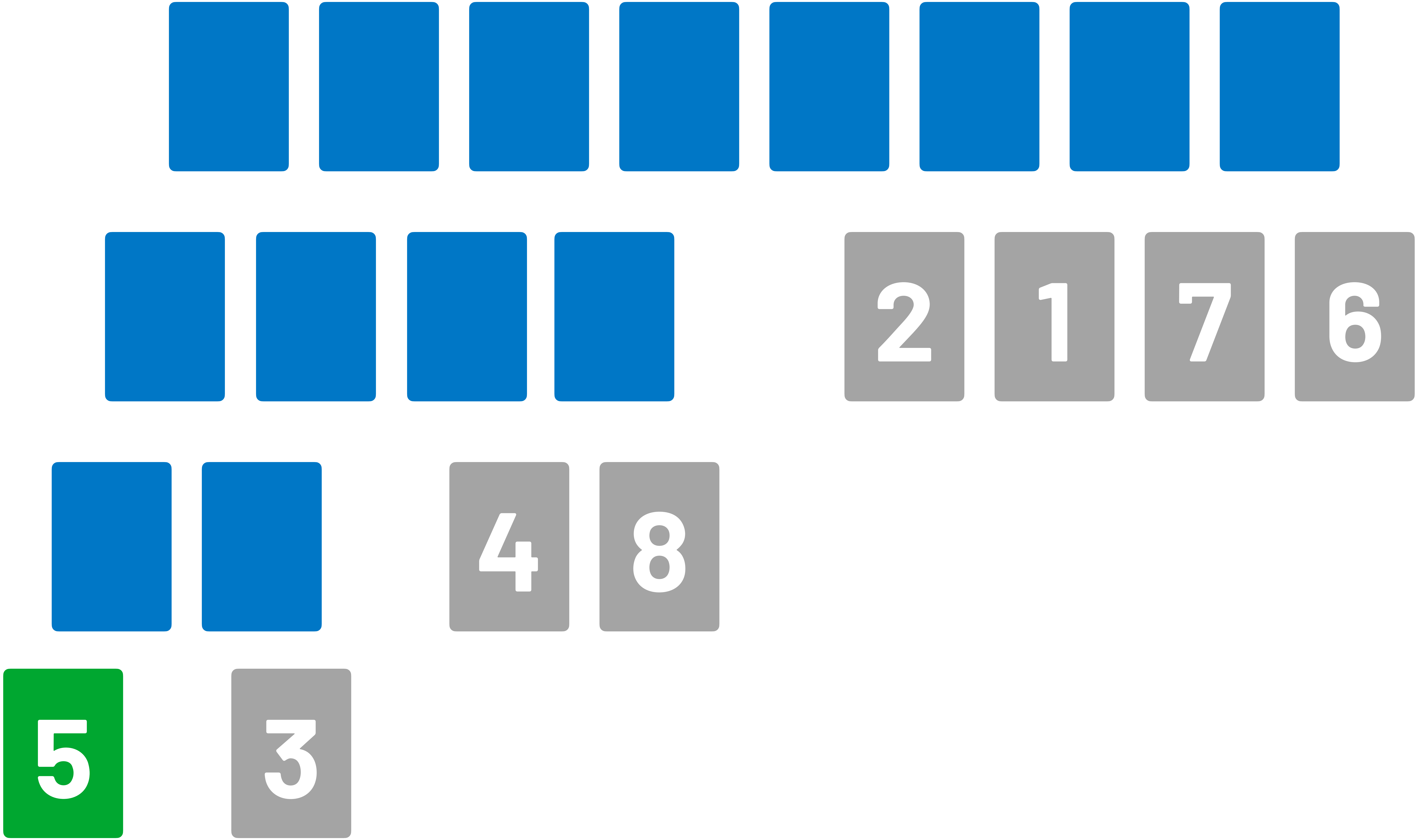

5
3
4
8
2
1
7
6

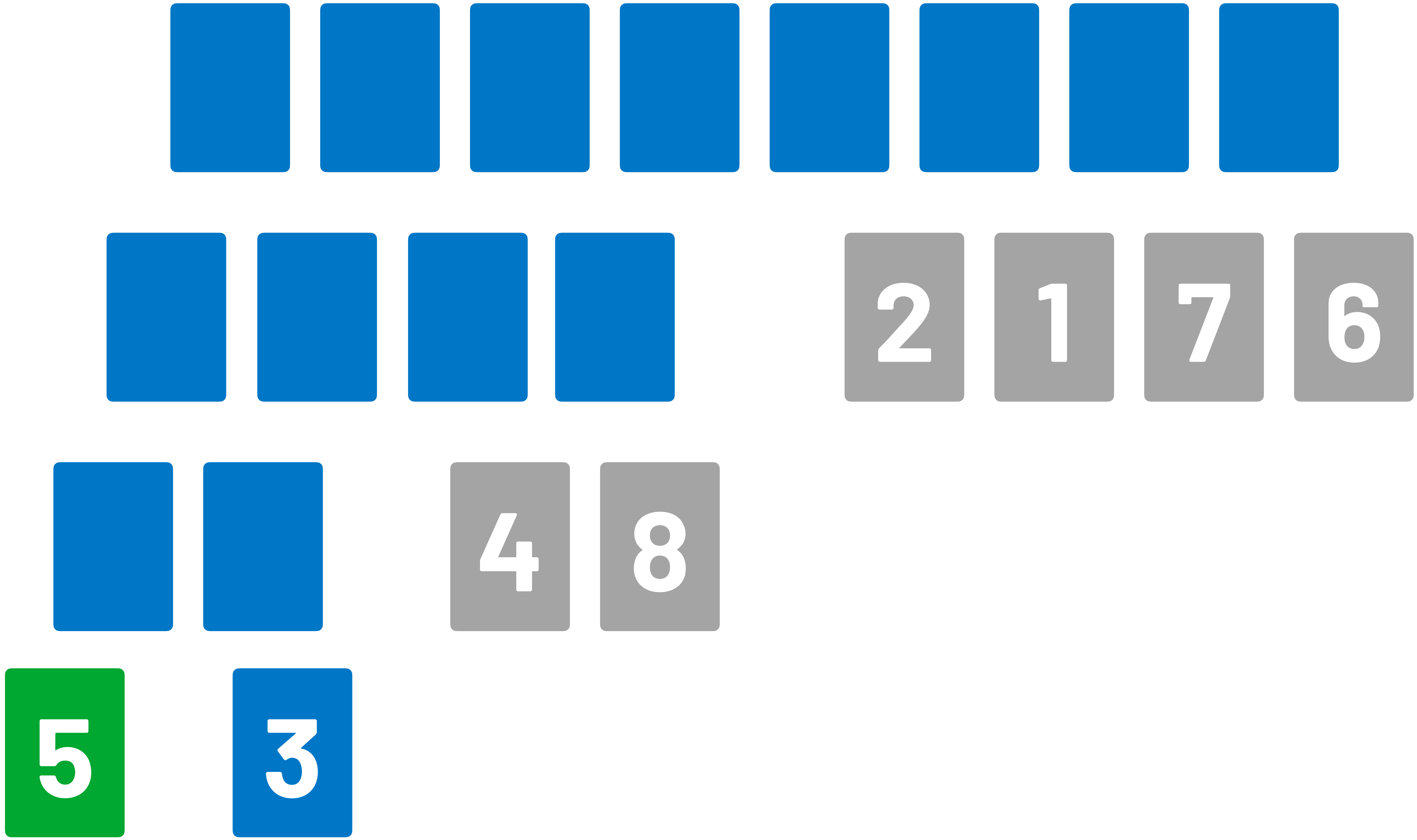

2
1
7
6
5
3
4
8

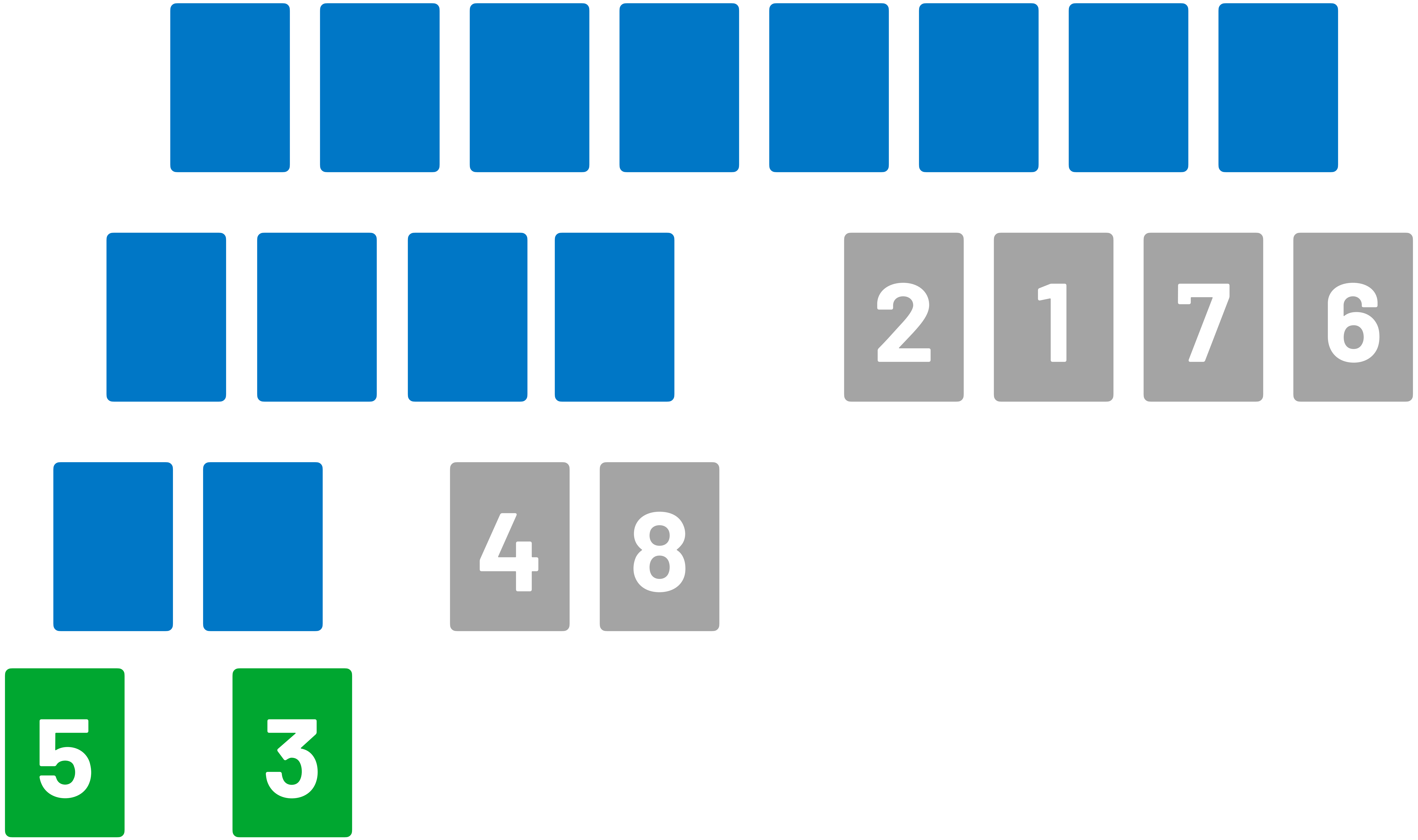

2
1
7
6
5
3
4
8


2
1
7
6
4
8
5
3

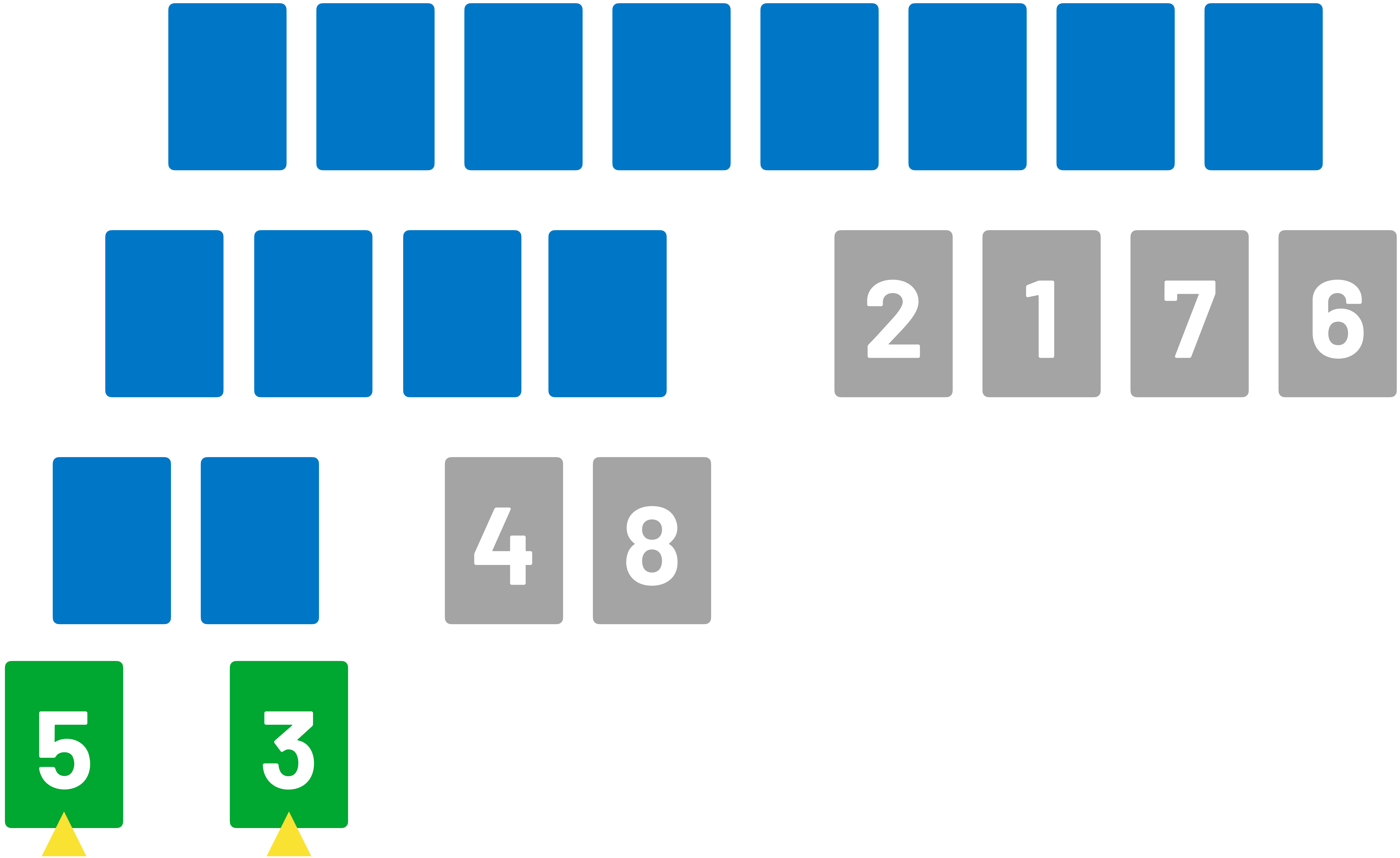

2
1
7
6
4
8
5
3

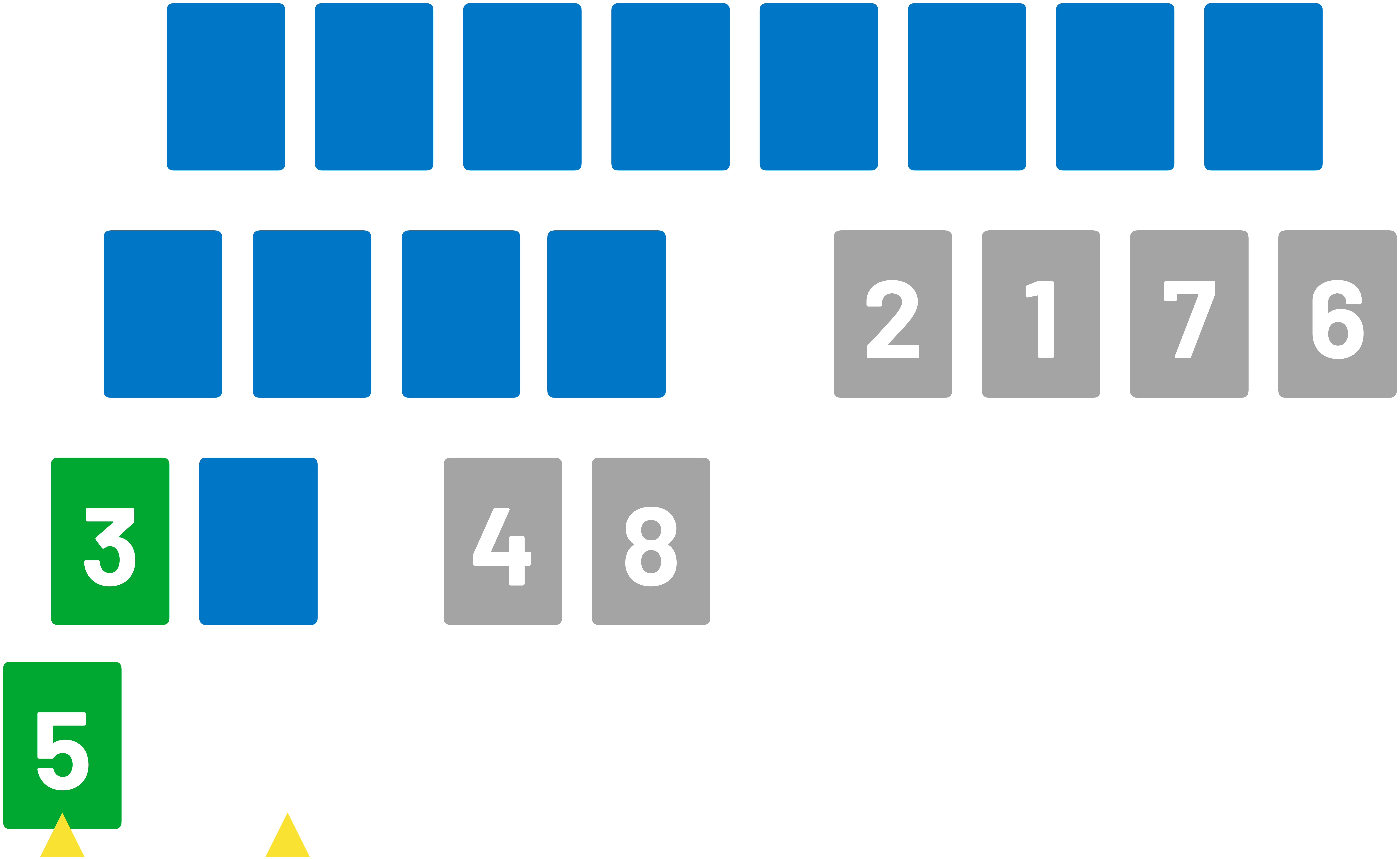

2
1
7
6
4
8
5
3


2 1 7 6
4 8
5
3

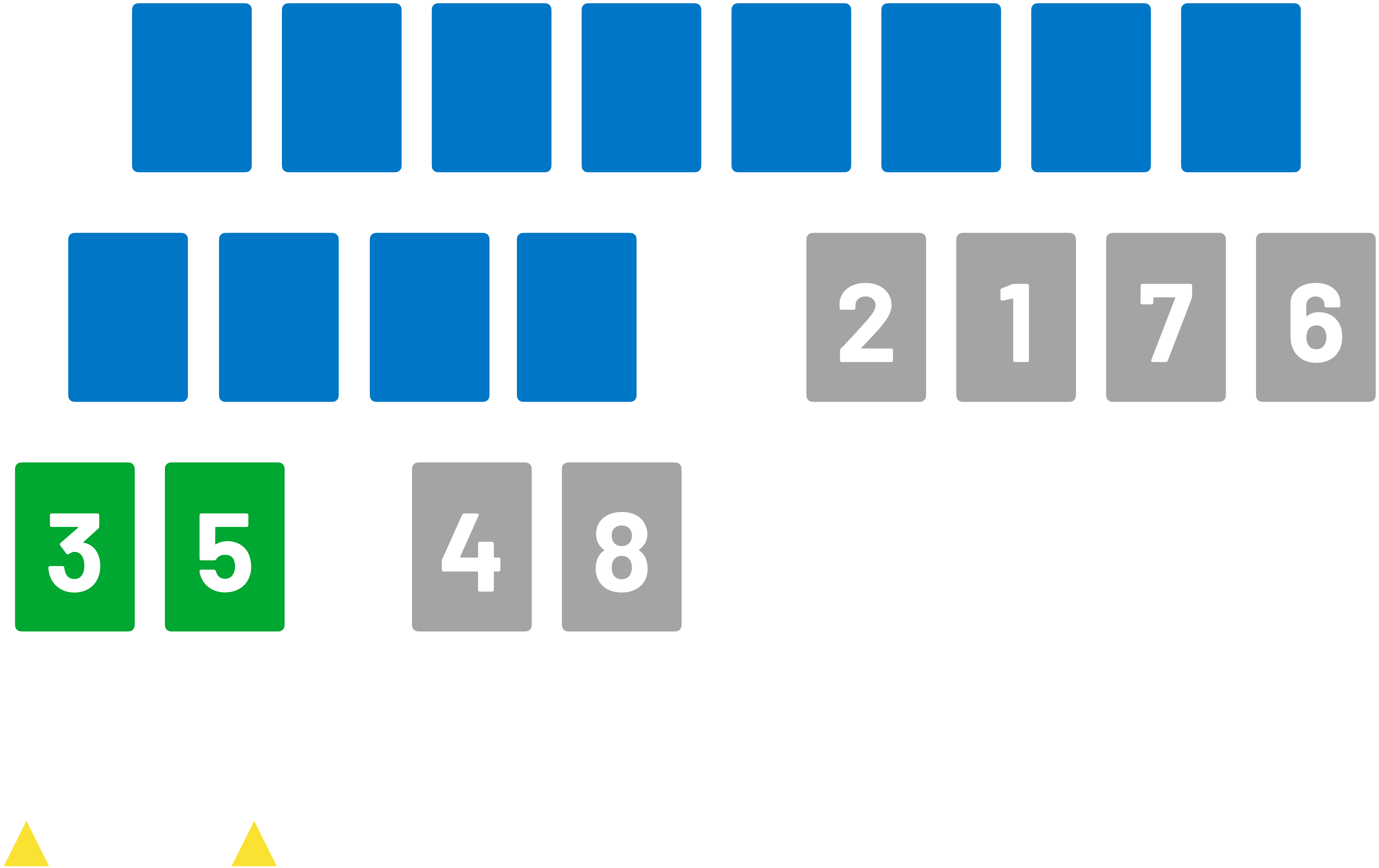


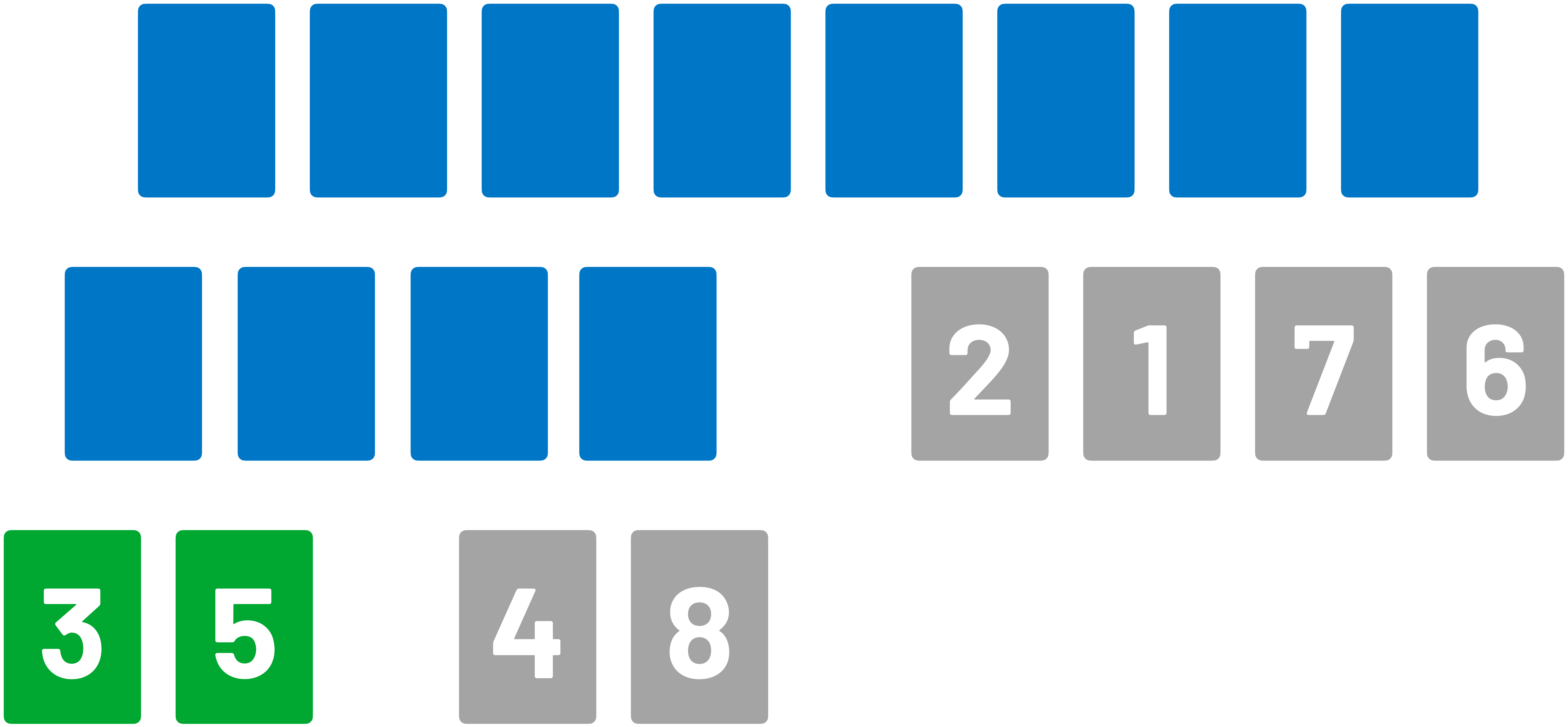
2
1
7
6
4
8
5
3

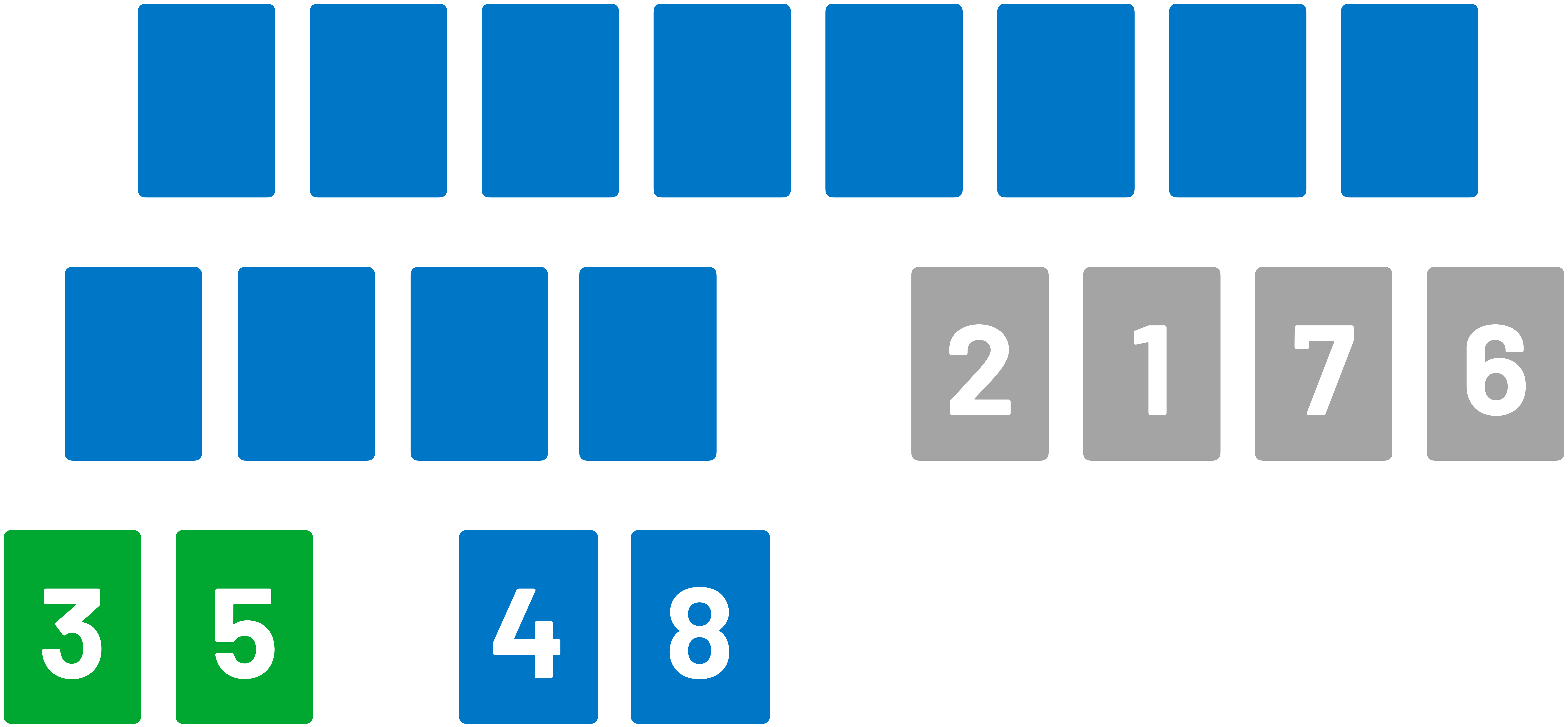

2
1
7
6
4
8
5
3

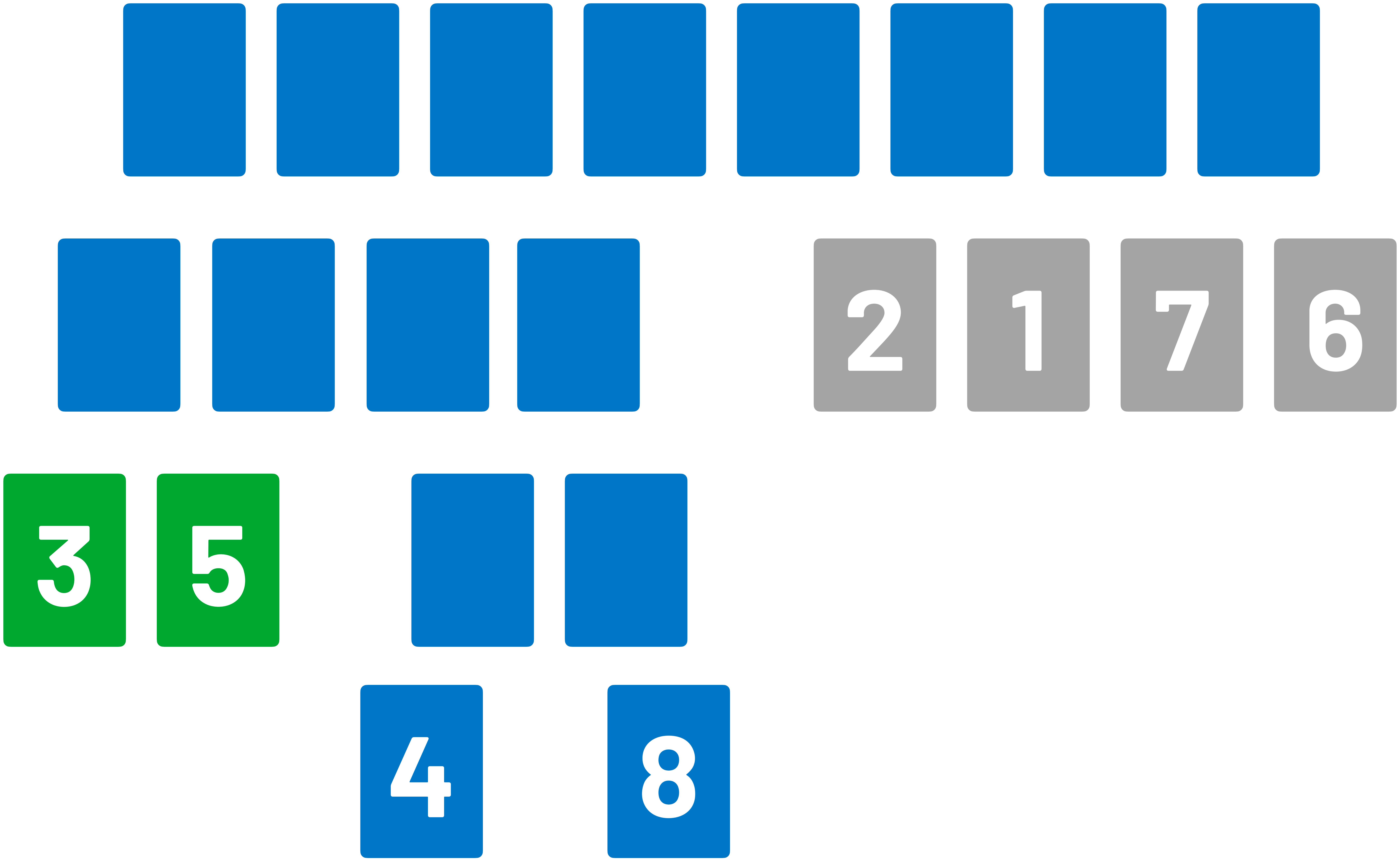


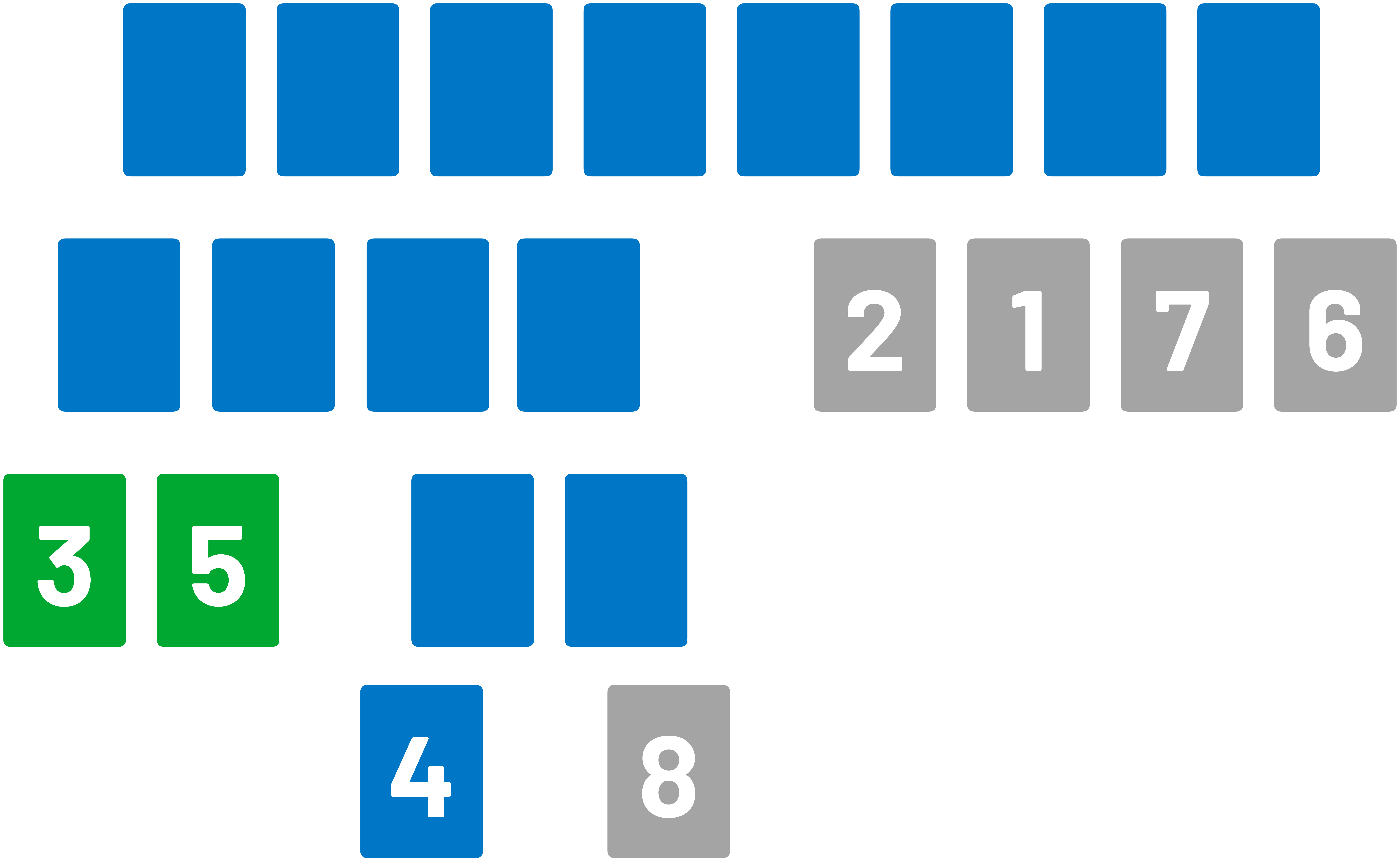
2
1
7
6
3
4
8
5

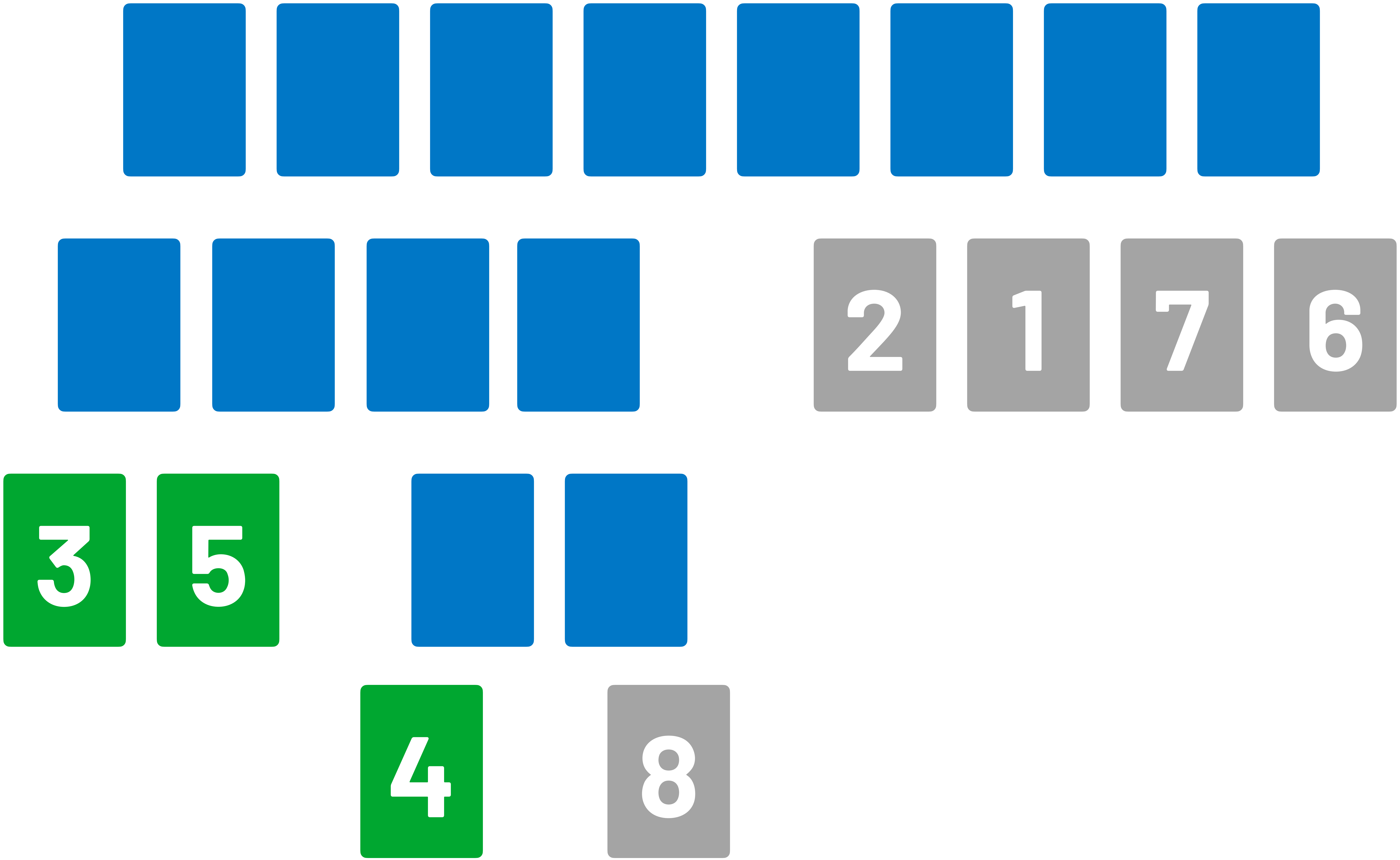

2
1
7
6
3
5
4
8

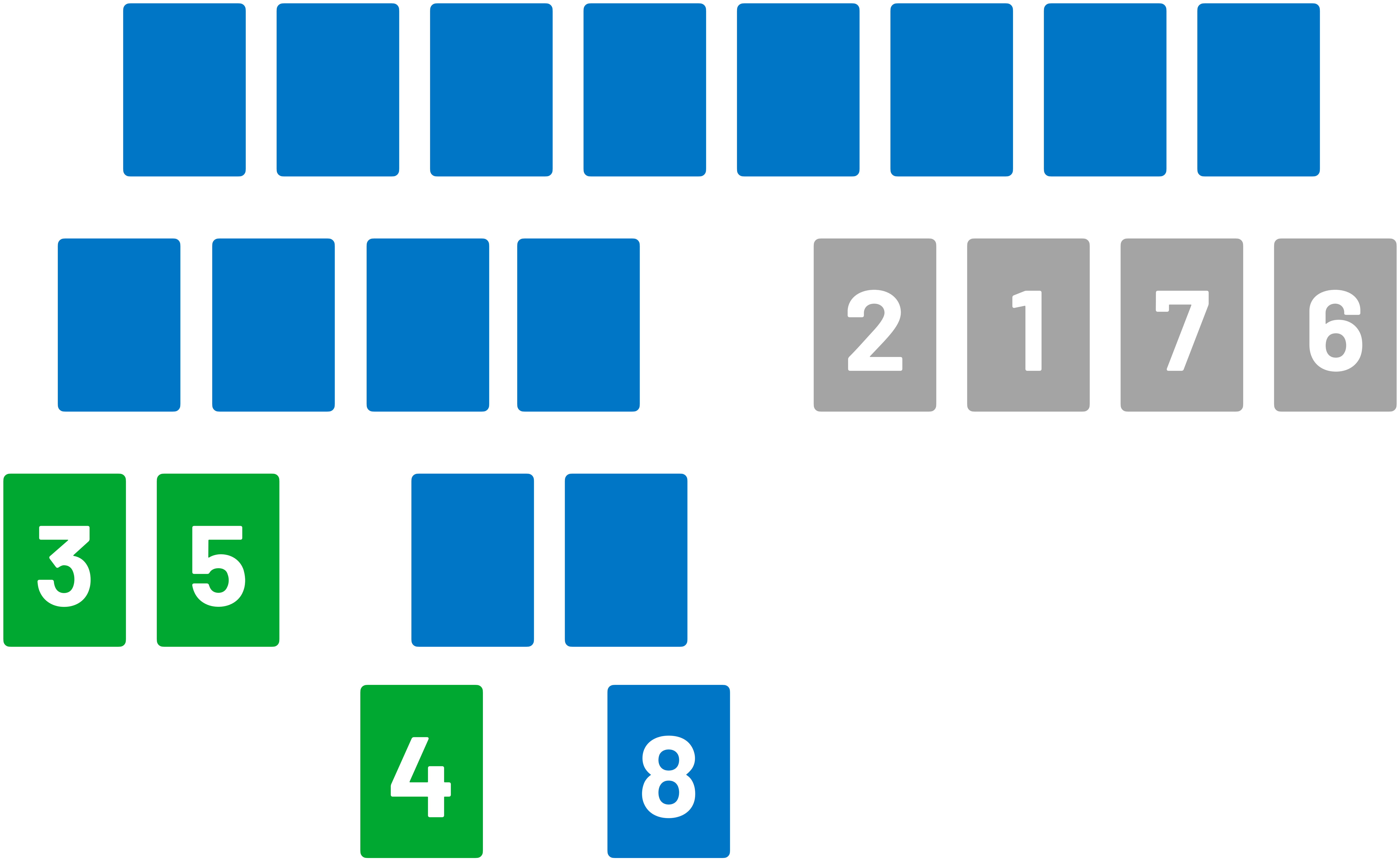

2
1
7
6
3
5
4
8

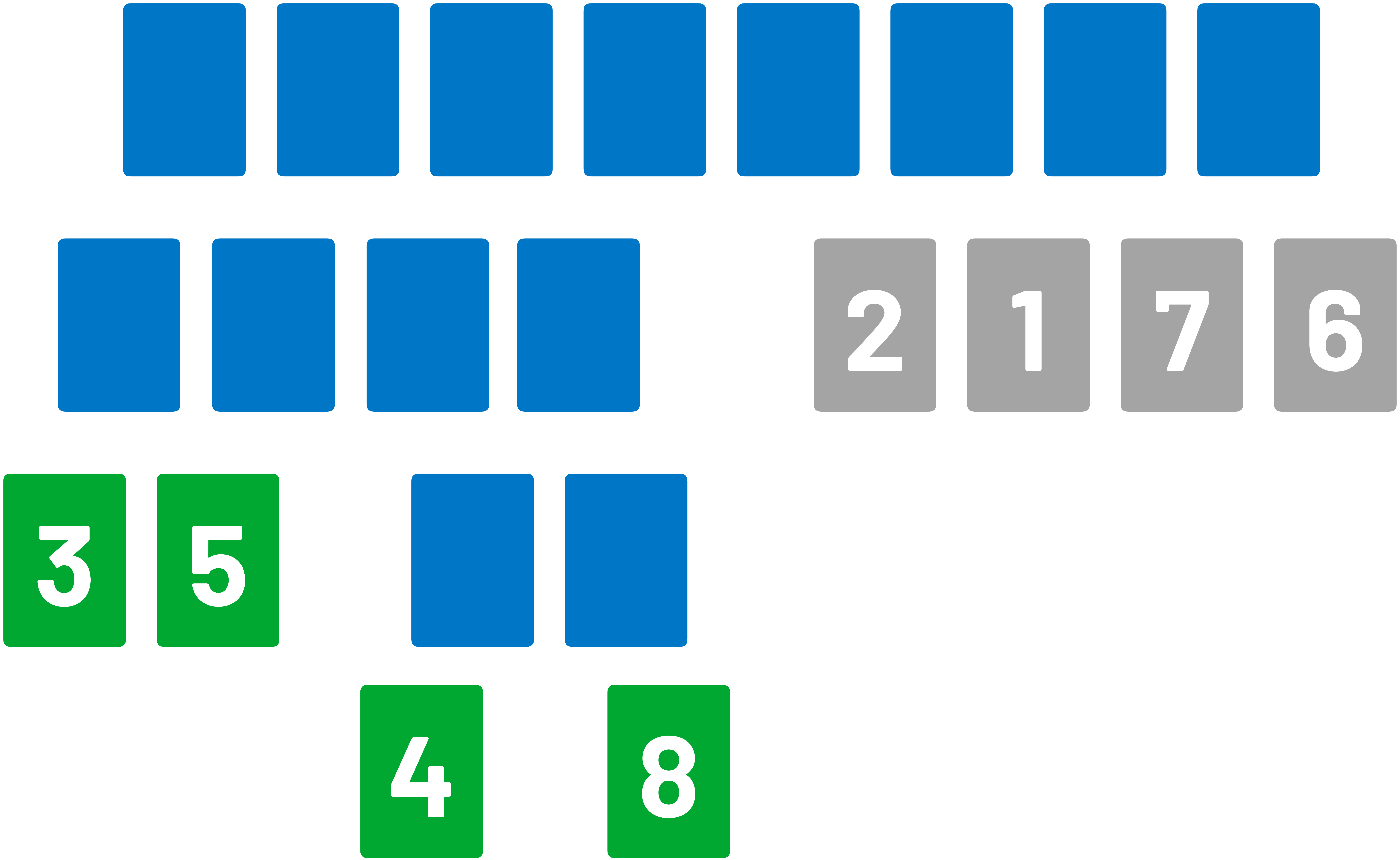

2
1
7
6
3
5
4
8

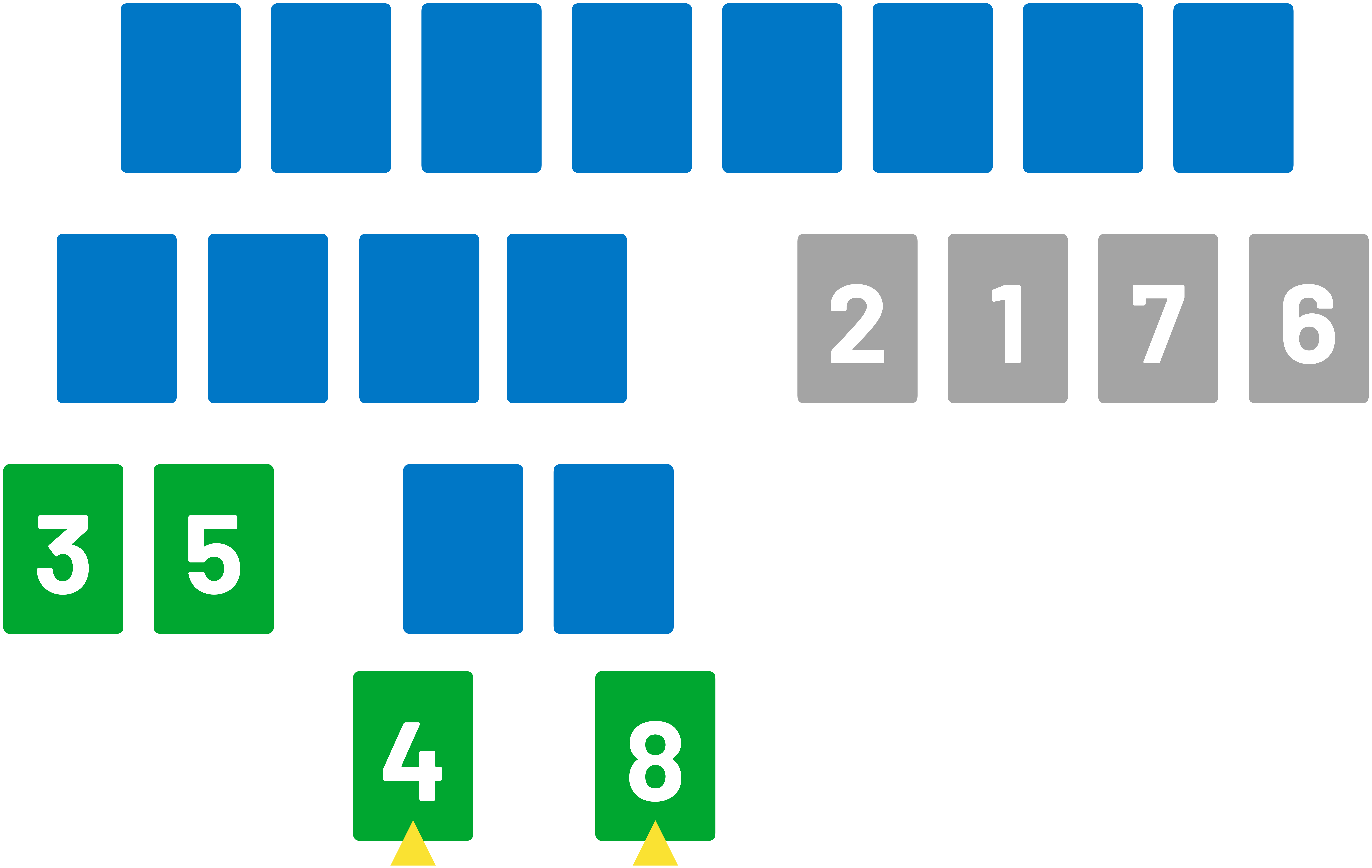

2
1
7
6
3
5
4
8

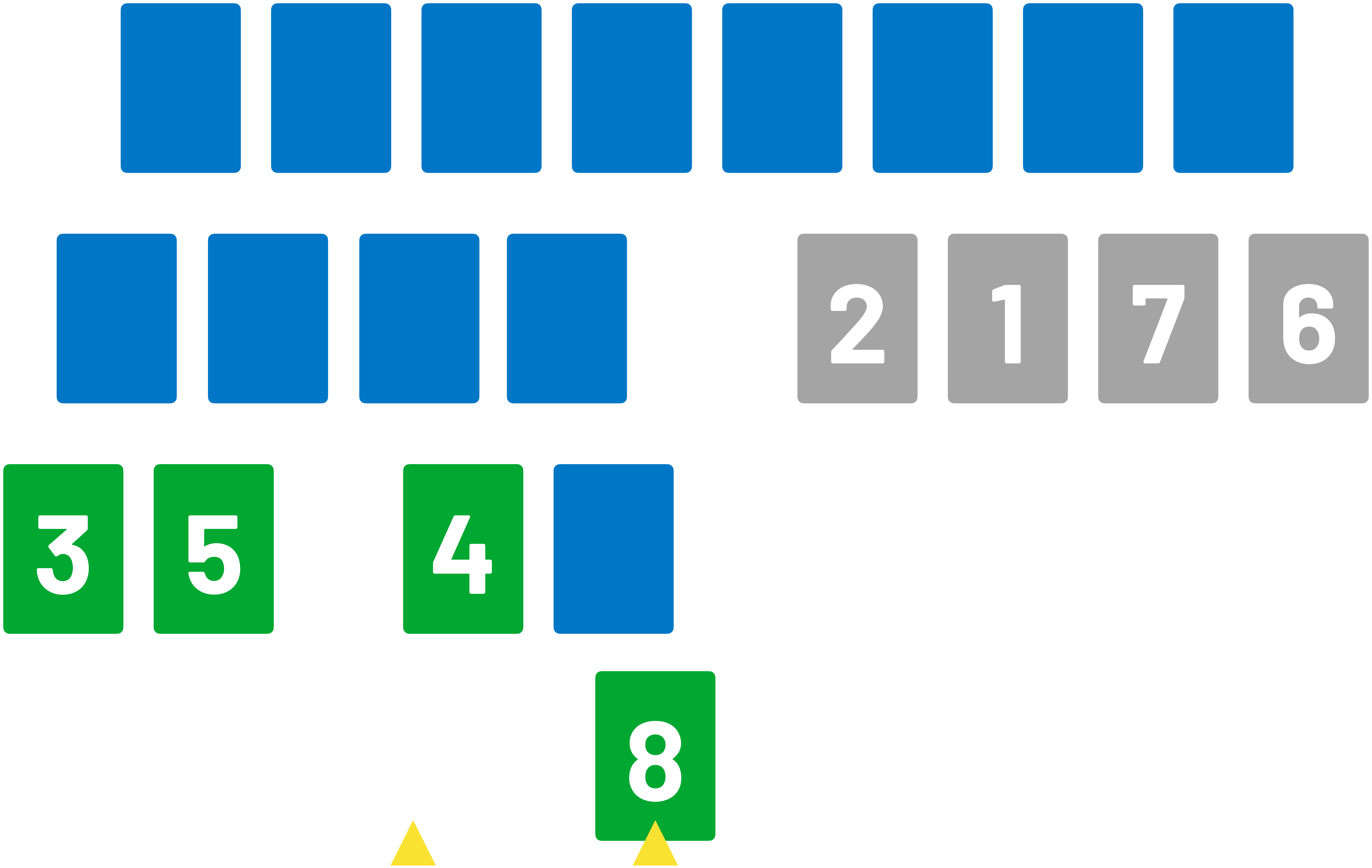

2
1
7
6
3
5
4
8

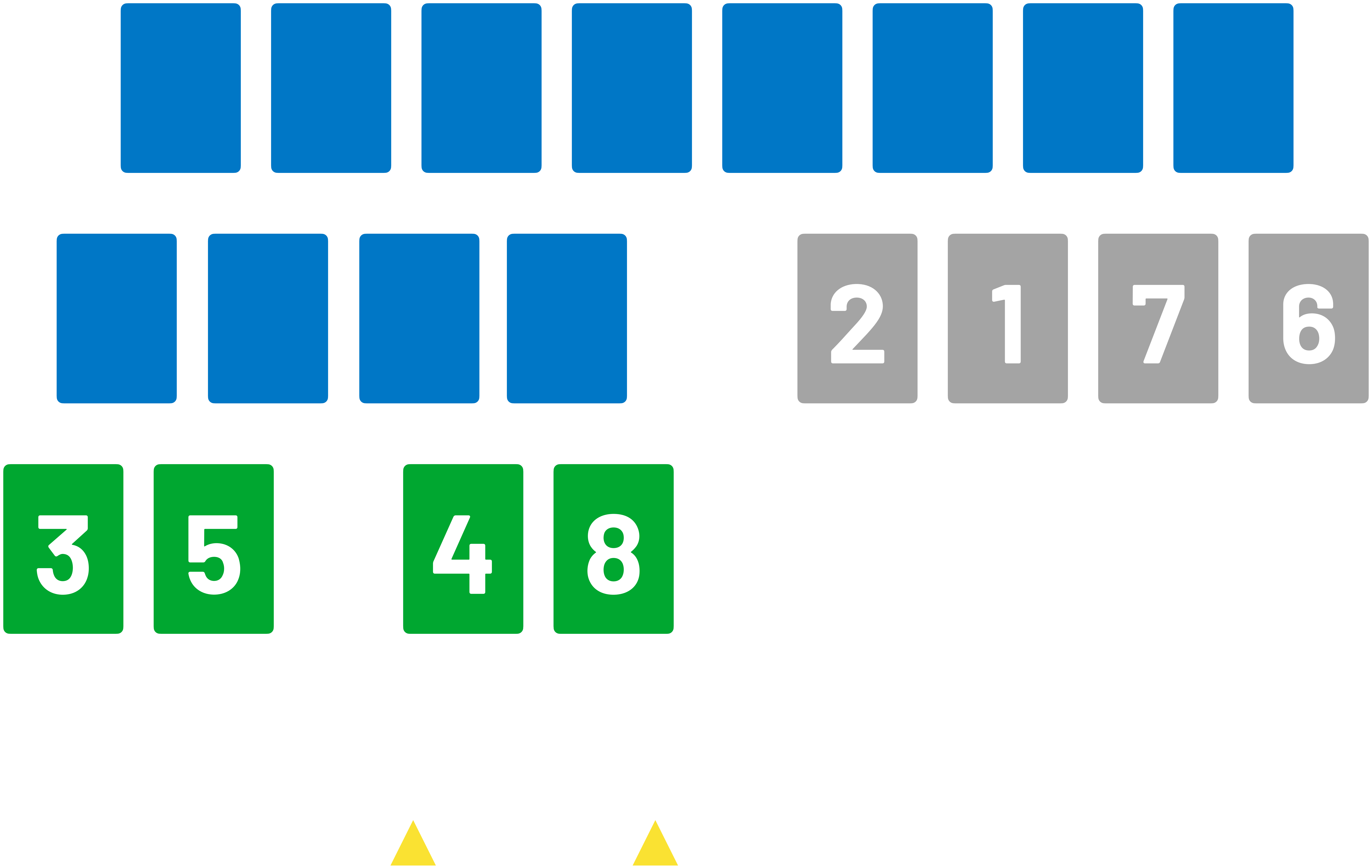

2
1
7
6
3
5
4
8

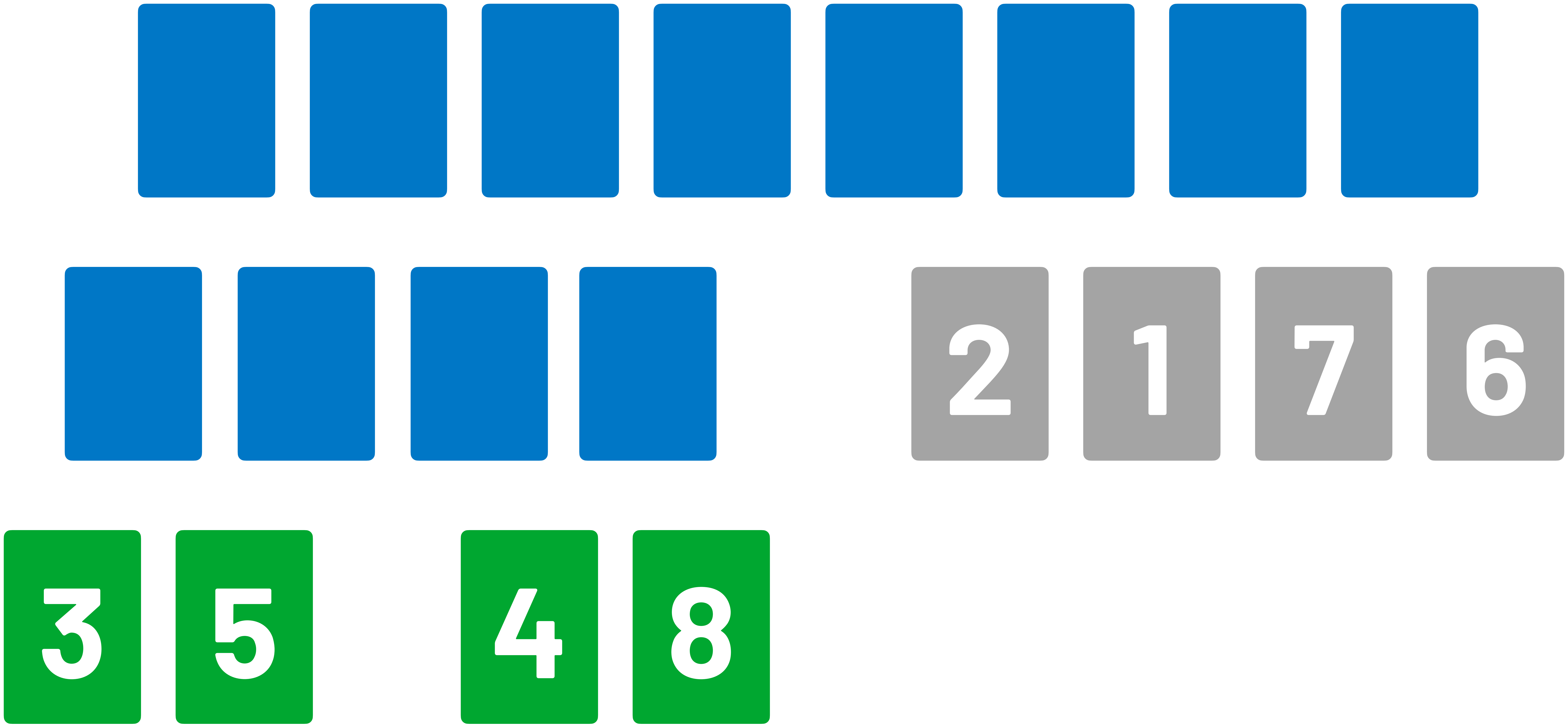

2
1
7
6
3
5
4
8

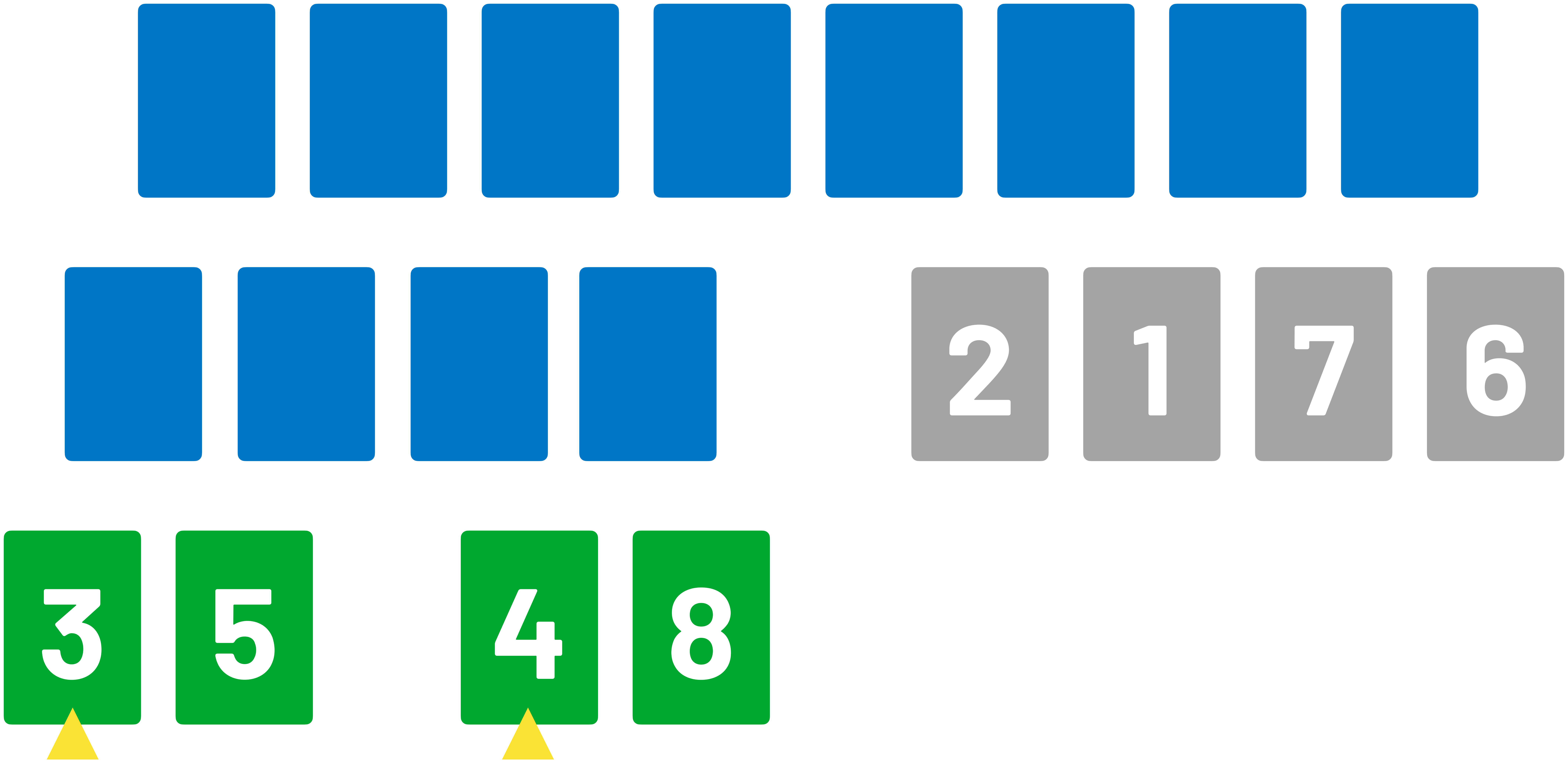

2
1
7
6
3
5
4
8

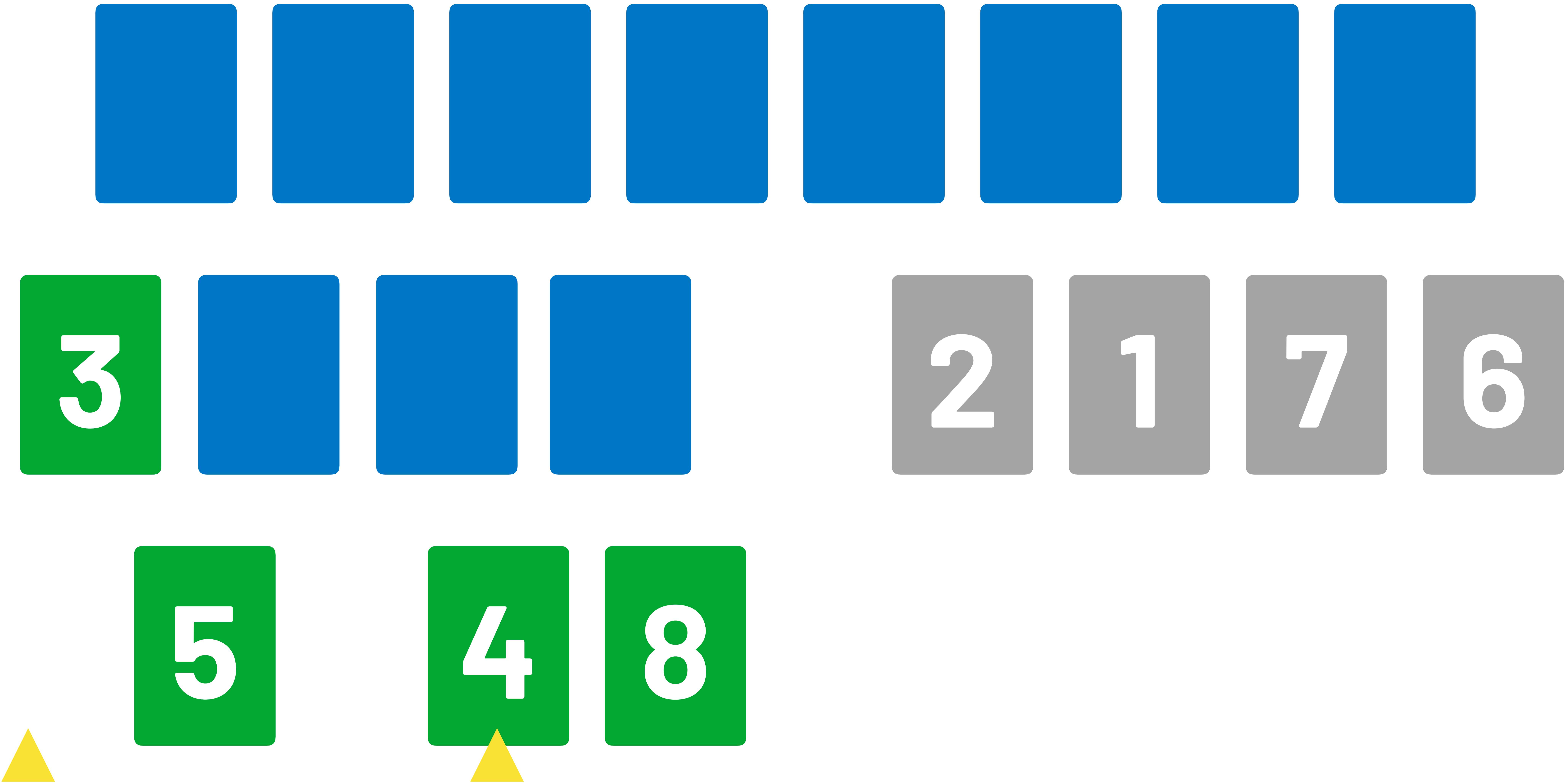

2
1
7
6
3
5
4
8

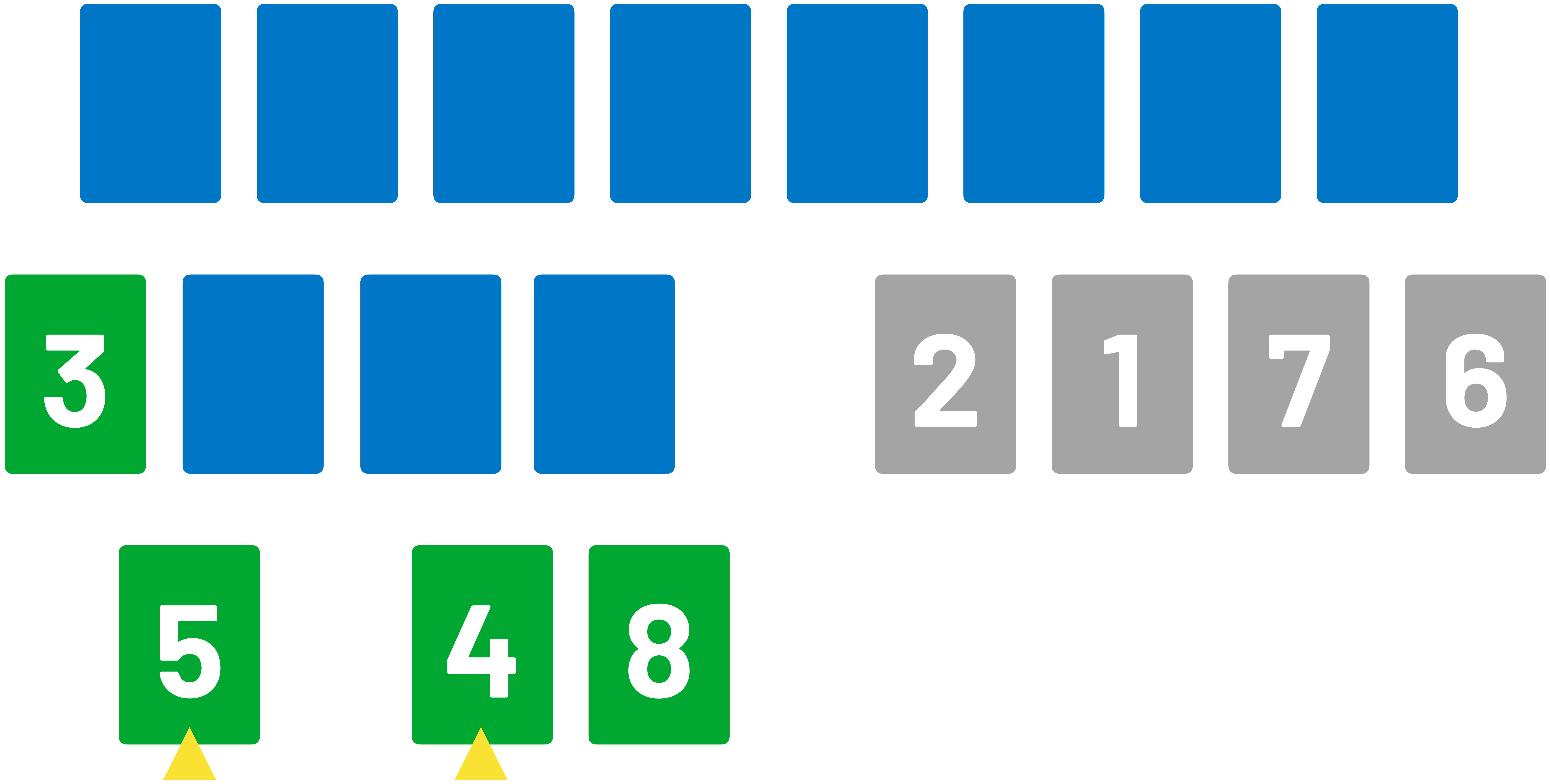

2
1
7
6
3
5
4
8

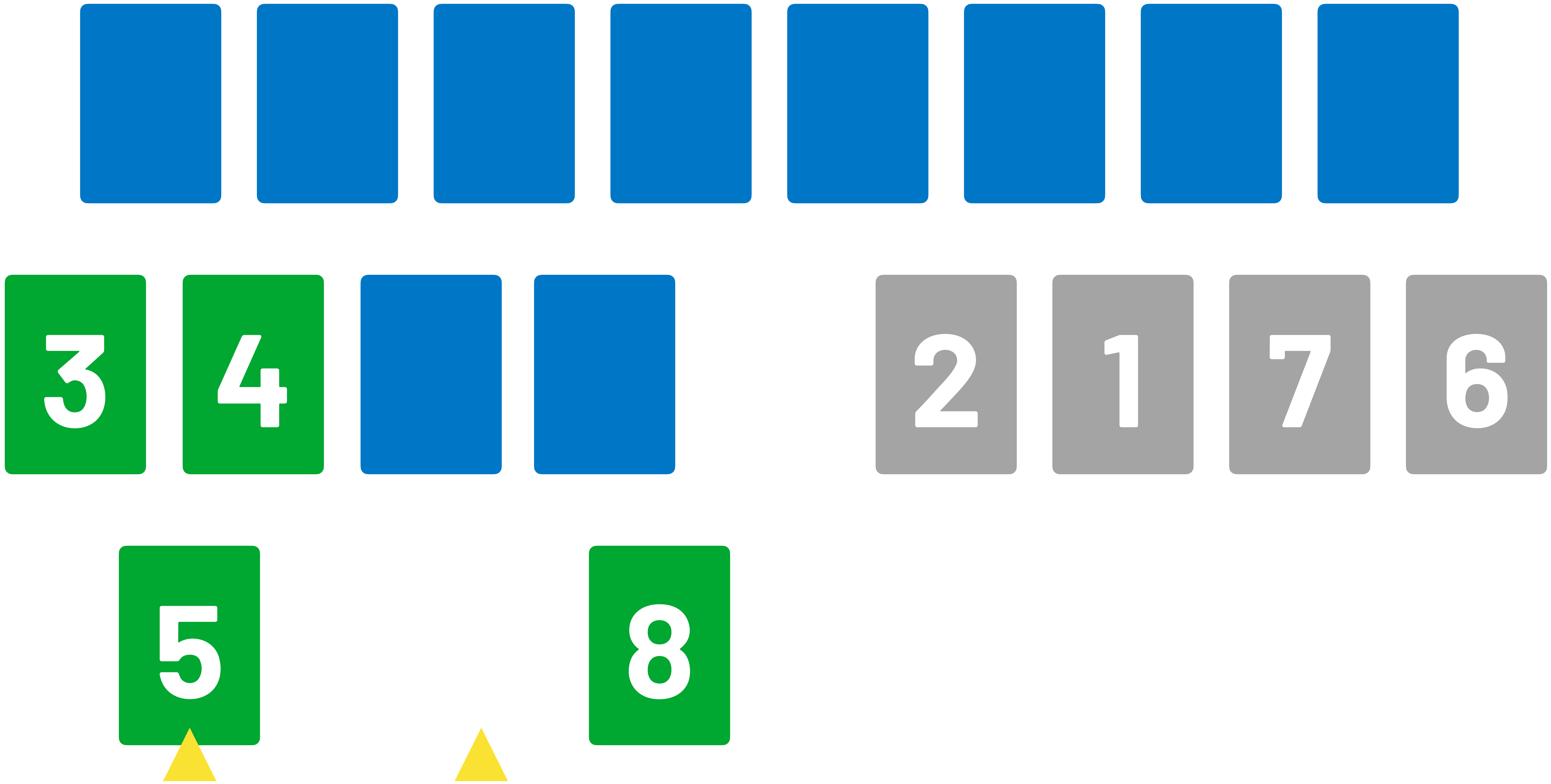

2
1
7
6
3
5
4
8

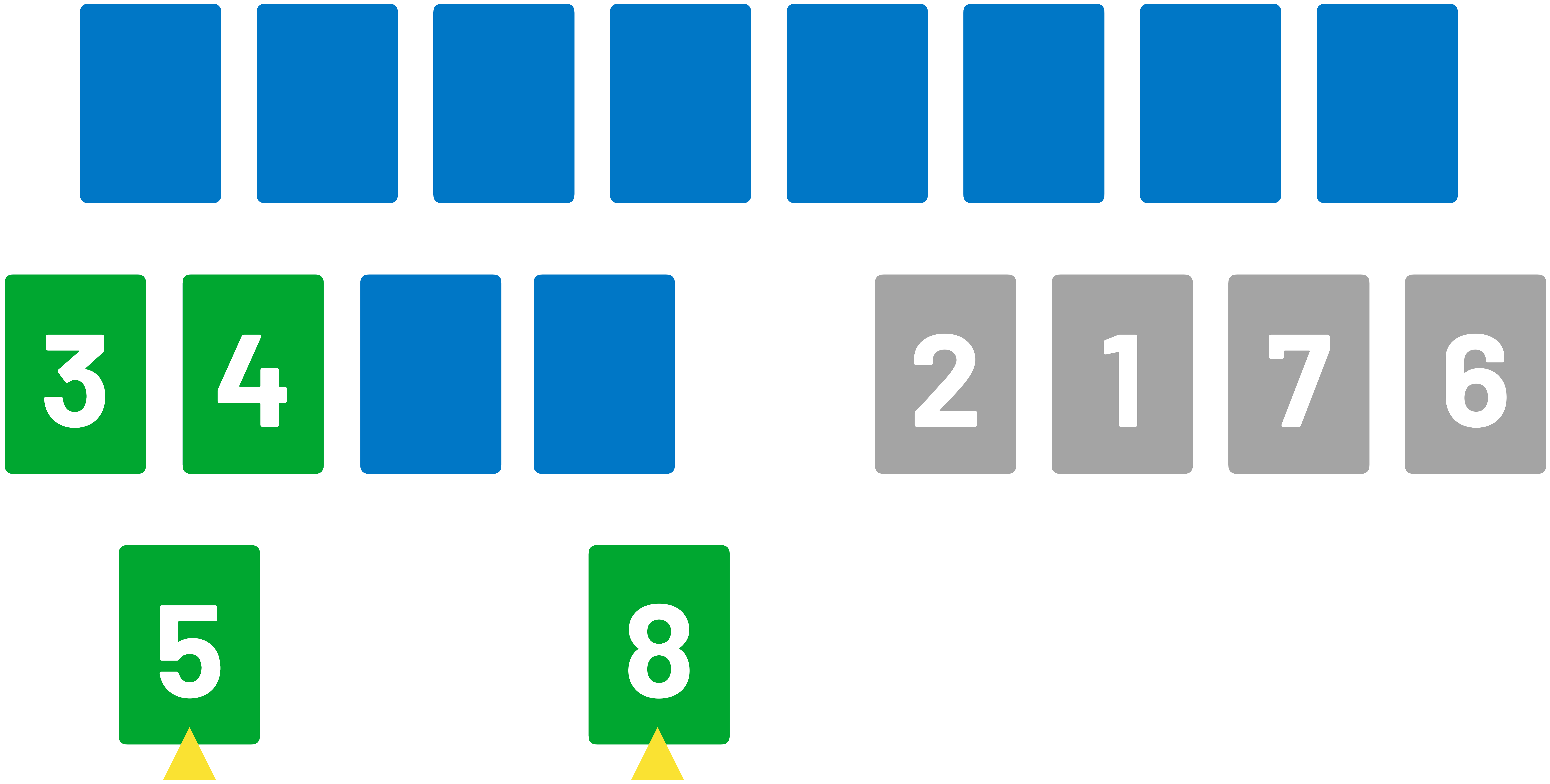

2 1 7 6
3 5
4 8


2
1
7
6
3
5
4
8

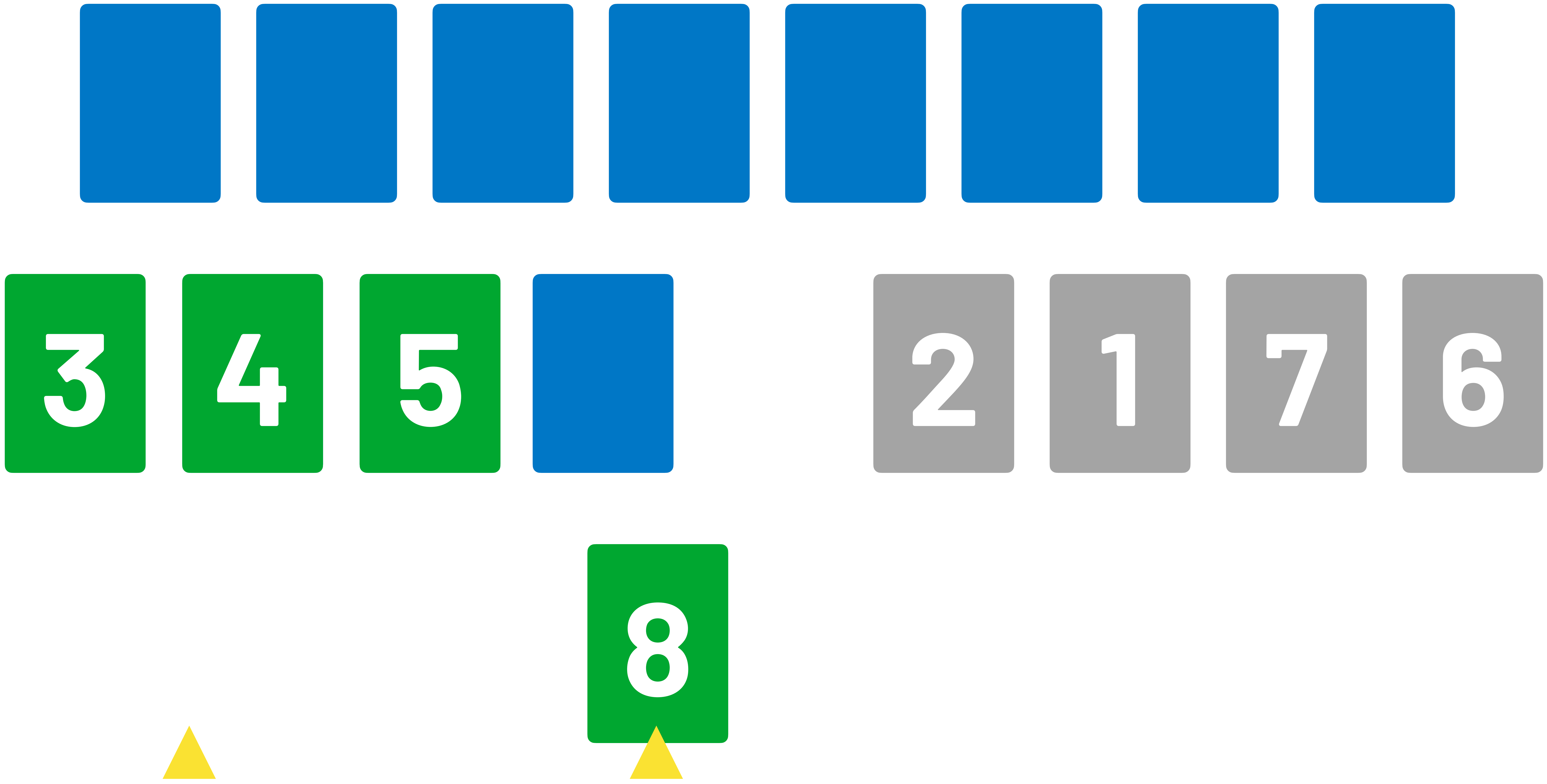

3
2
1
7
6
5
4
8

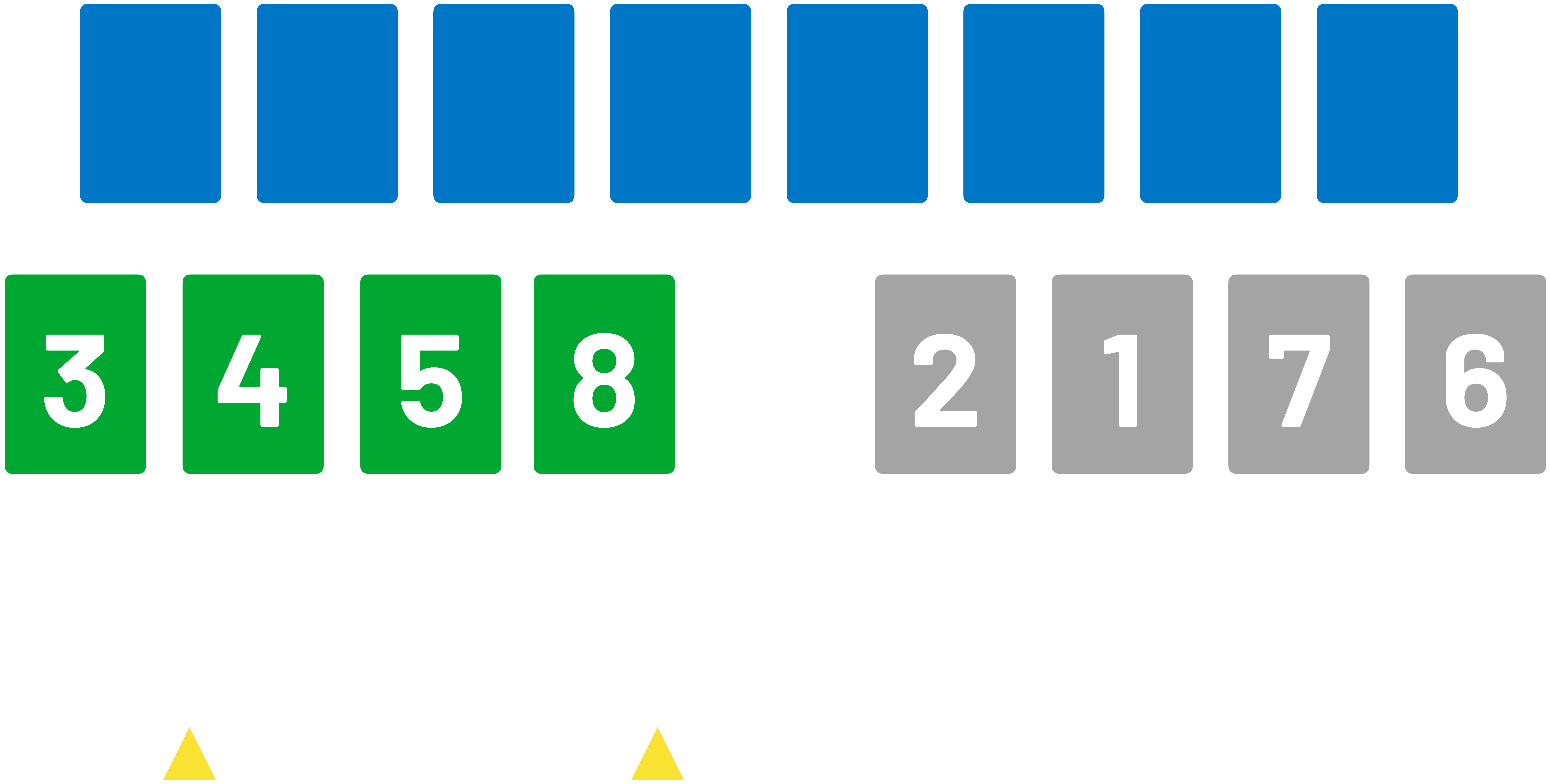

3
2
1
7
6
5
4
8

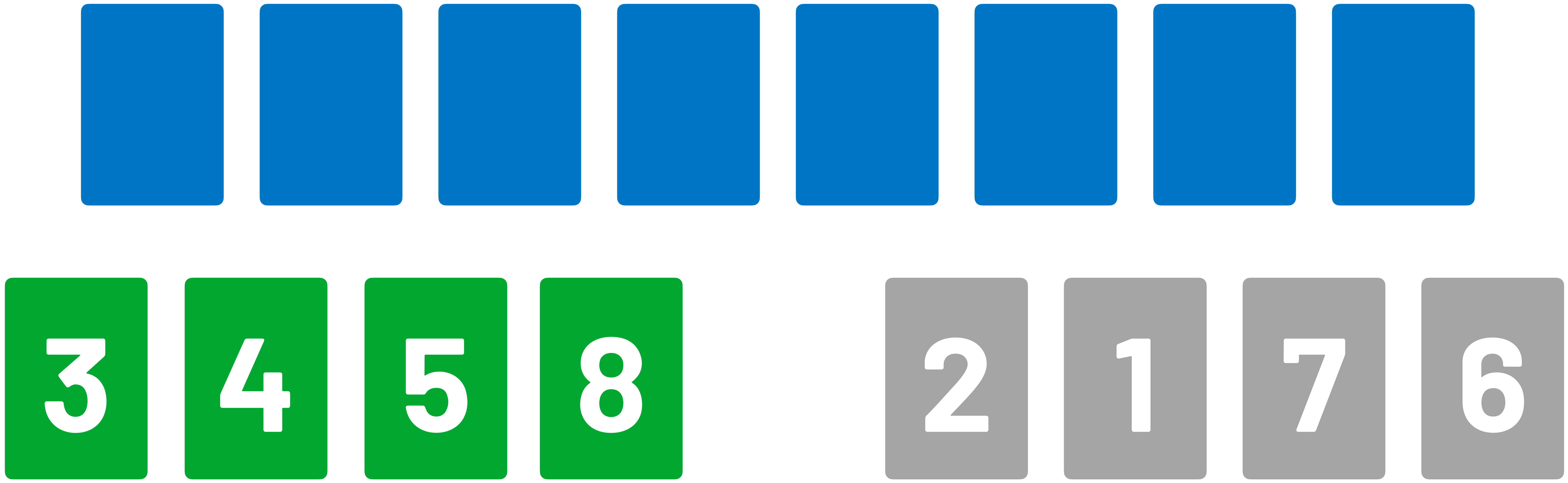

3
4
2
1
7
6
5
8

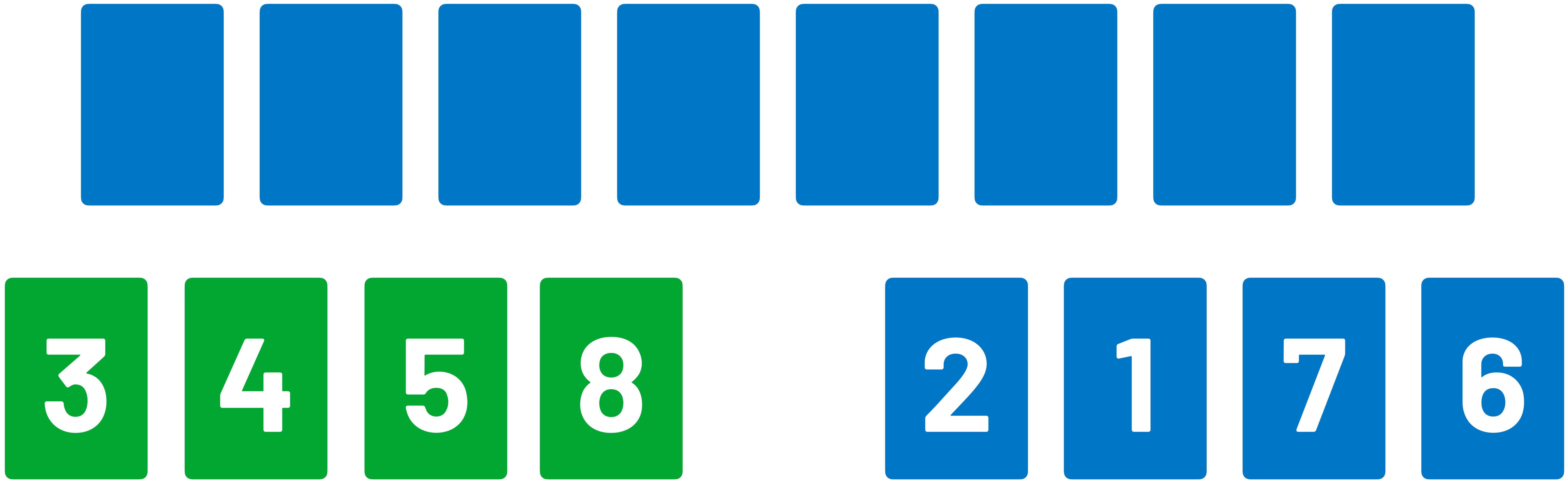


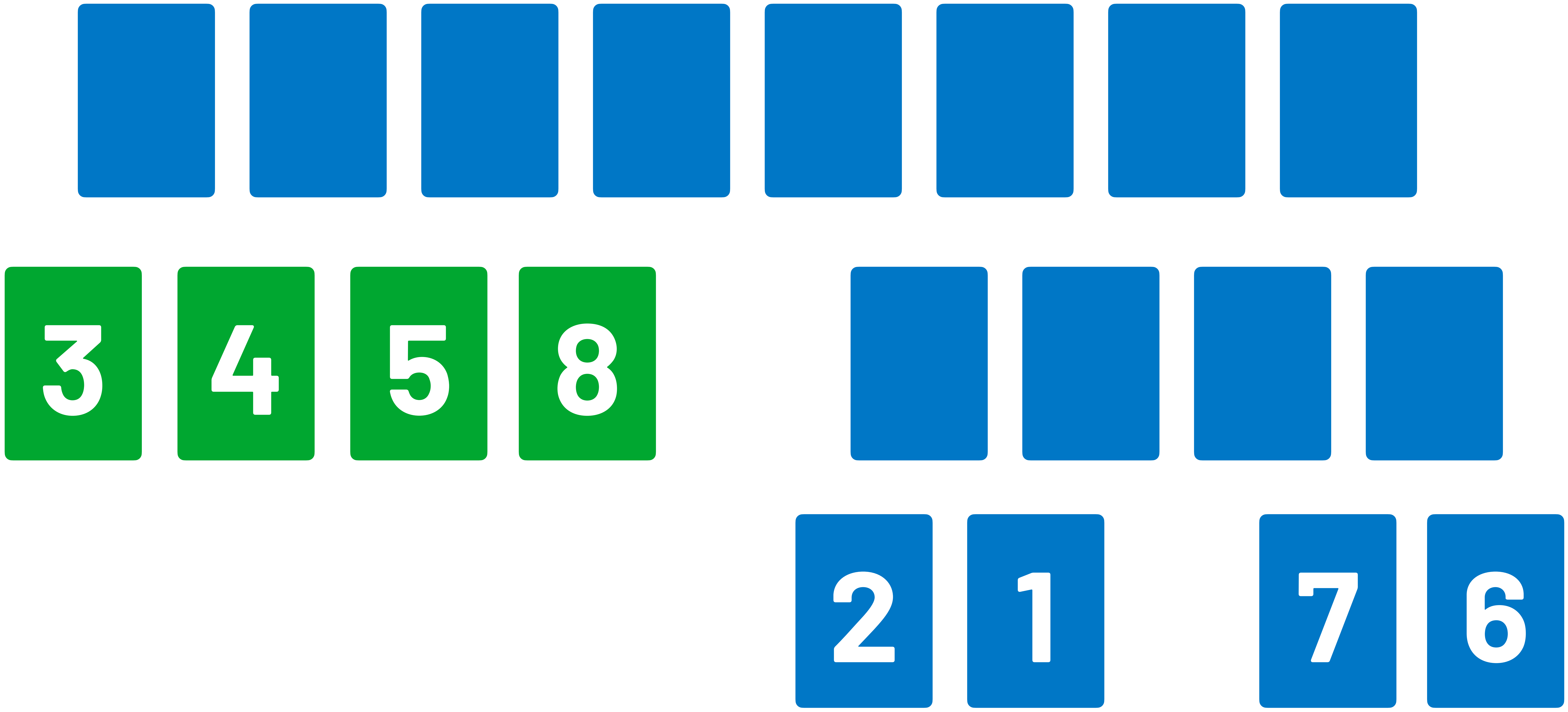
3
4
2
1
7
6
5
8

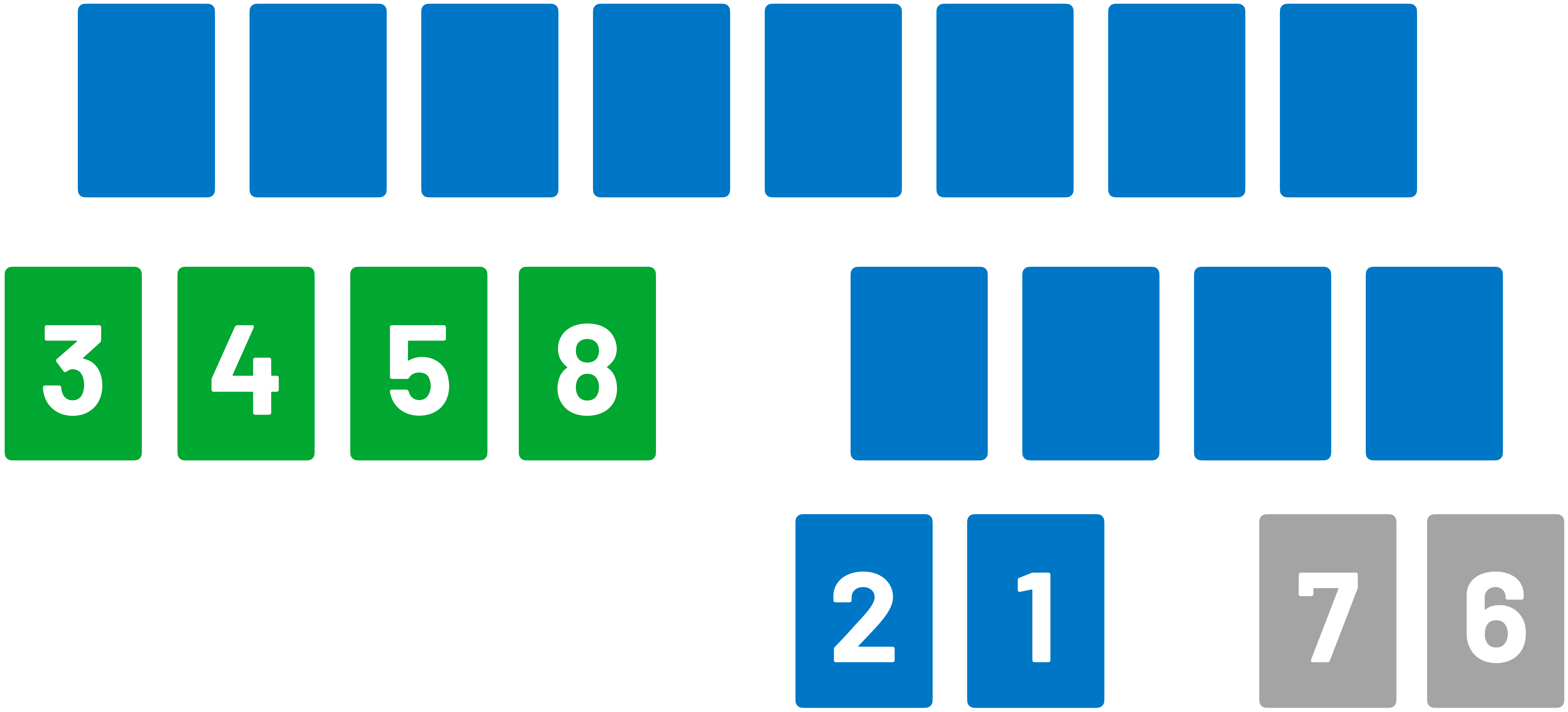

3
4
5
2
1
7
6
8

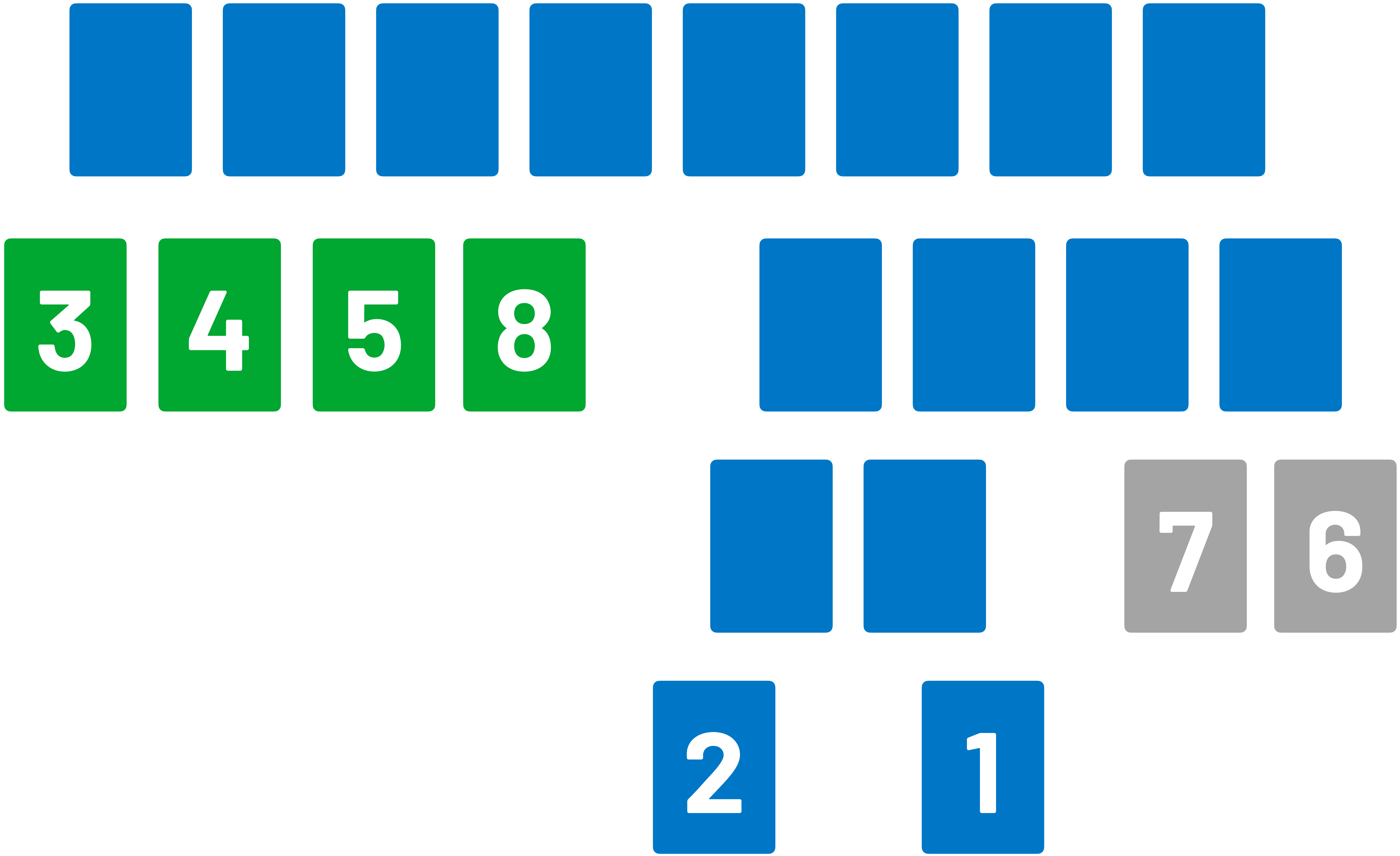

3
4
5
8
2
1
7
6

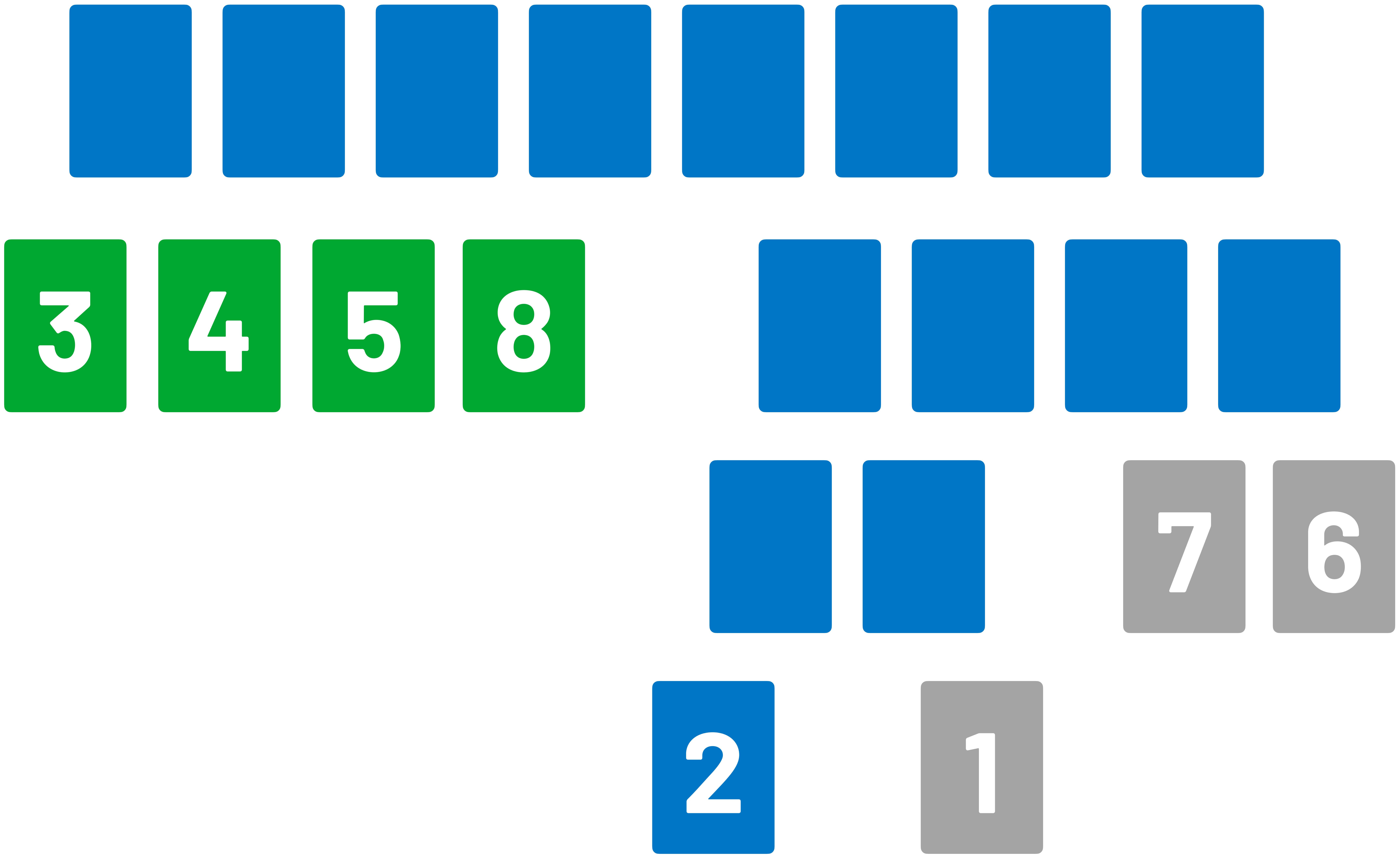

3 4 5 8
2 1 7 6

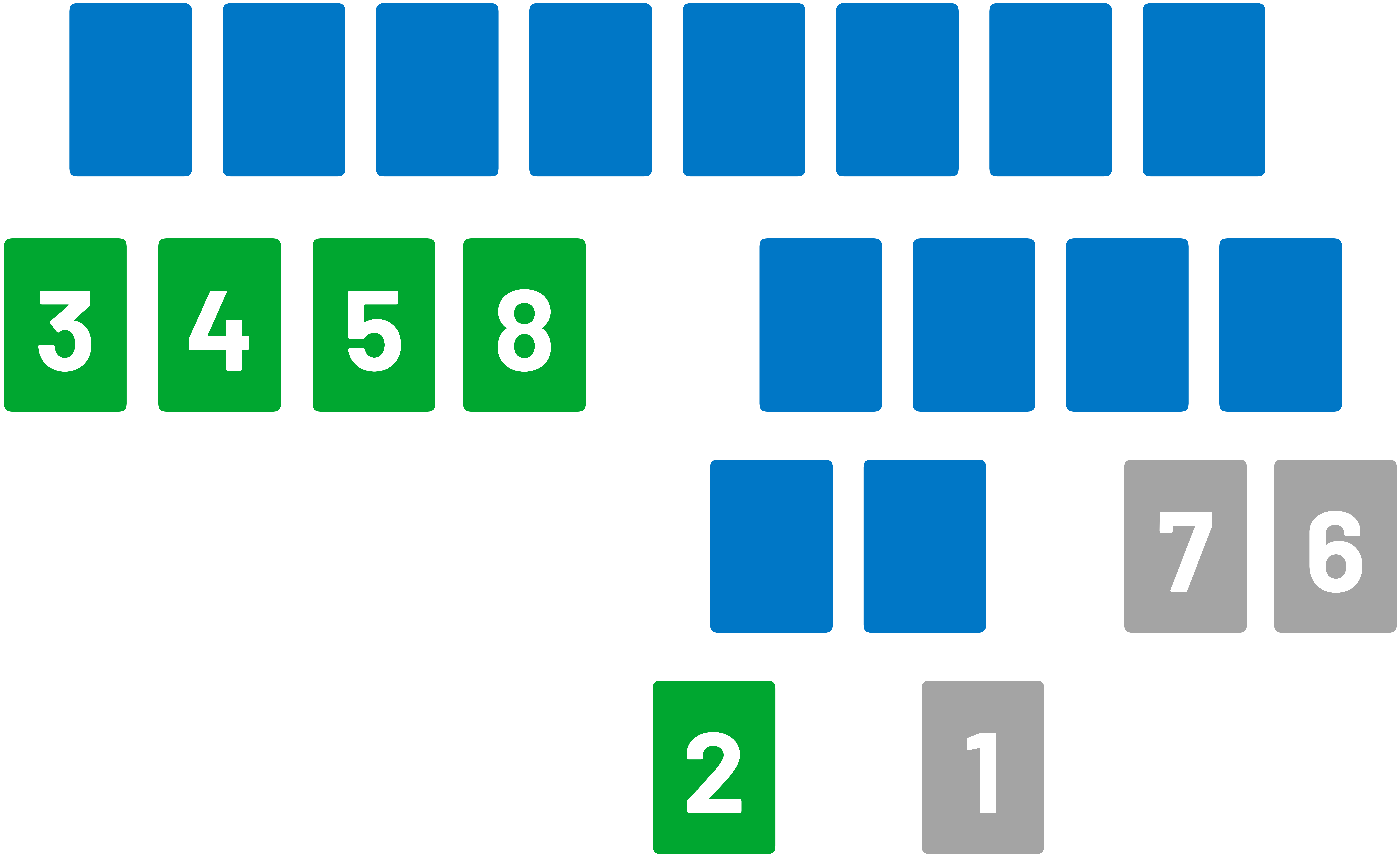

3
4
5
8
2
1
7
6

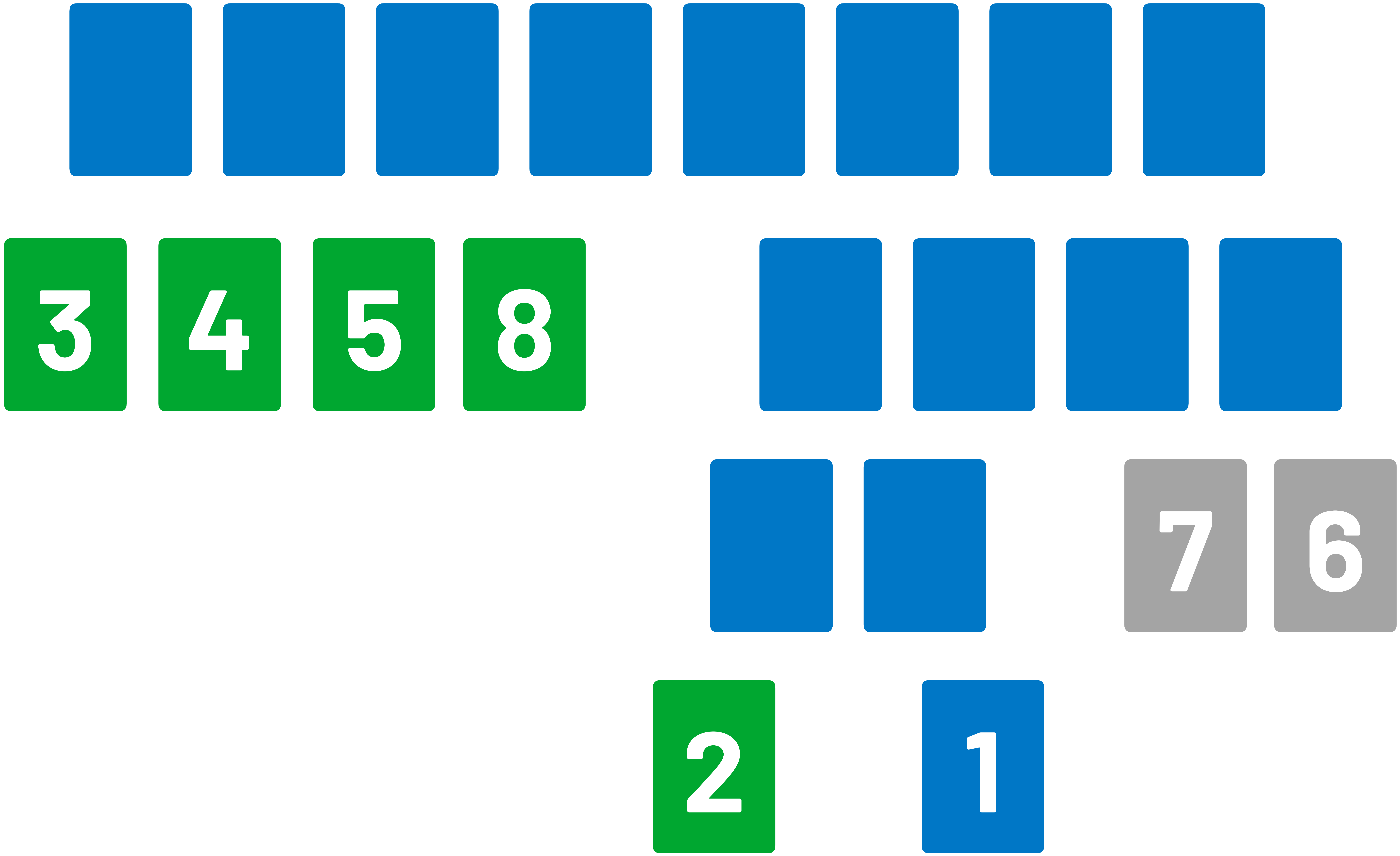

3
4
5
8
2
1
7
6

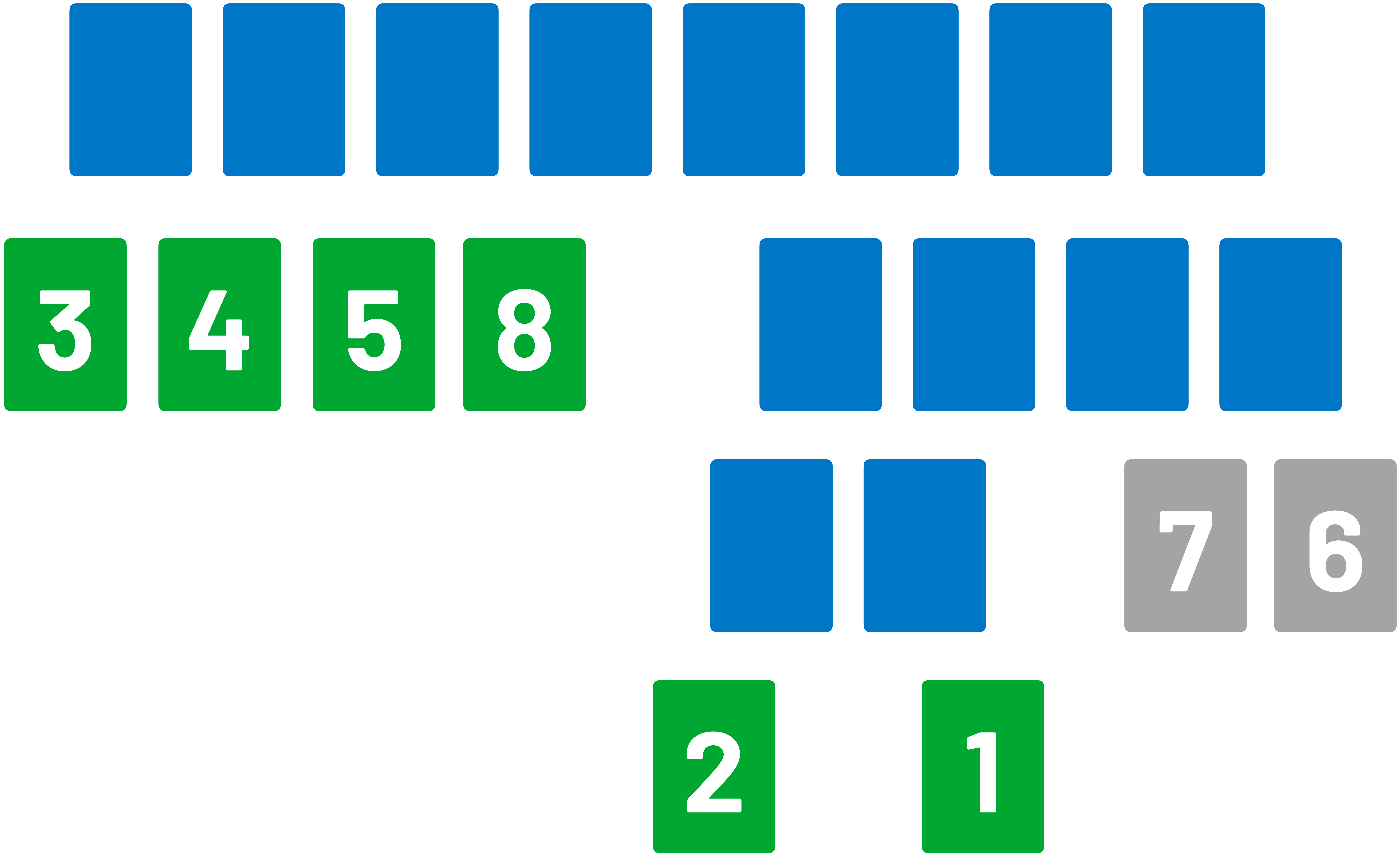

3
4
5
8
2
1
7
6

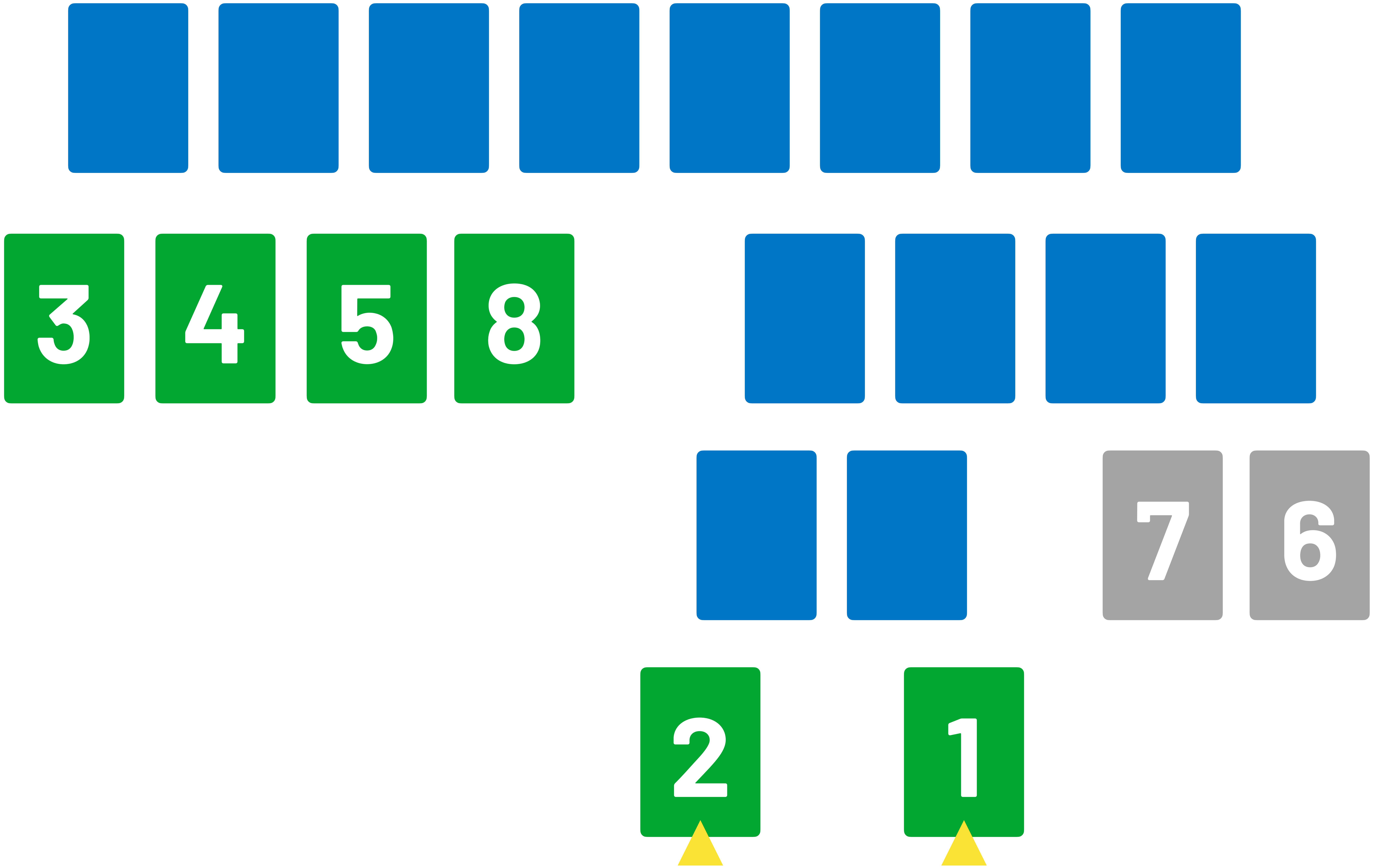

3
4
5
8
7
6
2
1

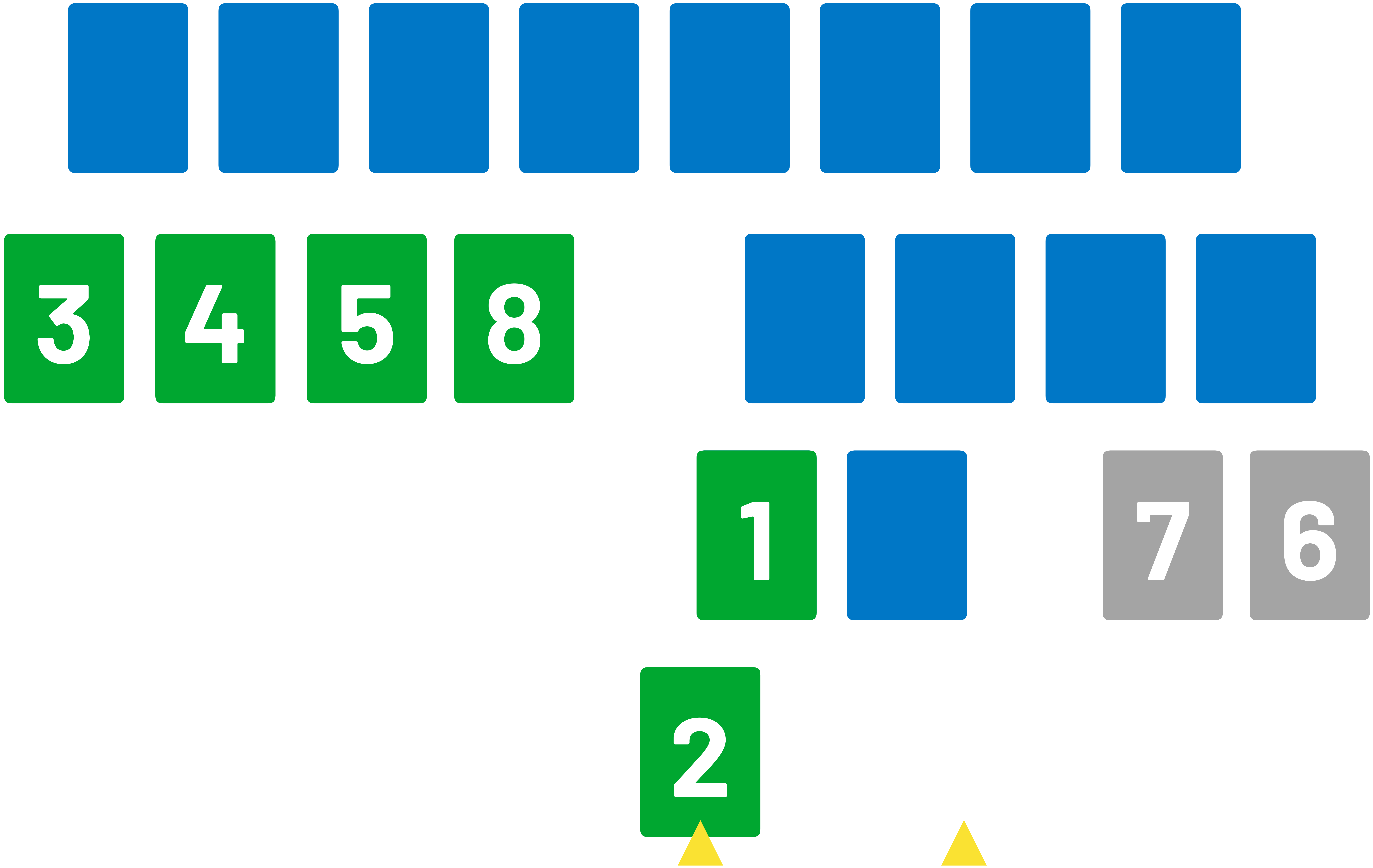

3
4
5
8
7
6
2
1

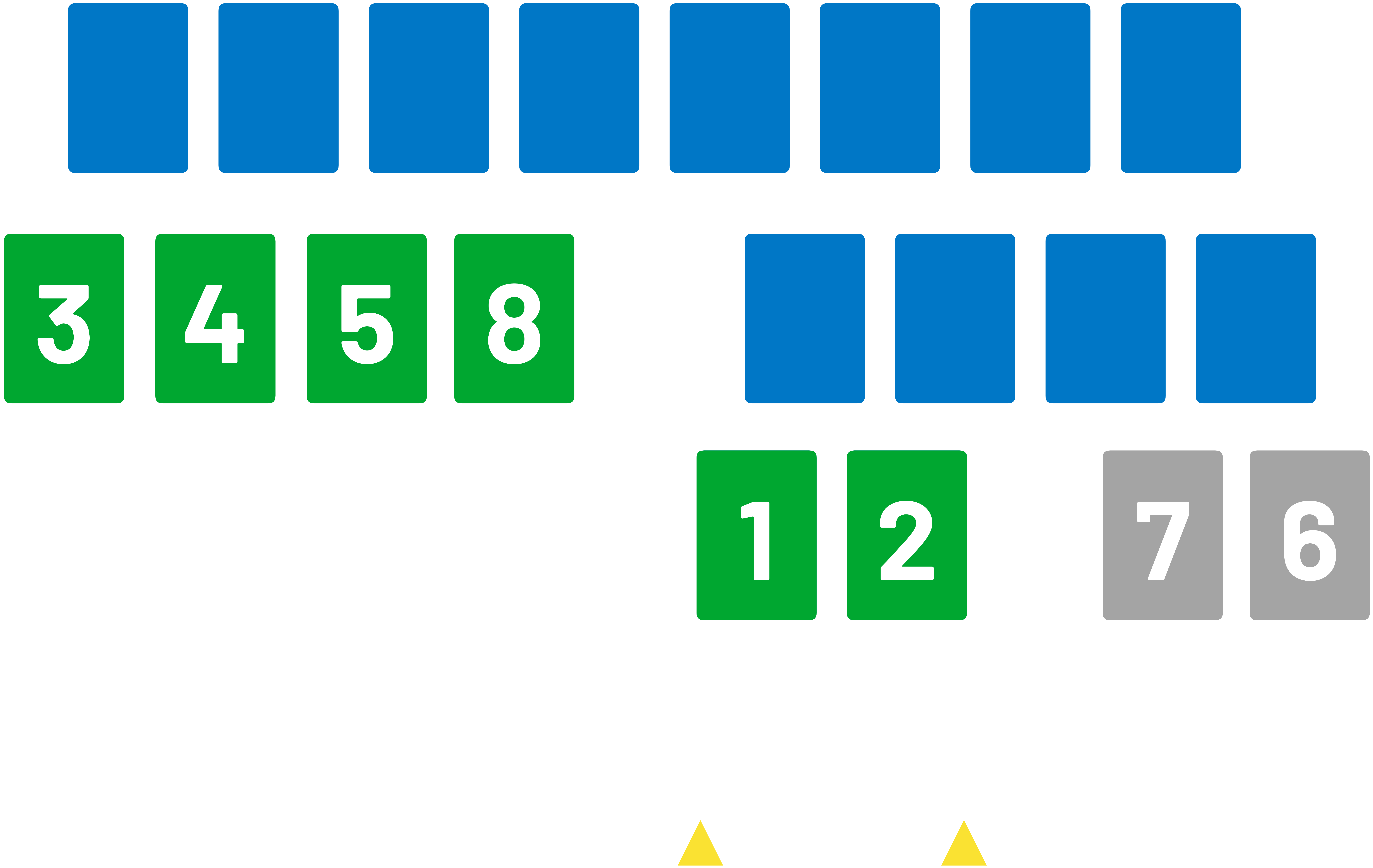

3
4
5
8
7
6
2
1

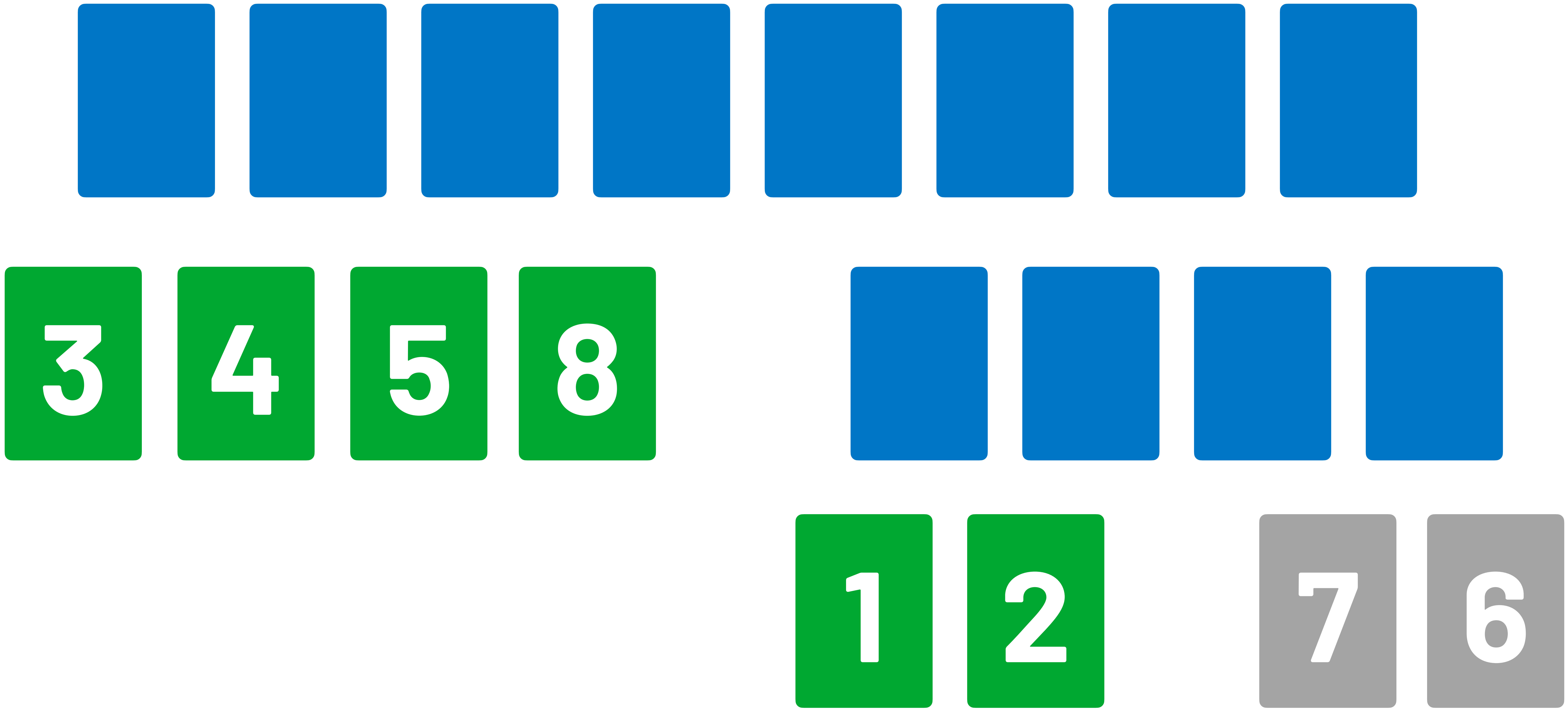

3
4
5
8
7
6
2
1

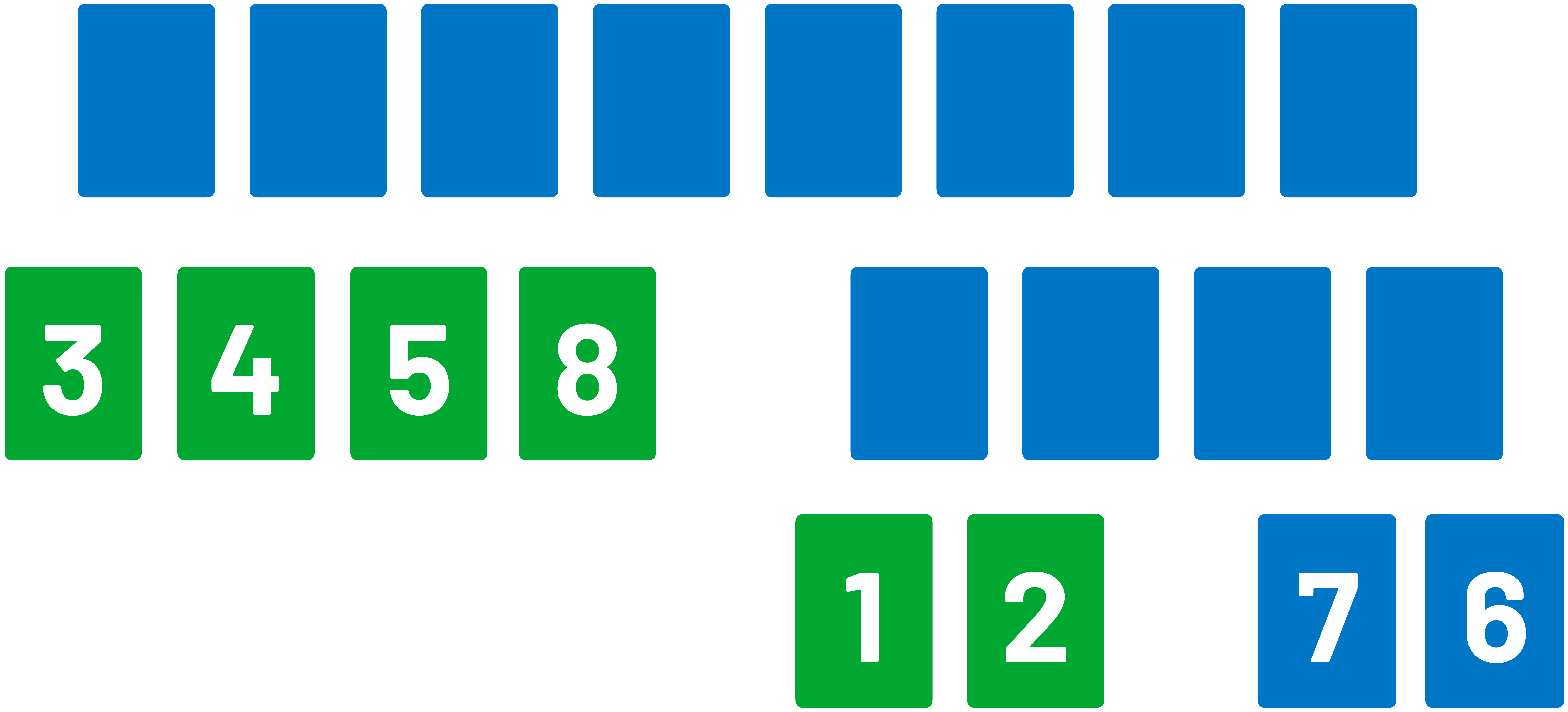

3
4
5
8
7
6
2
1

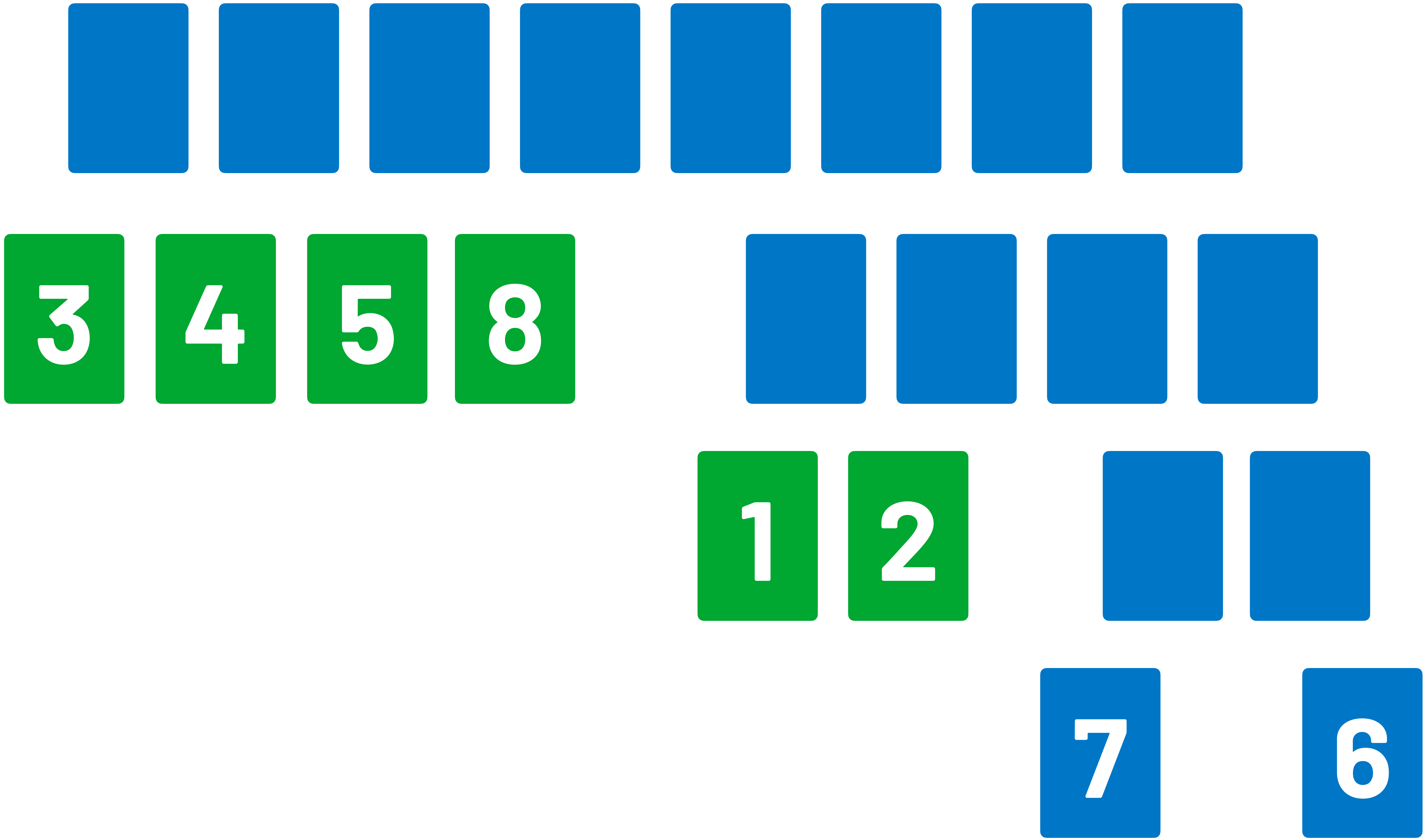

3
4
5
8
7
6
2
1

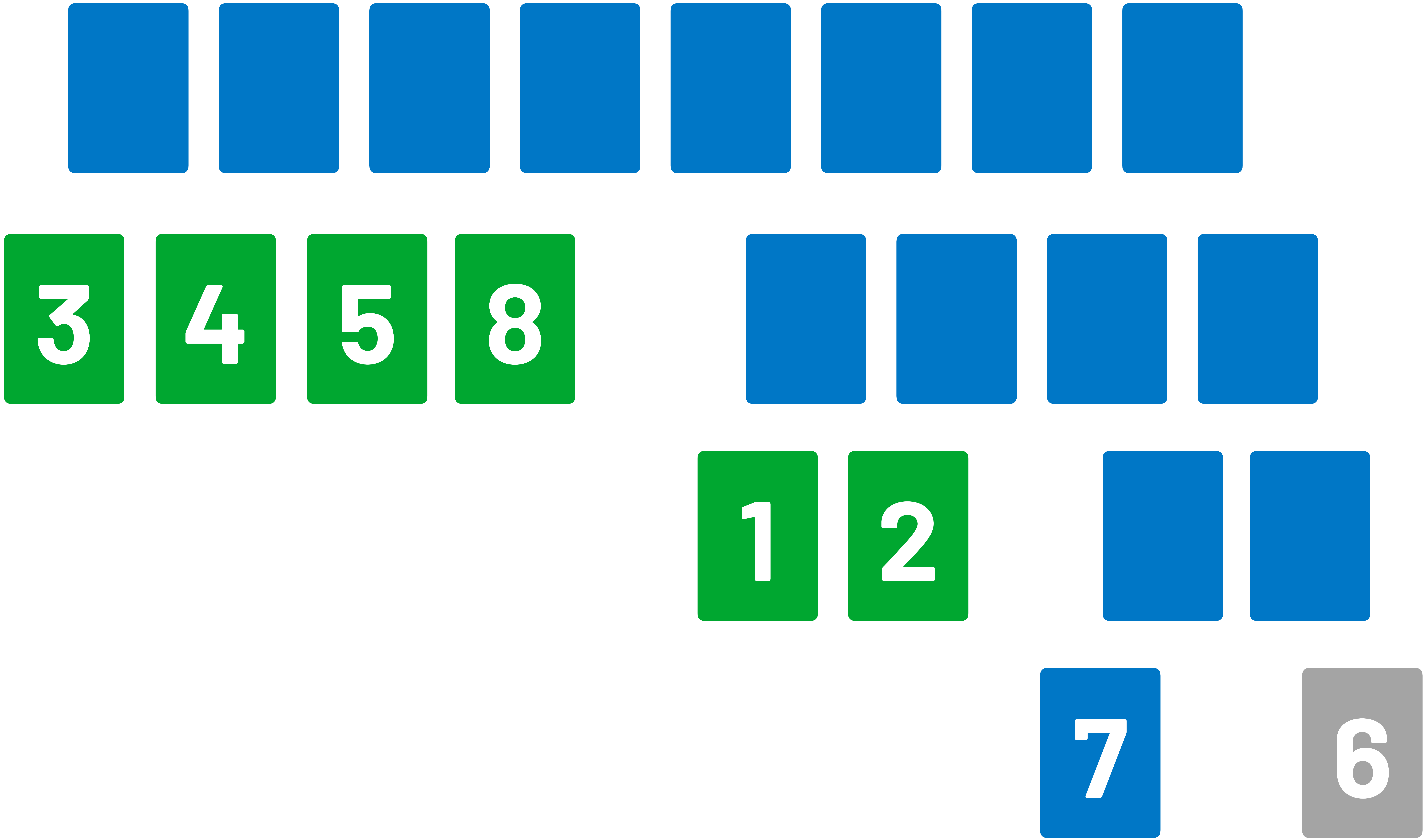

3
4
5
8
1
7
6
2

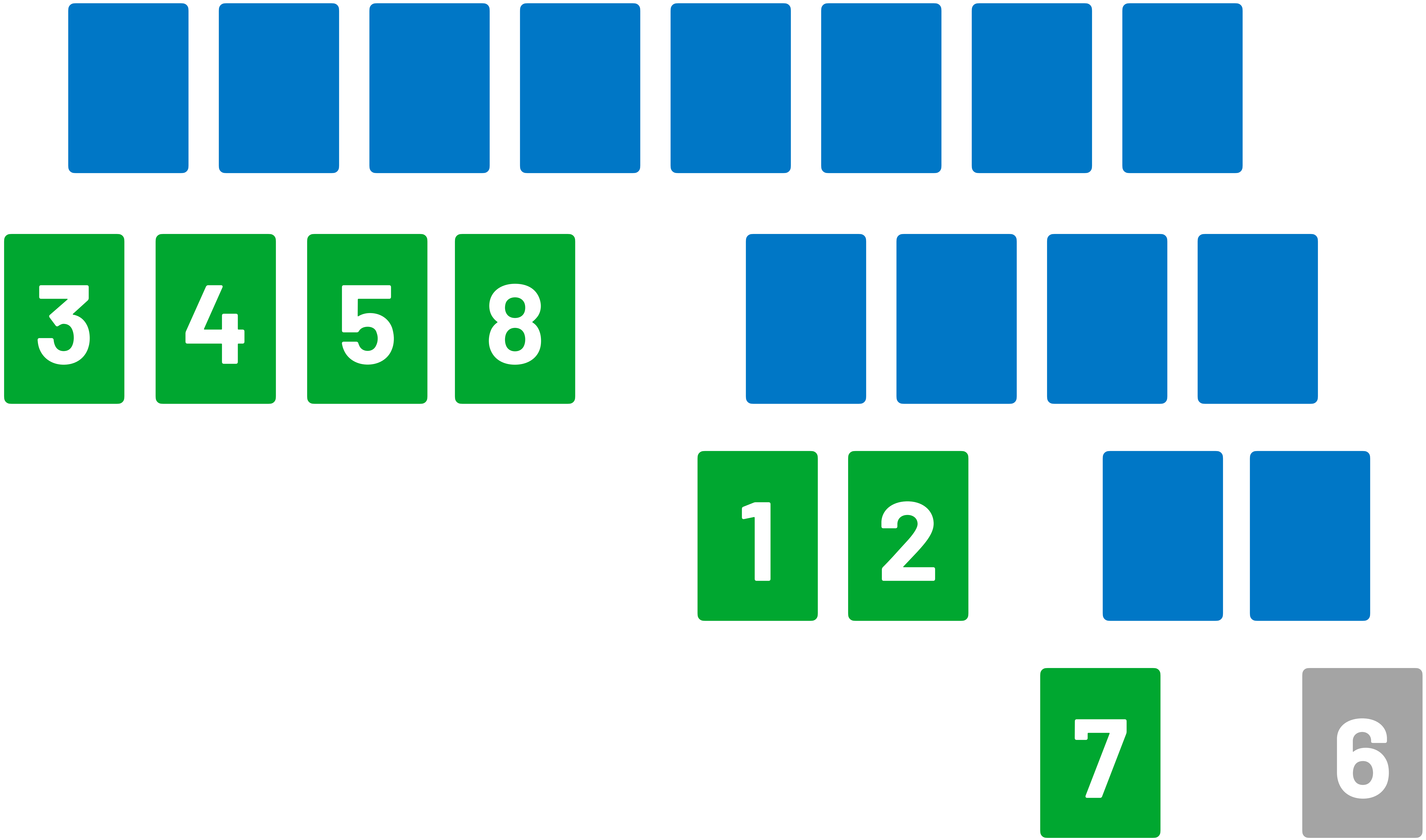

3
4
5
8
1
2
7
6

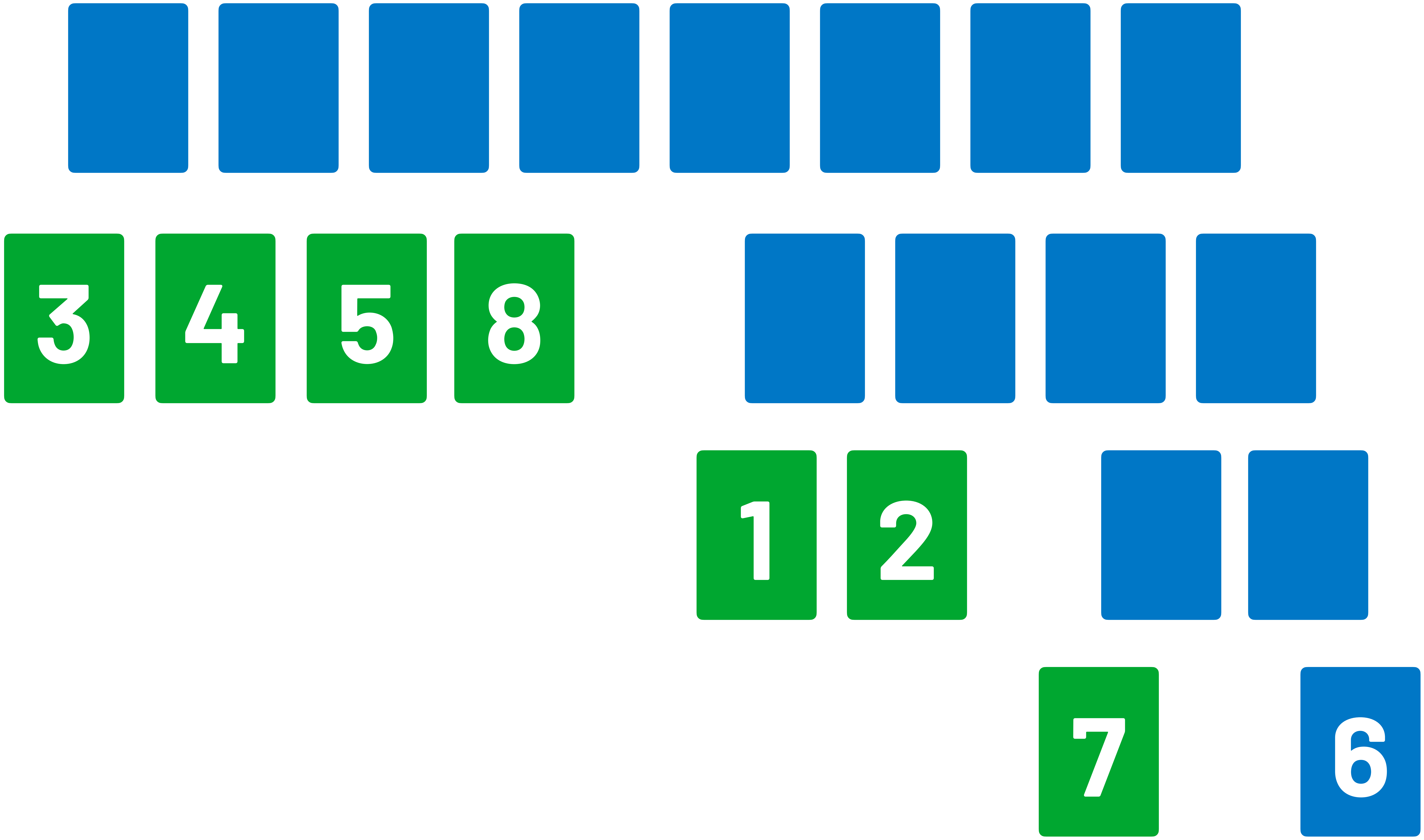

3
4
5
8
1
2
7
6

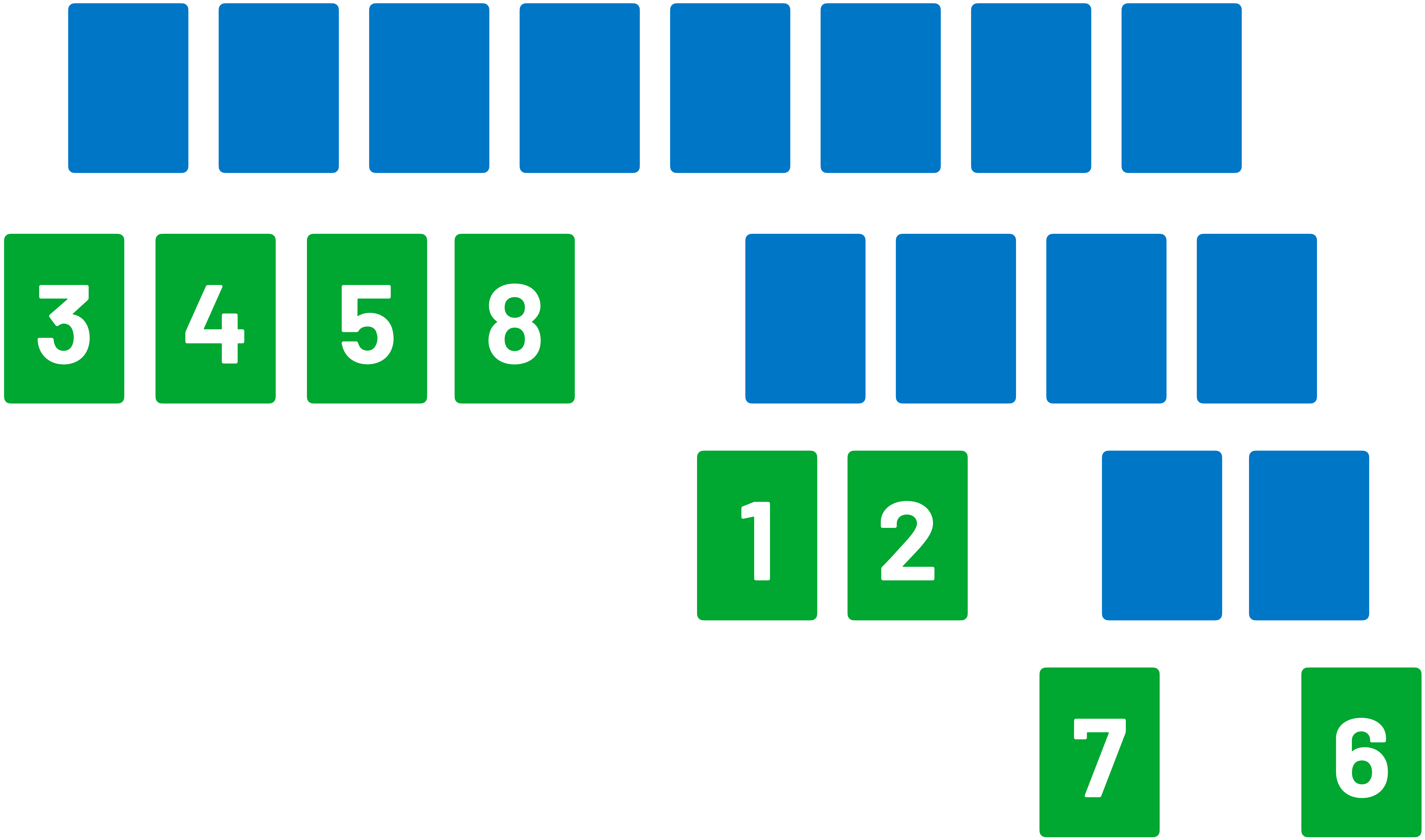

3
4
5
8
1
2
7
6

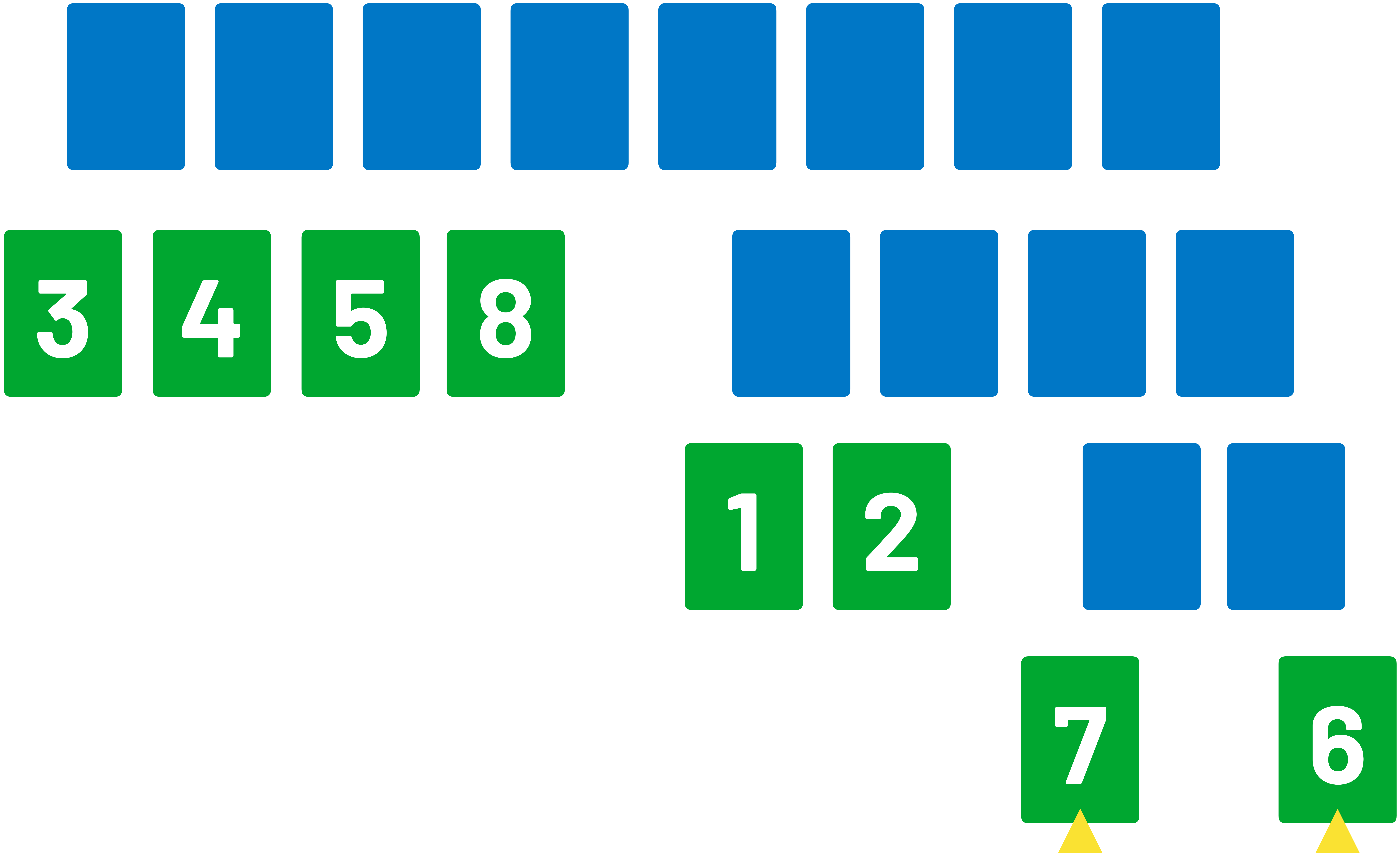

3
4
5
8
1
2
7
6

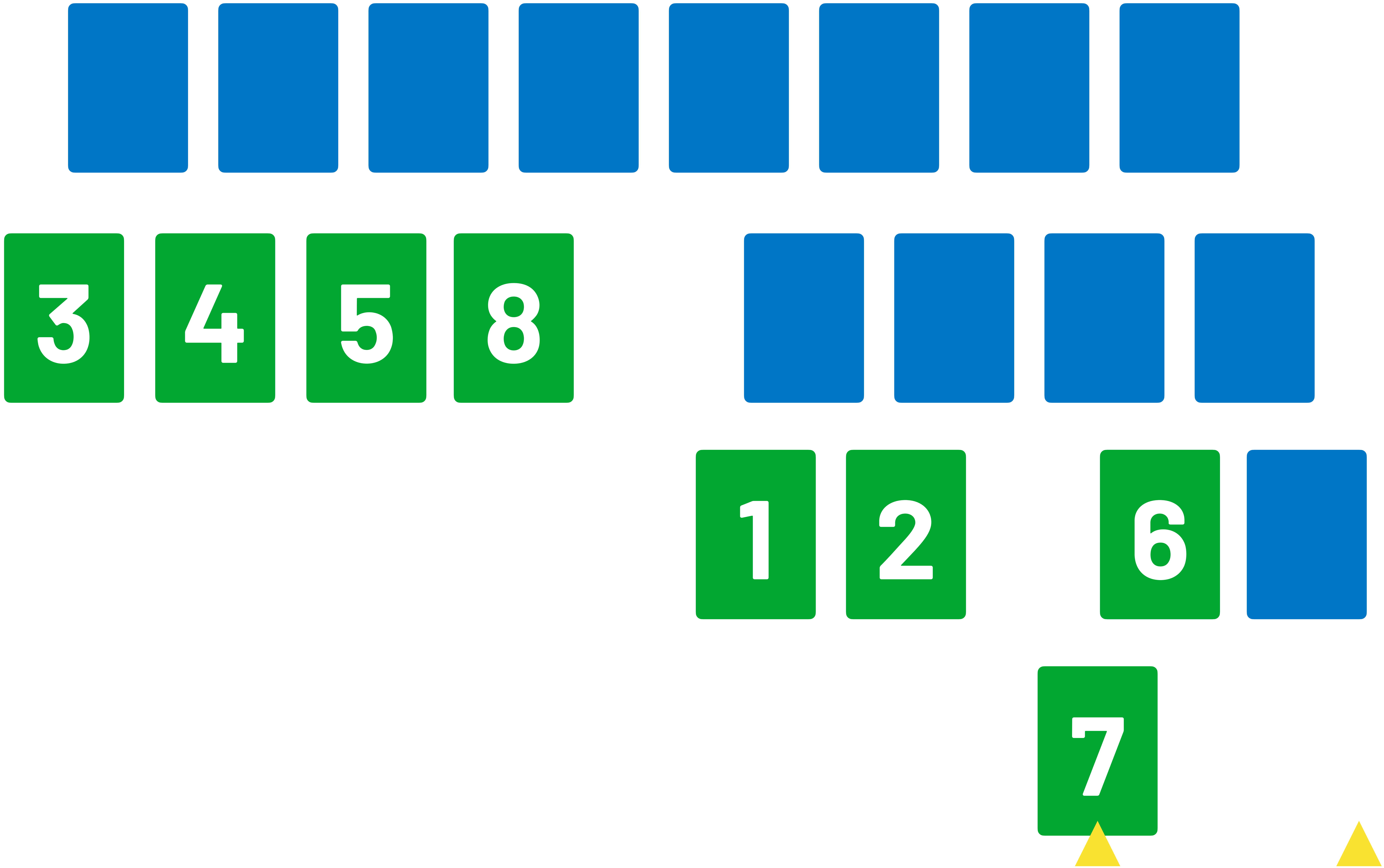

3
4
5
8
1
2
7
6

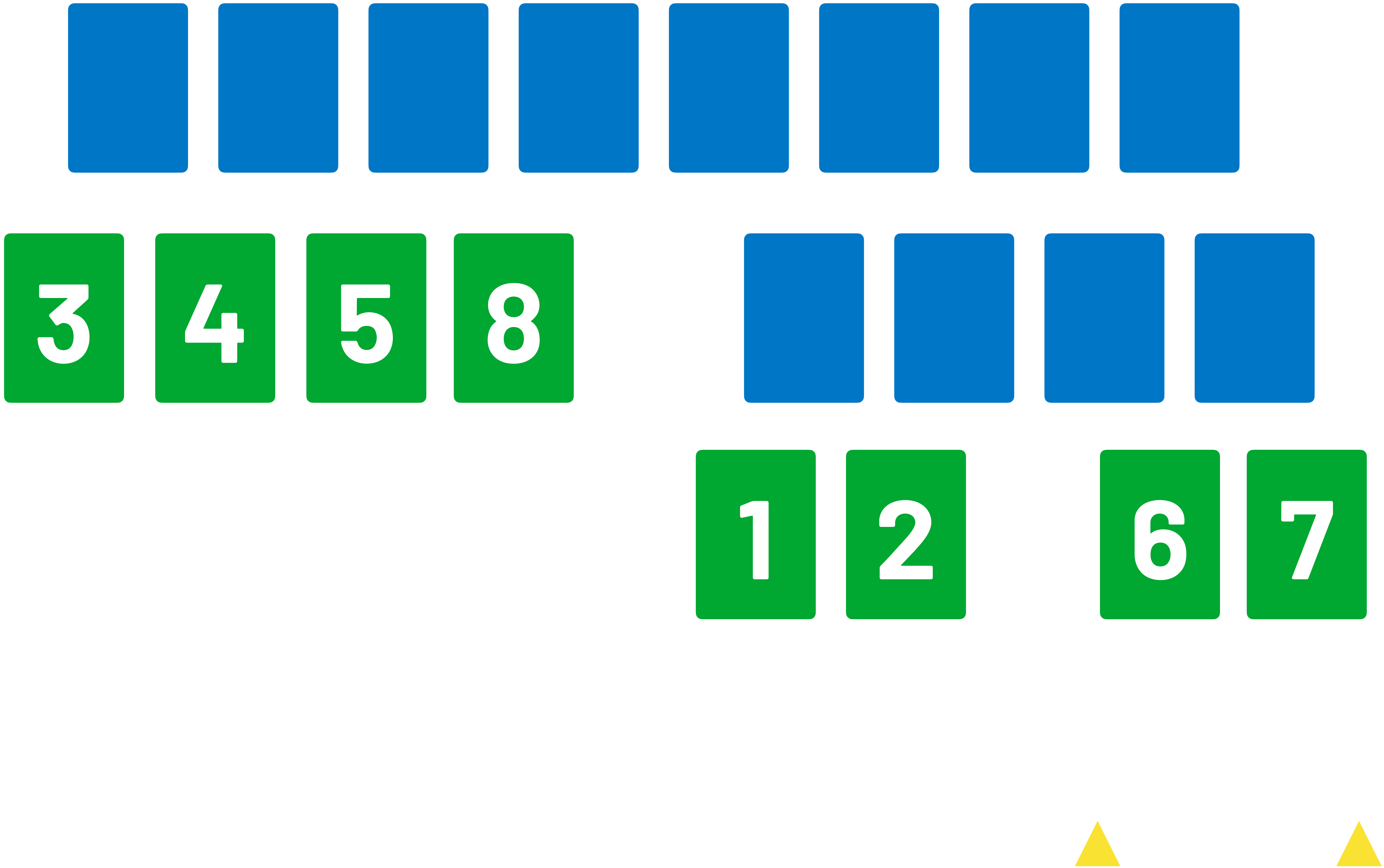

3
4
5
8
1
2
7
6


3
4
5
8
1
2
7
6

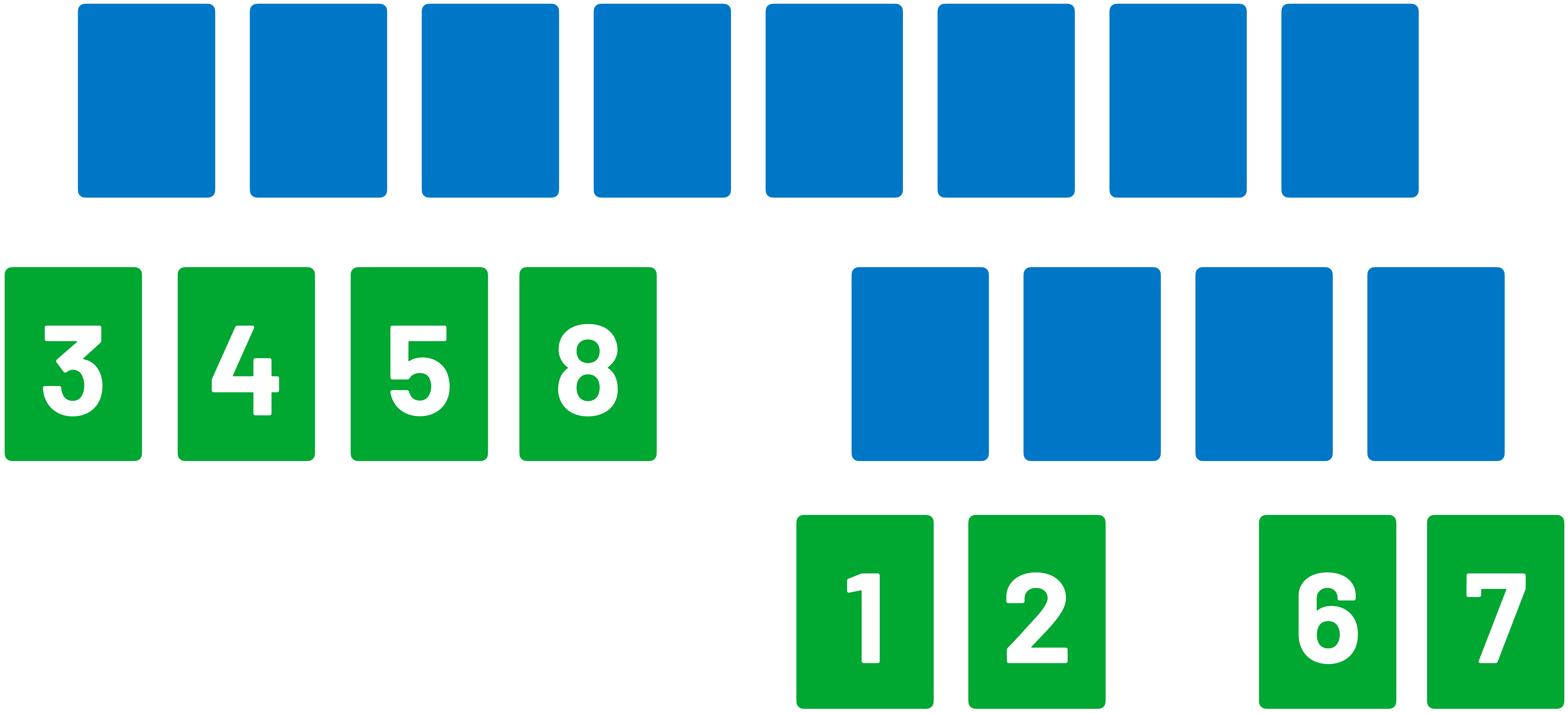

3
4
5
8
1
2
7
6

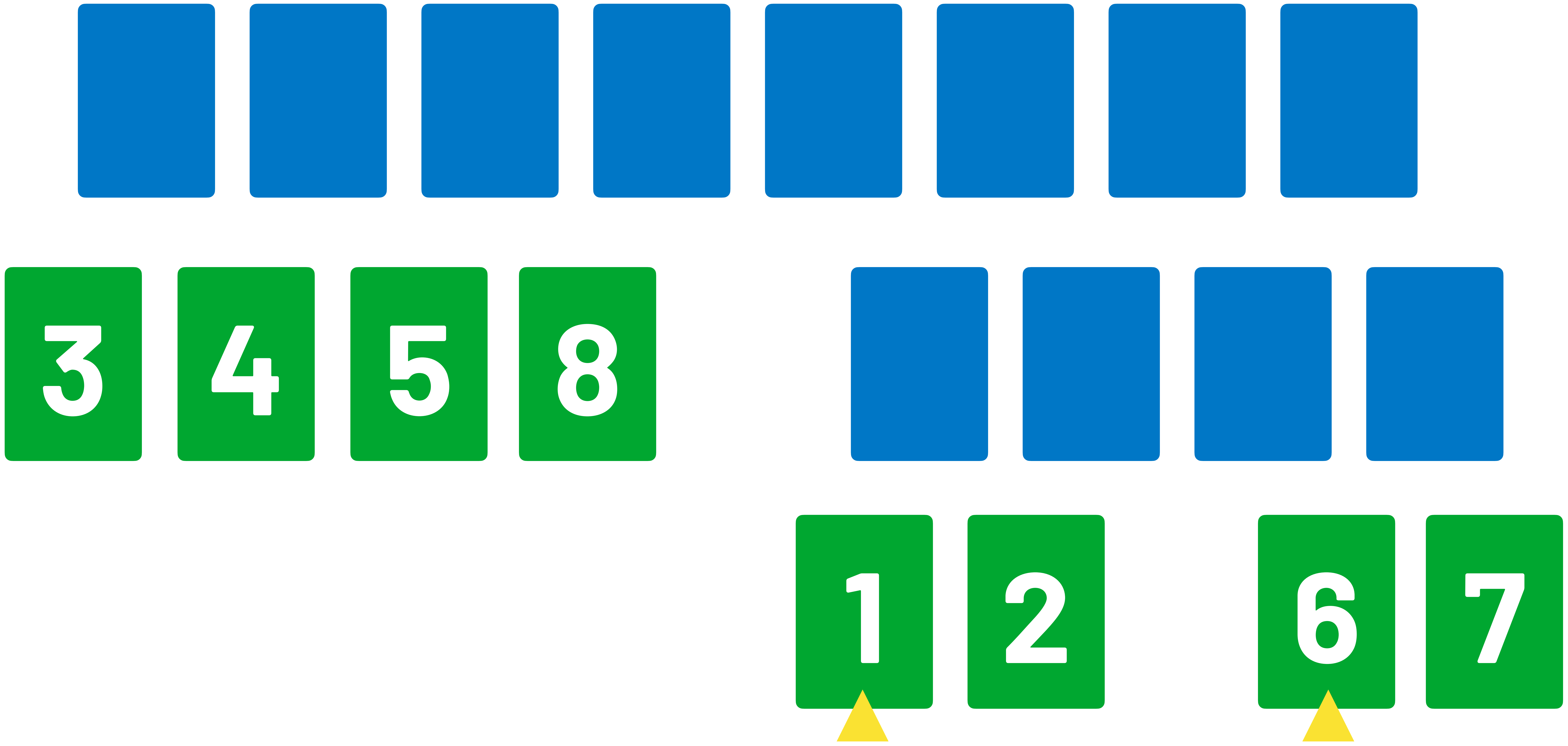

3
4
5
8
1
2
7
6

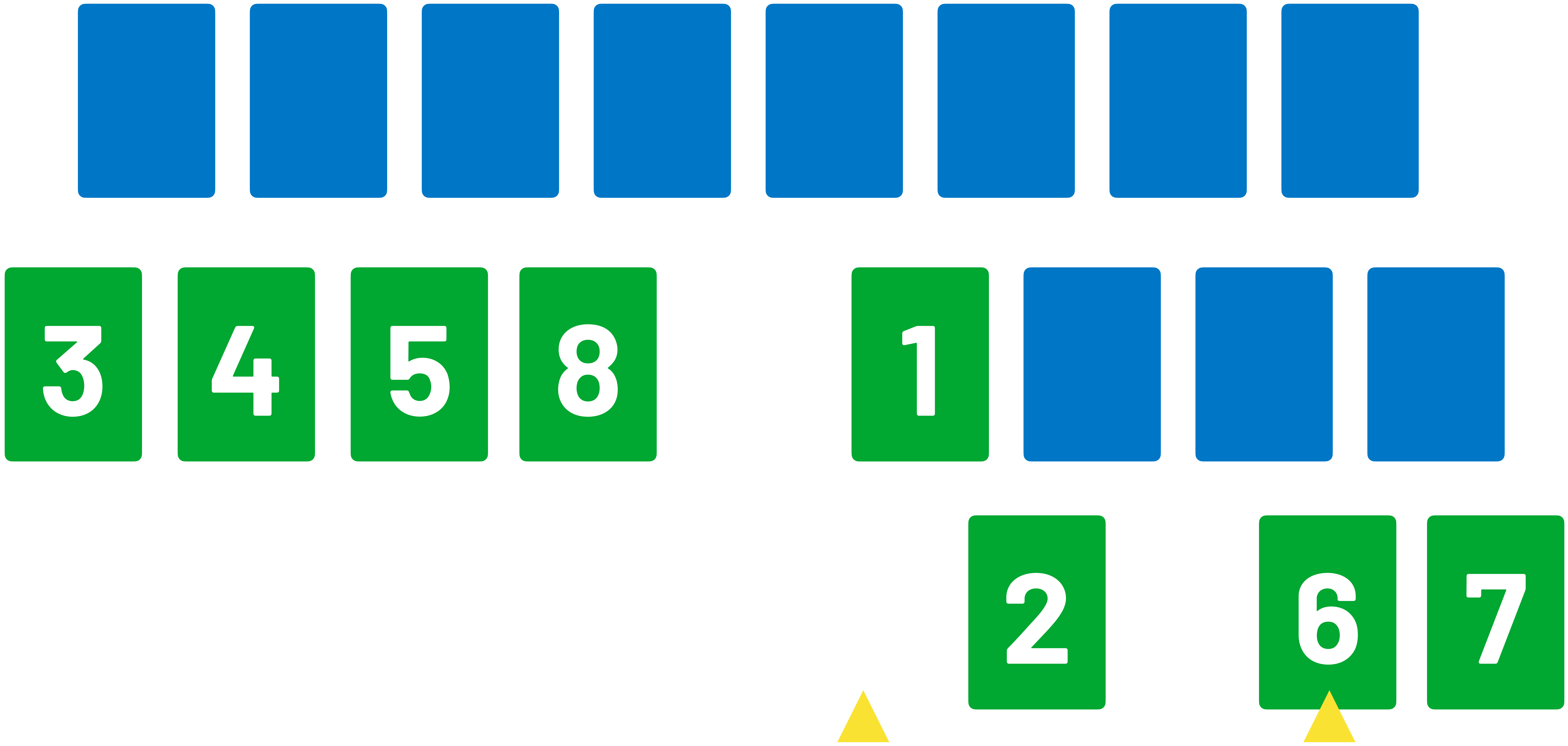

3
4
5
8
1
2
6
7

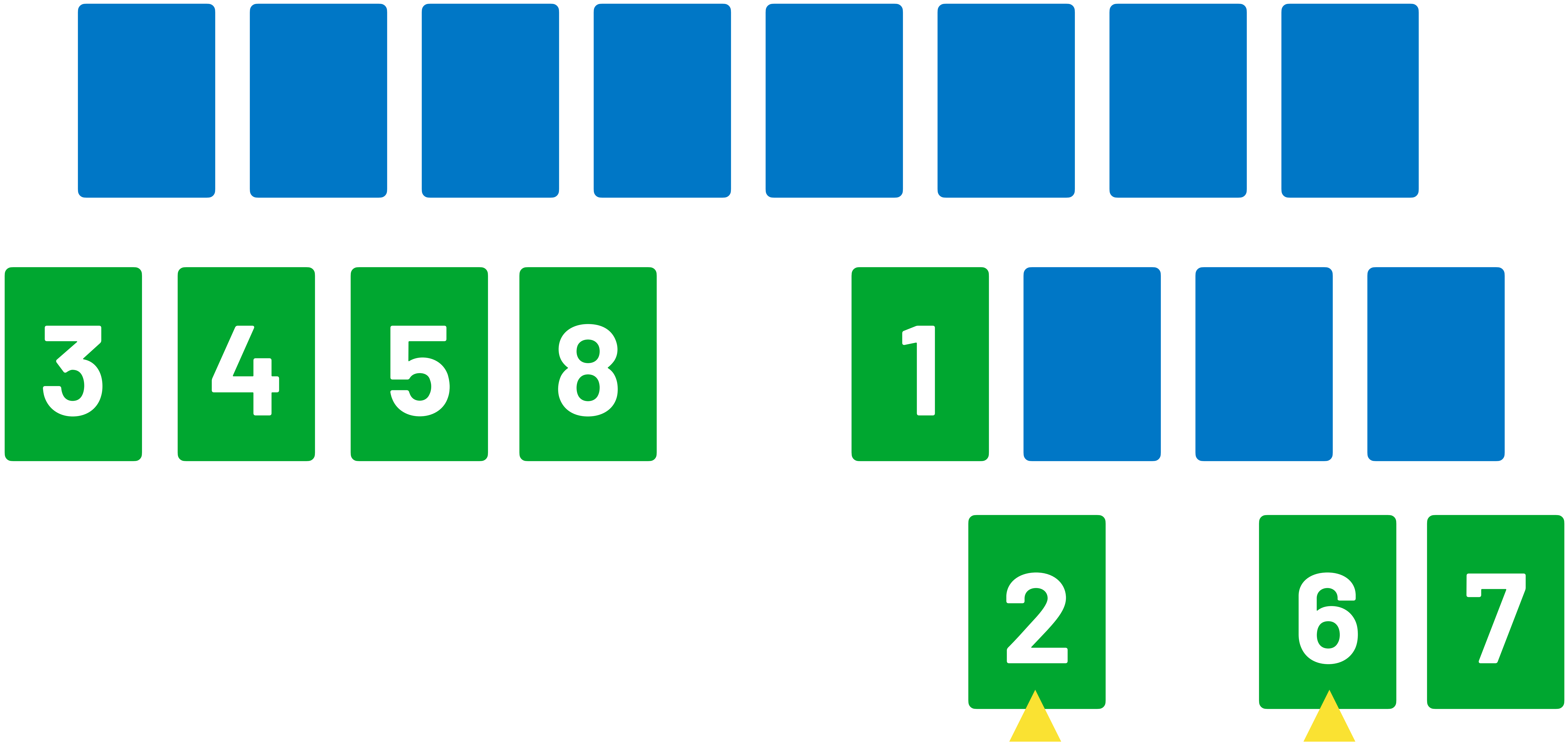


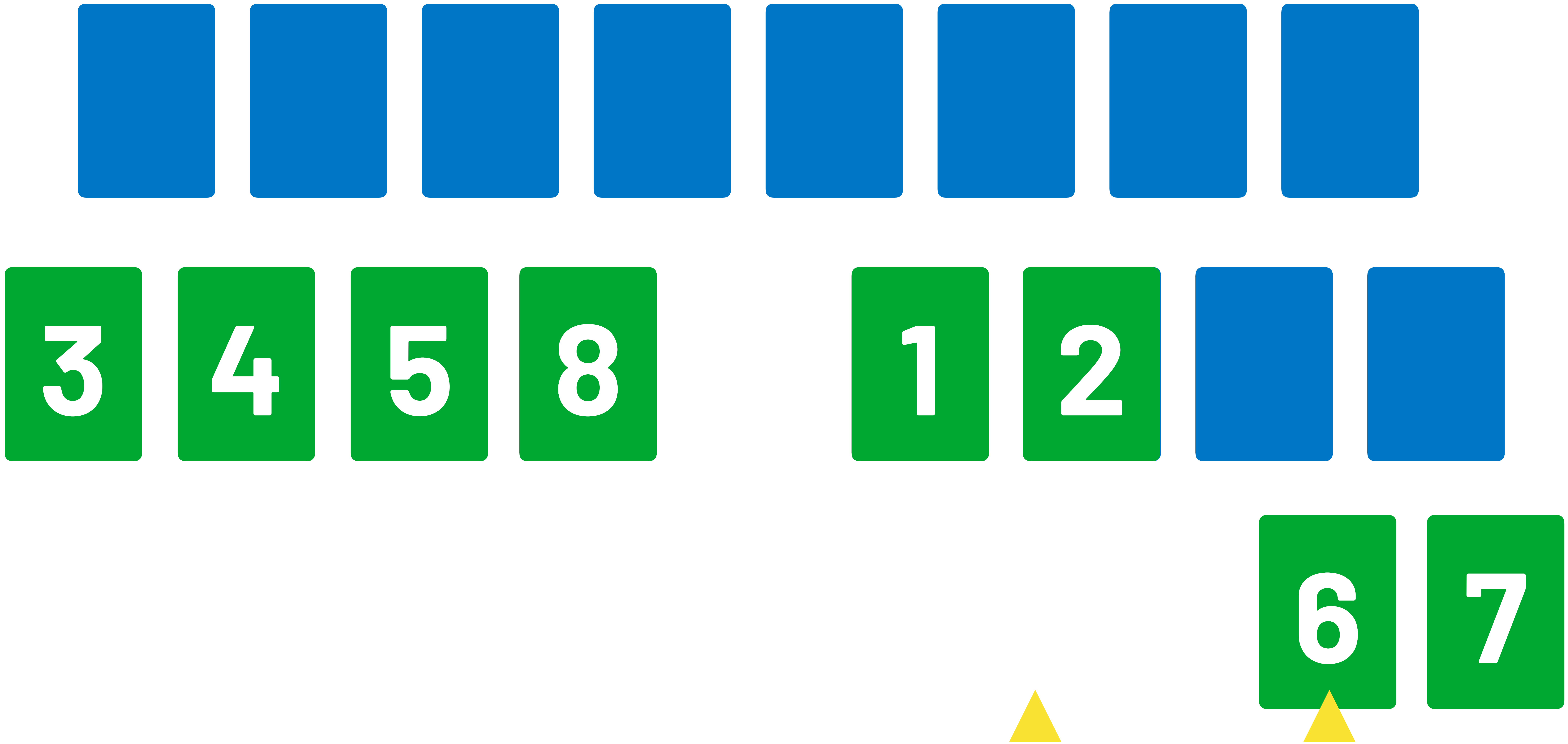
3
4
5
8
1
2
6
7

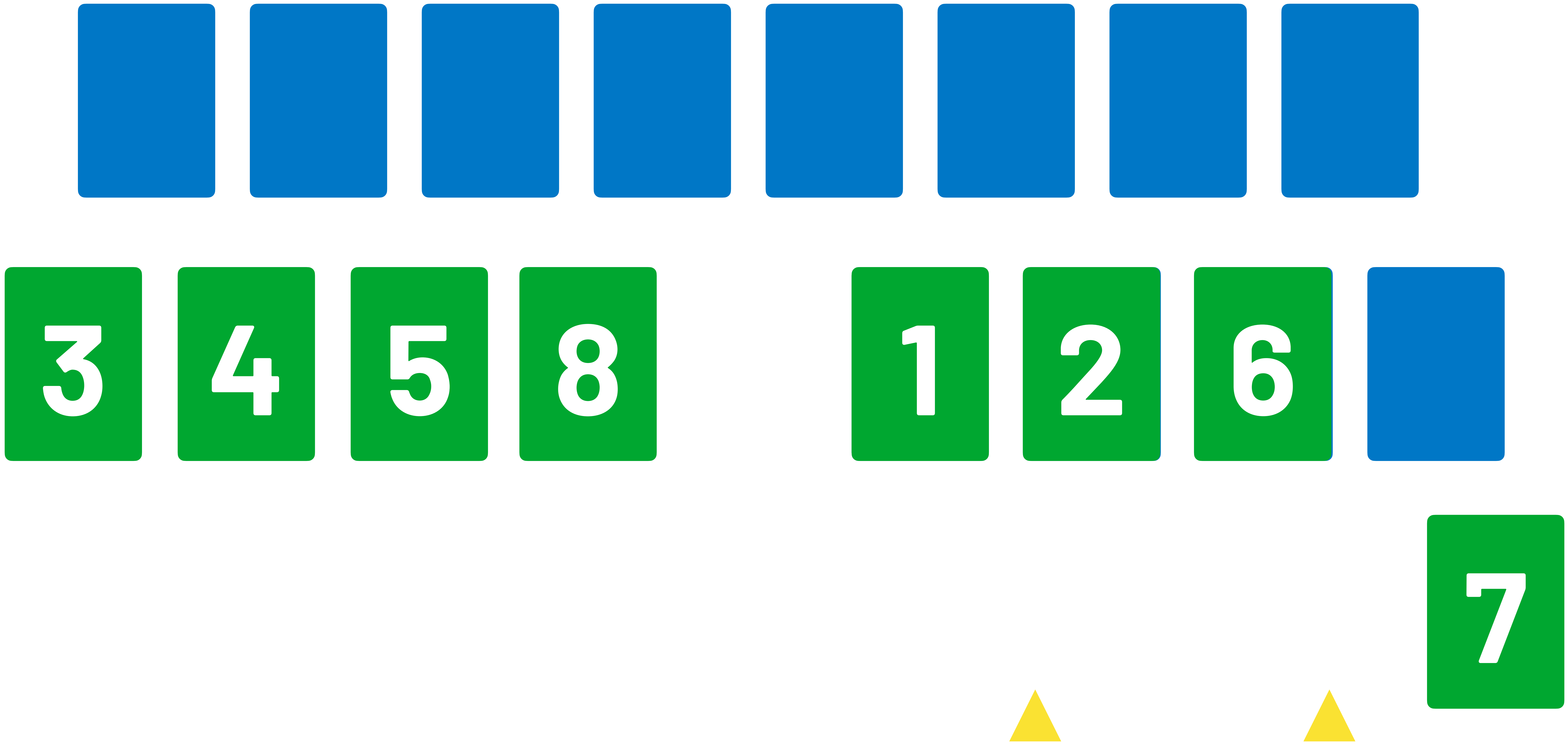

3
4
5
8
1
2
6
7

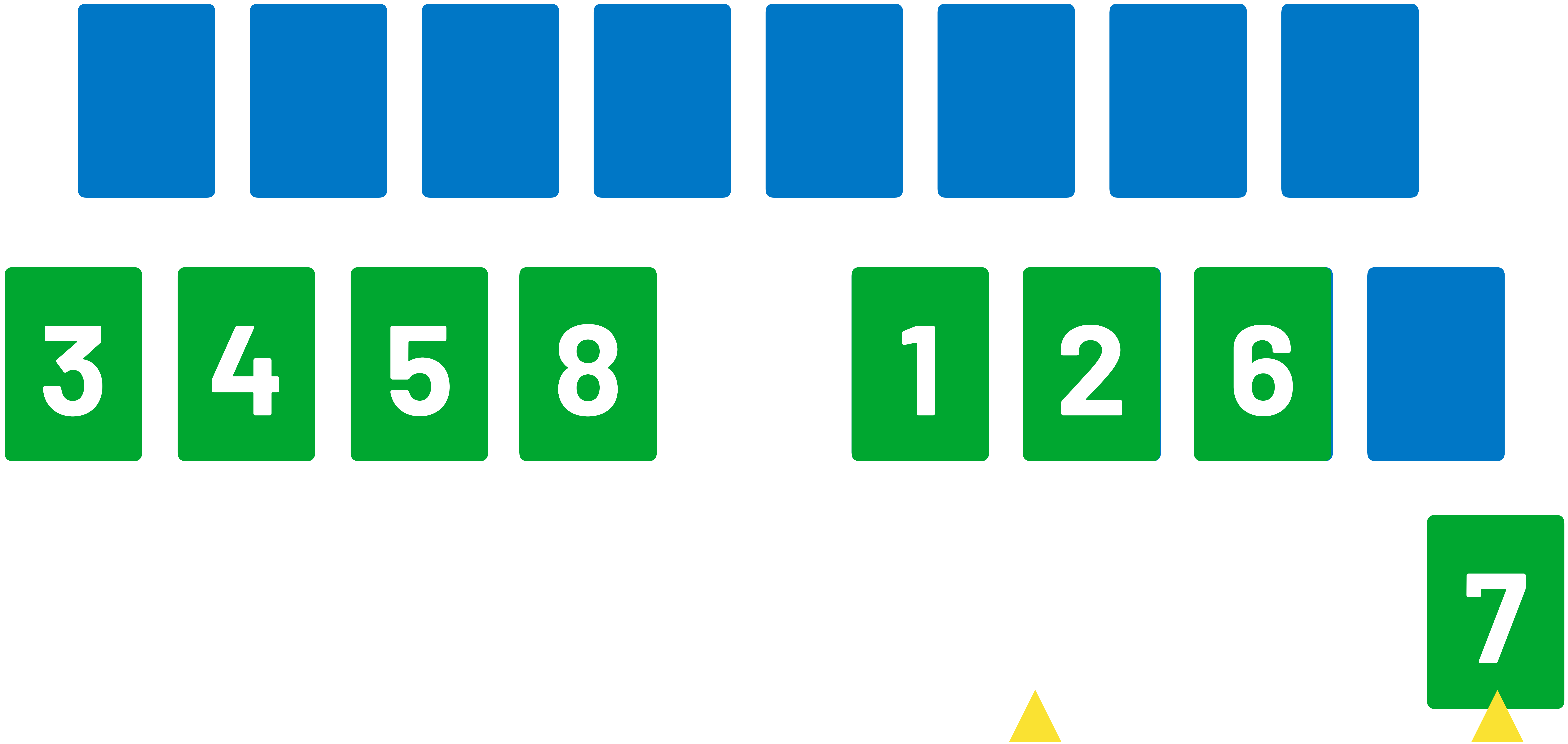

3
4
5
8
1
2
6
7

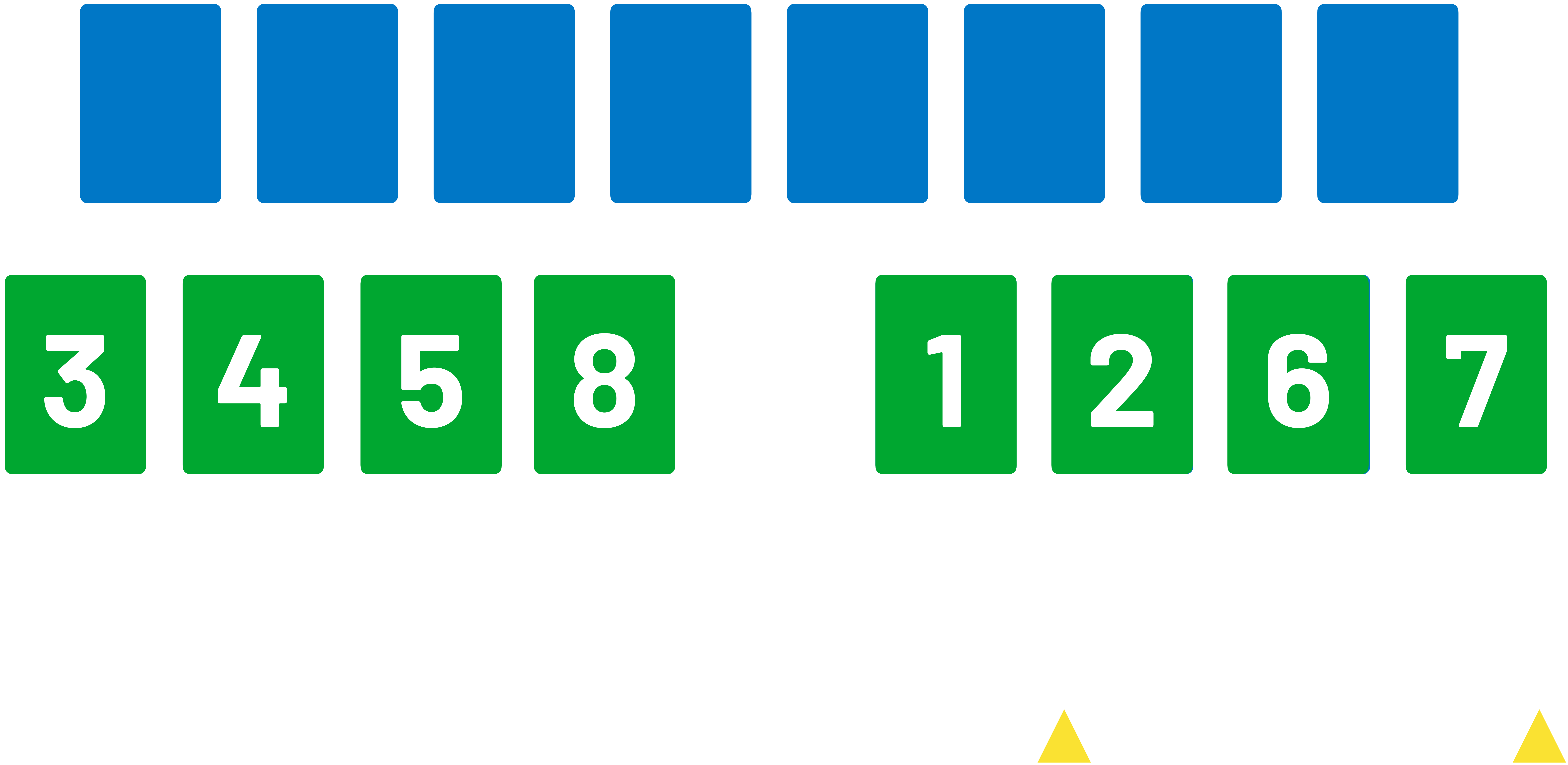

3
4
5
8
1
2
6
7

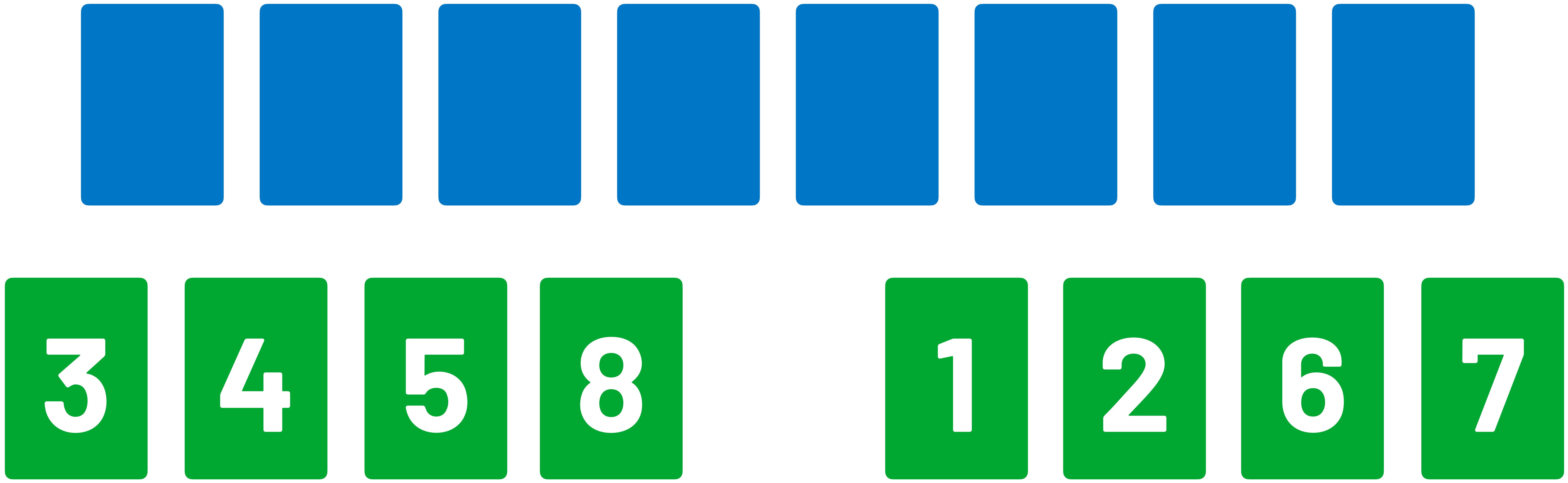

3
4
5
8
1
2
6
7

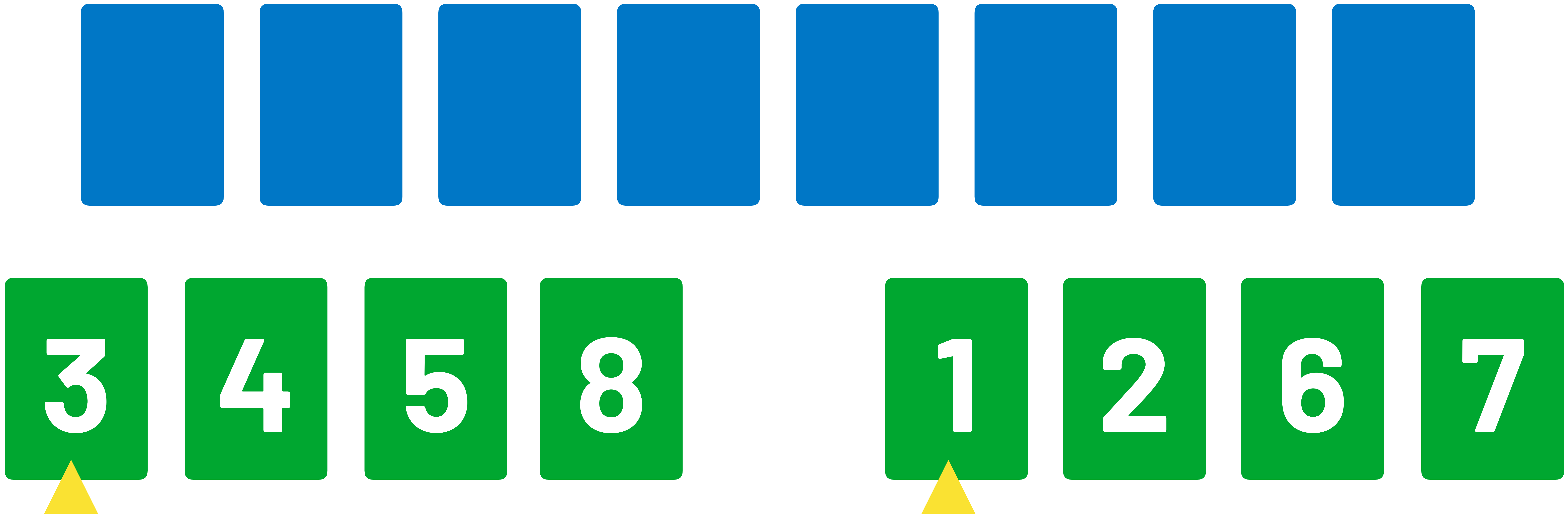

3
4
5
8
1
2
6
7


3 4 5 8
1 2 6
7


3
4
5
8
1
2
6
7


3
4
5
8
1
2
6
7

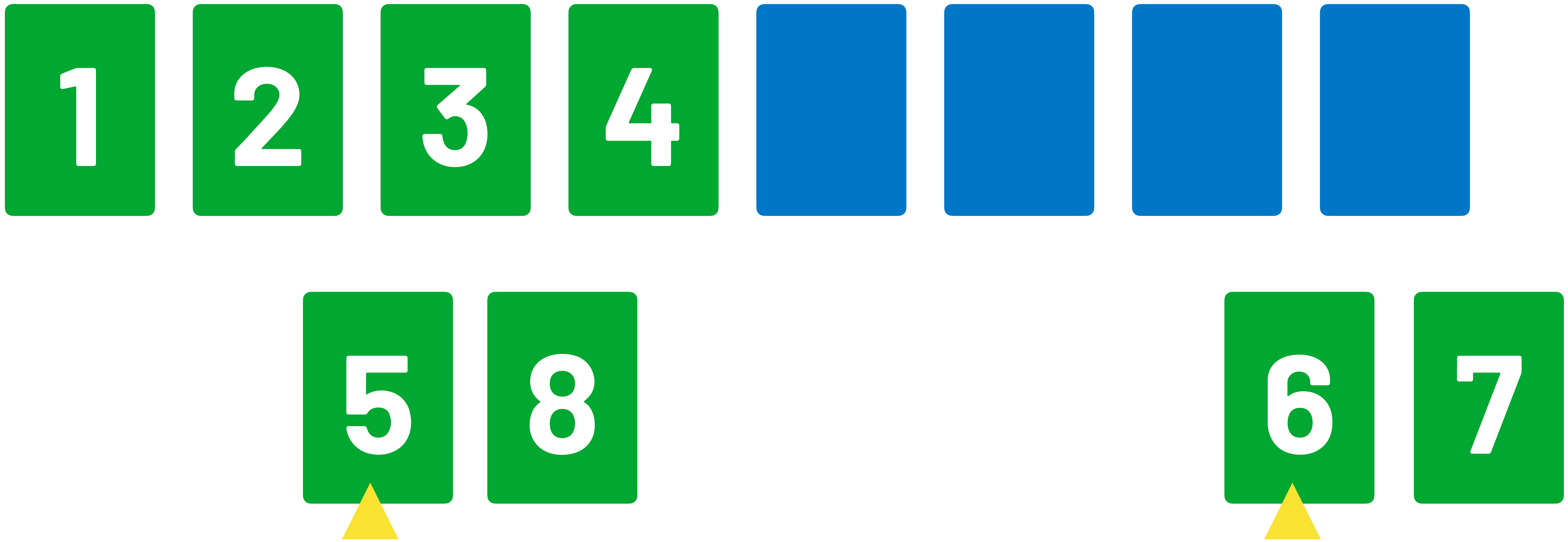

3
4
5
8
1
2
6
7

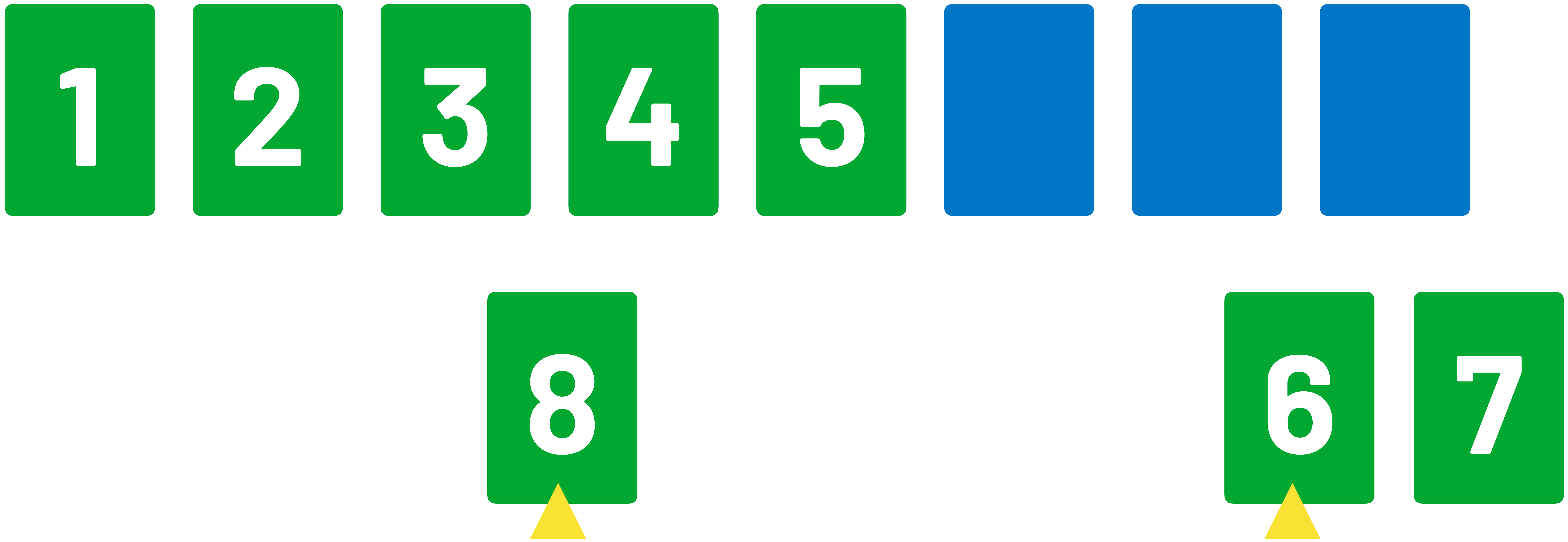

3
4
5
8
1
2
6
7

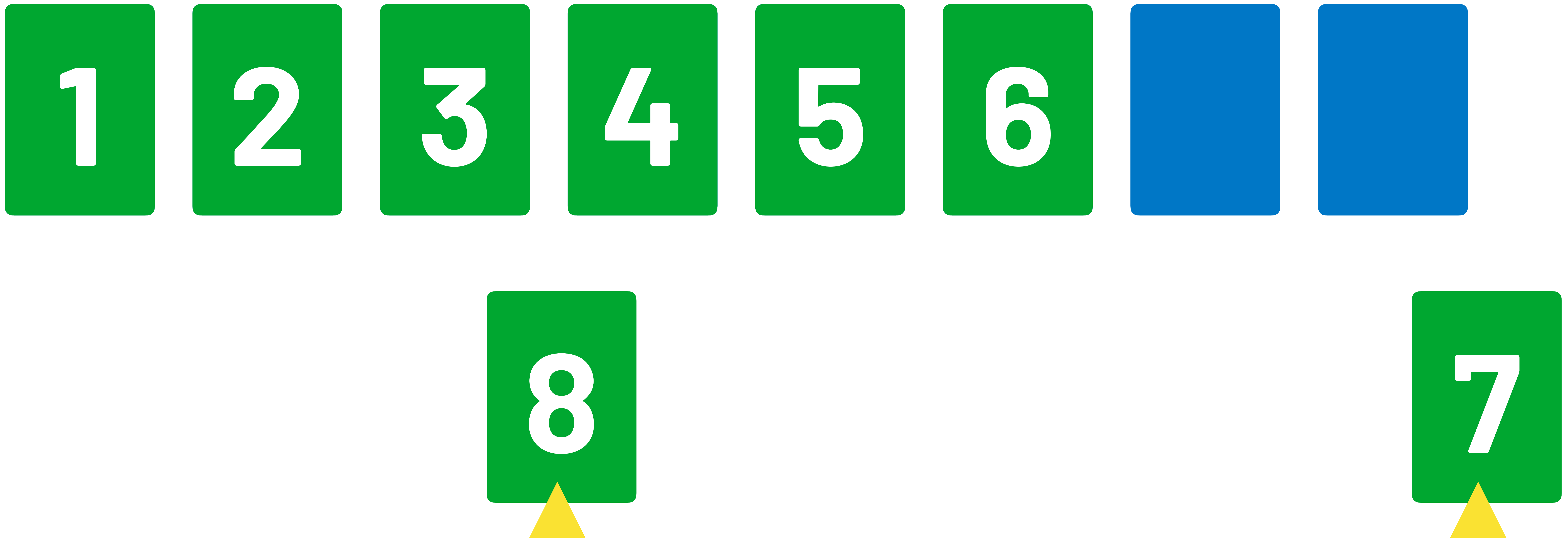

1
3
4
5
8
2
6
7

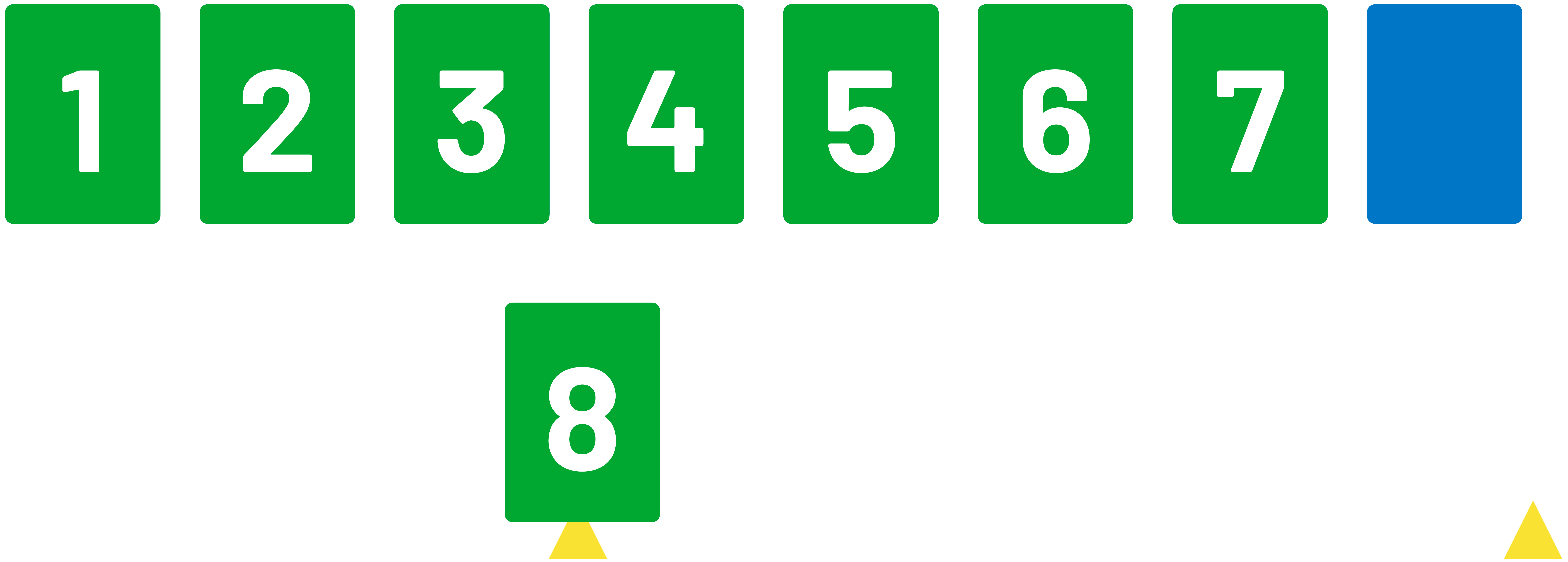

1
2
3
4
5
8
6
7

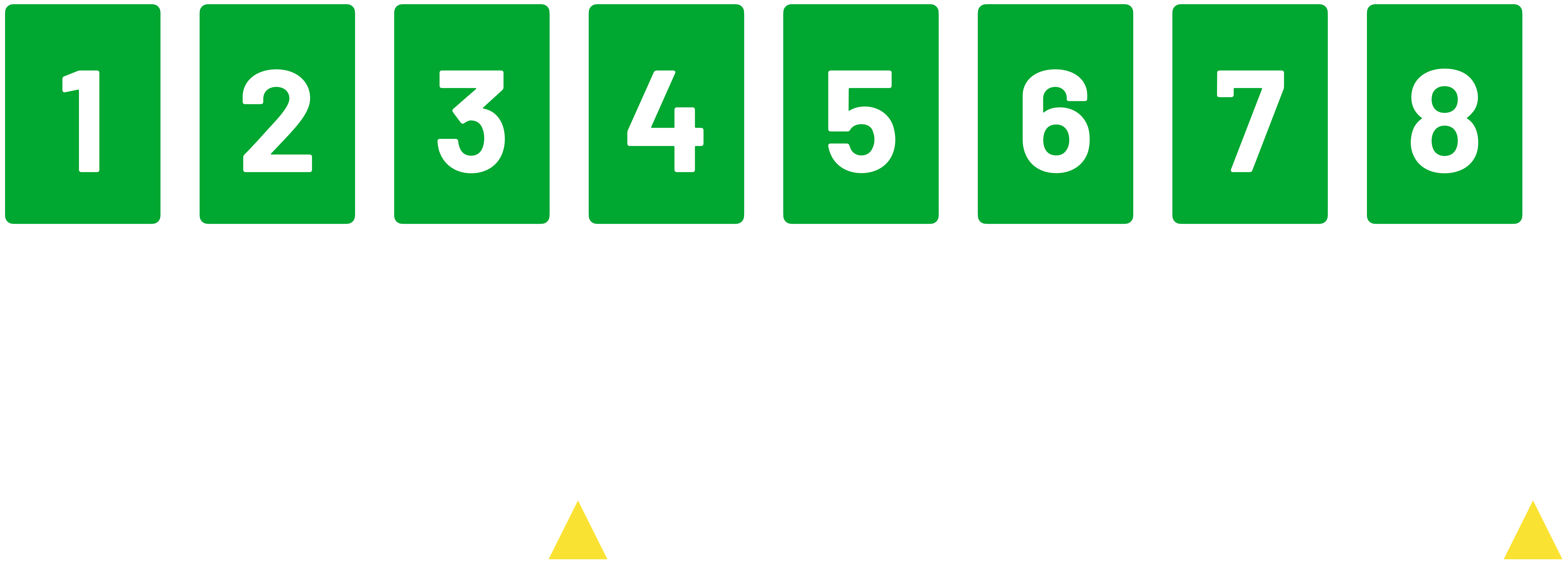

1
2
3
4
5
8
6
7


1
2
3
4
5
8
6
7


1
2
3
4
5
8
6
7


1
2
3
4
5
6
8
7


1 2 3 4 5 6 7
8


1
2
3
4
5
6
7
8

1 2 3 4 5 6 7 8

Mergesort

1 2 3 4 5 6 7 8

Algorithm	Upper Bound	Lower Bound
Linear Search	$O(n)$	$\Omega(1)$
Binary Search	$O(\log n)$	$\Omega(1)$
Selection Sort	$O(n^2)$	$\Omega(n^2)$
Bubble Sort	$O(n^2)$	$\Omega(n)$
Mergesort	$O(n \log n)$	$\Omega(n \log n)$

Practice Problem

Less is More

https://cs50.harvard.edu/college/2019/fall/test/less/

AAAAAAATTTTTAAAGGGGCCCCCCAAACCCC

- 7 As
- 5 Ts
- 3 As
- 4 Gs
- 6 Cs
- 3 As
- 4 Cs

A7T5A3G4C6A3C4

1. Consider the DNA sequence below.

```
TTTAAAACCGAAA
```

How could that sequence instead be encoded using run-length encoding?

```
T3A4C2G1A3
```

2. For what types of sequences would this algorithm likely increase the file size, rather than decrease it? Include in your answer an example of a DNA sequence for which the “compressed” version actually requires more characters than the original.

ATGC → A1T1G1C1

3. Imagine what might happen if we compressed BMPs of the flags of Romania and Germany, below, and compressed each using run-length encoding for each row of pixels. Assuming both flags have the same resolution (i.e., rows and columns of pixels), which image could be compressed more (i.e., take up less space once compressed)? Why?

Flag of Romania

Flag of Germany

A	01000001	0
C	01000011	1
G	01000111	10
T	01010100	11

4. Using this new encoding (A = 0, C = 1, G = 10, T = 11), how would you represent this sequence in binary?

```
CCCAGTTA
```

```
11101011110
```

5. Using this new encoding (A = 0, C = 1, G = 10, T = 11), how would you represent this sequence in binary?

```
TGCATCCA
```

```
11101011110
```

6. What problem does this encoding thus have? Propose another encoding for DNA nucleotides that fixes that problem, while still using fewer bits than storing nucleotides as ASCII characters.

A	00
C	01
G	10
T	11

CS50 Test Review

Algorithms and Programming Languages