Bubble Sort

Bubble Sort

- In bubble sort, the idea of the algorithm is to move higher valued elements generally towards the right and lower value elements generally towards the left.

<u>In pseudocode:</u>

```
• Set swap counter to a non-zero value
• Repeat until the swap counter is 0:
    • Reset swap counter to 0
    • Look at each adjacent pair
        • If two adjacent elements are not in order, swap them and add one to the swap
          counter
```

Bubble Sort

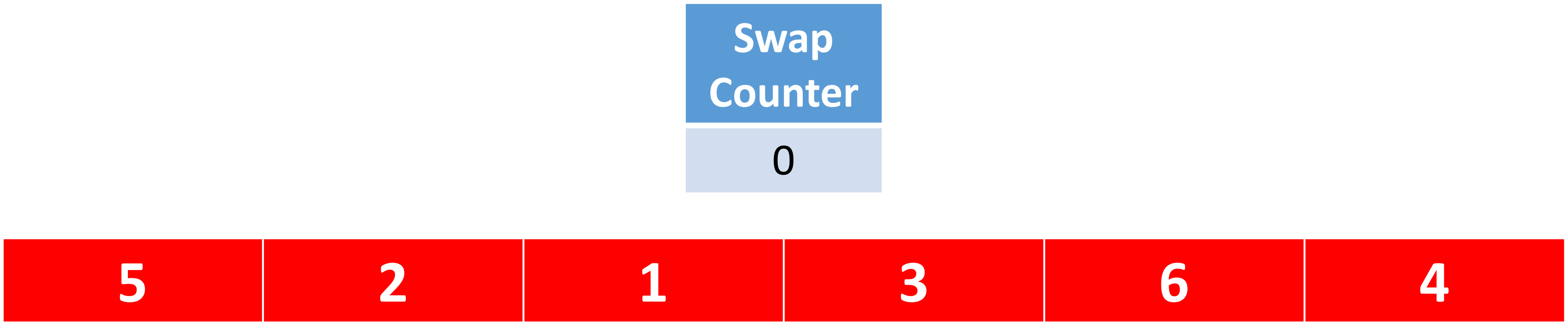
Swap Counter
-1

5	2	1	3	6	4

In pseudocode:

```
Set swap counter to a non-zero value
Repeat until the swap counter is 0:
    Reset swap counter to 0
    Look at each adjacent pair
        If two adjacent elements are not in order, swap them and add one to the swap counter
```

Bubble Sort



Swap Counter
0

5	2	1	3	6	4

In pseudocode:

```
Set swap counter to a non-zero value
Repeat until the swap counter is 0:
    Reset swap counter to 0
    Look at each adjacent pair
        If two adjacent elements are not in order, swap them and add one to the swap counter
```

Bubble Sort

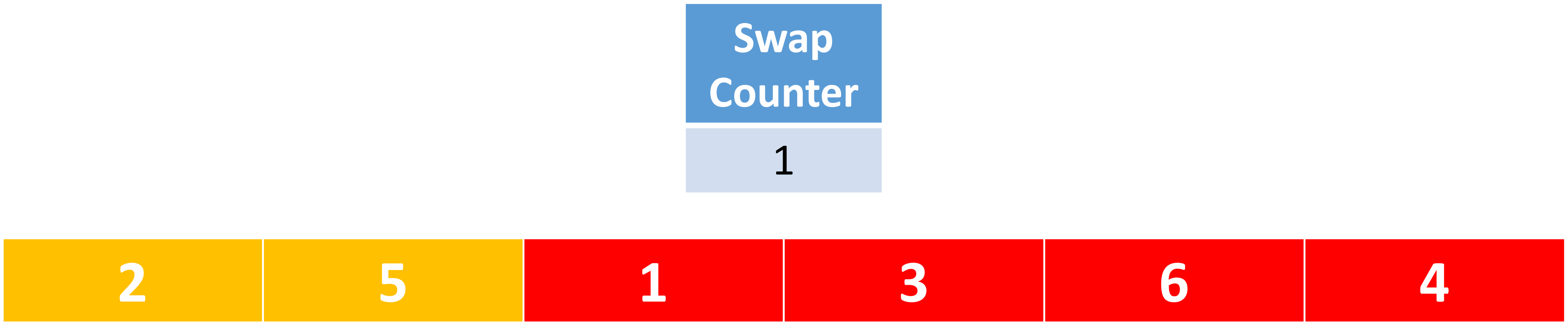
Swap Counter
0

5	2	1	3	6	4

In pseudocode:

```
Set swap counter to a non-zero value
Repeat until the swap counter is 0:
    Reset swap counter to 0
    Look at each adjacent pair
        If two adjacent elements are not in order, swap them and add one to the swap counter
```

Bubble Sort



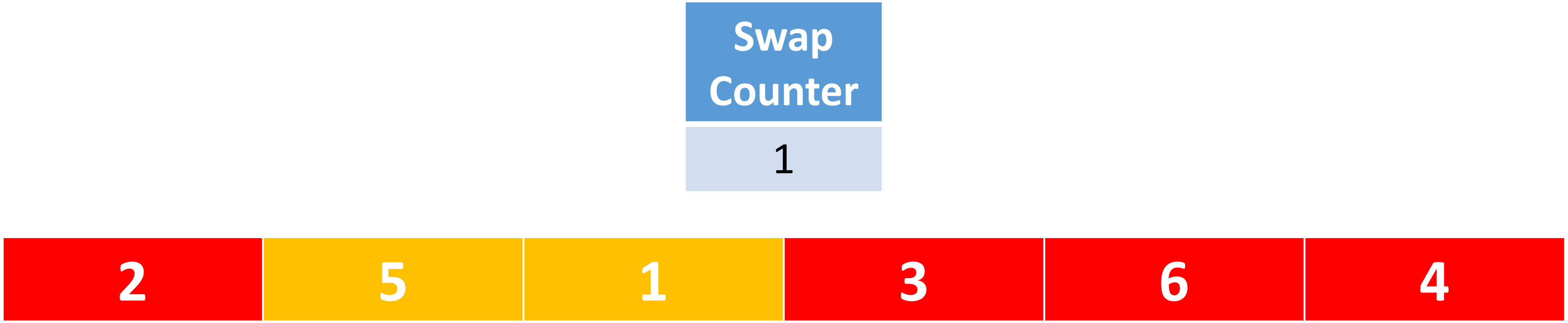
Swap Counter
1

2	5	1	3	6	4

<u>In pseudocode:</u>

```
Set swap counter to a non-zero value
Repeat until the swap counter is 0:
    Reset swap counter to 0
    Look at each adjacent pair
        If two adjacent elements are not in order, swap them and add one to the swap counter
```

Bubble Sort



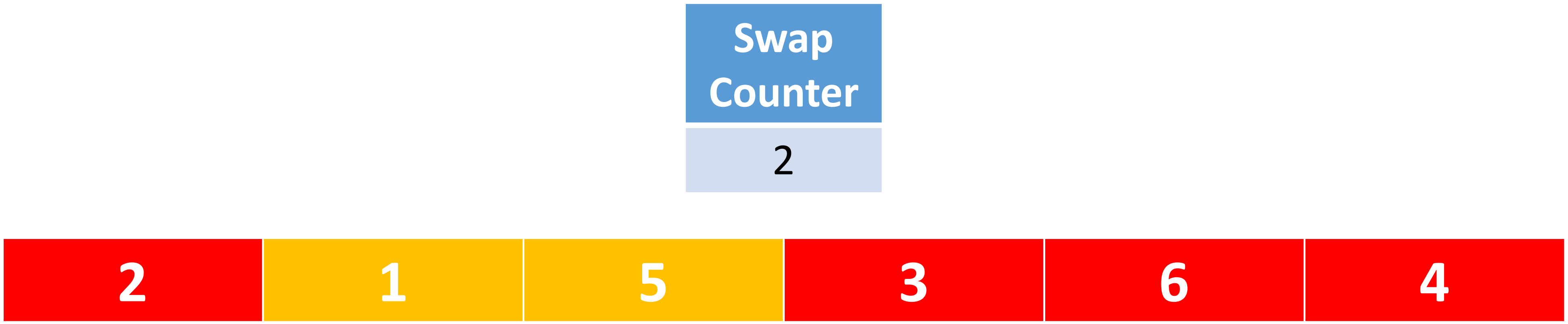
Swap Counter
1

2	5	1	3	6	4

In pseudocode:

```
Set swap counter to a non-zero value
Repeat until the swap counter is 0:
    Reset swap counter to 0
    Look at each adjacent pair
        If two adjacent elements are not in order, swap them and add one to the swap counter
```

Bubble Sort



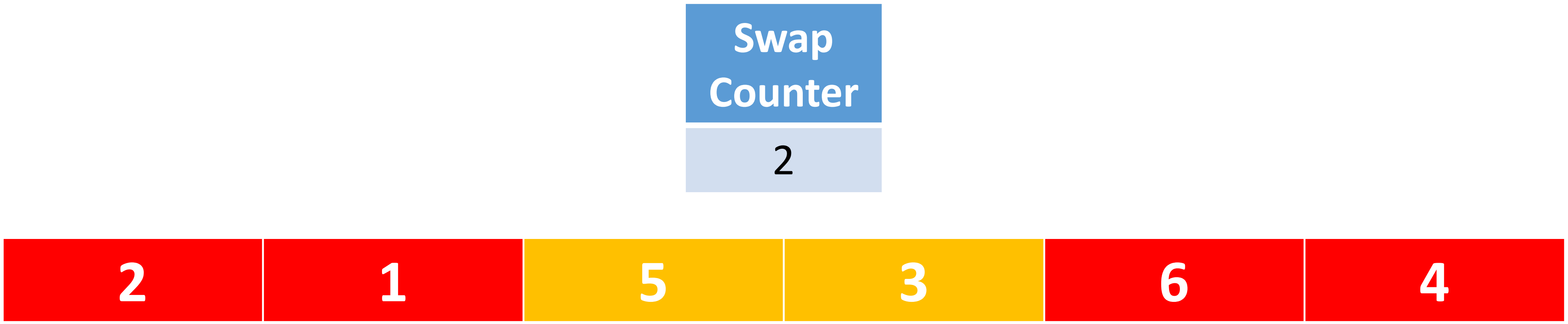
Swap Counter
2

🟥 2	🟨 1	🟨 5	🟥 3	🟥 6	🟥 4

<u>In pseudocode:</u>

```
Set swap counter to a non-zero value
Repeat until the swap counter is 0:
    Reset swap counter to 0
    Look at each adjacent pair
        If two adjacent elements are not in order, swap them and add one to the swap counter
```

Bubble Sort



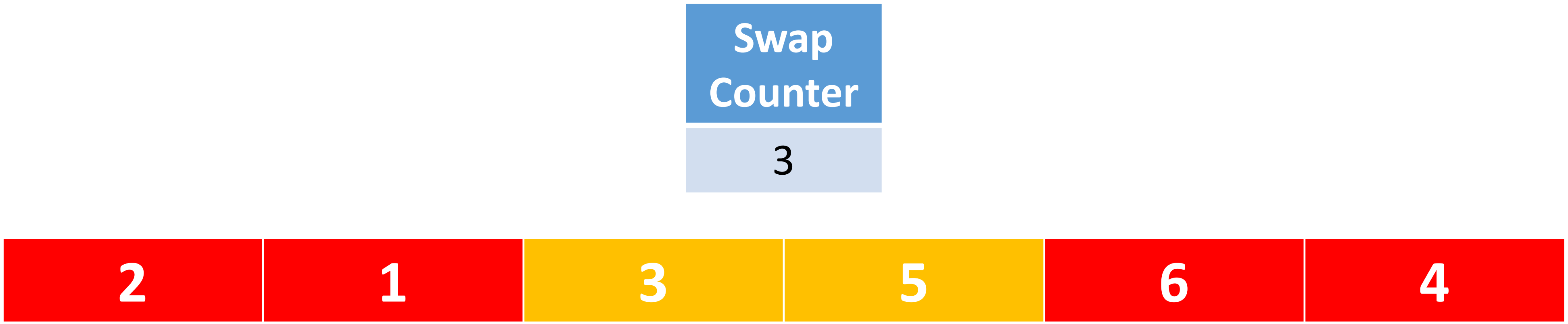
Swap Counter
2

2	1	5	3	6	4

In pseudocode:

```
Set swap counter to a non-zero value
Repeat until the swap counter is 0:
    Reset swap counter to 0
    Look at each adjacent pair
        If two adjacent elements are not in order, swap them and add one to the swap counter
```

Bubble Sort



Swap Counter

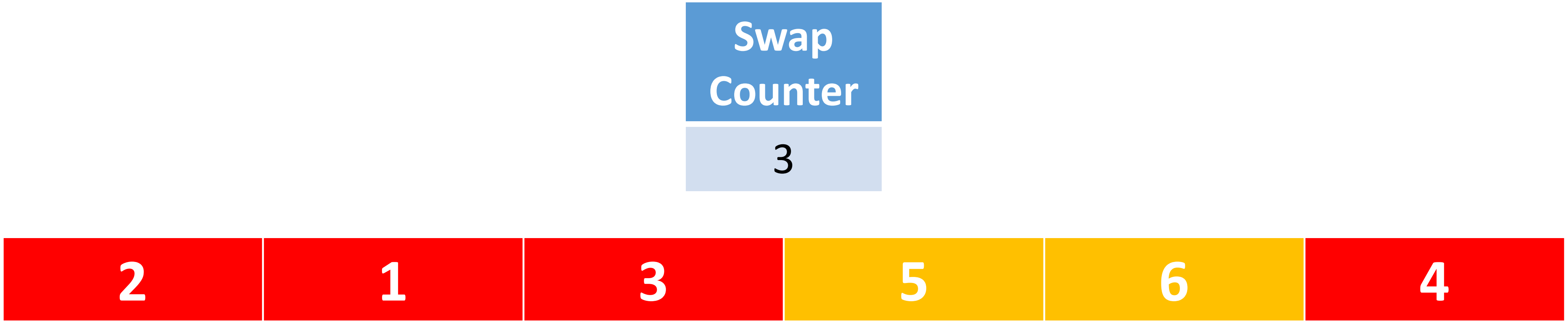
3

2	1	3	5	6	4

In pseudocode:

```
Set swap counter to a non-zero value
Repeat until the swap counter is 0:
    Reset swap counter to 0
    Look at each adjacent pair
        If two adjacent elements are not in order, swap them and add one to the swap counter
```

Bubble Sort



Swap Counter

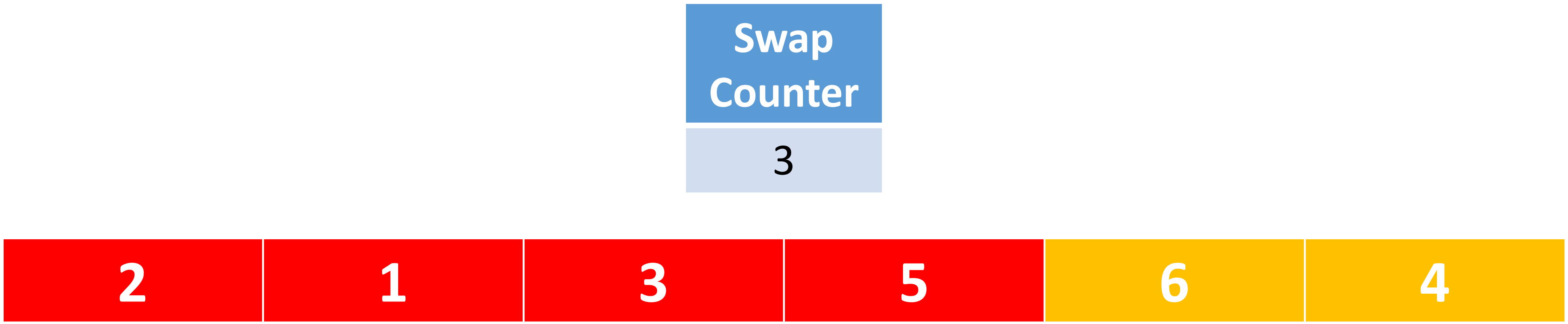
3

2	1	3	5	6	4

In pseudocode:

```
Set swap counter to a non-zero value
Repeat until the swap counter is 0:
    Reset swap counter to 0
    Look at each adjacent pair
        If two adjacent elements are not in order, swap them and add one to the swap counter
```

Bubble Sort



Swap Counter

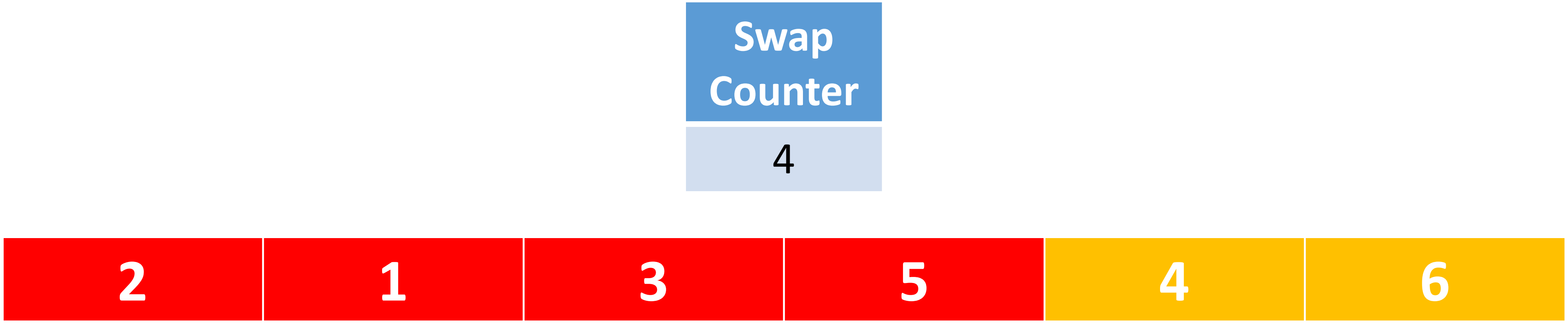
3

2	1	3	5	6	4

<u>In pseudocode:</u>

```
Set swap counter to a non-zero value
Repeat until the swap counter is 0:
    Reset swap counter to 0
    Look at each adjacent pair
        If two adjacent elements are not in order, swap them and add one to the swap counter
```

Bubble Sort



Swap Counter

4

2	1	3	5	4	6

In pseudocode:

```
Set swap counter to a non-zero value
Repeat until the swap counter is 0:
    Reset swap counter to 0
    Look at each adjacent pair
        If two adjacent elements are not in order, swap them and add one to the swap counter
```

Bubble Sort

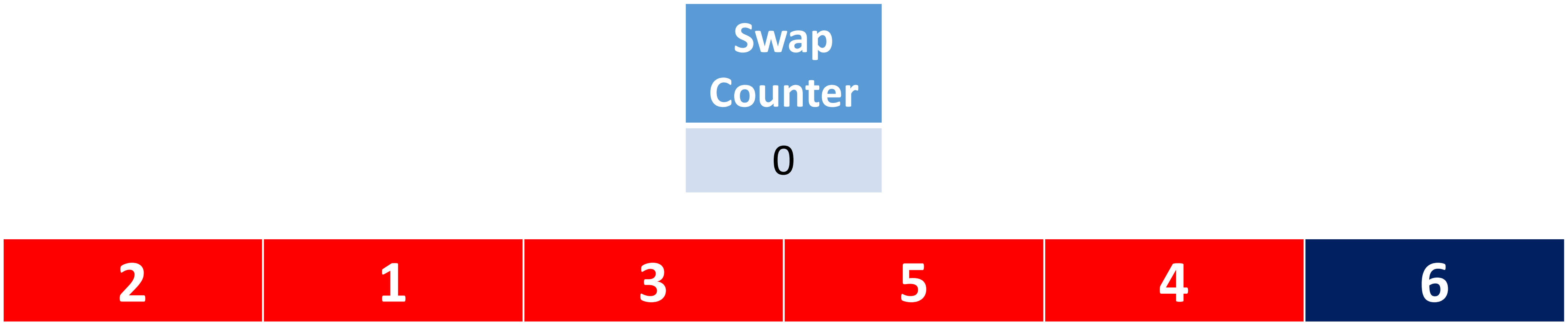
Swap Counter
4

2	1	3	5	4	6

In pseudocode:

```
Set swap counter to a non-zero value
Repeat until the swap counter is 0:
    Reset swap counter to 0
    Look at each adjacent pair
        If two adjacent elements are not in order, swap them and add one to the swap counter
```

Bubble Sort



Swap Counter

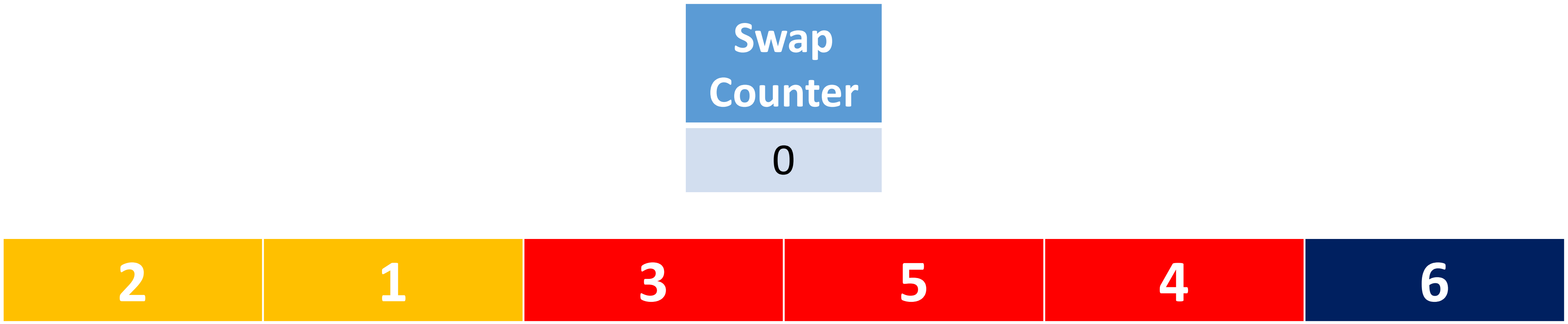
0

2 1 3 5 4 6

<u>In pseudocode:</u>

```
Set swap counter to a non-zero value
Repeat until the swap counter is 0:
    Reset swap counter to 0
    Look at each adjacent pair
        If two adjacent elements are not in order, swap them and add one to the swap counter
```

Bubble Sort



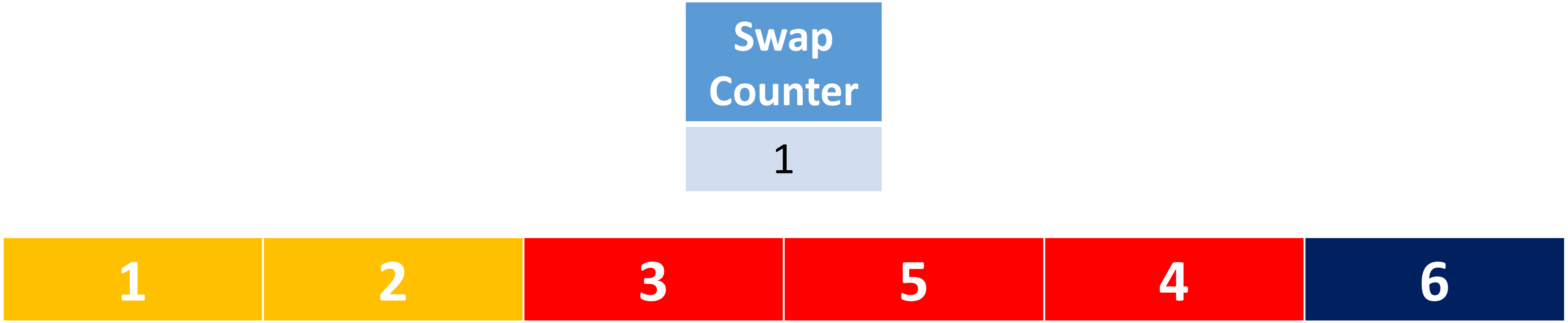
Swap Counter
0

2	1	3	5	4	6

<u>In pseudocode:</u>

```
Set swap counter to a non-zero value
Repeat until the swap counter is 0:
    Reset swap counter to 0
    Look at each adjacent pair
        If two adjacent elements are not in order, swap them and add one to the swap counter
```

Bubble Sort



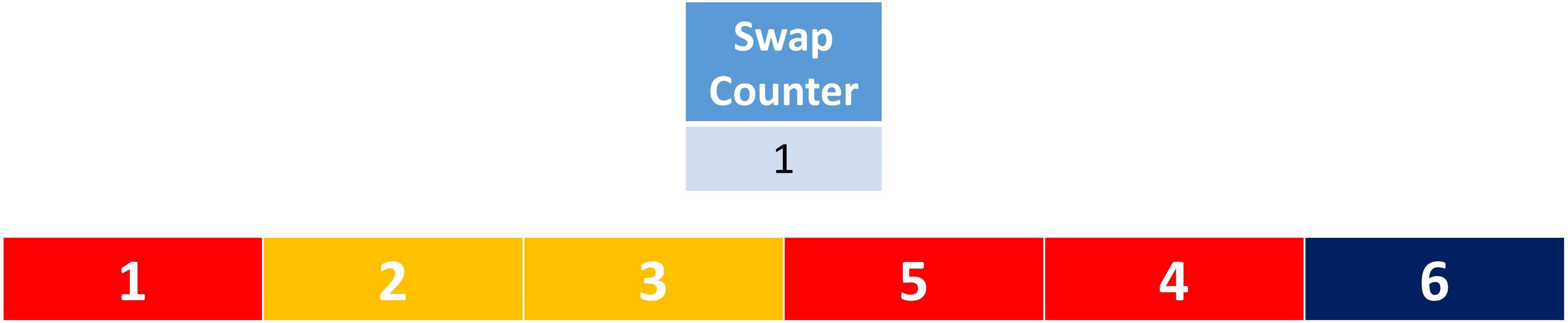
Swap Counter
1

1	2	3	5	4	6

<u>In pseudocode:</u>

```
Set swap counter to a non-zero value
Repeat until the swap counter is 0:
    Reset swap counter to 0
    Look at each adjacent pair
        If two adjacent elements are not in order, swap them and add one to the swap counter
```

Bubble Sort



Swap Counter

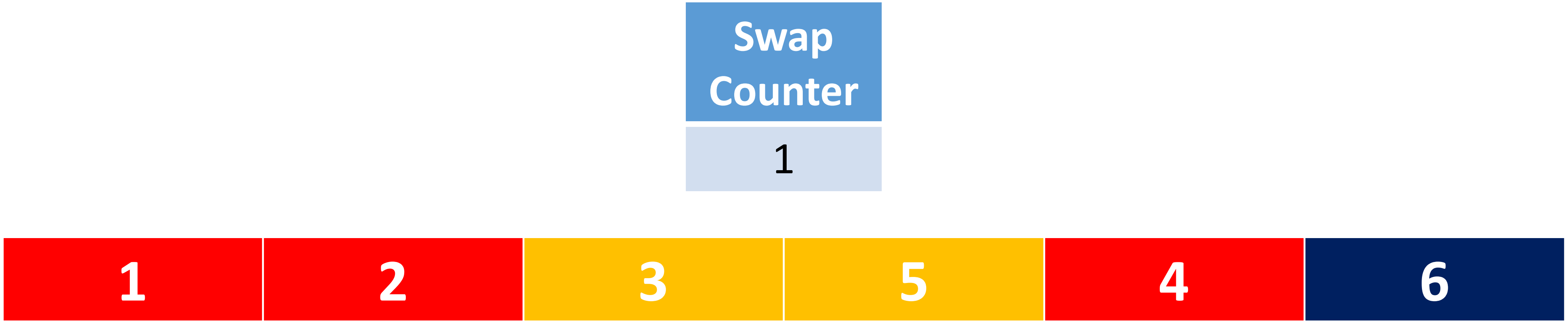
1

1	2	3	5	4	6

<u>In pseudocode:</u>

```
Set swap counter to a non-zero value
Repeat until the swap counter is 0:
    Reset swap counter to 0
    Look at each adjacent pair
        If two adjacent elements are not in order, swap them and add one to the swap counter
```

Bubble Sort



Swap Counter

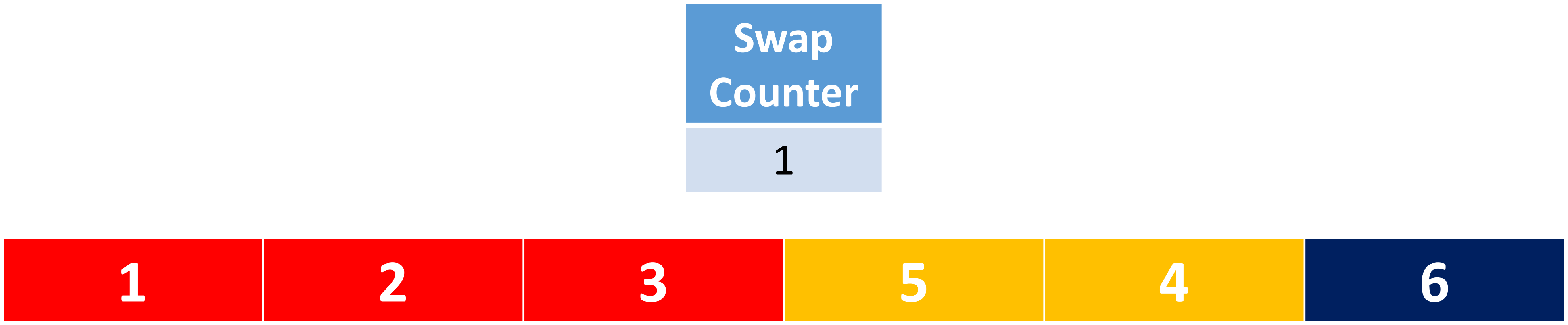
1

1	2	3	5	4	6

In pseudocode:

```
Set swap counter to a non-zero value
Repeat until the swap counter is 0:
    Reset swap counter to 0
    Look at each adjacent pair
        If two adjacent elements are not in order, swap them and add one to the swap counter
```

Bubble Sort



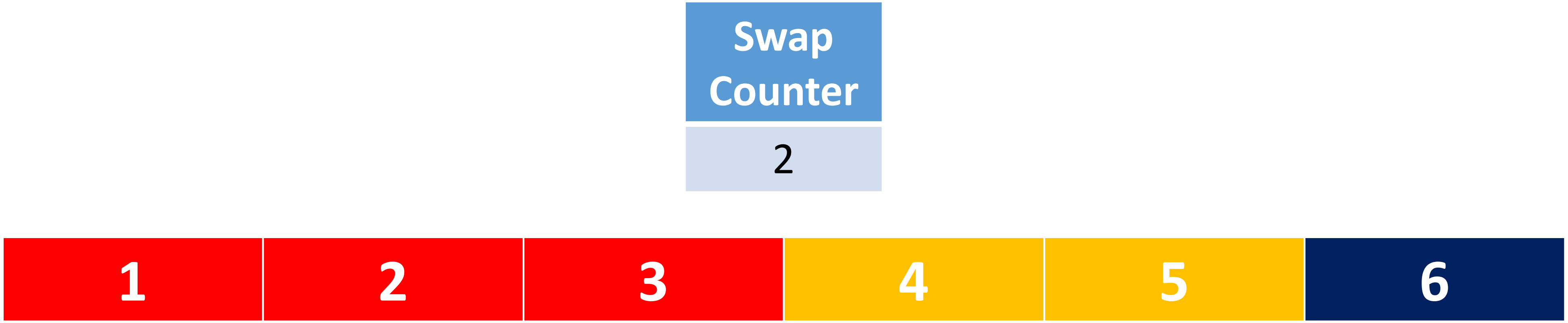
Swap Counter
1

1	2	3	5	4	6

<u>In pseudocode:</u>

```
Set swap counter to a non-zero value
Repeat until the swap counter is 0:
    Reset swap counter to 0
    Look at each adjacent pair
        If two adjacent elements are not in order, swap them and add one to the swap counter
```

Bubble Sort



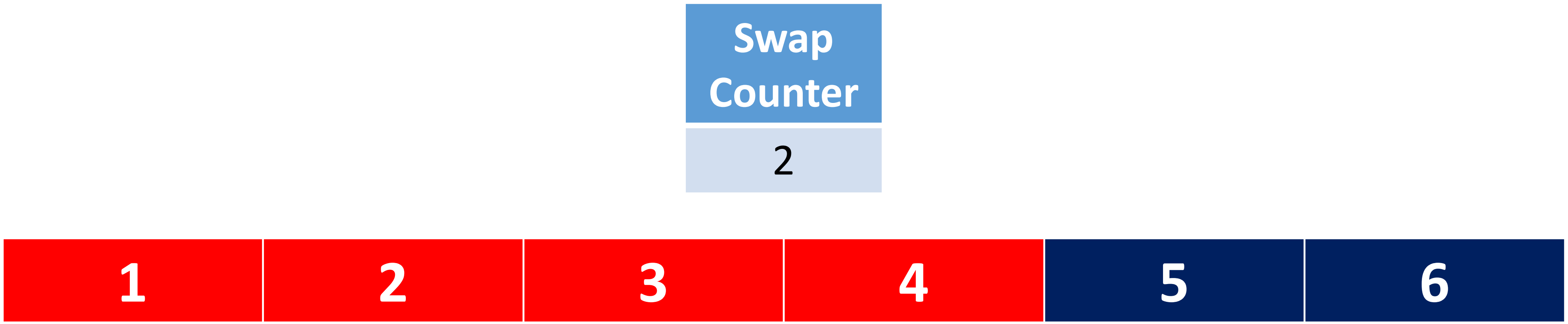
Swap Counter
2

1	2	3	4	5	6

In pseudocode:

```
Set swap counter to a non-zero value
Repeat until the swap counter is 0:
    Reset swap counter to 0
    Look at each adjacent pair
        If two adjacent elements are not in order, swap them and add one to the swap counter
```

Bubble Sort



Swap Counter

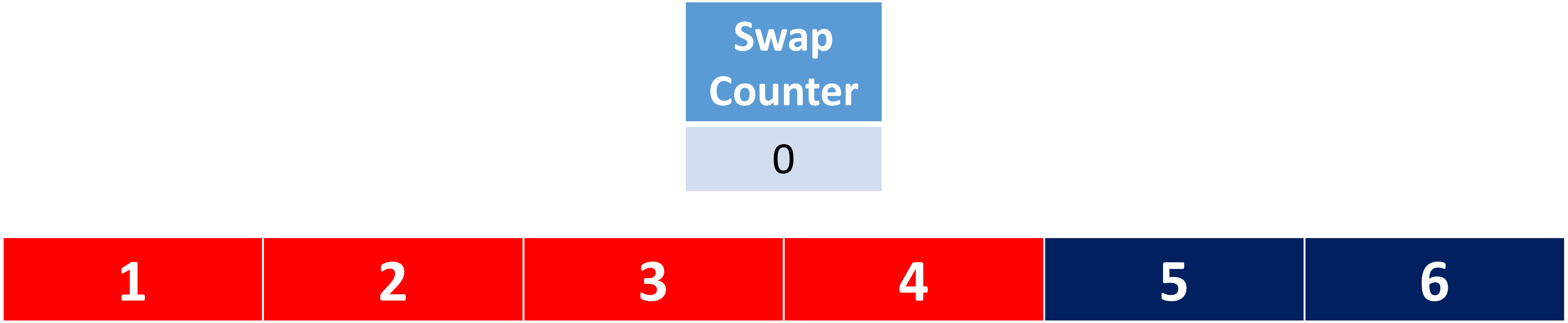
2

1	2	3	4	5	6

<u>In pseudocode:</u>

```
Set swap counter to a non-zero value
Repeat until the swap counter is 0:
    Reset swap counter to 0
    Look at each adjacent pair
        If two adjacent elements are not in order, swap them and add one to the swap counter
```

Bubble Sort



Swap Counter

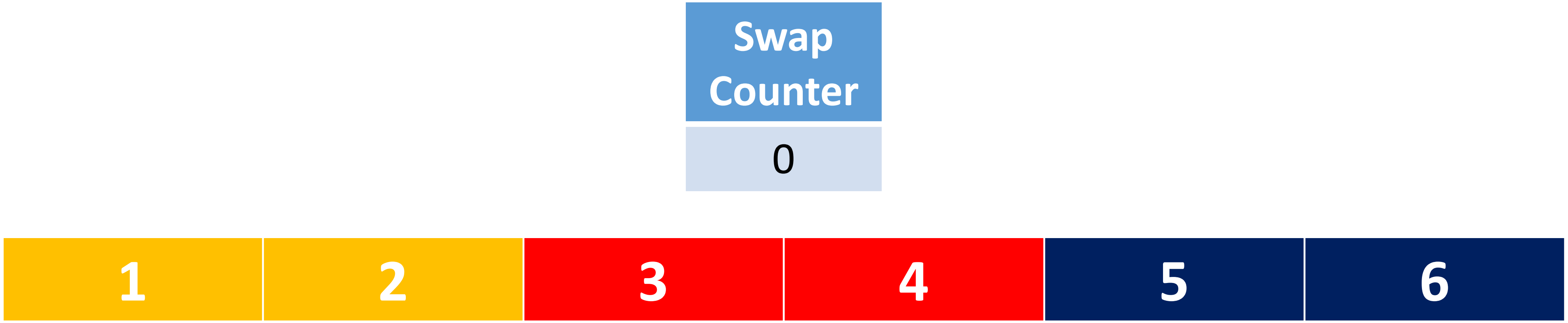
0

1	2	3	4	5	6

<u>In pseudocode:</u>

```
Set swap counter to a non-zero value
Repeat until the swap counter is 0:
    Reset swap counter to 0
    Look at each adjacent pair
        If two adjacent elements are not in order, swap them and add one to the swap counter
```

Bubble Sort



Swap Counter

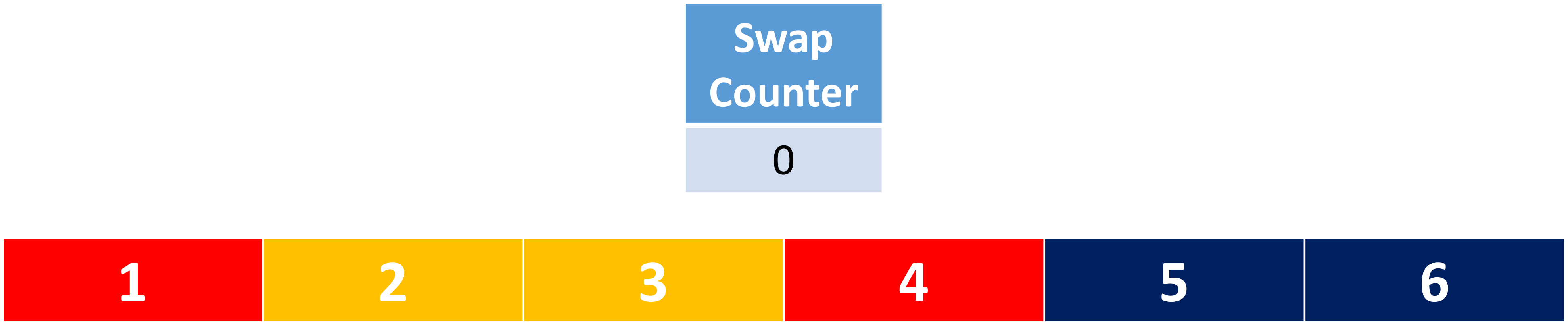
0

1	2	3	4	5	6

In pseudocode:

```
Set swap counter to a non-zero value
Repeat until the swap counter is 0:
    Reset swap counter to 0
    Look at each adjacent pair
        If two adjacent elements are not in order, swap them and add one to the swap counter
```

Bubble Sort



Swap Counter

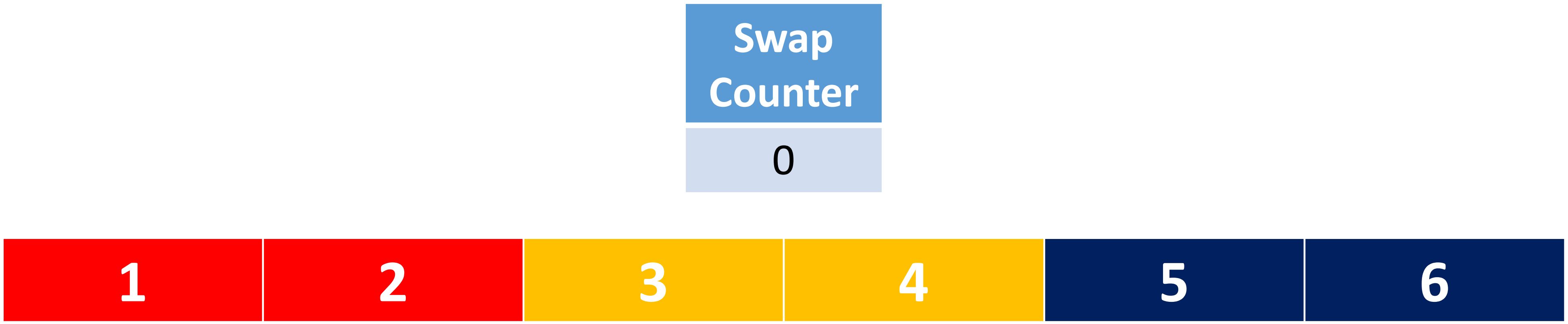
0

1	2	3	4	5	6

<u>In pseudocode:</u>

```
Set swap counter to a non-zero value
Repeat until the swap counter is 0:
    Reset swap counter to 0
    Look at each adjacent pair
        If two adjacent elements are not in order, swap them and add one to the swap counter
```

Bubble Sort



Swap Counter
0

1	2	3	4	5	6

In pseudocode:

```
Set swap counter to a non-zero value
Repeat until the swap counter is 0:
    Reset swap counter to 0
    Look at each adjacent pair
        If two adjacent elements are not in order, swap them and add one to the swap counter
```

Bubble Sort

Swap Counter
0

1	2	3	4	5	6

In pseudocode:

```
Set swap counter to a non-zero value
Repeat until the swap counter is 0:
    Reset swap counter to 0
    Look at each adjacent pair
        If two adjacent elements are not in order, swap them and add one to the swap counter
```

Bubble Sort

- **Worst-case scenario**: The array is in reverse order; we have to “bubble” each of the *n* elements all the way across the array, and since we can only fully bubble one element into position per pass, we must do this *n* times.

- **Best-case scenario**: The array is already perfectly sorted, and we make no swaps on the first pass.

Bubble Sort

$$O(n^2)$$

$$\Omega(n)$$