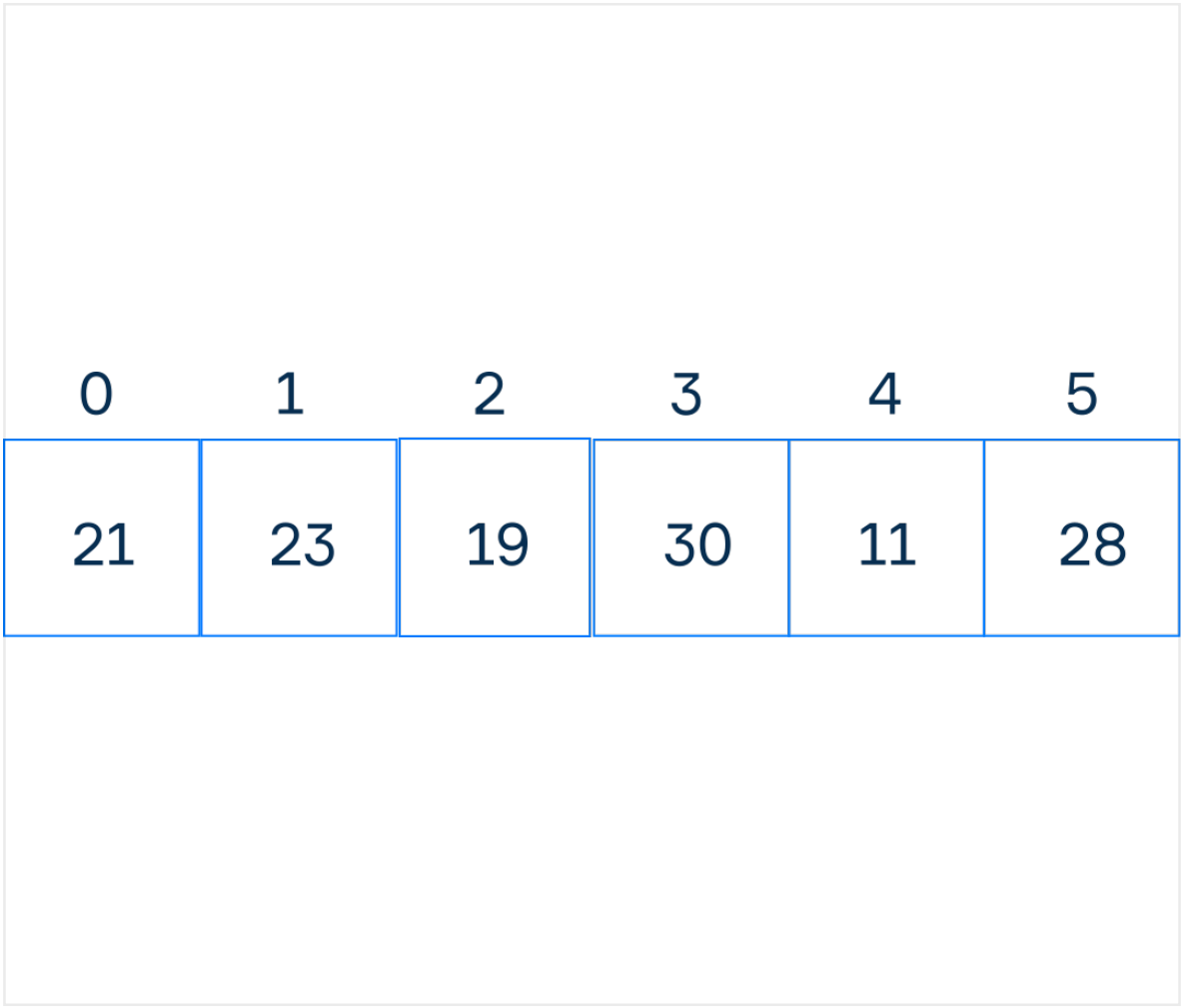
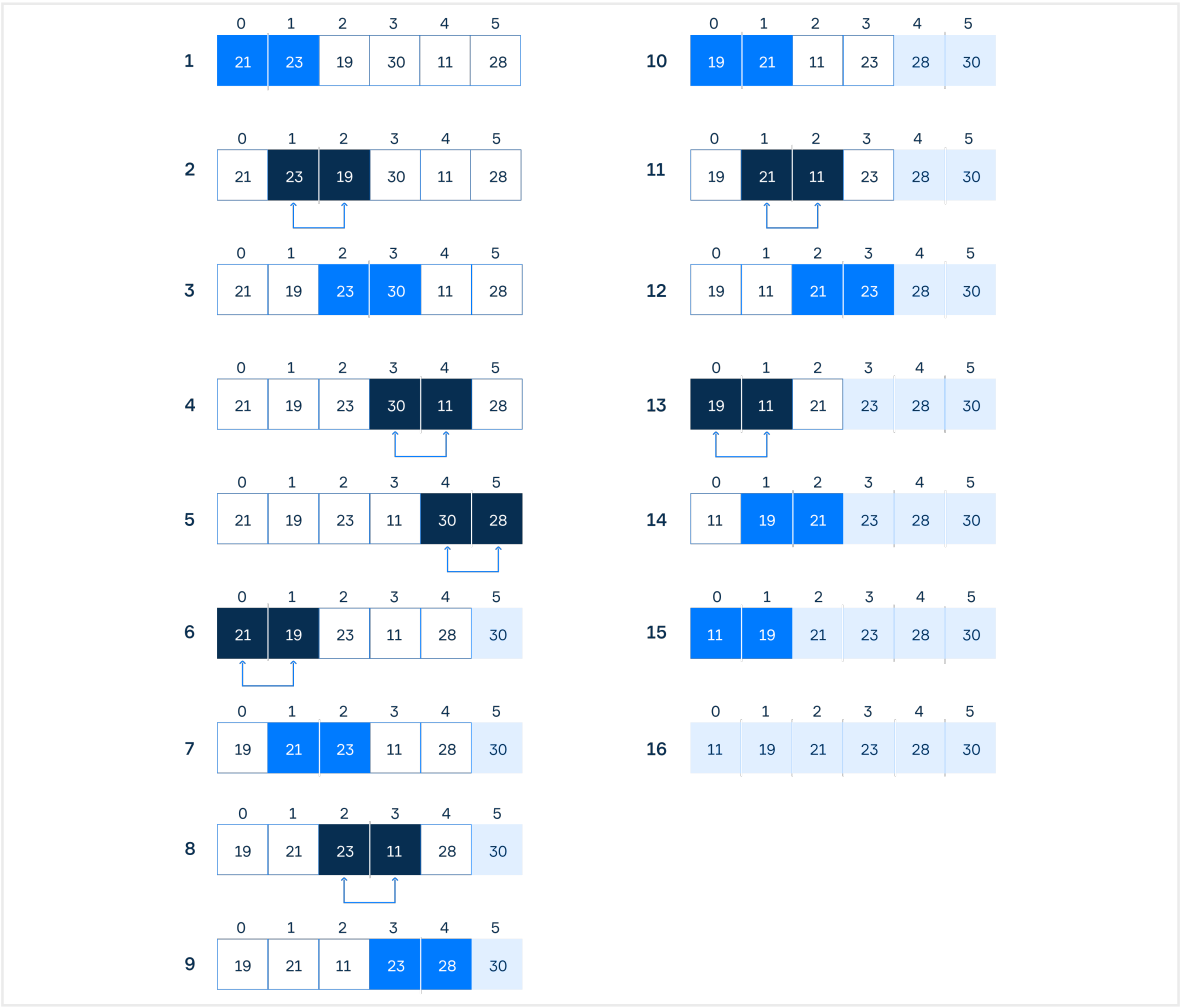
Bubble sort

Bubble sort is one of the simplest sorting algorithms. It repeatedly goes through the array we want to sort, compares each pair of adjacent elements, and swaps them if they are in the wrong order.
The wrong and correct orders depend on the required sorting order. If we need to sort the array in ascending order, the wrong order is when the previous element is greater than the next one. If we need to sort the array in descending order, the wrong order is the case when the previous element is smaller than the next one.
This algorithm is not suitable for large arrays, since its average and worst-case time complexity is
O(*n*^2)
, where n is the array length.
The algorithm is **stable**: it doesn't change the relative order of identical elements.

Example
Suppose we have an unsorted array of integers and we want to sort it in ascending order.



The array has six elements, the first element has the index 0, the last one has the index 5.
The following image illustrates how the bubble sort algorithm works step by step. The array is sorted in ascending order.



In a loop, the algorithm compares each pair of adjacent array elements and swaps them if they are in the wrong order (dark blue color). If the order is correct (blue color), it does not do anything to the elements.
As you can see, the max elements gradually float to the end of the array, which justifies the name of the algorithm. The algorithm doesn't swap elements in the right sorted part.

Pseudocode

In this section, we present the pseudocode for the Bubble Sort algorithm, illustrating its simple yet effective approach to sorting. The pseudocode follows the principles discussed earlier in the explanation.

```
// Swap Function
function swap(arr, i, j):
    temp = arr[i]
    arr[i] = arr[j]
    arr[j] = temp

// Bubble Sort Function
```

```
function bubbleSort(arr):
    // Get the length of the array
    n = length(arr)

    // Initialize a flag for swaps
    swapped = true

    // Continue until no more swaps are needed
    while swapped is true:
        swapped = false
        // Iterate through the array
        for i in [2, n]:
            // Compare adjacent elements and swap if necessary (for ascending
order)
            if arr[i - 1] > arr[i] then:
                swap(arr, i - 1, i)
                swapped = true
            // For descending order, change the condition to arr[i - 1] < arr[i]
```