

WILEY

ENABLING DISCOVERY | POWERING EDUCATION | SHAPING WORKFORCES

Data Structure and Algorithm

The background of the slide is a vibrant, futuristic digital landscape. It features a central laptop with a screen displaying various data visualizations, including line graphs, bar charts, and circular progress indicators. The laptop is surrounded by several glowing blue squares, each with a white circular outline and a grid pattern, resembling circuit boards or data nodes. These squares are connected by glowing orange lines and dots, suggesting a network or data flow. The entire scene is set against a dark blue background filled with binary code (0s and 1s) and a bright, diagonal white light beam.



Objectives

By the end of this module, you will be able to:



Define sorting



Identify the need for sorting



Identify the types of sorting and its usage

Let Us Discuss

While purchasing clothes in a store, how do you find the correct size for yourself?



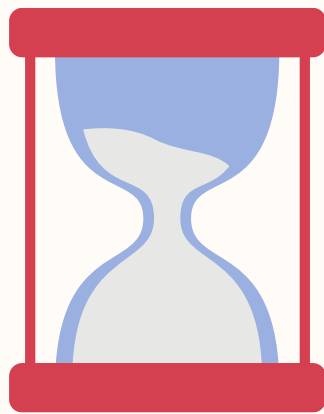
Sorting Data

- **Sorting** is an efficient solution to search records investing a smaller amount of time.
- Sorting is the process of **arranging data in some predefined order or sequence**.
- If the data is sequenced, you can reduce the number of records to be traversed.
- When the data is large, selecting a **sorting algorithm** makes the most efficient use of time or memory.

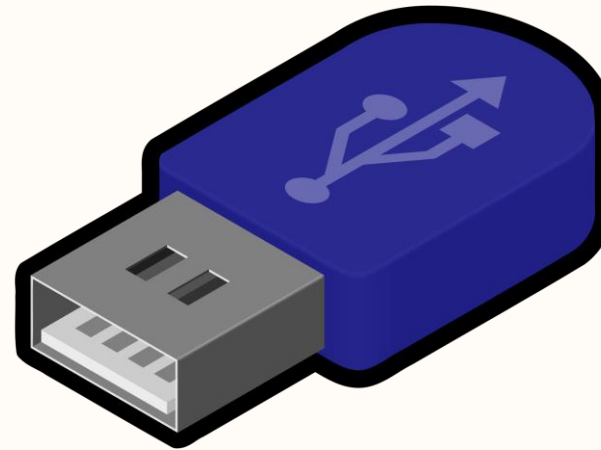


Selecting a Sorting Algorithm

To select an appropriate algorithm, you need to consider the following factors:



Execution time



Storage Space

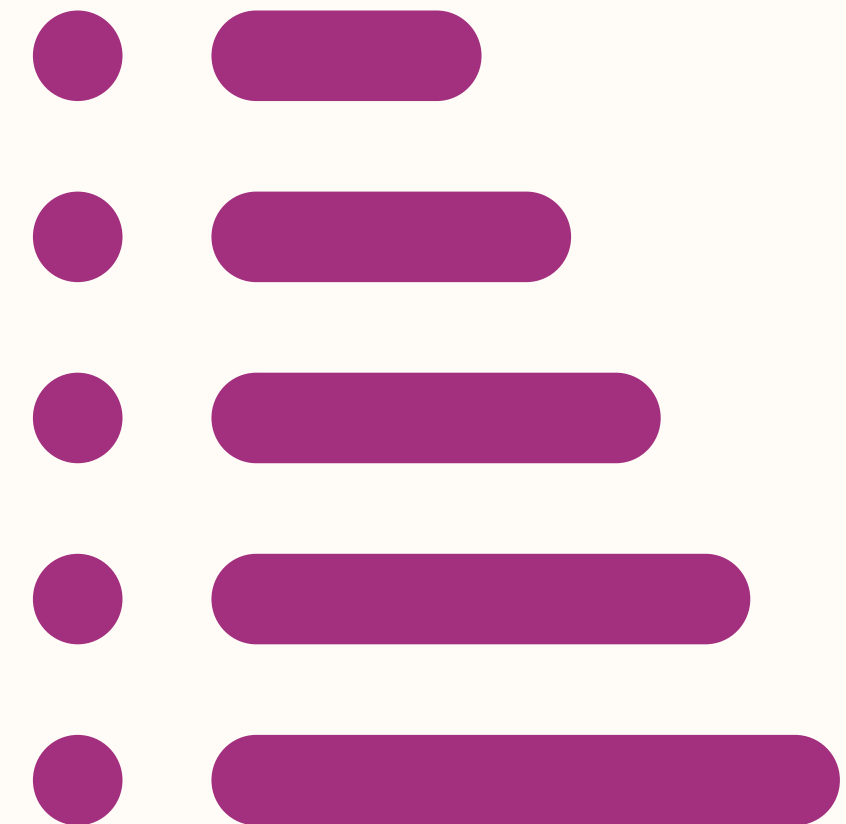


Programming Effort

Types of Sorting Algorithms

There are various sorting algorithms that are used to sort data. Some of these are:

- **Bubble sort**
- **Insertion sort**
- **Quick sort**



Sorting Data by Using Bubble Sort

- **Bubble Sort Algorithm:**
 - Is one of the simplest sorting algorithms.
 - Has a quadratic order of growth and is therefore suitable for sorting small lists only.
 - Works by repeatedly scanning through the list, comparing adjacent elements, and swapping them if they are in the wrong order.

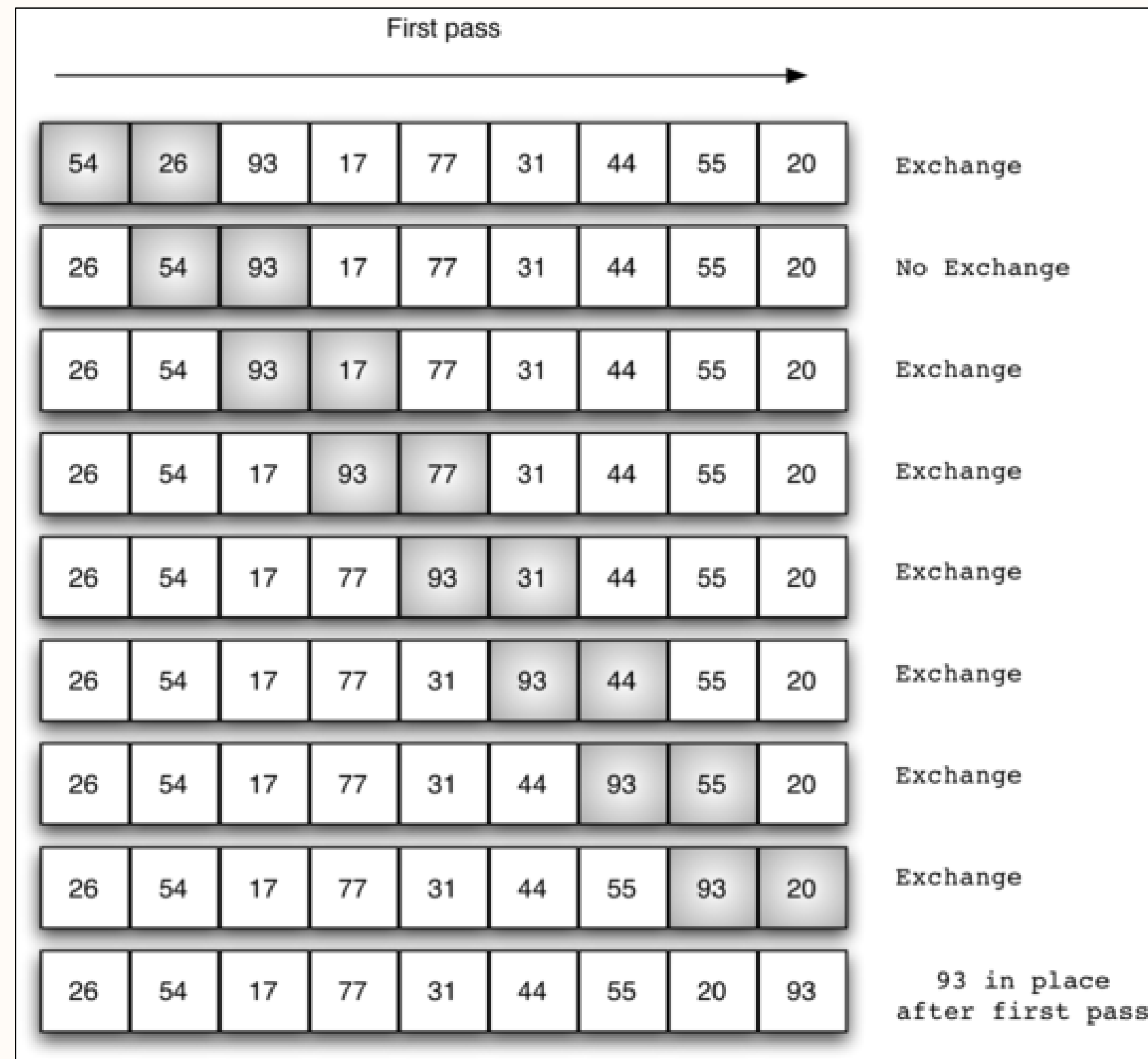
6 5 3 1 8 7 2 4

Implementing the Bubble Sort Algorithm

Algorithm for bubble sort:

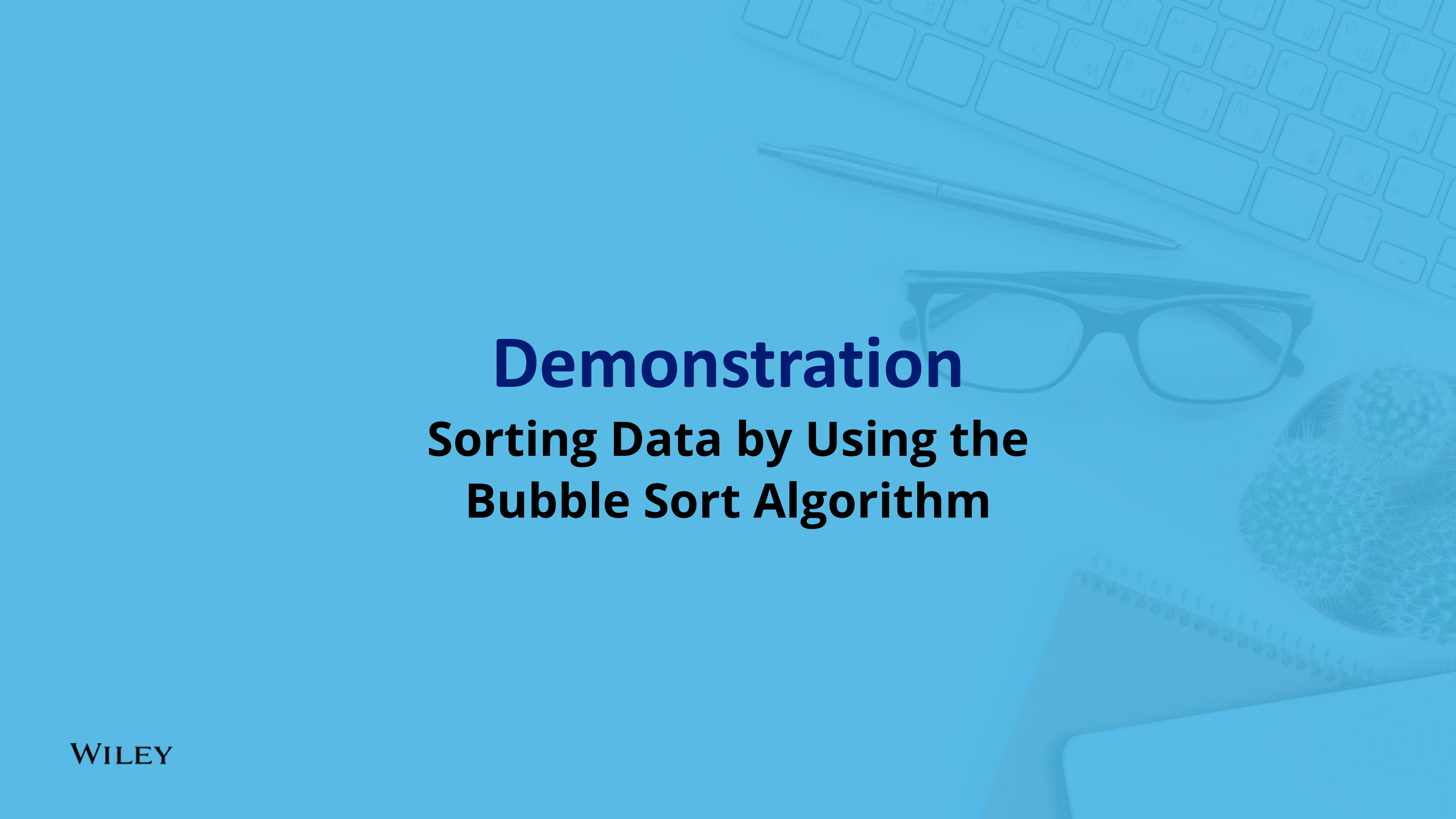
1. Set $\text{pass} = 1$.
2. Repeat step 3 varying j from 0 to $n - 1 - \text{pass}$.
3. If the element at index j is greater than the element at index $j + 1$, swap the two elements.
4. Increment pass by 1.
5. If $\text{pass} \leq n - 1$, go to step 2.

Implementing the Bubble Sort Algorithm



Determining the Efficiency of the Bubble Sort Algorithm

- The efficiency of a sorting algorithm is measured in terms of number of comparisons.
- There are $n-1$ comparisons in Pass 1, $n-2$ comparisons in Pass 2, and so on.
- The total number of comparisons will be calculated by using the following formula:
 - $\text{Sum} = (n - 1) + (n - 2) + (n - 3) + \dots + 3 + 2 + 1$
 - $\text{Sum} = n/2 [2a + (n - 1)d]$
 $\text{Sum} = (n - 1)/2 [2 \times 1 + (n - 1 - 1) \times 1]$
 $\text{Sum} = (n - 1)/2 [2 + (n - 2)]$
 $\text{Sum} = (n - 1)/2 (n)$
 $\text{Sum} = n(n - 1)/2$
- The bubble sort algorithm is of the order $O(n^2)$.

The background of the slide is a light blue overlay on a photograph of a desk. On the desk, there is a white computer keyboard in the upper right, a silver pen lying horizontally, a pair of black-rimmed glasses, and a small, round, spiky cactus in a pot. A spiral-bound notebook is visible in the bottom right corner.

Demonstration

Sorting Data by Using the Bubble Sort Algorithm

Sorting Data by Using Insertion Sort

Insertion Sort Algorithm:

- Has a **quadratic order of growth** and is therefore suitable for sorting small lists only.
- Is much more efficient than bubble sort, if the list that needs to be sorted is nearly sorted.

Implementing the Insertion Sort Algorithm

- Algorithm for insertion sort:
 - Repeat steps 2, 3, 4, and 5 varying i from 1 to $n - 1$.
 - Set $\text{temp} = \text{arr}[i]$.
 - Set $j = i - 1$.
 - Repeat until j becomes less than 0 or $\text{arr}[j]$ becomes less than or equal to temp :
 - a. Shift the value at index j to index $j + 1$.
 - b. Decrement j by 1.
 - Store temp at index $j + 1$.

Implementing the Insertion Sort Algorithm

3 7 5 6 9

Compare 7 with 3, $3 < 7$ so skip

3 7 5 6 9

Compare 5 with 7 and 3, $5 < 7$ so move 5 left

3 5 7 6 9

Compare 5 with 3, $5 > 3$ keep it same

3 5 7 6 9

Compare 6 with 7, 5 and 3, $6 < 7$ so swap both

3 5 6 7 9

Compare 6 with 5 and 3, $6 > 5$ so skip

3 5 6 7 9

Compare 9 with 7, 6, 5 and 3, since $9 > 7$, keep it same

The background of the slide is a light blue overlay on a photograph of a desk. On the desk, there is a white computer keyboard with Cyrillic characters, a silver pen, a pair of black-rimmed glasses, and a small, round, spiky cactus. A spiral-bound notebook is also visible in the bottom right corner.

Demonstration

Sorting Data by Using the Insertion Sort Algorithm

Sorting Data by Using Quick Sort

Quick Sort Algorithm:

- is one of the most efficient sorting algorithms.
- is based on the divide and conquer approach.
- successively divides the problem into smaller parts until the problems become so small that they can be directly solved.

Implementing the Quick Sort Algorithm

In quick sort algorithm:

- Select an element from the list called as **pivot**
- **Partition the list** into two parts such that:
 - All the elements towards the **left end of the list are smaller** than the **pivot**
 - All the elements towards the **right end of the list are greater** than the **pivot**
- Store the **pivot** at its correct position **between the two parts of the list**
- Repeat this process for each of the two sub-lists created after partitioning

This process continues until one element is left in each sub-list.

Implementing the Quick Sort Algorithm

The following algorithm depicts the logic of quick sort:

```
QuickSort(low,high)
```

```
1. If (low > high):
```

```
    Return.
```

```
2. Set pivot = arr[low].
```

```
3. Set i = low + 1.
```

```
4. Set j = high.
```

```
5. Repeat step 6 until i > high or arr[i] > pivot.
```

```
// Search for an element greater than the pivot.
```


Implementing the Quick Sort Algorithm (cont.)

6. Increment i by 1.

7. Repeat step 8 until $j < \text{low}$ or $\text{arr}[j] < \text{pivot}$.
//Search for an element smaller than pivot

8. Decrement j by 1.

9. If $i < j$:

 // If greater element is on the left of
 //smaller element

 Swap $\text{arr}[i]$ with $\text{arr}[j]$.

10. If $i \leq j$:

 Go to step 5. // Continue the search

11. If $\text{low} < j$:

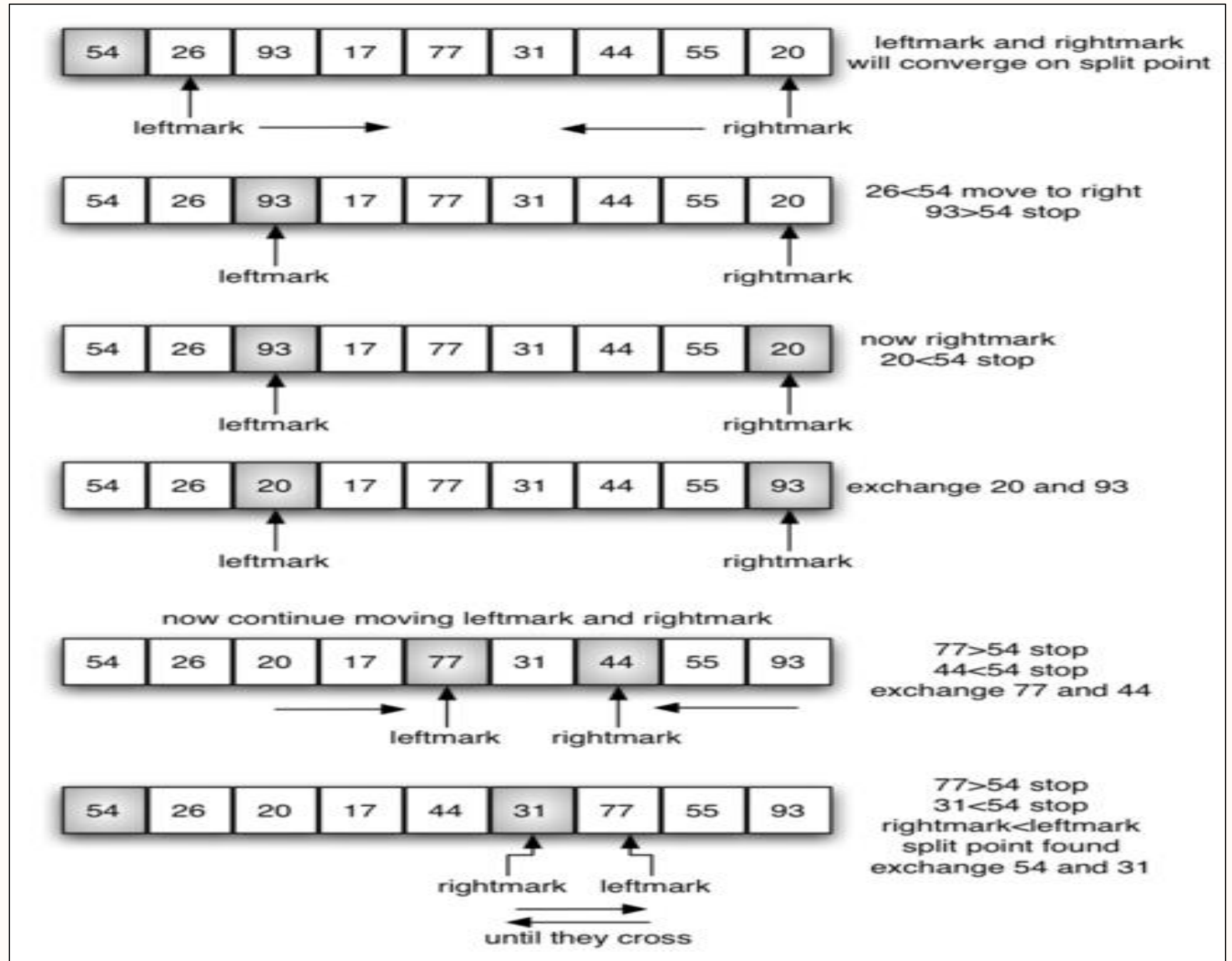
 Swap $\text{arr}[\text{low}]$ with $\text{arr}[j]$.

 //Swap pivot with last element in first part of
 //the list

Implementing the Quick Sort Algorithm (Cont.)

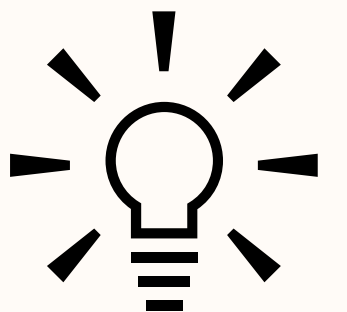
```
QuickSort(low, j - 1). // Apply quicksort on          //list left to pivot  
13. QuickSort(j + 1, high). // Apply quicksort on  
    //list right to pivot
```

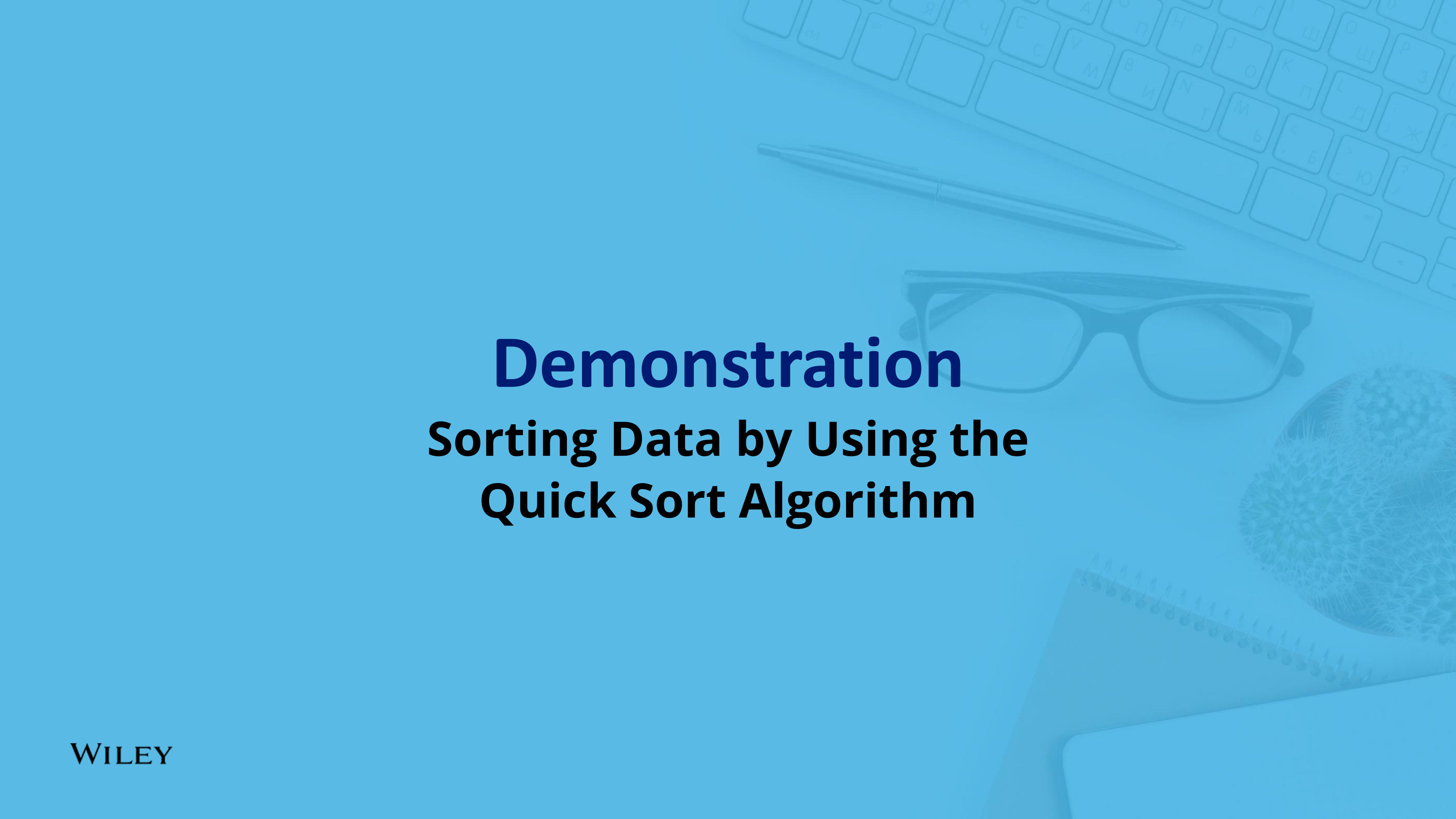
Implementing the Quick Sort Algorithm



Activity

Write a program that stores 10 numbers in an array and sorts them by using the quick sort algorithm. In addition, the program should also calculate the number of comparisons and the number of data movements.



The background of the slide is a light blue overlay on a photograph of a desk. On the desk, there is a white computer keyboard in the upper right, a silver pen lying horizontally, a pair of black-rimmed glasses, and a spiral-bound notebook in the lower right. A small, round, spiky object, possibly a cactus or a stress ball, is also visible near the glasses.

Demonstration

Sorting Data by Using the Quick Sort Algorithm

Question

What is the order of growth for the bubble sort algorithm?

- Linear
- Loglinear
- Constant
- Quadratic





Questions?

Summary

In this session, you learned to:

- Implement Bubble sort
 - Bubble sort algorithm has a quadratic order of growth and is therefore suitable for sorting small lists only.
 - Understand the efficiency of bubble sort
- Implement Insertion Sort
 - Insertion sort performs a different number of comparisons depending on the initial ordering of elements.
 - Understand the efficiency of bubble sort
- Implement Quick Sort.
 - Understand the efficiency of bubble sort
 - The quick sort algorithm recursively divides the list into two sub lists.