Algorithm and Data Structures
Week 8 and Week 9

Endang Wahyu Pamungkas, Ph.D.

The Concept of Sorting

- **Sorting** is a basic building block that many other algorithms are built upon. It's related to several exciting ideas that you'll see throughout your programming career.
- Understanding how sorting algorithms in Python work behind the scenes is a fundamental step toward implementing correct and efficient algorithms that solve **real-world problems**.

The Importance of Sorting Algorithm

- Sorting is one of the most thoroughly studied algorithms in computer science. There are dozens of different sorting implementations and applications that you can use to make your code more efficient and effective:
  - **Searching**: Searching for an item on a list works much faster if the list is sorted.
  - **Selection**: Selecting items from a list based on their relationship to the rest of the items is easier with sorted data.
  - **Duplicates**: Finding duplicate values on a list can be done very quickly when the list is sorted.
  - **Distribution**: Analyzing the frequency distribution of items on a list is very fast if the list is sorted.

Python's Built-In Sorting Algorithm

- The Python language, like many other high-level programming languages, offers the ability to sort data out of the box using sorted().

```
>>> array = [8, 2, 6, 4, 5]
>>> sorted(array)
[2, 4, 5, 6, 8]
```

Sorting Algorithm

- There are several sorting algorithms that we can implement to solve the sorting problem, including:
  - Selection Sort
  - Bubble Sort
  - Insertion Sort
  - Quick Sort
  - Merge Sort

Selection Sort

- In selection sort we start by finding the minimum value in a given list and move it to a sorted list. Then we repeat the process for each of the remaining elements in the unsorted list.
- The next element entering the sorted list is compared with the existing elements and placed at its correct position.
- So, at the end all the elements from the unsorted list are sorted.

Selection Sort

Selection Sort

```
def selection_sort(unsorted_list):
  sorted_list = []
  for idx in range(len(unsorted_list)):
    min_idx = 0
    min_value = 1000
    for idx2 in range(len(unsorted_list)):
      if unsorted_list[idx2] < min_value:
        min_idx = idx2
        min_value = unsorted_list[idx2]
    del unsorted_list[min_idx]
    sorted_list.append(min_value)
  return sorted_list
```

Selection Sort

```
def selection_sort(input_list):
    for idx in range(len(input_list)):
        min_idx = idx
        for j in range( idx +1, len(input_list)):
            if input_list[min_idx] > input_list[j]:
             min_idx = j
# Swap the minimum value with the compared value
        input_list[idx], input_list[min_idx] =
input_list[min_idx], input_list[idx]
```

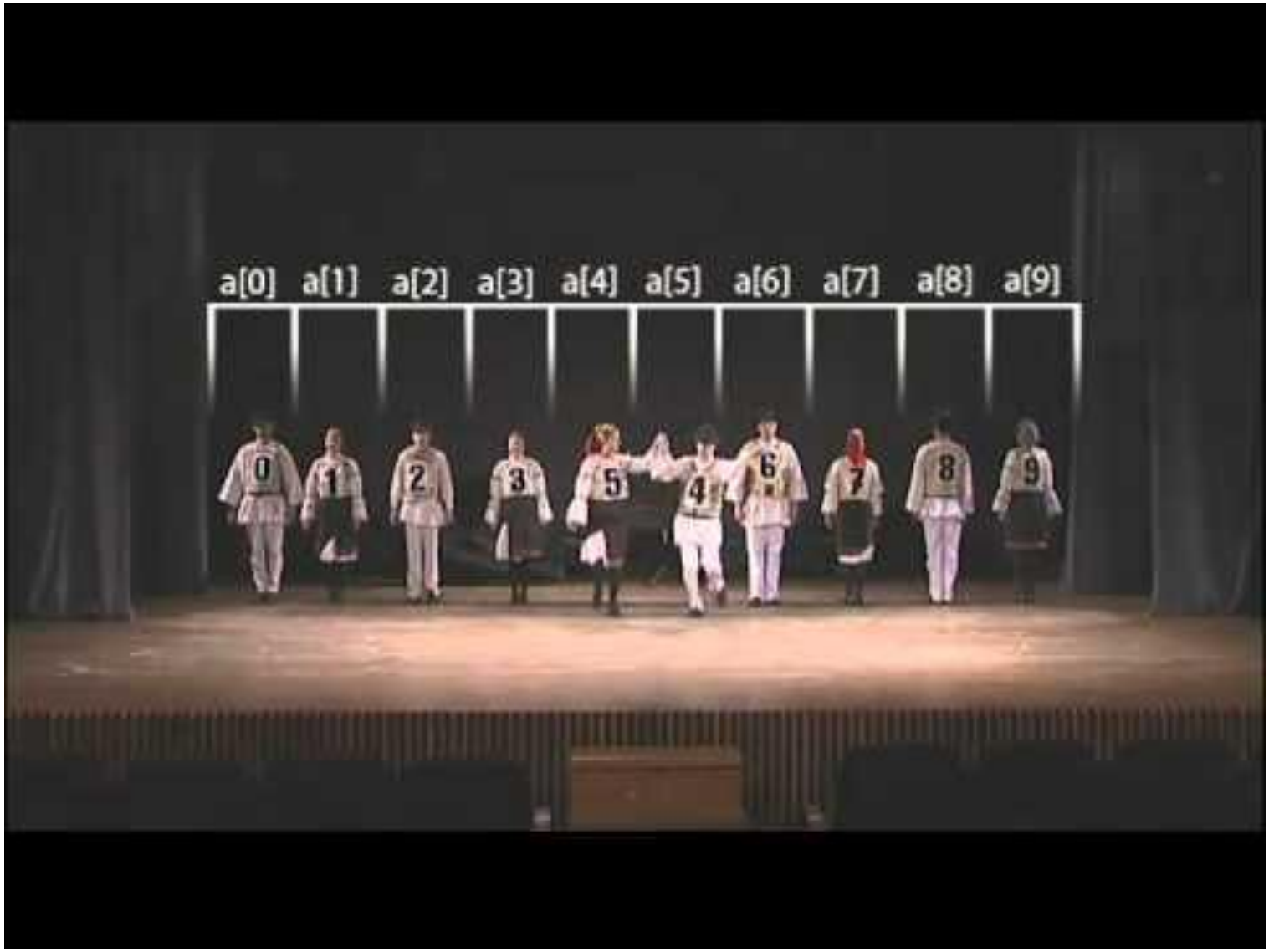
Bubble Sort

- Bubble Sort is one of the most straightforward sorting algorithms. Its name comes from the way the algorithm works: With every new pass, the largest element in the list “bubbles up” toward its correct position.
- Bubble sort consists of making multiple passes through a list, comparing elements one by one, and swapping adjacent items that are out of order.

Bubble Sort

- Bubble Sort is one of the most straightforward sorting algorithms. Its name comes from the way the algorithm works: With every new pass, the largest element in the list “bubbles up” toward its correct position.
- Bubble sort consists of making multiple passes through a list, comparing elements one by one, and swapping adjacent items that are out of order.
- The idea is that to sort a group of data, just compare two adjacent data. If the left is smaller than the right, it means it’s okay. But if the left is greater than the right, the two data are swapped.

Bubble Sort



Bubble Sort

- First Iteration (Compare and Swap)
  - Starting from the first index, compare the first and the second elements.
  - If the first element is greater than the second element, they are swapped.
  - Now, compare the second and the third elements. Swap them if they are not in order.
  - The above process goes on until the last element.

Exercise

- I have the following number sequence :
  3, 6, 4, 7, 1, 2, 5, 8
- Could you please demonstrate step by step how this number sequence can be sorted by using bubble sort algorithm?

Bubble Sort

- Now, try to implement it !

Bubble Sort

```
def bubbleSort(array):
  for i in range(len(array)):
    for j in range(0, len(array) - i - 1):
      if array[j] > array[j + 1]:
        temp = array[j]
        array[j] = array[j+1]
        array[j+1] = temp
```

Bubble Sort

- How about the implementation of descending Bubble Sort?

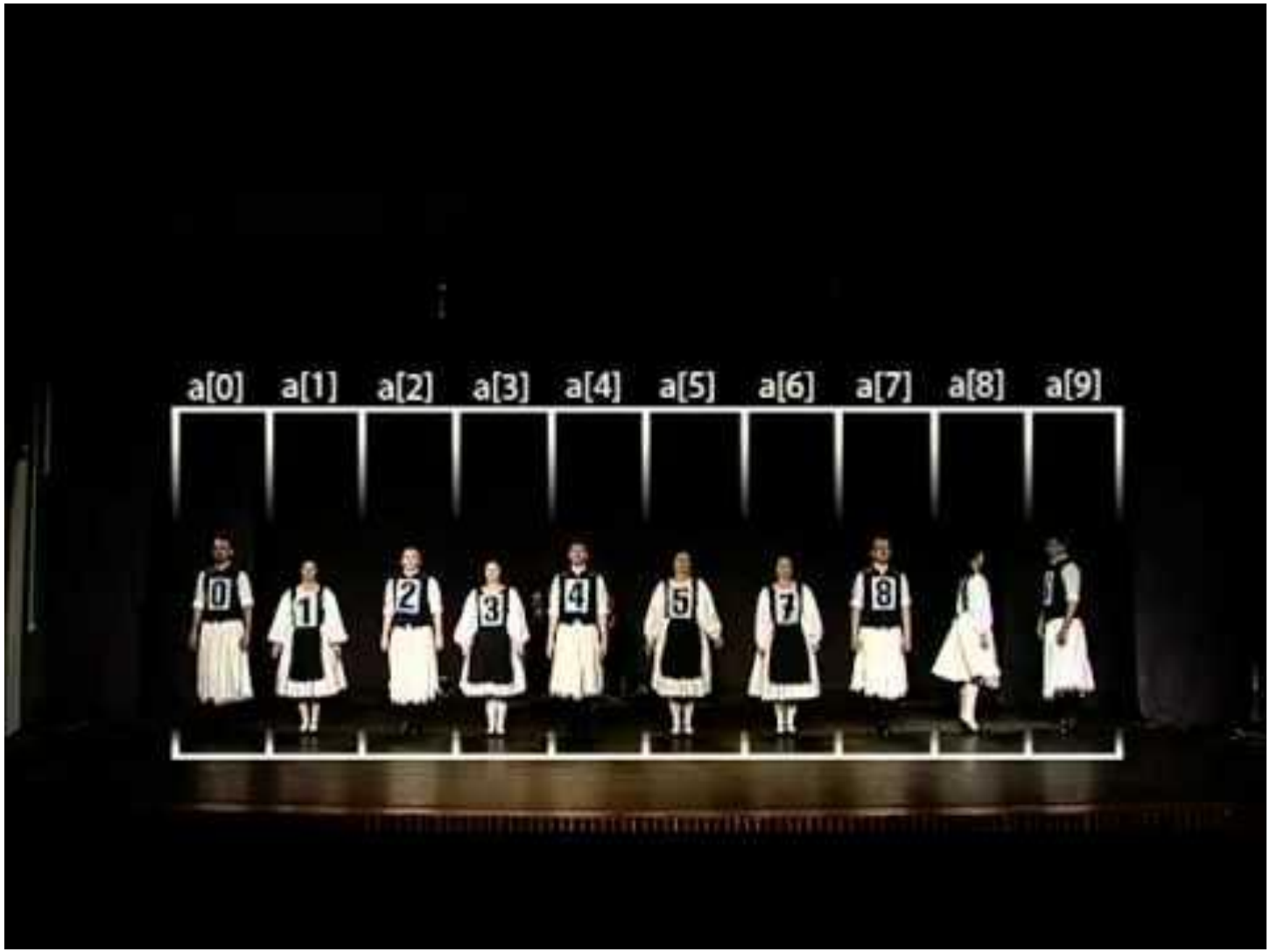
Insertion Sort

- Like bubble sort, the insertion sort algorithm is straightforward to implement and understand.
- But unlike bubble sort, it builds the sorted list one element at a time by comparing each item with the rest of the list and inserting it into its correct position.
- This “insertion” procedure gives the algorithm its name.

Insertion Sort

- An excellent analogy to explain insertion sort is the way you would sort a deck of cards. Imagine that you're holding a group of cards in your hands, and you want to arrange them in order.
- You'd start by comparing a single card step by step with the rest of the cards until you find its correct position.
- At that point, you'd insert the card in the correct location and start over with a new card, repeating until all the cards in your hand were sorted.

Insertion Sort



Insertion Sort

- The first element in the array is assumed to be sorted. Take the second element and store it separately in key.
- Compare key with the first element. If the first element is greater than key, then key is placed in front of the first element.
- Take the third element and compare it with the elements on the left of it. Placed it just behind the element smaller than it. If there is no element smaller than it, then place it at the beginning of the array.
- Do the same for the rest element.

Exercise

- I have the following number sequence :
  3, 6, 4, 7, 1, 2, 5, 8
- Could you please demonstrate step by step how this number sequence can be sorted by using insertion sort algorithm?

Insertion Sort

```
def insertionSort(array):
    for step in range(1, len(array)):
        key = array[step]
        j = step - 1
        while j >= 0 and key < array[j]:
           array[j + 1] = array[j]
           j = j - 1
           array[j + 1] = key
```

Insertion Sort

- Again, how if we want to sort in descending order?

Exercise

A school is trying to take an annual photo of all the students. The students are asked to stand in a single line in non-decreasing order by height. Return the number of students who did not stand in the right position!

Example :
Input: heights = [1,1,4,2,1,3]
Output: 3
Input: heights = [5,1,2,3,4]
Output: 5
Input: heights = [1,2,4,5,3]
Output: 3