Lesson 5: Sorting Algorithms and Introduction to Recursion

Marcin Kurzyna

Lecture Goals

This lecture introduces:

- The Bubble Sort algorithm.
- What recursion is and how it works in Python.
- Classic recursive problems: factorial and Fibonacci numbers.

1 Bubble Sort

Bubble Sort is one of the simplest sorting algorithms. It repeatedly goes through the list, compares pairs of elements, and swaps them if they are in the wrong order.

How It Works

- Compare element i with element $i + 1$.
- If they are in the wrong order, swap them.
- Repeat this for all pairs.
- With each full pass, the largest remaining element “bubbles up” to the end.

Example Code

```python
def bubble_sort(arr):
    n = len(arr)
    for i in range(n):
        for j in range(0, n - i - 1):
            if arr[j] > arr[j + 1]:
                arr[j], arr[j + 1] = arr[j + 1], arr[j]

data = [5, 2, 9, 1, 5, 6]
bubble_sort(data)
print(data)
```

2 Introduction to Recursion

Recursion is a technique where a function calls itself to solve a smaller version of the same problem.

Key Components of Recursion

- **Base case** – stops the recursion.
- **Recursive case** – reduces the problem into a smaller one.

2.1 Example: Factorial

The factorial of a number n is defined as:

$$n! = n \cdot (n-1)!$$

with:

$$0! = 1$$

```python
def factorial(n):
    if n == 0:
        return 1
    return n * factorial(n - 1)

print(factorial(5))  # 120
```

3 Fibonacci Numbers

The Fibonacci sequence is defined as:

$$F(0) = 0, \quad F(1) = 1$$

$$F(n) = F(n-1) + F(n-2)$$

```python
def fibonacci(n):
    if n <= 1:
        return n
    return fibonacci(n - 1) + fibonacci(n - 2)

print(fibonacci(6))  # 8
```

Summary

In this lecture, we covered:

- How Bubble Sort works and how to implement it.
- What recursion is and how Python functions can call themselves.
- Two classic recursive examples: factorial and the Fibonacci sequence.

4 Exercises

1. Implement Bubble Sort so that it sorts numbers in **descending** order.
2. Modify Bubble Sort to detect if the list is already sorted and stop early.
3. Write a recursive function `power(a, b)` that computes:

   $$a^b$$

4. Write a recursive function that counts how many digits a number has.
5. Write a function using recursion that returns the sum of all numbers from 1 to n.
6. Write a function `reverse_recursive(s)` that reverses a string recursively.
7. Write a recursive version of the greatest common divisor function (GCD) using Euclid's algorithm.
8. Challenge: Write an efficient Fibonacci function using **memoization**.