# PG4200 Algorithms and Data Structures

# Re-sit exam

## 6/8/2025

Candidate: 31

### Question 5:

### LO4: Traversing Graphs Algorithms

The Fibonacci sequence is defined as:

F(0) = 0, F(1) =1

F(n) = f(n-1) + F(n-2) for n≥2

(Goodrich & Tamassia, 2006, Ch.4.3; Sedgewick & Wayne, 2011, Ch.2.3)

It can be computed in two common ways:

1. Recursive approach, directly applies the mathmatical recurrence relation.
2. Iterative approach, uses a loop to build the result from the bottom up.

Screenshot of the Fibonacci Recursive Algorithm from IntelliJ *(based on Sedgewick & Wayne, 2011, Ch.2.3):*

Et bilde som inneholder tekst, skjermbilde, programvare, display

KI-generert innhold kan være feil.

Screenshot of the output:

Et bilde som inneholder tekst, Font, skjermbilde, design

KI-generert innhold kan være feil.

Screenshot of the Fibonacci Iterative Algorithm from IntelliJ:

Et bilde som inneholder tekst, skjermbilde, programvare, Multimedieprogramvare

KI-generert innhold kan være feil.

Screenshot of the output:

Et bilde som inneholder tekst, Font, skjermbilde, design

KI-generert innhold kan være feil.

## Efficiency justification

#### Recursive approach:

Logic: Breaks problem into smaller subproblems

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

Negative: Recalculate the same Fibonacci values many times. For example:  
fib(5) computes fib(3) twice, fib(2) three times, etc.

Time complexity: O(2^n), because each call generates two more calls, leading to exponential growth.

Space complexity: O(n), due to call stack repth in recursion.

#### Iterative approach:

Logic: Builds Fibonacci sequence from the bottom up using a loop.

Positive: Computes each Fobonacci number exactly once.

Time complexity: O(n), because it loops once per term.

Space complexity: O(1), because it only stores a few variables.

#### Experimental comparison for n = 1 🡪 5

Both produce the same output:

1, 1, 2, 3, 5

But:

Recursive approach:

* Makes many repeated calls even for small n
* Uses more memory because of call stack
* Gets much slower as n increases

Iterative approach:

* One simple loop
* No repeated computation
* Minimal memory use

### Conclusion

The iterative algorithm is more efficien bacuse:

1. Time complexity: O(n) vs O(2^n)
2. Space complexity O(1) vs =(n)
3. No redundant calculations
4. Scales better for large n

## Referances

Goodrich, M. T., & Tamassia, R. (2006). Data Structures and Algorithms in Java (4th ed., Ch. 4.3). Wiley.

Sedgewick, R., & Wayne, K. (2011). Algorithms (4th ed., Ch. 2.3). Addison‑Wesley.

Gupta, R. (n.d.). LO 4: Traversing Graphs Algorithms [Lecture slides]. Kristiania University College.