Introduktion til Programmering og Problemløsning (PoP)

Jon Sporring
Department of Computer Science
2022/09/02

UNIVERSITY OF COPENHAGEN

F# er en lommeregner

```
% dotnet fsi


Microsoft (R) F# Interactive version
12.0.4.0 for F# 6.0

Copyright (c) Microsoft Corporation.
All Rights Reserved.


For help type #help;;


> 357;;

val it: int = 357


> 864;;

val it: int = 864


> 357+864;;

val it: int = 1221
```

```
> let a = 357;;

val a: int = 357


> let b = 864

- let c = a+b

- printfn "%A + %A = %A" a b c;;

357 + 864 = 1221

val b: int = 864

val c: int = 1221

val it: unit = ()
```

Leksikografisk virkefelt

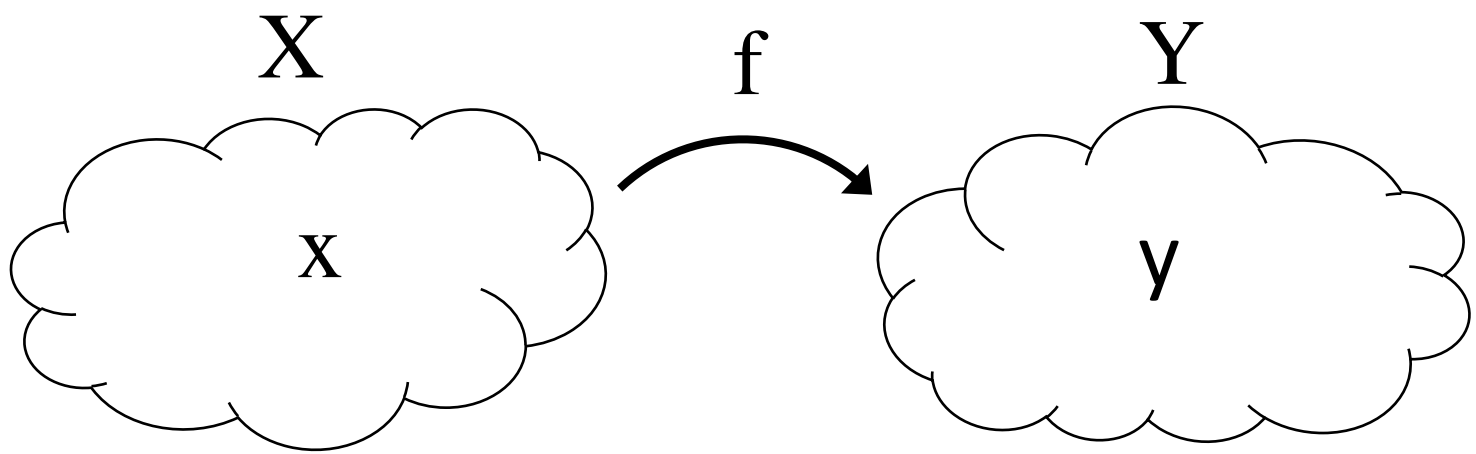
3 måder at køre (execute / run) programmet på:

- dotnet fsi -> indtast myFirstFsharp.fsx
- dotnet fsi myFirstFsharp.fsx
- dotnet run

F# funktioner

```
% dotnet fsi

Microsoft (R) F# Interactive version 12.0.4.0 for F# 6.0
Copyright (c) Microsoft Corporation. All Rights Reserved.

For help type #help;;

> let m = 3*4;;
val m: int = 12

> let mul x = 3*x;;
val mul: x: int -> int

> let y = mul 4;;
val y: int = 12
```

```
> let a = 3
- let b = 4
- let mul () = a*b;;
val a: int = 3
val b: int = 4
val mul: unit -> int

> let y = mul ();;
val y: int = 12
```



Nedtælling med Rekursion

```
let rec countDown n =
    printfn "%A" n
    match n with
    0 -> ()
    | _ -> countDown (n-1)
```

sideeffekt

countDown 5

1. countDown 5
2. ~>countDown (countDown 4)
3. ~>countDown (countDown (countDown 3))
4. ~>countDown (countDown (countDown (countDown 2)))
5. ~>countDown (countDown (countDown (countDown (countDown 1))))
6. ~>countDown (countDown (countDown (countDown (countDown (countDown 0)))))
7. ~>countDown (countDown (countDown (countDown (countDown ()))))
8. ~>countDown (countDown (countDown (countDown ())))
9. ~>countDown (countDown (countDown ()))
10. ~>countDown (countDown ())
11. ~>countDown ( )
12. ~> ( )

Regler for rekursion
1. Der skal være et basetilfælde
2. Funktionen skal kalde sig selv evt. indirekte
3. Rækken af rekursive kald skal ramme basetilfældet indenfor endelig tid

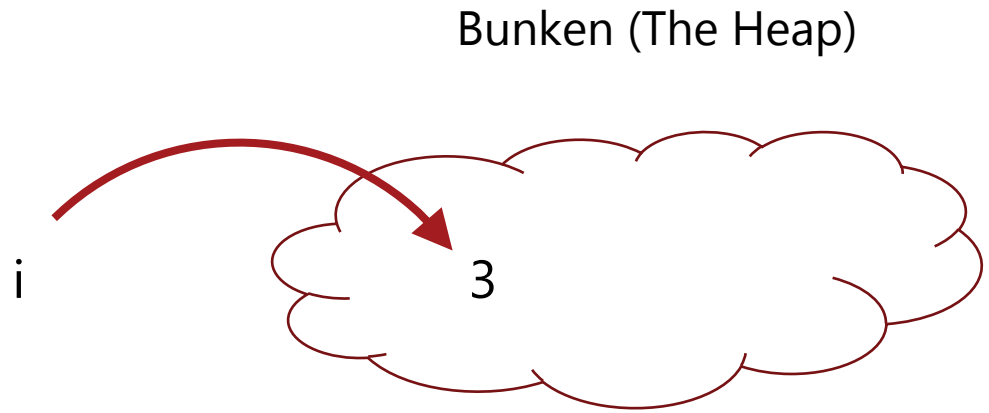
Nedtælling med while-løkke

```
let countDown n =
   let mutable i = n
   while i >= 0 do
      printfn "%A" i
      i <- i - 1
```

Dynamisk virkefelt

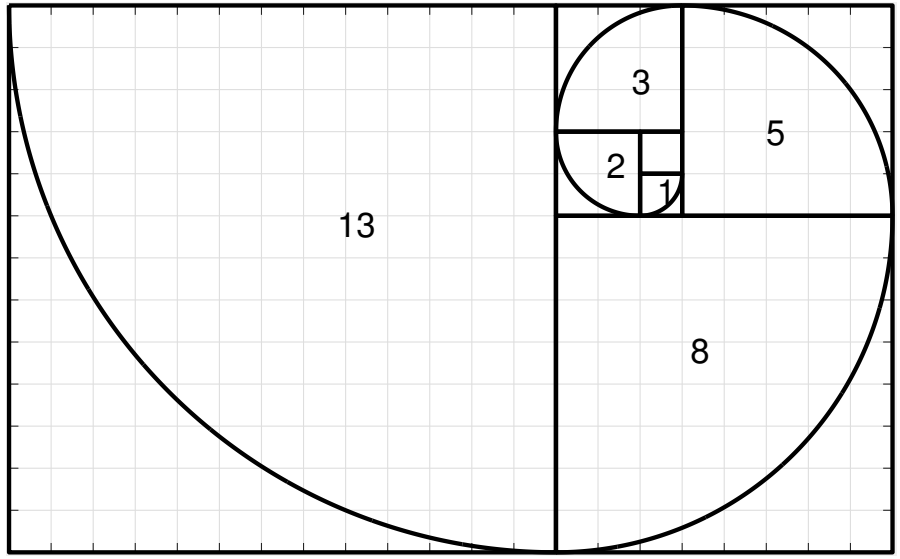
countDown 5

Regler for while-løkker
1. Der skal være en slutbetingelse
2. Løkken skal helst nærme sig slutbetingelsen

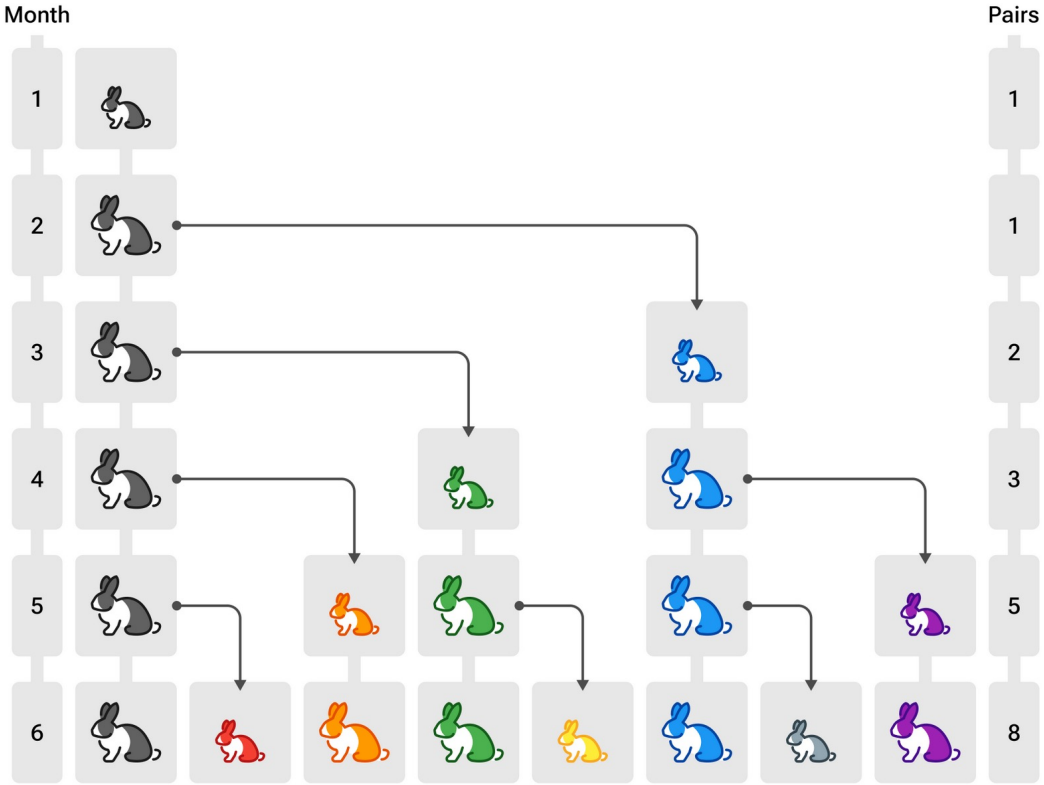


1. countDown 5
2. i = 5
3. i <- 4
4. i <- 3
5. i <- 2
6. i <- 1
7. i <- 0
8. i <- -1
9. ~> ( )

Fibonacci: 0 1 1 2 3 5 8 13 21 …



Kanin par: Nyt par efter 2 måneder

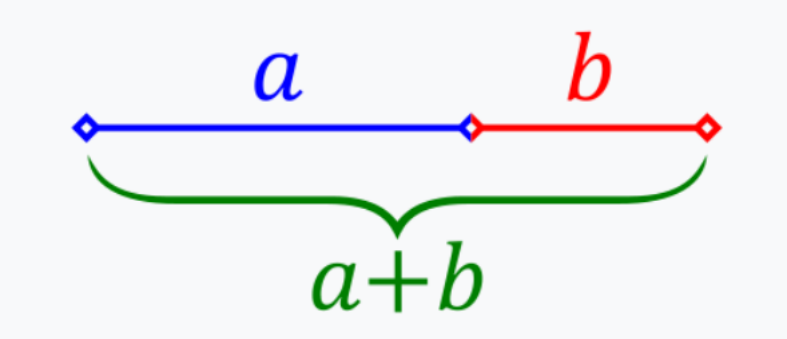


F(0) = 0

F(1) = 1

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

Det gyldne snit



b = F(n-2)

a = F(n-1)

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

$$\frac{a+b}{a} = \frac{a}{b}$$

$$1 + \frac{b}{a} = \frac{a}{b}$$

$$\phi = \frac{a}{b}$$

$$1 + \frac{1}{\phi} = \phi$$

$$\phi + 1 = \phi^2$$

$$\phi = \frac{1 \pm \sqrt{5}}{2}$$

$$= (-0.618\,, 1.618)$$

Fibonacci: 0 1 1 2 3 5 8 13 21 …

Recursive

```
let rec fib n =
  match n with
    0 -> 0
    | 1 -> 1
    | _ ->
      fib (n - 1) + fib (n - 2)
```

```
let mutable i = 0
while i <= 45 do
  printfn "fib(%d) = %d" i (fib i)
  i <- i + 1
```

$F(0) = 0$

$F(1) = 1$

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

```
% time dotnet fsi fibRecursive.fsx
fib(0) = 0
fib(1) = 1
fib(2) = 1
fib(3) = 2
fib(4) = 3
fib(5) = 5
…
fib(44) = 701408733
fib(45) = 1134903170
```

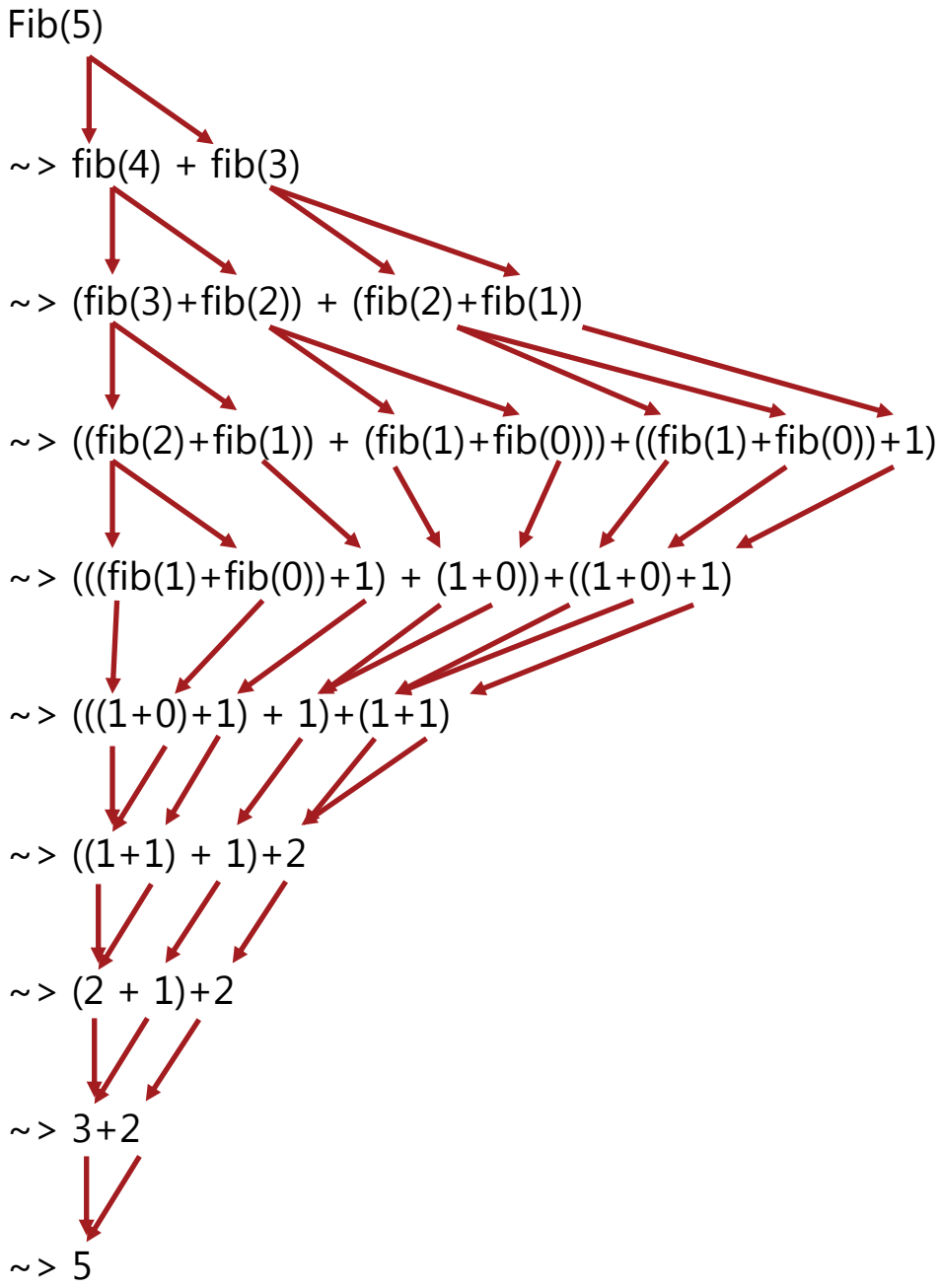
dotnet fsi fibRecursive.fsx 14.34s user …

Rekursion

$$F(0) = 0$$

$$F(1) = 1$$

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

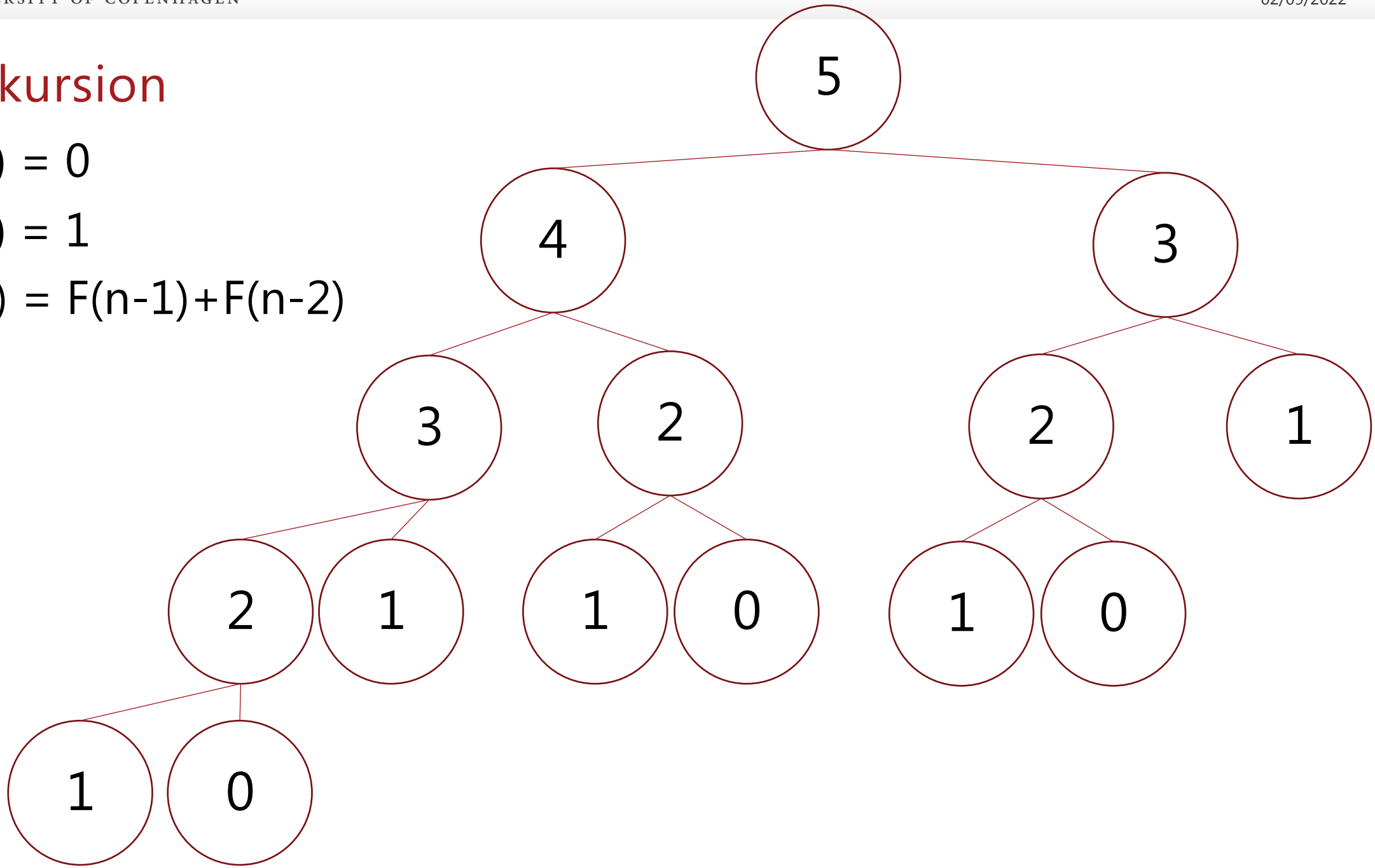


Rekursion

F(0) = 0

F(1) = 1

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



Fibonacci: 0 1 1 2 3 5 8 13 21 …

F(0) = 0

F(1) = 1

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

% time dotnet fsi fibImperative.fsx
…
fib(44) = 701408733
fib(45) = 1134903170
dotnet fsi fibImperative.fsx 0.98s user …

Imperative

```
let fib n =
  match n with
    0 -> 0
    | 1 -> 1
    | _ ->
      let mutable prevPrev = 0
      let mutable prev = 1
      let mutable i = 2;

      while i <= n do
        let curr = prev + prevPrev
        prevPrev <- prev
        prev <- curr
        i <- i + 1
      prev

let mutable i = 0
while i <= 45 do
  printfn "fib(%d) = %d" i (fib i)
  i <- i + 1
```

1. fib 5
2. prevPrev = 0
3. prev = 1
4. i = 2
5. curr = 1
6. prevPrev <- 1
7. prev <- 1
8. i <- 3
9. curr = 2
10. prevPrev <- 1
11. prev <- 2
12. i <- 4
13. curr = 3
14. prevPrev <- 2
15. prev <- 3
16. i <- 5
17. curr = 5
18. prevPrev <- 3
19. prev <- 5
20. i <- 6
21. ~> 5

Resumé

Denne video fortalte om:

- F# som en lommeregner
- Konstanter, typer, bindinger, betingelser, løkker, printfn, mutérbare værdier
- Fibonaccis talrække og sammenlignen mellem rekursiv og while-løkke