NTNU

Kunnskap for en bedre verden

Kunnskap for en bedre verden

TDT4105 Informasjonsteknologi, grunnkurs m/Matlab

Mer om funksjoner:

- rekursive funksjoner

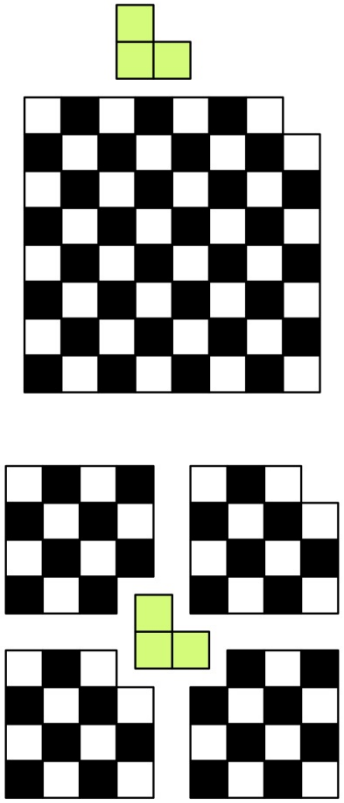
Pensum: **10.5 i Matlab-boka**

10.1-10.4 er orienteringsstoff og ikke aktuelt til eksamen

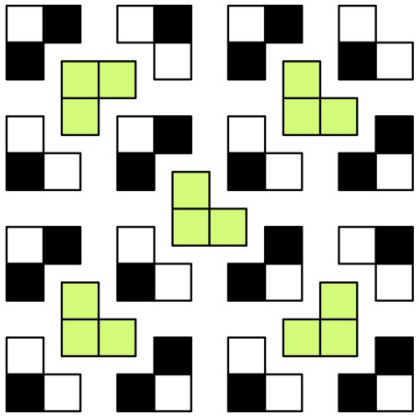
Anders Christensen (anders@ntnu.no)

Rune Sætre (satre@ntnu.no)

Rekursive funksjoner



- En rekursiv funksjon er en funksjon som kaller seg selv:
  - For å løse en enklere utgave av det samme problemet
  - Med modifiserte («enklere») parametere
  - Må ha stopptilfeller der vi vet løsningen
- Noen problemer løses enklest ved å løse et mindre (enklere) problem av den samme typen
  - n! = n * (n-1)! for alle n > 0
  - 0! = 1
- Rekursjon gir ofte enkle programmer
  - Men ikke-rekursive løsninger er ofte mer effektive



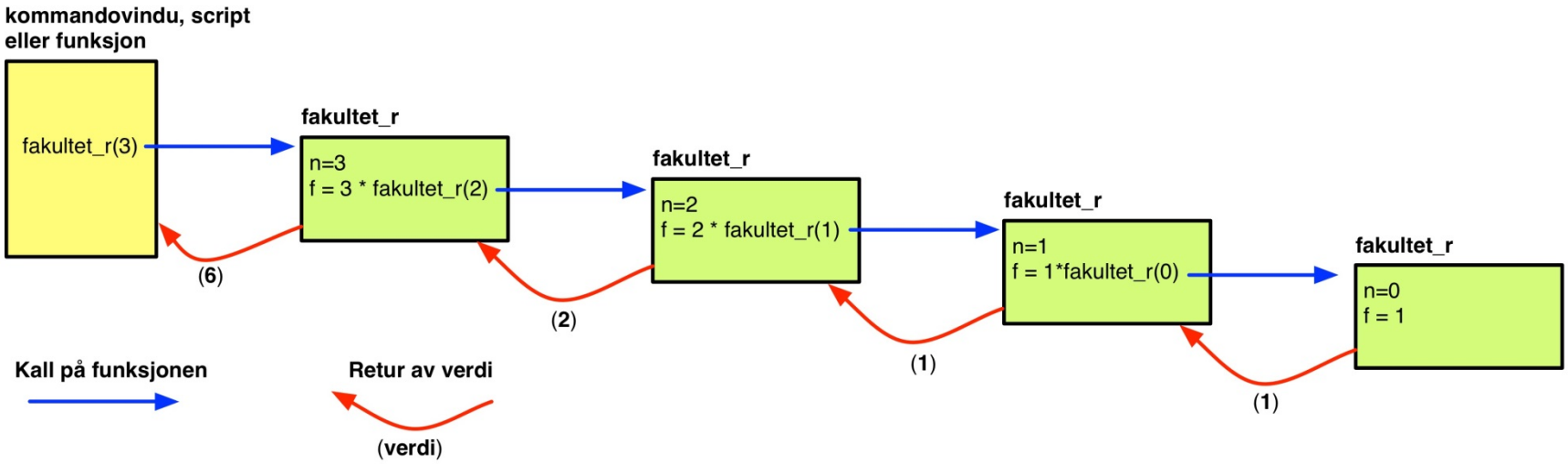
fakultet_r.m

```
function f = fakultet_r(n)
% Fakultet av n, rekursivt

    if n == 0
        f = 1;
    else
        f = n*fakultet_r(n-1);
    end

end % function
```

Eksempel på kjøring: Kall-tre for 3!



Fibonacci-tallene

- 0, 1, 1, 2, 3, 5, 8, 13, 21, …
- Sjekk wikipedia for eksempler på forekomst i naturen.
- Definisjon:
  - f(0) = 0
  - f(1) = 1
  - f(n) = f(n-1) + f(n-2), for n > 1
- Kan programmeres uten rekursjon (iterativt)
  - Lagre en liste med en for-løkke som løper over 2:n
- Rekursiv løsning følger definisjonen direkte

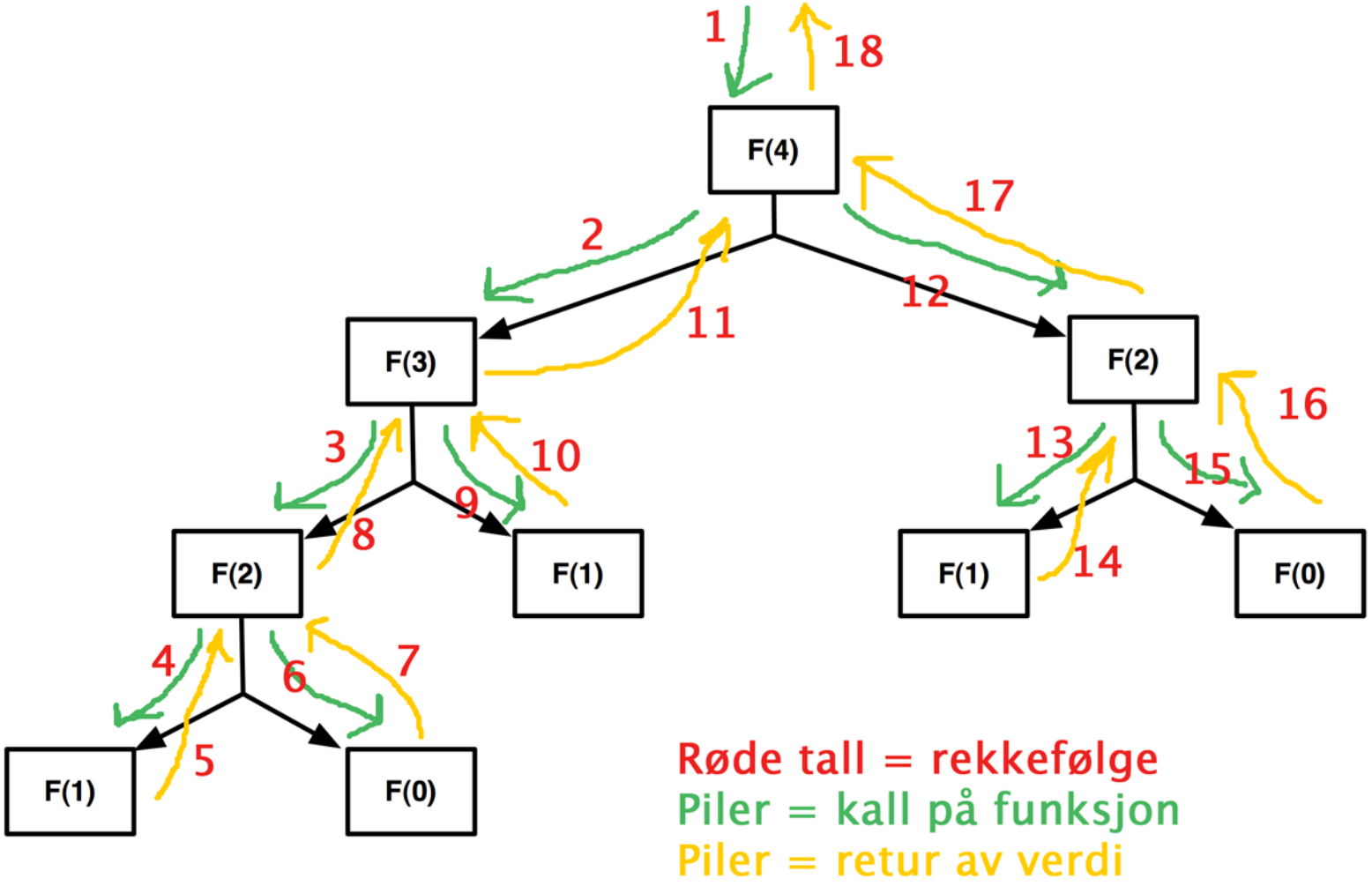
fibonacci.m

```
function f = fibonacci(n)
% Regner ut fibonaccitall n

    if n == 0
        f = 0;
    elseif n == 1
        f = 1;
    else
        f = fibonacci(n-1) + fibonacci(n-2);
    end

end % function
```

Fibonacci(4)



Tidsbruk

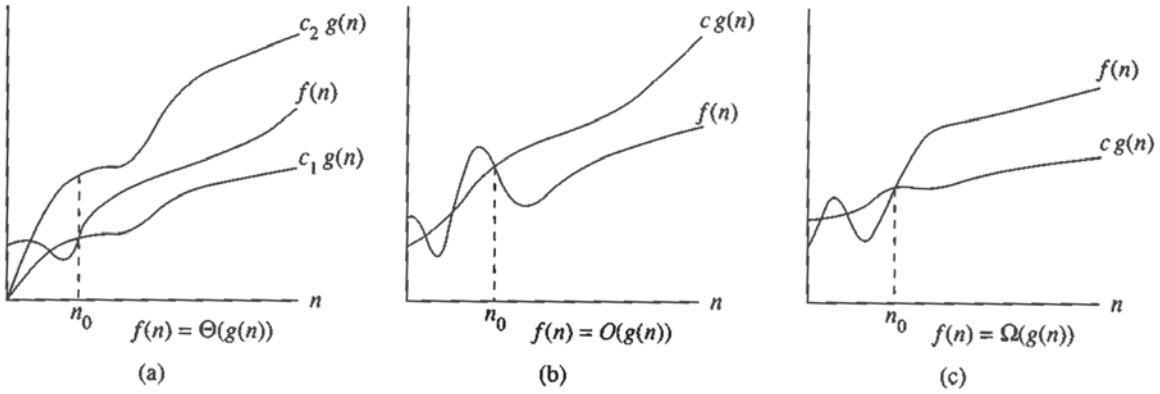
- Har kompleksitet $O(2^N)$!
- Noen eksempler på kjøretider:
  - N=10 0.002 s
  - N=20 0.2 s
  - N=25 1,8 s
  - N=32 51 s
  - N=35 217 s (3 minutter og 37 sekunder)
- Elegant, enkel løsning
- Kostnaden i kjøretid øker VELDIG raskt
- Iterativ løsning er O(N)!
- Funksjonskall er ganske kostbare

O, Ω og Θ fra Algoritme-Teorien

- O(g(n)) er mengden av funksjoner (f(n)) der:
  - Det finnes konstanter c og n_0 slik at $0 \leq f(n) \leq cg(n)$ for alle $n \geq n_0$
  - g(n) er en **asymptotisk øvre grense** for f(n)
- Ω(g(n)) er mengden …
  - Det finnes konstanter c og n_0 slik at $0 \leq cg(n) \leq f(n)$ for alle $n \geq n_0$
  - g(n) er en **asymptotisk nedre grense** for f(n)
- Θ(g(n)): både O(g(n)) og Ω(g(n)) **asymptotisk grense**

O, Ω og Θ (forts.)

- Θ(g(n)): både O(g(n)) og Ω(g(n))
- Det finnes konstanter c_1, c_2 og n_0 slik at $0 \leq c_1 g(n) \leq f(n) \leq c_2 g(n)$ for alle $n \geq n_0$



Kilde: Intro. to Algorithms, Cormen et al., MIT Press 1990

Funksjoners vekst

n	n^2	n^3	2^n	log(n)	nlog(n)
1	1	1	2	0	0
2	4	8	4	1	2
3	9	27	8	2	5
4	16	64	16	2	8
5	25	125	32	2	12
6	36	216	64	3	16
7	49	343	128	3	20
8	64	512	256	3	24
9	81	729	512	3	29
10	100	1000	1024	3	33
100	10000	1000000	1,2677E+30	7	664
1000	1000000	1000000000	1,072E+301	10	9966
10000	100000000	1000000000000	#NUM!	13	132877
100000	10000000000	1000000000000000	#NUM!	17	1660964
1000000	1000000000000	1000000000000000000	#NUM!	20	19931569
10000000	100000000000000	10000000000000000000000	#NUM!	23	232534967
100000000	10000000000000000	1E+24	#NUM!	27	2657542476
1000000000	1000000000000000000	1E+27	#NUM!	30	29897352854
10000000000	100000000000000000000	1E+30	#NUM!	33	332192809489
100000000000	1E+22	1E+33	#NUM!	37	3654120904376
1000000000000	1E+24	1E+36	#NUM!	40	39863137138648
10000000000000	1E+26	1E+39	#NUM!	43	431850652335357
100000000000000	1E+28	1E+42	#NUM!	47	4650699332842310
1000000000000000	1E+30	1E+45	#NUM!	50	49828921423310400
10000000000000000	1E+32	1E+48	#NUM!	53	531508495181978000
100000000000000000	1E+34	1E+51	#NUM!	56	5647277761308520000
1000000000000000000	1E+36	1E+54	#NUM!	60	59794705707972500000
10000000000000000000	1E+38	1E+57	#NUM!	63	631166338028599000000
100000000000000000000	1E+40	1E+60	#NUM!	66	6643856189774730000000
1000000000000000000000	1E+42	1E+63	#NUM!	70	69760489992634600000000
10000000000000000000000	1E+44	1E+66	#NUM!	73	730824180875220000000000
100000000000000000000000	1E+46	1E+69	#NUM!	76	7640434618240930000000000
1000000000000000000000000	1E+48	1E+72	#NUM!	80	797262742772967000000000000
10000000000000000000000000	1E+50	1E+75	#NUM!	83	8304820237218410000000000000
100000000000000000000000000	1E+52	1E+78	#NUM!	86	863701304670714000000000000000
1000000000000000000000000000	1E+54	1E+81	#NUM!	90	89692058619588000000000000000
10000000000000000000000000000	1E+56	1E+84	#NUM!	93	930139866568462000000000000000
100000000000000000000000000000	1E+58	1E+87	#NUM!	96	963359147517335000000000000000

Iterativ løsning: fibonacci_i.m

```
function f = fibonacci_i(n)
% Regner ut Fibonacci-tallet F(n)

    if n == 0
        f = 0;
    elseif n == 1
        f = 1;
    else
        % initialiserer
        minusTo = 0;
        minusEn = 1;
        % regner ut tallet
        for i = 2:n
            f = minusEn + minusTo;
            minusTo = minusEn;
            minusEn = f;
        end % for
    end %if

end % function
```

50 første Fibonacci-tallene
fibonacci_test.m

```
clear, clc

tic % startertidtaking
kolonne = 1;

for tall=0:50

    fprintf('%15d', fibonacci_i(tall) );

    if kolonne == 5
        fprintf('\n'); % ny linje
        kolonne = 1;
    else
        kolonne = kolonne + 1;
    end;
end

fprintf('\n');
toc % avslutter tidtaking
```

Utskrift og kjøretid

```
           0           1           1           2           3
           5           8          13          21          34
          55          89         144         233         377
         610         987        1597        2584        4181
        6765       10946       17711       28657       46368
       75025      121393      196418      317811      514229
      832040     1346269     2178309     3524578     5702887
     9227465    14930352    24157817    39088169    63245986
   102334155   165580141   267914296   433494437   701408733
  1134903170  1836311903  2971215073  4807526976  7778742049
 12586269025
Elapsed time is 0.002617 seconds.
>>
```

- Iterativ eller rekursiv?
  - Et enkelt valg i dette tilfellet

Tekna

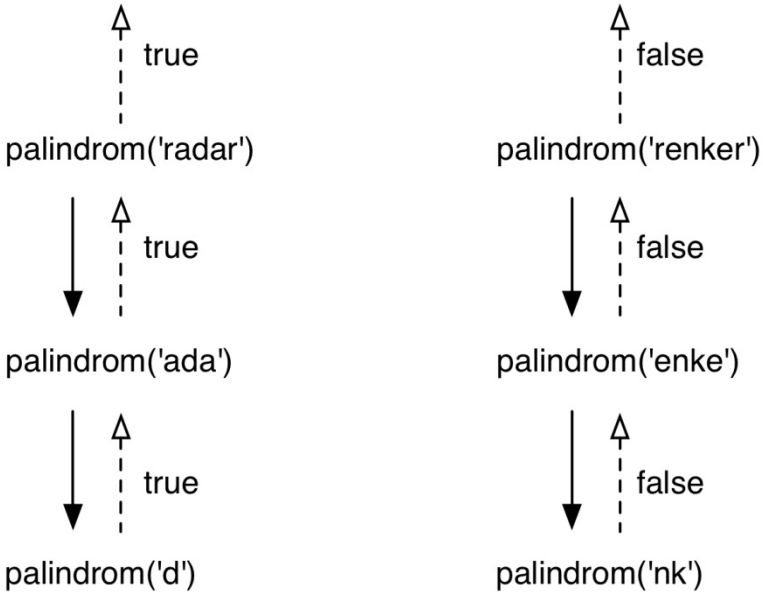
Kunnskap for en bedre verden

Blæsting i Pausen:
Tekna holder eksamensrettet kræsjkurs i matlab

- Dato: 23.11.2016 (Onsdag)
- Sted: R1
- Klokkeslett: 18:00-21:00
- Kursholder: Chris-Mikael Rom Hansen (innleid av Tekna)
- Kun for Teknamedlemmer, men Tekna har gratis innmelding ut året!
- Påmelding: https://www.tekna.no/kurs/krasjkurs-i-matlab-32792/

Palindrom?

- Def: Tekststreng som kan leses likt begge veier
  - Otto, Radar, Agnes i senga, …
- Stopptilfelle: 0 eller 1 tegn er et palindrom
- Test flanke-tegnene
  - **Ulike**: Ikke et palindrom
  - **Like**: Sjekk om teksten uten flanke-tegnene er et palindrom



palindrom.m

```
function p = palindrom(tekst)
% sjekker om tekst er et palindrom (leses likt begge veier)

    lengde = length(tekst);
    if lengde < 2
        p = true;
    elseif upper(tekst(1)) == upper(tekst(lengde))
        p = palindrom( tekst(2:lengde-1) );
    else
        p = false;
    end

end % function
```

Bruker upper-funksjonen for å gjøre
tegnsammenligningene uavhengig av
små/store bokstaver, slik at f. eks. ’RADar’ blir et palindrom

BoB

- https://vimeo.com/44024588