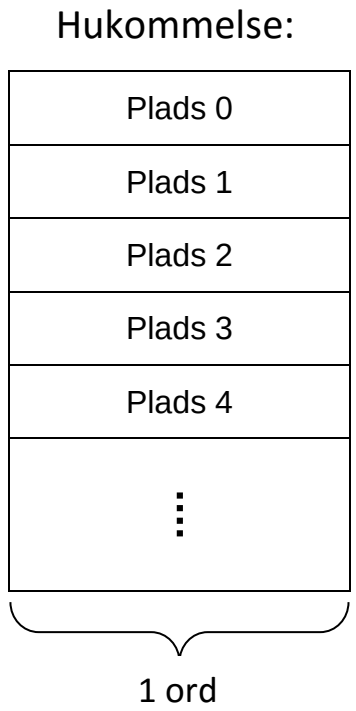
1.1-2 introduktion til pseudokode
Noter CLRS 1-2

- RAM modellen
- Algoritmer og datastrukturer
- Pseudokode
- Tidsanalyse

Disse noter er stærkt inspireret af noter af Philip Bille og Inge Li Gørtz til kuset Algoritmer og Datastrukturer, på DTU,
http://www2.compute.dtu.dk/courses/02105+02326/2015/#generelinfo

RAM model

- Abstrakt model af computere
- Hukommelse modelleret som en stor tabel af ord



Operationer:

Følgende operationer kan udføres i konstant tid:

- Simple aritmetiske operationer: +, -, *, /
- Simple binære operationer tager: &, |, ^, ~
- Læse et ord fra hukommelsen: A[i] (læser ord nummer i fra tabellen A)
- Gemme et ord i hukommelsen: A[i] = x (gemmer værdien x i A[i])

RAM model (fortsat)

Hvad er et ord?

- Et antal bits (*w* bits), der kan indeholde ”hvad som helst”
- Antag at basale objekter kan være i et ord (tal, tegn, mv.)
- Et ord kan ikke indeholde store objekter! (tekststrenge, tabeller, mv.)
  - Sådanne objekter laves ved at sammensætte flere ord!

Algoritmer og datastrukturer

- Algoritmisk problem. Præcist defineret relation mellem input og output.

- Algoritme. Metode til at løse et algoritmisk problem.
  - Beskrevet i diskrete og entydige skridt. (i en bestemt model, f.eks. RAM)
  - Matematisk abstraktion af program.

- Datastruktur. Metode til at organise data så det kan søges i eller manipuleres.

Eksempel 1: Sum

Beregn summen af to tal:

- Input: To tal *a* og *b*
- Output: Et tal *c* således at *c* = *a*+*b*

***A*lgoritme:**

- Returnér *a*+*b* (kan udregnes nemt i f.eks. RAM modellen)

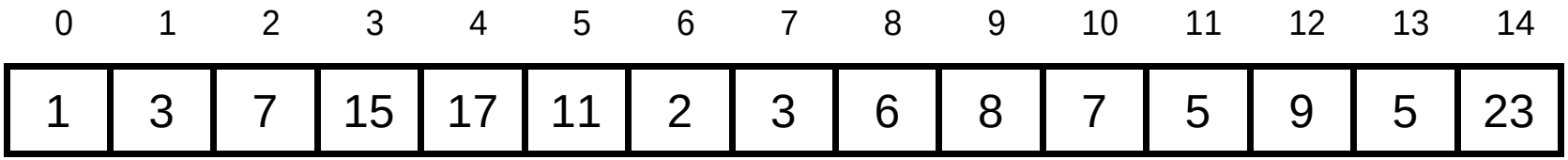
Eksempel 2: Maksimalt tal

Givet en tabel A[0..n-1], find et tal i, således at A[i] er maksimalt.

- Input. Tabel A[0..n-1].
- Output. Et tal i, $0 \le i < n$, så $A[i] \ge A[j]$ for alle indgange $j \ne i$.

Algoritme:

- Gennemløb A og vedligehold indeks af nuværende maksimale indgang.
- Returner indeks.



Beskrivelse af algoritmer

- Naturligt sprog.
  - Gennemløb A og vedligehold indeks af nuværende maksimale indgang.
  - Returner indeks.
- Program.

```
public static int findMax(int[] A, int n){
   int max = 0;
   for(i = 0; i < n; i++)
        if (A[i] > A[max]) max = i;
   return max;
}
```

- Pseudokode.

```
FINDMAX(A, n):
    max = 0
    for i = 0 to n-1
        if (A[i] > A[max]) max = i
    return max
```

Pseudokode

Hvad er pseudokode?

- Lige som ”rigtig” kode, men uden besværet!
- Giver en måde til samtidig at beskrive en algoritme præcist men stadig abstrahere fra unødvendige detaljer.

Hvad kan man med pseudokode?

- Definere funktioner:

```
1  GetANumber():
2     Return 8
```

Bemærk:
1 og 2 er linjenumre indsat for læsbarhed

- Variable:

```
1 x = 5
2 y = 8
```

- Aritmetik:

```
1 x = 5
2 y = 8 + 7
3 z = y * x
```

Pseudokode (fortsat)

- Funktionskald:

```
1 X = GETANUMBER()
2 PRINT(X)
```

- Kommentarer

```
1 X = GETANUMBER()   // x is now 8
2 PRINT(X)           // prints 8
```

- Tabeller:

```
1 A = array of size 100
2 Initialize A to all zeroes
3 A[5] = 1
4 A[8] = 41
5 PRINT(SUM(A)) //prints 42
```

- Tekst:

```
1 X = ”Hej verden!”
2 PRINT(X)
```

Bemærk:
I linje 1 og 2 bruger vi blot en engelsk beskrivelse. Således skal vi ikke tænke på hvordan vores tabel faktisk nulstilles.
I linje 5 antager vi at der findes en funktion ved navn SUM*. Generelt er dette okay for funktioner der virker ”rimelige”.*

Pseudokode (programflow)

- Pseudokode læses som udgangspunkt linje for linje
- Eksempel med programmet fra forrige slide: Først laves en tabel, dernæst sættes hele tabellen til 0. Dernæst sætter vi først A[5] til 1 og dernæst A[8] til 41. Til sidst udskriver vi summen af alle indgangene i tabellen.
- Programflow kan dog ændres!

If-sætning

- Deler koden op i blokke der udføres på en betingelse:

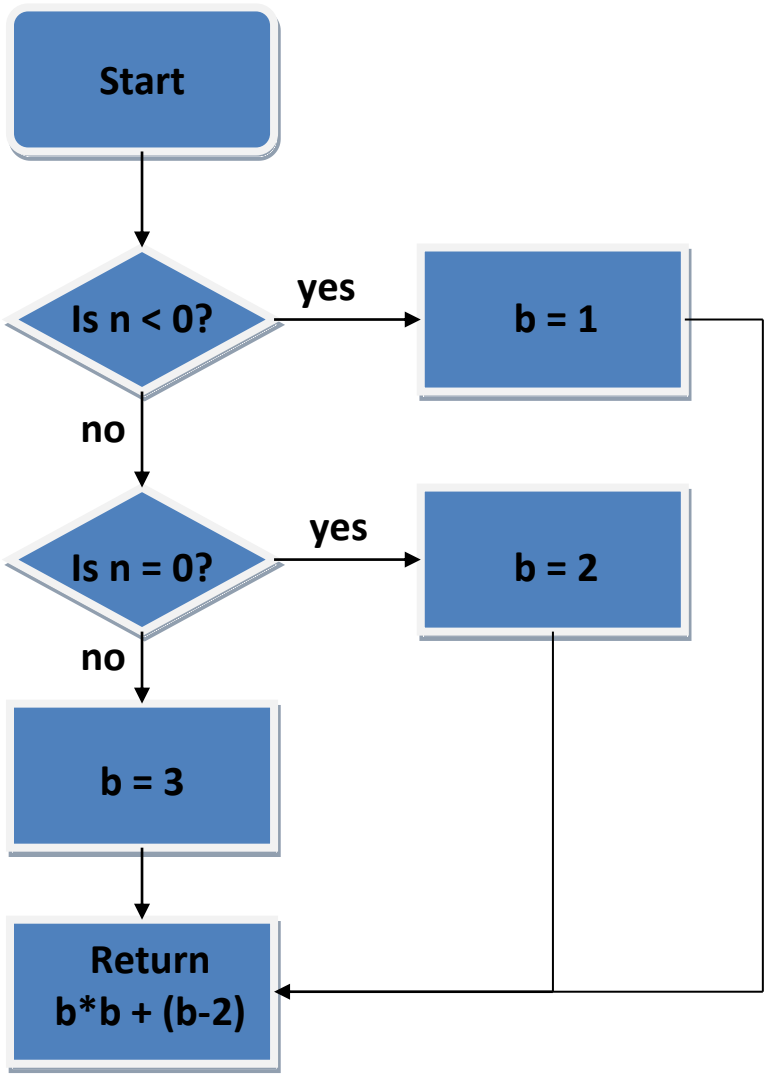
```
1  LIGEULIGE(n):
2    If n is even:
3      Return "Lige"
4    Else:
5      Return "Ulige"
```

Bemærk:
Vi bruger indentering af linjerne til at angive hvad der hører til if-sætningen (indikeret af farverne): Linje 3 udføres kun, hvis n er et lige tal, og linje 5 kun hvis det ikke er tilfældet.

Pseudokode (programflow fortsat)

- If-sætning struktur kan være mere kompliceret
- Og kan illustreres med flowcharts

```
1  UdregnNoget(n):
2    b = 0
3    If n < 0:
4      b = 1
5    Elseif n == 0:
6      b = 2
7    Else:
8      b = 3
9    Return b * b + (b - 2)
```



Øvelse: Hvad returnerer `UdregnNoget(8)`?

Pseudokode (programflow fortsat)

Har set if-sætninger, men de har begrænsninger.

Løkker

- Vi vil ofte gøre det samme mange gange!
- To slags løkker vi vil bruge: `while` og `for`
- De to programmer herunder udskriver begge tallene 0, 1, 2, ..., 9, 15

```
1  i = 0
2  While i < 10:
3     PRINT(i)
4     i = i + 1
5  PRINT(15)
```

```
1  For i = 0 to 9:
2     PRINT(i)
3  PRINT(15)
```

Bemærk: 15 bliver kun udskrevet én gang, da den sidste linje ikke er en del af løkken! Dette kan vi se på indenteringen ligesom med if-else-sætningerne.

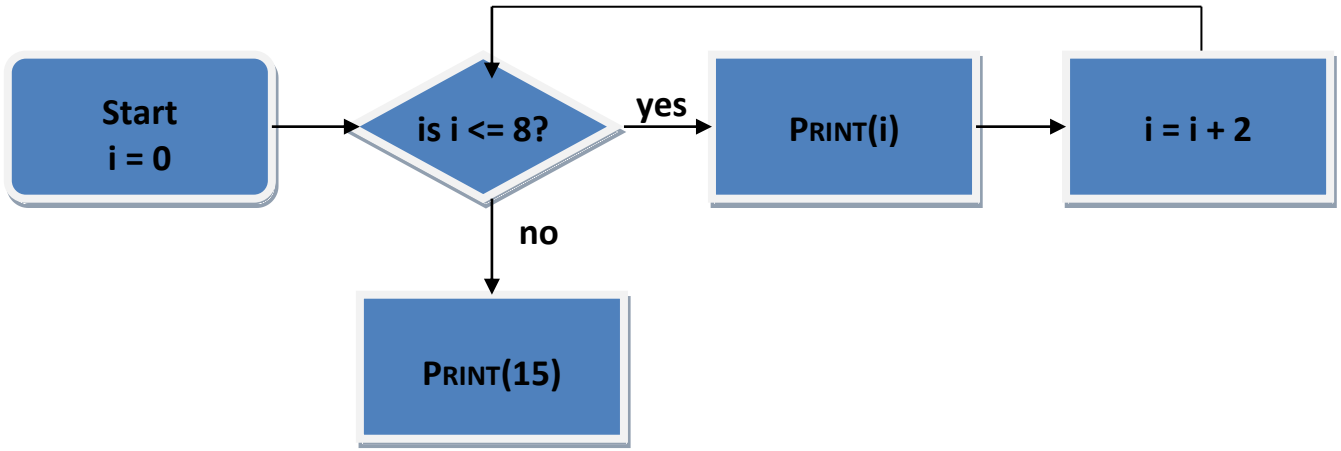
Pseudokode (programflow fortsat)

- I while-løkken kan vi selv bestemme hvordan vi manipulerer *i*.
- Vi må også beskrive hvordan *i* ændres i en for-løkke. Programmerne herunder udskriver begge tallene 0, 2, 4, 6, 8, 15

```
1  i = 0
2  While i < 10:
3    PRINT(i)
4     i = i + 2
5  PRINT(15)
```

```
1  For i = 0 to 8 by 2:
2    PRINT(i)
3  PRINT(15)
```

- Vi kan illustrere for-løkken herover med et flowchart:



Pseudokode (flere eksempler)

- Vi kan bruge løkker og if-sætninger inde i hinanden. Vi kan også lave løkker over tabeller. Følgende kode udskriver summen af alle lige tal i en tabel:

```
1  SumEven(A):
2    S = 0
3    For i = 0 to length(A) - 1:
4      If A[i] is even:
5        S = S + A[i]
6    Return A[i]
```

- Med disse simple udtryk kan vi skrive pseudokode, der beskriver meget sofistikerede algoritmer!

Husk: Der er mange forskellige måder at skrive pseudokode på. Det vigtigste er at man er i stand til at formidle en algoritme! Selve formatet er ikke det vigtige!

I vil derfor se mange forskellige måder at skrive det samme på. Nogle gange er ting med stort, nogle gange med fed, nogle gange er der kolonner, nogle gange parenteser.

Teoretisk analyse

- Køretid/tidskompleksitet.
  - T(n) = antallet af skridt som algoritmen udfører på et input af størrelse n.
- Skridt.
  - Læsning/skrivning til hukommelse (x := y, A[i], i = i + 1, ...)
  - Aritmetiske/boolske operationer (+, -, *, /, %, &&, ||, &, |, ^, ~)
  - Sammenligninger (<, >, =<, =>, =, ≠)
  - Programflow (if-then-else, while, for, goto, funktionskald, ..)
- Værstefaldstidskompleksitet (worst-case complexity).
  - Interesseret (næsten altid) i værstefaldstidskompleksitet = maksimal køretid over alle input af størrelse n.

Teoretisk analyse

- Køretid. Hvad er køretiden T(n) for algoritmen?

```
FINDMAX(A, n)
    max = 0
    For i = 0 to n-1:
        If (A[i] > A[max]): max = i
    Return max
```

c_4

$n \cdot c_5$

c_6

$$T(n) = c_4 + n \cdot c_5 + c_6 = \Theta(n)$$

- T(n) er en *lineær funktion* af n: T(n) = an + b, for passende konstanter a og b.
- Asymptotisk notation. T(n) = Θ(n)