Ugeseddel 1

Bemærk det er en foreløbig plan. Der kan forekomme ændringer.

Litteratur:

Uge 1 vil vi primært bruge på:

- Velkomstnote
- Få overblik over kursets side på Absalon og FB.
- CLRS preface
- CLRS Part I Foundations. Introduction. Side 3-4
- CLRS kapitel 1
- Visse begreber fra uge 1 er mere formelt defineret i CLRS kapitel 2, som gennemgås næste uge.
- Kursusinformation
- https://b.socrative.com/

Noter:

- 1.1-2 introduktion til pseudokode Noter CLRS 1-2
- 1.3 toppunkter CLRS intro

Mål for ugen:

- Forstå Pseudokode: if, else, while, variable, tabeller, osv.
- Forstå RAM maskinmodellen med elementær tidsanalyse
- Stifte bekendtskab med rekursion, toppunkter,…

Forelæsninger (og spørgertimer):

Tirsdag:	Algoritme, input/output, pseudokode, m.m.
	Instruktorerne gennemgår mere om pseudokode (if/else, for, while osv.)
	Spørgetimen, vil have mest fokus på overordnet ting om kurset
Torsdag:	Mere om pseudokode – inkl. rekursion
	Toppunkter (se note)
	Spørgetimen vil have mest fokus på det faglige

Opgaver tirsdag:

Opgaver markeret med (ekstra) kan gemmes til resten af opgaverne er regnet. I bunden af ugesedlen finder du nogle svære opgaver markeret med stjerne hvis du mangler udfordring eller er færdig med dagens opgaver.

Pas på: Opgaver markeret med "*" er svære, "" er meget svære, og "***" har du ikke en chance for at løse.**

1 Find relevant information

1.1 Login på kursets hjemmeside på Absalon:
https://absalon.ku.dk/courses/2578

1.2 Anmod om medlemskab i FB gruppen:
https://www.facebook.com/groups/DMA16/

2 Hvor finder vi algoritmer og hvad bruges de til

2.1 Nævn eksempler på algoritmer i har brugt.

2.2 Nævn eksempler på hvor algoritmer bliver brugt (eller burde være bedre)

2.3 Ud over tid og plads hvad bruger vi så algoritmer til at minimere/maksimere?

2.4 Hvilke algoritmer skal der bruges for at lave en musikafspilningstjeneste som f.eks. Spotify?

3 Pseudokode del A: Betragt funktionerne i figur 1.

3.1 Hvad returnerer funktionerne A1 og A2

3.2 Hvad returnerer funktionerne A3, A4, og A5 for n=4, n=10 og n=100?

3.3 Ændr funktionerne A1-5 (skriv pseudokode) så de returnerer en værdi der er dobbelt så stor

4 Pseudokode del B: Betragt funktionerne i figur 2.

4.1 Hvad returnerer funktionerne for n=4, n= 10 og n=100. Og udtryk hvad de returnerer som funktion af n. (fx *f(n) = 3n*).

4.2 Ændr disse funktioner (skriv pseudokode) så de returnerer en værdi der er dobbelt så stor

5 Løkker Hvad returnerer funktionerne `loop0`, `loop1`, `loop2`, `loop3`, `loop4` og `loop5` i figur 3 og 4 når

5.1 n= 4?

5.2 n= 10?

5.3 n= 1000?

5.4 Udtrykt som en funktion af n? Bemærk at 1+2+..+n= n * (n+1) /2

6 Mere løkker (ekstra) Betragt funktionerne loop6 og loop7 i figur 4

6.1 Hvad returnerer funktionerne for n=4 og n=16?

6.2 Beskriv som funktion af n, hvilken matematisk funktion disse funktioner er relateret til**.**

7 Lav selv funktioner (ekstra)

7.1 Lav 3 funktioner:

- `f1(x,y)` der returnerer summen af x og y.
- `f2(x,y)` der returnerer gennemsnittet af x.
- `f3(x,y)` der returnerer x^y (der må kun bruges gange).

7.2 Lav en funktion `f(x,y,z)` der returnerer 1 såfremt to af tallene summer til det tredje og ellers returnerer 0

Opgaver torsdag formiddag:

1. **Tabeller** Betragt funktionerne i figur 5. Læg mærke til at n er antallet af elementer.
   - **1.1** Hvad returnerer T1([2,3,1,4,3],5) og T1([0,-1,1],3)? Beskriv mere generelt hvad T1 gør.
   - **1.2** Hvad returnerer T2([2,3,4,2],4,2) og T2([2,3,4,2],4,3)? Beskriv mere generelt hvad T2 gør.
   - **1.3** Hvad returnerer T3([7,10,12,2],4)? Beskriv mere generelt hvad T3 gør.
2. **Lav selv tabelfunktioner**
   - **2.1** Lav en funktion `mindste(A,n)` der returnerer det mindste tal i en tabel A med n tal
   - **2.2** [*] Lav en funktion der bestemmer om to tal fra tabellen summer til 100
3. **Køretider** Løs følgende opgaver.
   - **3.1** CLRS 1.2-2.

3.2 CLRS 1.2-3.
3.3 CLRS 1-1 (udfyld kun tabellen for log n, n og 2^n).

4. **Zombieduellering** Du har en hær af *n* hjernetomme zombier. Du vil gerne finde den stærkeste og svageste zombie blandt gruppen. Ved at sætte to zombier sammen i et bur med en luns kød kan du hurtigt afgøre hvilken af de to er stærkest. Desværre slider det på zombier at slås, så du vil gerne minimere antallet af dueller. Løs følgende opgaver.
   4.1 Vis hvordan du finder den stærkeste zombie med højst $n - 1$ dueller.
   4.2 [*] Vis hvordan du finder både den stærkeste og svageste zombie med højst $3n/2$ dueller.
   4.3 [**] Vis hvordan du finder både den stærkeste og næststærkeste zombie med højst $n + \log_2 n$ dueller.

Opgaver torsdag eftermiddag:

1. **Rekursion og iteration** En funktion er *rekursiv* hvis den kalder sig selv. F.eks. er funktionen `f(A,n)` i figur 6 rekursiv. Løs følgende opgaver:
   1.1 Hvad beregner `f(A,n)` hvis `A` er en tabel af heltal af længde `n`?
   1.2 Omskriv `f(A,n)` til at være *iterativ* i stedet, dvs. lav en funktion der gør det samme som `f(A,n)` uden at kalde sig selv.

2. **Mere rekursion** Betragt funktionerne r1, r2, r3, r4, r5 og r6 i figur 6
   2.1 Hvad beregner funktionerne for n= 2, n=4 og n=5?
   2.2 Udtryk som funktion af n hvad funktionerne beregner
   2.3 Omskriv funktionerne så de bliver iterative

3. **Lineartrose** Du har netop ansat 128 programmører til din nyopstartede high-tech virksomhed. Desværre lider en af dem af den frygtede *lineartrose* sygdom, der får alle i nærheden af vedkommende til at skrive langsomme programmer. For at finde den syge programmør har du lejet et særligt kammer, du kan bruge til at afprøve om en delmængde af dine programmører har en syg iblandt sig. Det er dyrt at leje lokalet og processen med at teste gruppen er meget omstændigt (lange og komplicerede programmeringstest er nødvendige). Derfor vil du gerne minimere antallet af gange du skal bruge lokalet til at finde den syge. Løs følgende opgaver.
   3.1 Vis at du kan finde den syge programmør med højst 7 tests.
   3.2 Hvor mange test kan du klare dig med `generelt` hvis du har *n* programmører?
   3.3 [*] Antag du lejer $k > 1$ kamre som du kan bruge til at teste k grupper af programmører samtidig. Hvor mange *runder* af tests kan du klare dig med til at finde den syge programmør? I hver runde kan du teste k grupper parallelt.

4. **Find toppunkter**
   Indgang A[i] i en tabel er et toppunkt hvis A[i] er mindst ligeså stort som dets naboer:
   - A[i] toppunkt hvis A[i-1] ≤ A[i] ≥ A[i+1] for 1 <= i <= n-2
   - Specielt har vi A[0] toppunkt A[0] ≥ A[1] og A[n-1] er toppunkt hvis A[n-2] ≤ A[n-1]. (Tænk A[-1] = A[n] = -∞).

   Lad $A = [2, 1, 3, 7, 3, 11, 1, 5, 7, 10]$ være en tabel. Løs følgende opgaver.
   4.1 Angiv alle toppunkter i *A*.
   4.2 Angiv hvilke toppunkter de to lineærtidsalgoritmer finder i figur 7 og 8.
   4.3 Angiv sekvensen af rekursive kald, som den rekursive algoritme producerer. Se figur 9. Antag først at rekursionen fortsætter med venstre halvdel af tabellen hvis der er mulighed for at gå begge veje. Angiv derefter alle mulige sekvenser af rekursive kald der kan opnås ved frit valg når man kan gå begge veje. En iterativ version findes i figur 10.

Pseudokode Part A

```
A1()
   x = 1
   return x


A2()
    x = 1
    x = x + 1
    return x


A3(n)
    x = n
    return x


A4(n)
    If n > 10 return 1
     else return 0


A5(n)
     x = 0
    For i = 1 to n
         x = x+2
    return x
```

Figur 1:

Pseudokode Part B

```
B1(n)
   x = n
   x = x + x
   return x


B2(n)
    If n > 10
      x = 1
    Elseif n == 10
      x = 0
    Else x = -1
    return x


B3(n)
    x = n
    y = 2 * n
    x = 2*x + 3*y
    return x


B4(n)
    x = n
    y = 2 * x
    x = 2 * y
    return x
```

Figur 2

Pseudokode Løkker 1

```
loop0(n)
    x = 0
    For i = 1 to n
        x = x+2
    return x


loop1(n)
   x = 0
   for i = 1 to n
          for j = 1 to n
                 x = x +1
   return x


loop2(n)
   x = 0
   for i = 1 to n
          x = x + 1
   for j = 0 to n
          x = x +1
   return x


loop3(n)
   x = 0
   for i = 1 to n
      if i >= n-1
           for j = 1 to n
             x= x+1
   return x
```

Figur 3

Pseudokode løkker 2

```
loop4(n)
   x = 0
   for i = 1 to n
      for j = i to n
           x = x + 1
   return x


loop5(n)
   x = 0
   while n > 0
     x = x +1
     n = n -1
   return x
```

loop6(n) // antag n = 2^k for et heltal k

```
   x = 0
   while n > 1
     x = x + 1
     n = n / 2
   return x


loop7(n)
   x = 1
   while n > 0
     x = x * 2
     n = n -1
   return x
```

Figur 4

Tabeller

```
T1(A,n)
  x = 0
  for i = 0 to n-1
    x = x + A[i]
  return x


  T2(A,n,y)
    x = 0
    For i = 0 to n-1
      If A[i]==y
        x = x +1
    return x


  T3(A,n)
    For i=0 to n-2
      if A[i] > A[i+1]
        temp = A[i+1]
        A[i+1] = A[i]
        A[i] = temp
```

Figur 5

Rekursion

```
R1(n)
   If n > 0 return R1(n-1)
   else return 0

R2(n)
    If n > 1 return 2+R2(n-1)
    Else return 2

R3(n)
   if n == 1 return 1
   else return n+R3(n-1)

R4(n)
   if n <= 1 return 0
   else return 1+R4(n/10)

R5(n)
   if n == 1 return 1
   else return 2*R5(n-1)

R6(n)
   if n == 1 return 1
   else return R6(n-1)+R6(n-1)
```

f(A,n) // Tabel A med n elementer

```
    if n == 0 return 0
    else return f(A, n - 1) + A[n-1]
```

Figur 6

Toppunktsalgoitmer

```
TOPPUNKT1(A, n)
      if A[0] ≥ A[1] return 0
      for i = 1 to n-2
        if A[i-1] ≤ A[i] ≥ A[i+1]
          return i
```

Figur 7

```
Toppunkt2(A, n)
      max = 0
      for i = 0 to n-1
        if A[i] > A[max]
          max = i
      return max
```

Figur 8

```
TOPPUNKT3(A,i,j)// rekursiv version
          m = rundop((i+j)/2))
      if A[m] ≥ naboer
        return m
      elseif A[m-1] > A[m]
        return TOPPUNKT3(A,i,m-1)
      elseif A[m] < A[m+1]
        return TOPPUNKT3(A,m+1,j)
```

Figur 9

```
TOPPUNKT3(A,n)// iterativ version
          i=0
          j=n-1
          while i < j
              m = ⌊ (i+j)/2)⌋
              if A[m] ≥ naboer
                 return m
               elseif A[m-1] > A[m]
                  j = m-1
               elseif A[m] < A[m+1]
                  i = m +1
```

Figur 10

Bemærkninger:

Nogle opgaver er stærkt inspireret af opgaver stillet af Philip Bille og Inge Li Gørtz i kurset Algoritmer og Datastrukturer, på DTU, http://www2.compute.dtu.dk/courses/02105+02326/2015/#generelinfo.

Bemærk "CLRS preface" skriver at det antages at de studerende kender til simple ting vedr. programmering som lister, tabeller osv. På kurset antager vi: Kompetencer svarende til at kurset "Programmering og problemløsning " følges senest samtidigt. Vi vil derfor bruge tid til at uddybe diverse begreber vedr. programmering. Bogen antager endvidere at den studerende f.eks. er kendt med visse matematiske begreber, f.eks. induktion, som først introduceres senere på kurset.