DMA ugeseddel 1

Litteratur

Uge 1 vil vi primært bruge på:

- Velkomst og kursusinformation.
- Få overblik over kursussiden på Absalon.
- CLRS Preface, side xii–xix.
- CLRS Part I Foundations. Introduction og kapitel 1, side 3–14.

Bemærk: Der står i "CLRS Preface" at det antages at de studerende kender til simple ting vedr. programmering som lister, arrays osv. I DMA 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 er kendt med visse matematiske begreber, f.eks. induktion, som først introduceres senere på kurset.

Mål for ugen

- Kende til pseudokode: if, else, for, while, variable, arrays, osv.
- Forstå RAM-maskinmodellen med elementær tidsanalyse.
- Stifte bekendtskab med simple algoritmer og rekursion.

Plan for ugen

- Mandag: Velkomst, algoritme, input/output, pseudokode.
- Tirsdag: Formaliteter om kurset, RAM-modellen, mere pseudokode.
- Fredag: Rekursion.

Opgaver til mandag

Opgaver markeret med (ekstra) kan gemmes til resten af opgaverne er regnet.

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

1. **Find relevant information.**

   Log in på kursussiden `https://absalon.ku.dk/courses/51660`, og brug noget tid på at danne dig et overblik over hvilken information siden indeholder.

2. Hvor finder vi algoritmer og hvad bruges de til?

   (a) Nævn eksempler på algoritmer I har brugt (fx til at udføre en konkret praktisk opgave).

   (b) Nævn eksempler på hvor algoritmer bliver brugt, eller hvor algoritmer burde være bedre end dem man bruger nu.

   (c) Ud over beregningstid og hukommelsesforbrug, hvad bruger vi så algoritmer til at minimere/maksimere?

   (d) Hvilke algoritmer skal der bruges for at lave en musikafspilningstjeneste som fx Spotify? (Tænk fx på søgefunktion, anbefaling af nye numre, datakompression, shuffle-funktion, etc.)

3. **Pseudokode.** Betragt funktionerne A1–A4 nedenfor.

   (a) Hvad returnerer funktionerne A1 og A2?

   (b) Hvad returnerer funktionerne A3 og A4 når $n = 4$, $n = 10$ og $n = 100$?

   (c) Ændr funktionerne A1–A4 (skriv pseudokode) så de returnerer en værdi der er dobbelt så stor.

```
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
```

4. **Mere pseudokode.** Betragt funktionerne B1–B4 nedenfor.

   (a) Hvad returnerer funktionerne for $n = 4$, $n = 10$ og $n = 100$.

   (b) Udtryk den returnerede værdi som funktion af n (fx $f(n) = 3n$).

   (c) Ændr disse funktioner (skriv pseudokode) så de returnerer en værdi der er halvt så stor.

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

```
B2(n)
  if n > 10
    x = 1
  else if 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 n + x + y
```

5. **Løkker.** Se funktionerne Loop0–Loop6 nedenfor.

   (a) Hvad returnerer funktionerne Loop0-Loop3 når $n = 4$, $n = 10$, $n = 1000$?

   (b) Loop2 og Loop3 ser umiddelbart ens ud: De er skrevet med de samme tegn i den samme rækkefølge. Forklar hvorfor indrykningen alligevel gør en forskel for hvad de returnerer.

   (c) Udtryk den returnerede værdi af Loop0-Loop3 som en funktion af n.

   (d) Gør det samme som i opgave (a) og (c) for Loop4-Loop6. *Hint:* Du kan få brug for *sumformlen*:

$$1 + 2 + \ldots + n = \frac{n \cdot (n+1)}{2}.$$

```
Loop0(n)
  x = 0
  for i = 0 to n − 1
    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 = 1 to n
    x = x + 1
  return x
```

```
Loop3(n)
  x = 0
  for i = 1 to n
    x = x + 1
    for j = 1 to n
      x = x + 1
  return x
```

```
Loop4(n)
  x = 0
  for i = 1 to n
    if i ≥ n − 1
      for j = 1 to n
        x = x + 1
  return x
```

```
Loop5(n)
  x = 0
  for i = 1 to n
    for j = i to n
      x = x + 1
  return x
```

```
Loop6(n)
  x = 0
  while n > 0
    x = x + 1
    n = n − 1
  return x
```

6. **Flere løkker (ekstra).** Se funktionerne Loop7–Loop9 nedenfor.

(a) Hvad returnerer Loop7–Loop8 når $n = 4$, $n = 10$, $n = 1000$?

(b) Udtryk den returnerede værdi af Loop7 og Loop8 som en funktion af n.

(c) Lav en liste over hvad Loop9 returnerer for hver værdi $n = 1, \ldots, 10$.

(d) (*) Argumentér for at hvis Loop9 returnerer en værdi x, så er $x \geq \log_2 n$. *Bonus-info: Collatz-formodningen* siger at Loop8 vil returnere en værdi for alle heltal $n \geq 1$, men man ved faktisk ikke, om der findes en værdi af n, som får løkken til at blive ved med at køre uendeligt længe, altså en såkaldt *uendelig løkke*. Det er et meget berømt spørgsmål. Læs evt. mere her `https://en.wikipedia.org/wiki/Collatz_conjecture`.

(e) (*) Hvis linje 8 i Loop9 ændres til $n = 3 \cdot n - 1$, fås da en funktion som altid returnerer?

```
Loop7(n) // antag n = 2^k for et heltal k
  x = 0
  while n > 1
    x = x + 1
    n = n/2
  return x
```

```
Loop8(n)
  x = 1
  while n > 0
    x = 2 · x
    n = n − 1
  return x
```

```
Loop9(n)
  x = 0
  while n > 1
    x = x + 1
    if n is even
      n = n/2
    else
      n = 3 · n + 1
  return x
```

7. **Lav selv funktioner (ekstra).** Opskriv pseudokode for funktioner specificeret som følger.

(a) f1(x,y), der returnerer summen af x og y.

(b) f2(x,y), der returnerer gennemsnittet af x og y.

(c) f3(x,y), der returnerer x^y, når det antages at y er et ikke-negativt heltal (af regneoperationer må der må kun bruges plus og gange).

(d) f4(x,y,z), der returnerer 1 såfremt to af tallene summer til det tredje og ellers returnerer 0.

Opgaver til tirsdag

1. **Arrays.** Betragt funktionerne T1–T3 nedenfor. I hvert tilfælde antages det at n er antallet af elementer i arrayet A.

   (a) Hvad returnerer T1([2,3,1,4,3],5) og T1([0,-1,1],3)? Beskriv mere generelt hvad T1 gør.

   (b) Hvad returnerer T2([2,3,4,2],4,2) og T2([2,3,4,2],4,3)? Beskriv mere generelt hvad T2 gør.

   (c) Hvad returnerer T3([7,10,12,2],4)? Beskriv mere generelt hvad T3 gør.

   (d) Hvad returnerer T4([7,10,12,2],4)? Beskriv mere generelt hvad T4 gør.

   (e) Argumentér for at linjen $B[i] = B[i] + A[j]$ i T4 udføres $1 + 2 + \ldots + n = \frac{n(n+1)}{2} = n^2/2 + n/2$ gange, så funktionen laver flere end $0.5 \cdot n^2$ operationer.

   (f) (*) Skriv pseudokode til en funktion der returnerer det samme som T4 for alle input, men som kun udfører $c \cdot n$ operationer, for en konstant $c > 0$.

```
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]
      t = A[i + 1]
      A[i + 1] = A[i]
      A[i] = t
  return A
```

```
T4(A,n)
  let B be a new array of size n
  for i = 0 to n − 1
    B[i] = 0
    for j = 0 to i
      B[i] = B[i] + A[j]
  return B
```

2. **Lav selv tabelfunktioner**

   (a) Lav en funktion mindste(A,n), der returnerer det mindste tal i en tabel A med n tal.

   (b) (*) Lav en funktion Sum100(A,n), der bestemmer om to tal fra A summer til 100.

   (c) (**) Antag nu at A er et *sorteret* array. Lav en funktion Sum100Sorted(A,n), der bestemmer om to tal fra A summer til 100, sådan at antallet af operationer som funktionen udfører er højst $c \cdot n$ for en konstant $c > 0$.

3. **Køretider.** Løs følgende opgaver.

   (a) CLRS 1.2-2. *Bemærk:* I uge 2 vil vi arbejde med de nævnte algoritmer "insertion sort" og "merge sort". For at lave denne opgave er det imidlertid ikke nødvendigt at kende dem.

   (b) CLRS 1.2-3.

   (c) CLRS1-1 (udfyld kun tabellen for $\log n$, n og 2^n).

4. **Zoombieduellering.** Du har en hær af 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.

   (a) Vis hvordan du finder den stærkeste zombie med højst $n - 1$ dueller.

   (b) (*) Vis hvordan du finder både den stærkeste og svageste zombie med højst $3n/2$ dueller.

   (c) (**) Vis hvordan du finder både den stærkeste og næststærkeste zombie med højst $n + \log_2 n$ dueller.

5. **Maksimal intervalsum.** Betragt et array A af størrelse n, hvor A kan indeholde både positive og negative tal. For to heltal a og b, hvor $0 \leq a \leq b \leq n$, definerer vi *intervalsummen* fra a til b som $A[a] + \ldots + A[b-1]$. (I det specielle tilfælde at $a = b$, definerer vi intervalsummen til at være 0.) Nedenstående funktion MaxIntervalSum beregner den maksimale intervalsum i A.

   (a) Håndkør funktionen når vi kalder MaxIntervalSum($[8, 4, -13, 10, 5], 5$).

   (b) (*) Antag at den første for-løkke er slut, så arrayet B er blevet udregnet. Argumentér for at $B[i] = A[0] + \ldots + A[i-1]$, for alle $0 \leq i \leq n$. Argumentér dernæst for at intervalsummen fra a til b er lig $B[b] - B[a]$.

   (c) (*) Argumentér for at funktionen MaxIntervalSum korrekt beregner den maksimale intervalsum i A.

   (d) (**) Opskriv pseudokode for en funktion der beregner den maksimale intervalsum uden at lave et ekstra array B, altså en mere pladsbesparende version.

```
MaxIntervalSum(A,n)
  let B be a new array of size n + 1
  B[0] = 0
  for i = 1 to n
    B[i] = B[i − 1] + A[i − 1]
  j = 0
  x = 0
  for i = 0 to n
    if B[i] < B[j]
      j = i
    x = max{x, B[i] − B[j]}
  return x
```

1 Opgaver til fredag

1. **Rekursion og iteration.** En funktion er rekursiv hvis den kalder sig selv. Fx er funktionen f nedenfor rekursiv. Løs følgende opgaver:

   (a) Hvad beregner f(A,n) hvis A er en tabel af heltal af længde n?

   (b) Omskriv f til at være iterativ i stedet, dvs. lav en funktion der gør det samme som f uden at kalde sig selv.

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

2. **Mere rekursion.** Betragt funktionerne R1–R6 nedenfor.

   (a) Hvad beregnerfunktionerne for $n = 2$, $n = 4$ og $n = 5$?

   (b) Udtryk den returnerede værdi som funktion af n.

   (c) Omskriv funktionerne så de bliver iterative.

```
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 == 0
    return 1
  else
    return 2·R5(n − 1)
```

```
R6(n)
  if n == 0
    return 1
  else
    return R6(n − 1)+R6(n − 1)
```

3. **Lineartrose.** Du har netop ansat 128 programmører til din nyopstartede high-tech virksomhed. Desværre lider en af dem af den frygtede sygdom *lineartrose*, 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 i blandt 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.

   (a) Vis at du kan finde den syge programmør med højst 7 tests.

   (b) Hvor mange test kan du klare dig med generelt hvis du har n programmører?

(c) (*) 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.** Lad A være et array af størrelse n, hvor $n \geq 2$. Indgang $A[i]$ er et toppunkt hvis $A[i]$ er mindst lige så stor som naboerne:

   - $A[i]$ er et toppunkt hvis $A[i-1] \leq A[i] \geq A[i+1]$ for $1 \leq i \leq n-2$.
   - $A[0]$ er et toppunkt hvis $A[0] \geq A[1]$, og $A[n-1]$ er et toppunkt hvis $A[n-2] \leq A[n-1]$. (Tænk $A[-1] = A[n] = -\infty$).

   Lad $A = [2, 1, 3, 7, 3, 11, 1, 5, 7, 10]$ være et array. Løs følgende opgaver.

   (a) Angiv alle toppunkter i A.

   (b) Angiv hvilke toppunkter de to lineærtidsalgoritmer Toppunkt1 og Toppunkt2 nedenfor finder.

   (c) Der er en fejl i funktionen Toppunkt1. Find et array som gør at den ikke returnerer et toppunkt, og foreslå en rettet version af funktionen som altid virker som den skal.

   (d) Angiv sekvensen af rekursive kald, der bliver foretaget når man kalder den rekursive algoritme Toppunkt3(A,0,9).

```
Toppunkt1(A,n)
  if A[0] ≥ A[1]
    return 0
  for i = 1 to n − 2
    if A[i − 1] ≤ A[i] and A[i] ≥ A[i + 1]
      return i
```

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

```
Toppunkt3(A,i,j)
  m = ⌈(i+j)/2⌉ // gennemsnittet af i og j rundet op til nærmeste heltal
  if A[m] is toppunkt
    return m
  else
    if A[m − 1] > A[m]
      return Toppunkt3(A,i,m − 1)
    else
      return Toppunkt3(A,m + 1,j)
```

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`.