Ugeseddel 2

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

Litteratur:

Uge 2 vil vi primært bruge på:

- CLRS kapitel 1 og 2
- Søgning og sortering noter

Note: 2.1-3 soegning og sortering Noter CLRS 2

Mål for ugen:

- Forståelse for vigtigheden af konstruktion af effektive algoritmer
- Sortering og binær søgning
- Del og hersk

Forelæsninger (og spørgertimer):

Tirsdag uge 2: Insertion sort, mere fra kapitel 1, Stephen.

Spørgertimen vil have mest fokus på det faglige, Stephen.

Torsdag uge 2: Del og hersk/rekursion, mest fra kapitel 2, Stephen.

Spørgetimen er aflyst denne gang

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.**

Bemærk: I nedenstående bruges flere gange udtryk som $\Theta(n)$, $\Theta(n \log n)$, og $\Theta(n^2)$. Disse udtryk er forklaret løst i pensum fra denne ugeseddel og til forelæsningen. De vil blive defineret mere præcist senere i kurset. Indtil videre kan du tænke på at f.eks. en $\Theta(n)$ tids algoritme højst må bruge c*n operationer, hvor c er en konstant, f.eks. c=100.

1. **Håndkøring og egenskaber** Løs følgende opgaver.
   - **1.1** CLRS 2.1-1.
   - **1.2** CLRS 2.1-2.
   - **1.3** CLRS 2.3-6

2. **Manglende tal** Lad A være en tabel af længde $n - 1$ således at indgangene i A indeholder et unikt tal fra $\{0,1,2,\ldots,n-1\}$, dvs., at A indeholder alle tal fra $\{0,1,\ldots,n-1\}$ på nær et enkelt tal m, som vi kalder det *manglende tal*. F. eks. med $A = [2, 0, 4, 3]$ ($n = 5$) er det manglende tal $m = 1$. Vi er interesserede i effektive algoritmer til at finde det manglende tal i A. Løs følgende opgaver.
   - **2.1** Giv en algoritme der løser problemet i $\Theta(n)$ tid. *Hint:* Brug en ekstra tabel af længde n.
   - **2.2** Vi vil nu gerne løse problemet hurtigt, men også begrænse pladsforbruget så meget som muligt. Giv en algoritme der løser problemet i $\Theta(n^2)$ tid og kun bruger et konstant antal ekstra variable.
   - **2.3** Giv en algoritme der løser problemet i $\Theta(n)$ tid og kun bruger et konstant antal ekstra variable.
     Hint: Husk at 1+2+..+n= n * (n+1) /2

3. **2Sum og 3Sum** Lad $A[0..n-1]$ være en tabel af heltal (positive og negative). Tabellen A har en 2-*sum*, hvis der findes to indgange i og j, så $A[i]+A[j]=0$. Tilsvarende, har A en 3-*sum*, hvis der findes tre indgange i, j og k så $A[i]+A[j]+A[k]=0$. Løs følgende opgaver.

   3.1 Giv en algoritme, der afgør om A har en 2-sum i $\Theta(n^2)$ tid.

   3.2 Giv en algoritme, der afgør om A har en 2-sum i $\Theta(n \log n)$ tid. *Hint:* benyt binær søgning.

   3.3 Giv en algoritme, der afgør om A har en 3-sum i $\Theta(n^3)$ tid.

   3.4 Giv en algoritme, der afgør om A har en 3-sum i $\Theta(n^2 \log n)$ tid. *Hint:* benyt binær søgning.

   3.5 [***] Giv en algoritme, der afgør om A har en 3-sum i $\Theta(n^2)$ tid.

Opgaver torsdag formiddag:

1. **Merge** Håndkør merge proceduren på følgende input:
   1.1 [1, 3, 4, 7, 8] og [2, 4, 5, 7, 8]
   1.2 [5, 8, 11, 14] og [7, 8, 13, 19]
   1.3 [2, 5, 7, 7, 9] og [11, 23, 41, 59, 89]

2. **Toppunkter** Løs følgende opgaver. Beskriv hvordan værstefaldsinput til hver af de 3 toppunktsalgoritmer i figur 7, 8 og 9 ser ud. Værstefaldsinput vil sige det input, som giver den værst tænkelige køretid.

3. **Duplikater og tætte naboer** Lad $A[0..n-1]$ være en tabel af heltal. Løs følgende opgaver.
   3.1 Et *duplikat* i A er et par af indgange i og j så $A[i]=A[j]$. Giv en algoritme der afgør om der er et duplikat i A i $\Theta(n^2)$ tid.
   3.2 Giv en algoritme der afgør om der er et duplikat i A i $\Theta(n \log n)$ tid. *Hint:* benyt merge sort som kan sortere i $\Theta(n \log n)$ tid.
   3.3 Et *tætteste par* i A er et par af indgange i og j så $|A[i]-A[j]|$ er minimal blandt alle par af indgange. Giv en algoritme der finder et tætteste par i A i $\Theta(n \log n)$ tid.

Opgaver torsdag eftermiddag:

1. **Mergesort** Håndkør mergesort med følgende input (illustrer som CLRS figur 2.4):
   1.1 [5, 8, 3, 1, 4, 7, 2, 6]
   1.2 [12, 53, 13, 64, 34, 9, 21, 51]

2. **Køretid** Antag du har algoritmer hvis køretid er $100n$, $10n^2$ og $5n^3$. Hvor meget langsommere kører algoritmerne hvis du fordobler inputstørrelsen n?

3. **Rekursion og fakultet** Denne opgave omhandler en algoritme for fakultetsfunktionen ($n! = n \cdot (n-1) \cdot \cdots 1$). Nedenfor er 3 forsøg på at lave en rekursiv algoritme der beregner $n!$.

```
Algoritme Fak1(n)
  if n = 1
    return n
  x = n · (Fak1(n) − 1)
  return x

Algoritme Fak2(n)
  if n = 0
    return n
  x = n · Fak2(n − 1)
  return x

Algoritme Fak3(n)
  if n = 1
    return n
  return n · Fak3(n − 1)
```

   3.1 Hvilken af algoritmerne beregner $n!$ korrekt når n er et positivt heltal?

3.2 Nedenstående algoritme tager et positivt heltal som input.

Algoritme Rekur(n)
  if $n \leq 0$
    return 0
  return n + Rekur(n − 1) + 2

Giv iterativ variant af algoritmen.

Stjerneopgaver (svære ekstraopgaver)

4. **2D toppunkter** Lad M være $n \times n$ matrix (2D-tabel). En indgang $M[i, j]$ er et *toppunkt* hvis det ikke er mindre end dets naboer i retning N, Ø, S og V (dvs. $M[i][j] \geq M[i-1][j]$, $M[i][j] \geq M[i][j-1]$, $M[i][j] \geq M[i+1][j]$ og $M[i][j] \geq M[i][j+1]$). Vi er interesseret i effektive algoritmer til at finde et toppunkt i A. Løs følgende opgaver.

4.1 Giv en algoritme der tager $\Theta(n^2)$ tid.

4.2 [***] Giv en algoritme der tager $\Theta(n \log n)$ tid. *Hint:* Start med at finde det maksimale tal i den midterste søjle og benyt det til at lave en rekursion.

4.3 [***] Giv en algoritme der tager $\Theta(n)$ tid. *Hint:* Konstruer en rekursion der inddeler M i 4 kvadranter.

5. **Udvælgelse, partitionering og kviksortering** Lad $A[0..n-1]$ være en tabel af heltal. Tallet med *rang* k i A er det tal der fremkommer på position k såfremt man sorterer A. *Medianen* af A er tallet i A med rang floor((n − 1)/2). Løs følgende opgaver.

5.1 Giv en algoritme, der givet et k, finder tallet med rang k i A i $\Theta(n \log n)$ tid.

En *partitionering* af A er en opdeling af A to tabeller A_{lav} og $A_{\text{høj}}$ således at A_{lav} indeholder alle tal fra A der er mindre end eller lig med medianen af A og $A_{\text{høj}}$ indeholder alle tal fra A der er større end medianen af A. Antag idet følgende at du har en lineærtidsalgoritme til at finde medianen af en tabel.

5.2 Giv en algoritme til at beregne en partitionering af A i $\Theta(n)$ tid.

5.3 [*] Giv en algoritme til at sortere A i $\Theta(n \log n)$ tid vha. rekursiv partitionering.

5.4 [**] Giv en algoritme, der givet et k, finder tallet med rang k i A i $\Theta(n)$ tid.

6. CLRS 2-1

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.