Institut for Matematik og Datalogi
Syddansk Universitet, Odense

18. april 2017
RF

DM507 – Opgaver uge 16

Eksaminatorier I

1. Cormen et al. øvelse 15.1-3 (side 370).
2. Cormen et al. øvelse 15.1-1 (side 369). Hint: dette er et simpelt induktionsbevis, med brug af formel A.5 (side 1147).
3. Cormen et al. øvelse 15.4-1 (side 396).
4. Cormen et al. øvelse 15.4-2 (side 396).
5. Cormen et al. øvelse 15.4-4 (side 396). Der menes her at I kan nøjes med en c-tabel af den angivne størrelse.
6. Eksamen juni 2010, opgave 4.
7. ($*$) Cormen et al. øvelse 15.4-5 (side 397). Hint: lav en (1-dimensional) tabel over $l(i)$ for $i = 1$ til n, hvor $l(i)$ angiver længden af en længste monotont stigende delsekvens *som ender* ved det i'te tal. Dvs. løs dette lidt modificerede problem med dynamisk programmering. Brug så tabellen med løsningerne for dette problem til at løse det oprindelige problem. At bruge ovenstående metode er meningen med opgaven. Man kan faktisk også finde længste monotont stigende delsekvens ved at se på inputsekvensen som en streng af tal, og derefter løse et LCS-problem mellem strengen selv og en udgave af den som er blevet sorteret, således at alle tallene kommer i stigende rækkefølge (($**$)-opgave: bevis at dette løser det samme problem). Derefter kan man bruge at vi jo allerede har en metode baseret på dynamisk programmering til at løse LCS. Er køretiden den samme eller er den forskellig for de to måder?

Eksaminatorier II

1. Eksamen juni 2013, opgave 7.

2. ($*$) Cormen et al. problem 15.2 (side 405). Hint: For at løse dette direkte med dynamisk programmering, lad delproblemer være givet ved delstrengen $x_i x_{i+1} \ldots x_{j-1} x_j$ for $i \leq j$, og se i analysen på begge ender (dvs. på x_i og x_j) af delproblemet, når man forsøger at relatere det til mindre delproblemer. Herudover ligner analysen lidt den for Longest Common Subsequence. Dette er meningen med opgaven. Man kan faktisk også løse dette palindrom-problem ved at løse LCS-problemet for strengen selv og dens omvendte (($**$)-opgave: bevis dette), hvilket vi jo allerede har en metode baseret på dynamisk programmering for. Er køretiden den samme eller er den forskellig for de to måder?

3. ($*$) Cormen et al. øvelse 16.2-5 (side 428). Hint: algoritmen er simpel, og ligner lidt den på side 421 i bogen, som løser activity selection problemet. For at bevise korrekthed, bevis følgende invariant: de hidtil valgte intervaller er en delmængde af en optimal dækning.

4. Eksamen januar 2007, opgave 2.

5. Opvarming til projektet del III: Læs først projektteksten. Lav så ved hjælp af `read()`-metoden fra Java-bibliotekets `FileInputStream` et Java-program som: 1) læser en fil byte for byte, 2) undervejs i et array tæller, hvor mange af hver af de 256 mulige bytes filen indeholder, og 3) til sidst udskriver en tabel i stil med:

   ```
   Byte 0: 0
   Byte 1: 0
   .
   .
   Byte 97: 7
   Byte 98: 4
   .
   .
   Byte 255: 0
   ```

   Prøv programmet på nogle simple `.txt`-filer (uden danske bogstaver). Find på nettet en tabel over ASCII-koden, og brug den til at checke output. Prøv også programmet på andet end tekstfiler, f.eks. `.jpg`-filer og `.doc`-filer.

6. Opvarming til projektet del III: Lav et Java-program som via kald til `readBit()`-metoden fra den udleverede `BitInputStream` læser de enkelte bits i en input `.txt`-fil én efter én, og undervejs udskriver disse bits som `'0'` og `'1'` tegn. Brug en tabel over ASCII-koden for at checke output fra en tekst `.txt`-fil.

7. Opvarming til projektet del III: Lav en `.txt`-fil med 16 tegn i. Lav et Java-program som via fire kald til `readInt()`-metoden fra den udleverede `BitInputStream` læser disse 16 bytes som fire `int`'s og udskriver hver af dem. [Med lidt viden fra nettet om two's complement repræsentationen (som Java og mange andre sprog bruger) for heltal samt programmet fra forrige opgave, kan man godt checke, om output passer.]

Studiegrupper

Forslag til fokus for arbejde i studiegrupper (hvis man er i en sådan):

- Forsøg at lave programmeringsopgaverne på forhånd. Diskuter og del løsningsmetoder i fælleskab, men sørg for at alle skriver deres egen kode.

- Diskuter, hvad der er ens og hvad der adskiller situationer, hvor henholdsvis divide-and-conquer og dynamisk programmering kan tænkes at være brugbart.

- Forsøg at lave opgaverne med dynamisk programmering på forhånd. Sammenlign svar i studiegruppen. Skiftes til at fremlægge jeres løsning. For de opgaver, hvor alle var gået i stå, forsøg at løse dem igen i fælleskab.