Institut for Matematik og Datalogi
Syddansk Universitet, Odense

25. februar 2008
LMF

DM507 Eksamen — Obligatorisk Opgave, Del 1 af 3
Et Pakningsproblem

1 Indledning

I denne note beskrives første del af den obligatoriske opgave, der skal løses i forbindelse med DM507, foråret 2008. Mandag d. 3. marts stilles anden del af opgaven. Tredje og sidste del af opgaven stilles mandag d. 21. april.

Afleveres de to første delopgaver samlet senest mandag d. 7. april, vil instruktorerne kigge dem igennem og give kommentarer. Hele opgaven (1. – 3. del) skal afleveres samlet senest tirsdag d. 13. maj kl 12:00. Det er frivilligt at aflevere delopgave 1 + 2 til instruktoren inden den endelige aflevering, men det anbefales kraftigt.

Efter en indledende beskrivelse af det problem, der skal arbejdes med, beskrives input/output-formater mm. Programmerne vil blive afprøvet automatisk, og det er derfor vigtigt at overholde kravene til input/output-format med videre. I teksten beskrives det, hvordan man selv kan kontrollere, at ens format er i orden.

2 Pakning-problemet

I denne opgave ser vi på et pakningsproblem, hvor vi er givet en mængde af kasser og en mængde af elementer, som skal pakkes i kasserne. Kasserne har alle størrelse B, men elementerne kan have varierende størrelse. Det gælder om at bruge så få kasser som muligt uden at overfylde nogen kasse; dvs. den samlede størrelse af elementerne pakket i en enkelt kasse må ikke overstige B.

3 Algoritmerne

Det beskrevne pakningsproblem er NP-fuldstændigt. Det betyder, at man regner med, at der ikke findes nogen polynomiel algoritme, som løser problemet optimalt. (At en algoritme er polynomiel betyder, at der findes en konstant c, så algoritmen har køretid $O(n^c)$.) Derfor må vi ty til algoritmer, som ikke altid finder den bedst mulige løsning. Algoritmen, som vi skal se på i denne delopgave, er en meget simpel algoritme, som hedder Next-Fit (NF). Denne algoritme pakker elementerne et efter et, og holder kun én kasse åben ad gangen. Kan det aktuelle element være i den åbne kasse, anbringes det der. Ellers lukkes den åbne kasse, en ny, tom kasse åbnes, og elementet lægges heri.

4 Programmet

Programmet skal skrives i Java og skal køre på IMADA's maskiner. Programmet skal være velstruktureret og kommenteret i passende omfang.

Next-Fit skal implementeres, så den har køretid $O(n)$, hvor n er antallet af elementer.

4.1 Input, output og kørsler

I dette afsnit beskrives formatet for input og output samt præcis, hvordan programmet skal kunne afvikles.

Input

Første linie i input-filen er et heltal, som angiver størrelsen B af kasserne. På de næste n linier står størrelserne af de n elementer, der skal pakkes. Disse størrelser er heltal mellem 1 og B. Nedenfor er vist et eksempel på input. Her er kassestørrelsen $B = 100$, og der er 10 elementer.

```
100
50
25
50
25
30
30
25
25
25
10
```

Figur 1: Illustration af input-formatet.

Du må gerne gå ud fra, at input overholder det beskrevne format. Dvs. dit program behøver ikke at kunne håndtere forkert input.

Output

Output består af en linie for hver kasse, som tages i brug. På linie i står størrelserne af de elementer som pakkes i den i'te kasse. Elementerne skal optræde i den rækkefølge, de blev pakket i. Der skal være præcis et mellemrum mellem hvert par af nabotal. Der må ikke være mellemrum efter det sidste tal på en linie. Hver linie (også den sidste) afsluttes med "\n".

I Figur 2 kan man se det korrekte output til input'et fra Figur 1.

```
50 25
50 25
30 30 25
25 25 10
```

Figur 2: Output svarende til input fra Figur 1.

Kørsler

Ud over input/output-format er der følgende krav til programmet:

- Alle filer skal ligge i ét katalog (directory) hos jer selv på IMADA's system.

- Hovedprogrammet skal hedde `BP.java` (for Bin Packing). Dvs. man skal kunne afvikle programmet på følgende måde:

  ```
  java BP <filename>
  ```

  hvor filen `filename` indeholder input.

- Programmet skal fungere på IMADAs system.

5 Test

Testen skal designes, inden du skriver programmet. Overvej, hvilke specialtilfælde man kan komme ud for, og hvilke fejl der kan opstå. Det er en god ide at teste de enkelte komponenter i programmet, efterhånden som du laver dem.

Du kan få et fingerpeg om, hvad vi vil sige til det output, dit program producerer, ved at skrive `DM507check` (`DM507check` kalder `/home/IMADA/courses/dm507/Tests/check.pl`). Du skal, før du afgiver kommandoen, placere dig i det katalog, som alle dine oversatte `Java`-programmer ligger i. Så vil dit program blive kørt på testfilerne i `/home/IMADA/courses/dm507/Tests`. Bemærk, at testen kun indeholder nogle få testeksempler, så hvis der ikke findes fejl, betyder det ikke nødvendigvis, at dit program fungerer korrekt.

6 Formalia

Bemærk, at den obligatoriske opgave er en del af eksamen, så reglerne for en eksamenssituation gælder også her. Dvs. du må ikke modtage hjælp fra andre, og du skal sørge for, at andre ikke kan læse dine filer. En god måde at beskytte sine filer på er følgende.

```
mkdir DM507projekt
chmod 700 DM507projekt
```

I må gerne snakke sammen om den overordnede løsning og lære af hinanden, men når I går i gang med at skrive ting ned, såvel program som rapport, skal I arbejde selvstændigt. Overtrædelse af dette er eksamenssnyd.

God arbejdslyst ☺