Institut for Matematik og Datalogi
Syddansk Universitet, Odense

24. september 2007
LMF

DM507 Eksamen — Obligatorisk Opgave, Del 1
Et Pakningsproblem

1 Indledning

I denne note beskrives første del af den obligatoriske opgave, der skal løses i forbindelse med DM507, efteråret 2007. Afleveres denne første del senest mandag d. 22. oktober, vil instruktorerne kigge den igennem og give kommentarer. Opgaven med instruktorens kommentarer kan hentes i det instruktorrum, som kaldes Balkonen, fra mandag d. 29. oktober. Hele opgaven (1. + 2. del) skal afleveres samlet senest mandag d. 26. november kl 12:00. Det er frivilligt at aflevere første delopgave 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. Til sidst beskrives afleveringsproceduren. Det er vigtigt at læse hele opgaven igennem, før man begynder sit arbejde.

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.

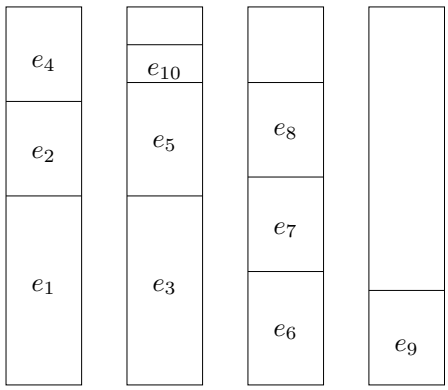
3 Algoritmerne

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. Algoritmerne, som vi skal se på i denne delopgave, er First-Fit (FF), First-Fit-Decreasing (FFD), Next-Fit (NF) og Next-Fit-Decreasing (NFD).

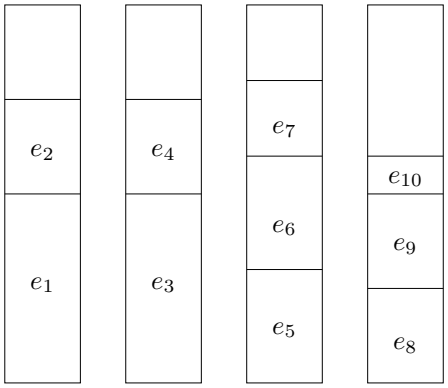
First-Fit nummererer kasserne efter den rækkefølge, den tager dem i brug. Den ser på elementerne et for et, og pakker hvert element i den første kasse, som har plads nok. Er der ingen af de allerede anvendte kasser, som har plads til elementet, tages en ny kasse i brug.

Next-Fit ser også på elementerne et efter et. Hvis det aktuelle element passer i den kasse, som er taget sidst i brug, pakkes det der. Ellers tages en ny kasse i brug.

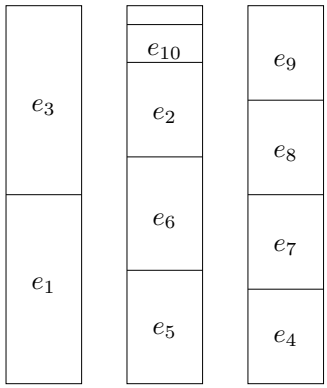
First-Fit-Decreasing gør det samme som First-Fit, bortset fra at den starter med at sortere elementerne efter størrelse, og pakker de største elementer først. Tilsvarende gør *Next-Fit-Decreasing* det samme som Next-Fit, bortset fra at den pakker elementerne i rækkefølge efter faldende størrelse.



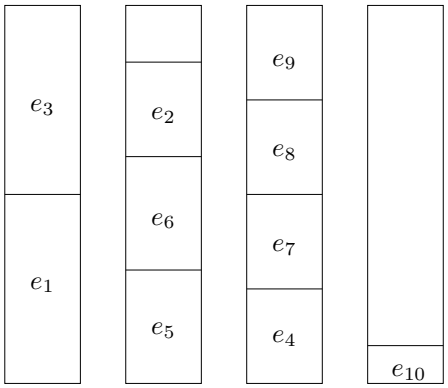
(a) First-Fit



(b) Next-Fit



(c) First-Fit-Decreasing



(d) Next-Fit-Decreasing

Figur 1: De fire algoritmer anvendt på sekvensen $\langle e_1, e_2, e_3, e_4, e_5, e_6, e_7, e_8, e_9, e_{10} \rangle$, hvor $|e_1| = |e_3| = 50$, $|e_5| = |e_6| = 30$, $|e_2| = |e_4| = |e_7| = |e_8| = |e_9| = 25$, $|e_{10}| = 10$ og $B = 100$.

I Figur 1 er vist et eksempel, hvor kasserne har størrelse 100, og elementerne har størrelse 50, 25, 50, 25, 30, 30, 25, 25, 25, 10.

4 Programmet

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

Algoritmerne skal implementeres sådan, at NF har køretid $O(n)$, NFD har køretid $O(n \log n)$, og FF og FFD har køretid $O(n^2)$, hvor n er antallet af elementer. Algoritmerne FF og FFD kan implementeres, så de har køretid $O(n \log n)$, men det er ikke et krav i denne opgave.

Skriv pseudokode for algoritmerne, inden du begynder at implementere dem.

Til at sortere elementerne må du gerne bruge JAVAs sorteringsmetode, men husk, at den sorterer elementerne i stigende orden, ikke i faldende.

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. Som en konkret illustration kan man se på input til eksemplet fra Figur 1.

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

Figur 2: Input svarende til eksemplet i Figur 1.

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

Output

Output'et består af fire dele svarende til de fire algoritmer FF, NF, FFD og NFD, i denne rækkefølge. Output'et fra de enkelte algoritmer adskilles med præcis en blank linie. Dvs. output'et indeholder præcis 3 blanke linier i alt.

For hver algoritme skal der være en linie for hver bin, som tages i brug. På linie i står størrelserne af de elementer som pakkes i den i'te bin. 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 3 kan man se det korrekte output til input'et fra Figur 2.

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.

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

50 25
50 25
30 30 25
25 25 10

50 50
30 30 25 10
25 25 25 25

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

Figur 3: Output svarende til eksemplet i Figur 1.

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.

Det er en rigtig god ide at køre DM507check, inden du afleverer. På den måde sikrer du bl.a., at dit program bruger det korrekte input- og output-format.

Programmet `DM507check` kan afbrydes med `Ctrl-c`.

6 Rapporten

Rapporten skal indeholde pseudokode for de fire algoritmer og gerne et klassediagram. Argumentér for, at algoritmerne har de ønskede køretider.

Rapporten skal desuden indeholde en udskrift af hele programmet. Denne udskrift skal være identisk med det elektronisk afleverede program. Der skal være en beskrivelse af de væsentligste valg, der er truffet i forbindelse med implementeringen, samt begrundelser herfor.

Endelig skal der være en overordnet beskrivelse af din teststrategi. Dvs. du skal *uden at referere til programmet* beskrive dine overvejelser i forbindelse med design af testen. Derudover skal selve testen dokumenteres, med reference til den overordnede teststrategi.

7 Aflevering

Programmet skal afleveres elektronisk. Den elektroniske aflevering foregår på følgende måde: Opret et katalog (directory), som indeholder alle dine `.java`-filer til opgaven *og intet andet*. Stil dig i dette katalog. Brug først `ls -la` for at sikre, at du står det rigtige sted, og at de rigtige filer er der. Afgiv derefter kommandoen `aflever DM507`.

D.v.s. gør følgende.

```
cd <opgave-katalog>
ls -la
aflever DM507
```

hvor `<opgave-katalog>` indeholder alle dine `.java`-filer til opgaven og intet andet.

Bemærk, at du (inden afleveringsfristen) kan aflevere flere gange. Kun den sidste aflevering tæller (de andre slettes).

Rapporten afleveres til forelæseren eller instruktoren. Denne opgaveformulering er vedhæftet en forside, som skal anvendes ved afleveringen.

8 Yderligere 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 og behandles som sådan.

God arbejdslyst 

Institut for Matematik og Datalogi
Syddansk Universitet, Odense

24. september 2007
LMF

DM507 eksamen, Efteråret 2007
Obligatorisk Opgave, Del 1

Skriv tydeligt (maskinskrift eller blokbogstaver)

Navn: .....

Fødselsdato: .....

Brugernavn (login): .....

Instruktor	Jacob	Mikkel
Sæt kryds		

Besvarelsen omfatter ..... nummererede sider.