Institut for Matematik og Datalogi
Syddansk Universitet, Odense

19. februar 2009
LMF

DM507 Eksamen — Obligatorisk Opgave, Del 1 af 3

Uafhængige mængder

1 Indledning

I denne note beskrives første del af den obligatoriske opgave, der skal løses i forbindelse med DM507, foråret 2009. Torsdag d. 5. marts stilles anden del af opgaven. Tredje og sidste del af opgaven stilles torsdag d. 19. marts.

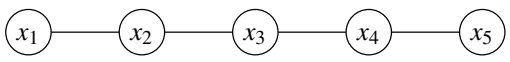
Delopgaverne skal ikke afleveres enkeltvis. Derimod skal hele opgaven afleveres samlet senest torsdag d. 30. april kl 12:00. Det er selvfølgelig en rigtig god ide at lave delopgaverne, efterhånden som de stilles — ideen med del-opgaverne er, at projektet deles op i overskuelige delprojekter, som kan testes selvstændigt, og at give jer bedre tid til at løse hele projektet.

2 Problemet

I denne opgave ser vi på et graf-teoretisk optimeringsproblem. En uafhængig mængde i en graf $G = (V,E)$ er en delmængde U af knudemængden V, hvor intet par af knuder i U er naboer i G. Dvs.

$$\forall x,y \in U : (x,y) \notin E$$

Vi antager, at hver knude $x \in V$ har en vægt $w(x)$, og opgaven er at finde en uafhængig mængde med størst mulig samlet vægt. Selv hvis alle knuder blot har vægt 1 (så opgaven blot er at finde en størst mulig uafhængig mængde), er problemet det, der kaldes NP-fuldstændigt. Det betyder, at man regner med, at der ikke findes en algoritme med polynomiel køretid (dvs. $O(n^k)$, hvor n er antallet af knuder i grafen, og k er en konstant), som altid løser problemet optimalt. Vi skal derfor fokusere på specielle typer af grafer, hvor problemet ikke er så svært at løse. Den første delopgave handler om stier, dvs. grafer hvor knuderne kan nummereres $x_1, x_2, \ldots, x_n$, så x_i og x_{i+1} er naboer, for $1 \leq i \leq n-1$, og ingen andre par af knuder er naboer. Eks.:



Hvis $w(x_1) = 1$, $w(x_2) = 3$, $w(x_3) = 1$, $w(x_4) = 4$ og $w(x_5) = 5$ i ovenstående eksempel, vil $\{x_2, x_5\}$ udgøre en optimal løsning.

3 Algoritmen

Når man for en given sti skal finde den bedste uafhængige mængde, kunne en første ide være at beregne vægten af samtlige uafhængige mængder og vælge den med størst vægt. Desværre har en sti med n knuder F_{n-2} uafhængige mængder, hvor F_i er det $i$te Fibonacci-tal. Dvs. denne strategi ville have eksponentiel køretid. I stedet skal vi bruge dynamisk programmering.

Lad G_i være delgrafen af G bestående af knuderne $x_1, x_2, \ldots, x_i$ og kanterne imellem dem. Dvs. $G_i = (V_i, E_i)$, hvor $V_i = \{x_1, x_2, \ldots, x_i\}$ og $E_i = \{(x_1, x_2), (x_2, x_3), \ldots, (x_{i-1}, x_i)\}$. Bemærk, at $G_n = G$.

For $i = 1, 2, \ldots, n$ skal der beregnes to værdier: $W^-(i)$ og $W^+(i)$. For hvert i udtrykker $W^-(i)$ den størst mulige vægt af en uafhængig mængde i G_i, hvis x_i *ikke* må være med i den uafhængige mængde. Tilsvarende udtrykker $W^+(i)$ den størst mulige vægt af en uafhængig mængde i G_i, hvis x_i *skal* være med. Disse værdier kan beregnes på følgende måde:

$$
\begin{aligned}
W^+(1) &= w(x_1) \\
W^-(1) &= 0 \\
\\
W^+(i) &= w(x_i) + W^-(i-1), && \text{for } 2 \le i \le n \\
W^-(i) &= \max\{W^+(i-1), W^-(i-1)\}, && \text{for } 2 \le i \le n
\end{aligned}
$$

Bemærk, at den samlede vægt af en optimal løsning er $\max\{W^-(n), W^+(n)\}$.

For stien i eksemplet fra Afsnit 2 ser W-værdierne sådan ud:

i	1	2	3	4	5
$w(i)$	1	3	1	4	5
$W^+(i)$	1	3	2	7	8
$W^-(i)$	0	1	3	3	7

4 Programmet

Programmet skal skrives i JAVA og skal køre på IMADAs maskiner. Alle filer skal ligge i ét katalog (directory).

Programmet skal være velstruktureret og kommenteret i passende omfang.

Køretiden skal være lineær, dvs. $O(n)$, hvor n er antallet af knuder i stien.

Input: Knudernes vægte angives i den rækkefølge, de optræder i stien. Hver linie indeholder én vægt. Der er ingen tomme linier og ingen blanke tegn. Vægtene er heltal. Nedenfor er vist et eksempel på input. Det svarer til eksemplet fra Afsnit 2.

```
1
3
1
4
5
```

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 blot af et tal, som angiver den samlede vægt af en optimal løsning. For ovenstående eksempel vil output altså se således ud:

```
8
```

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.

6 Rapporten

Der skal ikke skrives selvstændige rapporter om første og anden delopgave, men det er en god ide allerede nu at begynde på den del af rapporten, som omhandler første delopgave.

Argumentér for, at algoritmen har den ønskede køretid.

Rapporten skal desuden indeholde pseudokode og gerne et klassediagram. 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 Formalia

Bemærk, at den obligatoriske opgave er en del af eksamen, så reglerne for en eksamenssituation gælder også her. Dvs. du skal arbejde individuelt, 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
```

God arbejdslyst ☺