Institut for Matematik og Datalogi
Syddansk Universitet, Odense

21. februar 2011
LMF

DM507 Eksamen – Obligatorisk Opgave
Rejseplanlægning

1 Problemet

Denne opgave går ud på at lave et program, som ud fra en togkøreplan finder den bedste rute mellem to byer. Vi er selvfølgelig interesserede i, at turen tager så kort tid som muligt, men også at den indeholder så få togskift som muligt. Vi er villige til at forlænge turen med op til 15 minutter, hvis det sparer os for et togskift. Dvs. hvis vi kan vælge mellem to ruter r_1 og r_2 af varighed t_1 og t_2 med henholdsvis n_1 og n_2 togskift, vil vi vælge r_1, hvis $t_1+15n_1 < t_2+15n_2$. Hvis $t_1+15n_1 > t_2+15n_2$, vil vi vælge r_2, og hvis $t_1+15n_1 = t_2+15n_2$, er de to ruter lige gode.

Input er en køreplan, en start- og en slut-station. Output skal være varigheden og antal togskift for den bedste rute fra start- til slut-station. Hvis der findes to ruter r_1 og r_2 af varighed t_1 og t_2 med henholdsvis n_1 og n_2 togskift, hvor $t_1 + 15n_1 = t_2 + 15n_2$, vil både "$t_1$ n_1" og "t_2 n_2" være et rigtigt svar.

I Figur 1(a) er vist et eksempel på en køreplan. Hvis man beder om den bedste rute fra Lunderskov til Middelfart, vil det korrekte svar være "42 1".

For nemheds skyld lader vi i denne opgave som om ankomsttid og afgangstid er den samme på hver station og taler kun om afgangstider (i virkeligheden er der typisk et par minutters forskel på ankomsttid og afgangstid — dvs. denne forenkling betyder, at den beregnede rejsetid kan være et par minutter længere end den faktiske).

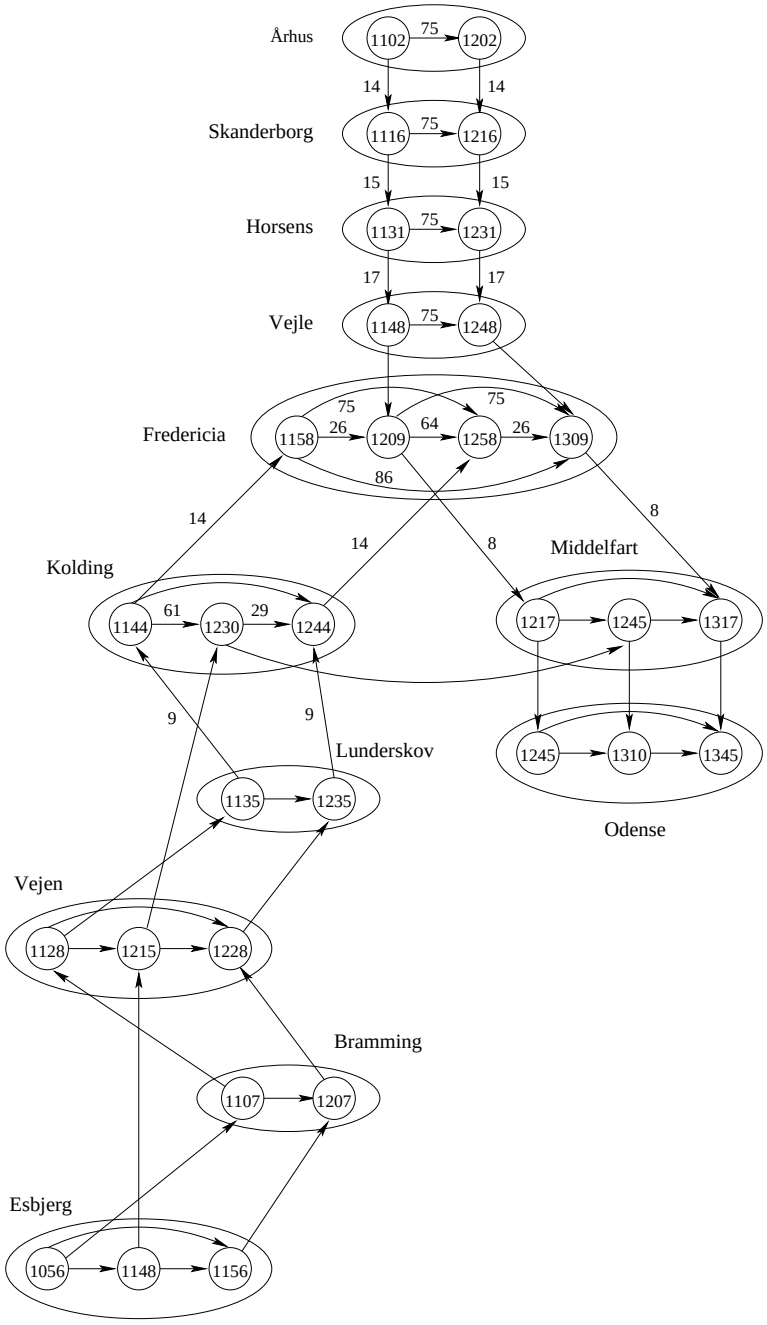
2 Overordnet fremgangsmåde

Der skal konstrueres en vægtet orienteret graf, som repræsenterer køreplanen. For hver toglinie gøres følgende (lad $s_1, s_2, \ldots, s_k$ være stationerne på linien, angivet i den rækkefølge de optræder på linien). For hver station s_i oprettes en knude $v^a_{s_i}$ for hver afgang a. For $1, 2, \ldots, k-1$ tilføjes en kant fra $v^a_{s_i}$ til $v^a_{s_{i+1}}$. Denne slags kanter kaldes *eksterne* kanter.

Når alle toglinier således er repræsenteret i grafen vha. eksterne kanter, gør man følgende for hver station s. Fra hver knude v^a_s tilføjes en kant til hver knude svarende til en afgang fra s, som er mindst 5 minutter senere end a (dvs. vi forudsætter, at der skal bruges mindst fem minutter på at skifte tog). Disse kanter kaldes *interne* kanter.

```
Århus H,       11:02
Skanderborg,   11:16
Horsens,       11:31
Vejle,         11:48
Fredericia,    12:09
Middelfart,    12:17
Odense,        12:45

Århus H,       12:02
Skanderborg,   12:16
Horsens,       12:31
Vejle,         12:48
Fredericia,    13:09
Middelfart,    13:17
Odense,        13:45

Esbjerg,       10:56
Bramming,      11:07
Vejen,         11:28
Lunderskov,    11:35
Kolding,       11:44
Fredericia,    11:58

Esbjerg,       11:56
Bramming,      12:07
Vejen,         12:28
Lunderskov,    12:35
Kolding,       12:44
Fredericia,    12:58

Esbjerg,       11:48
Vejen,         12:15
Kolding,       12:30
Middelfart,    12:45
Odense,        13:10
```

(a) En køreplan med tre linier



(b) Grafen svarende til køreplanen i (a). Kun nogle af kant-vægtene er angivet.

Figur 1: En køreplan og den tilsvarende graf

Vægten af en ekstern kant er forskellen i antal minutter på afgangstidspunkterne svarende til kantens to endepunkter. Vægten af en intern kant er forskellen i antal minutter på afgangstidspunkterne svarende til kantens to endepunkter + 15 minutter. Ud over vægten kan det være en god ide at opbevare information om, om kanten er intern eller ekstern, dvs. om den svarer til et togskift eller ej.

Figur 1(b) viser grafen svarende til køreplanen i Figur 1(a). For overskuelighedens skyld er kun nogle af kantvægtene angivet.

3 Programmet

Programmet skal være velstruktureret og kommenteret i passende omfang.

3.1 Afvikling

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

Input består af tre dele: en fil med køreplanen, navnet på startstationen og navnet på slutstationen. Køreplanen har samme format som vist i Figur 1(a). Dvs. hver linie starter med et stationsnavn efterfulgt af komma. Derefter kommer afgangstidspunktet, angivet som timer og minutter adskilt af et kolon. Efter hver endestation (bortset fra den sidste i filen) er der en blank linie.

Output skal være varighed, angivet i minutter, og antal skift for den fundne rute. De to tal skal adskilles af et enkelt mellemrum.

Hovedklassen skal hedde Rute. Man skal kunne afvikle programmet på følgende måde:

```
java Rute <inputfil> <startstation> <slutstation>
```

hvor filen `<inputfil>` indeholder køreplanen som beskrevet ovenfor.

Skriver man f.eks.

```
java Rute koereplan.txt Lunderskov Middelfart
```

hvor filen `koereplan.txt` indeholder køreplanen i Figur 1(a), skal output være

```
42 1
```

3.2 Køretid

Lad n betegne det samlede antal afgange fra alle stationerne tilsammen, og lad s være antallet af stationer. Lad a være det største antal afgange, der er fra nogen station. Grafen skal bygges i tid $O(\max\{n \log n, sa^2\})$. **Hint:** Lav f.eks. en ArrayList af knuder. Sorter denne efter stationsnavn og afgangstidspunkt (det er i orden at bruge en af JAVAs indbyggede sorteringsmetoder). Iterer over den sorterede ArrayList, når du tilføjer interne kanter.

Når grafen er bygget, skal programmet i tid $O(sa^2 \log(n))$ beregne varighed og antal skift for den ønskede rute. For at opnå denne køretid skal man bruge Dijkstras algoritme med prioritetskøen implementeret vha. en binær hob.

Den binære hob skal implementeres fra grunden vha. et array (eller evt. en ArrayList). Ligeledes skal du selv implementere datastrukturen for grafen. Dog er det i orden at bruge basale klasser fra JAVAs standardbibliotek, f.eks. hægtede lister til at implementere adjacenslisterne.

3.3 Et par yderligere hints

Der vil typisk være flere knuder svarende til start-stationen. Det kan være smart at starte med at sætte alle disse ind i prioritetskøen med prioritet 0. Dette svarer til at tilføje en ekstra knude s og en kant med vægt 0 fra s til hver knude, som repræsenterer start-stationen.

Et sidste lille hint: hvis der optræder kanter med negative vægte, skyldes det måske, at du har glemt at tage højde for kanter, der "krydser midnat". Det løses selvfølgelig ved at lægge 1440 til vægten (der er 1440 minutter på et døgn).

4 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`. 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`. Der skrives en linie for hvert test-eksempel. Hvis dit program giver et rigtigt resultat, skrives "OK" og antallet af sekunder, programmet var om at beregne resultatet. Hvis dit program ikke giver rigtigt resultat, skrives dit resultat og det rigtige (antal minutter, inkl. 15 minutter for hvert skift). Hvis dit program ikke giver et resultat inden for tre minutter, afbrydes det og køres på næste test-eksempel.

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

Det er vigtigt at køre DM507check, inden du afleverer. På den måde sikrer du bl.a., at dit program bruger det korrekte input- og output-format. Men bemærk, at testen kun indeholder nogle få test-eksempler, så hvis der ikke findes fejl, betyder det ikke nødvendigvis, at dit program fungerer korrekt.

5 Rapporten

Rapporten skal 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.

Argumentér for, at programmet har den ønskede køretid.

Der skal 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.

Eventuelle mangler i program eller rapport skal beskrives.

Endelig skal rapporten indeholde en udskrift af hele programmet. Denne udskrift skal være identisk med det elektronisk afleverede program.

På forsiden skal du skrive

- dit navn,
- dit brugernavn (login) på IMADAs maskiner samt
- navnet på din instruktor (Rojin eller Christian/Thies).

6 Aflevering

Programmet skal afleveres 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 skal afleveres via Blackboard. Husk, at dit navn og login samt din instruktors navn skal stå på forsiden af rapporten.

Begge dele skal afleveres senest **mandag d. 2. maj kl 10:00**.

7 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.

God arbejdslyst! ☺