Øving 8, algoritmer og datastrukturer

Vektede grafer

Implementer Edmonds-Karp algoritmen for maksimum flyt.

Format for graf-filer

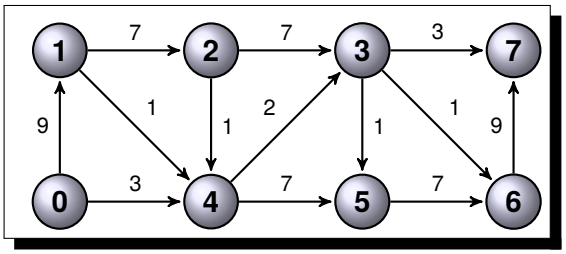
Filformat

```
Nodeantall Kantantall
franode tilnode kapasitet
franode tilnode kapasitet
franode tilnode kapasitet
...
```

flytgraf1

```
8 13
0 1 9
0 4 3
1 4 1
1 2 7
2 4 1
2 3 7
4 3 2
4 5 7
3 5 1
3 6 1
3 7 3
5 6 7
6 7 9
```

flytgraf1



Resultater

Programmet må legge frem resultatene fra algoritmen; hver flytøkende vei, hvor mye den øker flyten, og den maksimale flyten. Eksempel for grafen over, hvis node 0 er kilde og node 7 er sluk:

```
Maksimum flyt fra 0 til 7 med Edmond-Karp
Økning : Flytøkende vei
     2   0 4 3 7
     1   0 4 5 6 7
     1   0 1 2 3 7
     1   0 1 2 3 6 7
     1   0 1 4 5 6 7
     1   0 1 2 3 5 6 7
     1   0 1 2 4 5 6 7
     2   0 1 2 3 4 5 6 7
Maksimal flyt ble 10
```

Her ble det brukt 8 flytøkende veier. Legg merke til at flyten gjennom kant $4 \rightarrow 3$ snur til slutt.

Vektede grafer for maks flyt

flytgraf1	`http://www.iie.ntnu.no/fag/_alg/v-graf/flytgraf1`
flytgraf2	`http://www.iie.ntnu.no/fag/_alg/v-graf/flytgraf2`
flytgraf3	`http://www.iie.ntnu.no/fag/_alg/v-graf/flytgraf3`

Tips, feil i boka

Det er en feil i javakoden i boka på side 191. Linje 2 med «`Vkant neste;`» skal ikke være med. «Vkant» arver nemlig «neste» fra «Kant», og skal derfor ikke deklarere «neste» på nytt. (Slik det står, får «Vkant» *to* variabler kalt «neste». En av typen «kant», og en av typen «Vkant». Ved kjøring vil typisk innlesing fra fil oppdatere det ene feltet, mens algoritmen bruker det andre feltet som bare inneholder null. Dermed virker ingenting, og feilen er veldig vanskelig å begripe medmindre man forstår detaljene i hvordan arv virker i Java.)

Etter å ha fikset dette, vil Java protestere her og der på at «neste» er en «Kant» når det forventes en «Vkant». Dette ordnes med typecasting: `(Vkant)objekt.neste`

Tips om maks-flyt

Hver kant har en «kapasitet» som er vektingen i graf-fila. I tillegg må den ha et felt for «flyt», som kan oppdateres hver gang en ny flytøkende vei blir lagt til. Restkapasitet kan beregnes av en aksessmetode som trekker «flyt» fra «kapasitet».

Positiv flyt i en retning, er det samme som negativ flyt i motsatt retning. Derfor må alle kanter finnes i begge retninger, slik at flyt som legges til en vei, kan trekkes fra i motsatt retning. Da oppstår restkapasitet i den motsatte retningen, som programmet kanskje kan utnytte senere.

For at dette skal bli effektivt, er det lurt å la hver kant inneholde en referanse til den motsatte kanten. (Hvis en kant går fra node 2 til node 4, er den motsatte kanten fra node 4 til node 2.)

Ved innlesing kan man sjekke hver kant, for å se om den motsatte kanten finnes. I så fall kobles de sammen ved hjelp av referansene.

Etter innlesing må det opprettes motsatte kanter for alle kanter som ikke har noen motsatt kant fra før. Disse motsatte kantene får kapasitet 0. (De kan likevel få positiv RESTkapasitet, i tilfeller hvor de har negativ flyt.)

Det er lettere å finne den motsatte kanten i en graf implementert med nabotabell. Den motsatte kanten til kant[x][y] vil da være kant[y][x].

Edmond-Karp bruker bredde-først søk for å finne flytøkende veier. Dere kan ta utgangspunkt i BFS fra forrige øving. Men det må endres slik at det bare følger kanter som har restkapasitet > 0.

Når et BFS er ferdig, vil forgjengerkjeden fra sluknoden tilbake til kilden være den flytøkende veien. Den kanten langs veien som har lavest kapasitet, bestemmer hvor mye flyten kan økes langs veien.

For å øke flyten, økes flyten i alle kantene langs veien. For hver kant som får lagt til flyt, trekkes den samme flyten fra i den motsatte kanten.

Når flyten er øket, gjør man et nytt BFS for å finne en ny flytøkende vei, inntil det ikke er mulig å finne flere veier.