Notat for oblig 2, INF3/4130 h07

Dag Sverre Seljebotn

15. oktober 2007

Jeg har skrivd et noe langt notat for oblig 2 som interesserte kan se på. Merk at dette er kun for å gi et par tips (for oppgave 3 kanskje bare for spesielt interesserte), og det som står her skriver jeg for egen regning. Ting som står her har ingen innvirkning på hvordan en bør forstå obligen – ved motsetninger er det den offisielle oppgaveteksten som skal følges!

Oppgave 1

I oppgave 1 er det som nevnt bare ett riktig svar for hvert input. Det som kan være lurt for å sjekke svarene sine er å bruke en splaytresimulator som f.eks.

> `http://webpages.ull.es/users/jriera/Docencia/AVL/AVL%20tree%20applet.htm`

(NB! Jeg gir ingen garanti for at algoritmen her er identisk med den i boka, ved eventuelle avvik er det den i boka som skal følges. Kom gjerne med tilbakemeldinger på om det korresponderer godt nok.)

Oppgave 2

Som nevnt på forelesning og grupper har FordFulkerson-algoritmen den interessante egenskapen at den ikke bare gir en optimal løsning men et bevis for at denne er riktig, i form av et kutt der kapasiteten mellom delene er lik flyten (hvorfor dette beviser det kan en se i pensum). Da er det også, gitt et forsøk på løsning, ganske greit å validere hvorvidt det faktisk er en optimal løsning på $O(m^2)$ tid (lineær med hensyn på input):

- Løsningen må angi en flyt (dvs. flyt inn og ut av hver node må stemme).

- Flyten må passe med kapasitetsbegrensningene overalt.
- Flyten må være så stor som kuttet skulle tilsi.

I alle fall, poenget er at en greit kan lage et program som gitt et output fra algoritmen finner ut av om det er riktig svar, *uten* å bruke Ford-Fulkerson eller noen annen konstruktiv algoritme i det hele tatt. Og det har jeg altså gjort..last ned `oppgave2-validate.py` fra kurssidene (under "Ressurser for oblig 2"). Bruk:

```
$ ./oppgave2-validate.py myinput.txt mysolution.txt
Not OK. Errors:
        flow from source (7) != flow to sink (4)
        f does not obey capacity constraints of N
        flow size (7) != capacity over cut (4)
```

Ta en kikk på koden for detaljene.

Oppgave 3

For denne oppgaven er det lov å levere drøftingen på papir til gruppelærer dersom en synes det blir mye enklere å skrive eller tegne for hånd.

Når en skal løse oppgave 3 er det mulig å enten løse den med "muntlig", uformell drøfting eller bruke en mer formell og matematisk notasjon. Det er selvfølgelig opplagte fordeler og ulemper med begge metoder.

I denne oppgaven er det mye å holde rede på og som skal skilles på: Hva som ligger i tilstandsgrafen og hva som ligger i den "egentlige" grafen (som jeg vil kalle problemgrafen), ulikheten som skal vises for å vise monotonitet vil ha forskjellig n, k, $\mathbf{p}$ på hver side av ulikhetstegnet, og så videre.

Matematisk notasjon er godt egnet til å eliminere slike problemer som en ellers kanskje må slite litt for å holde rede på. Til gjengjeld kan det jo være uvant og få ting til å se uforståelig ut (men det er i høyeste grad en treningssak, ikke ulikt å lese programmeringsspråk). Så, for de som vil bruke matematisk notasjon men ønsker litt drahjelp har jeg skrivd dette notatet.

For de som bare vil drøfte oppgaven mer uformelt er det ikke sikkert at dette notatet vil gjøre annet enn å forvirre – i såfall, bare overse det! Det inneholder ikke egentlig noen flere tips til oppgaven, bare omformulerer den enda mer presist og innfører en notasjon en kan benytte i besvarelsen.

Tilsvarende kan det hende at dette er uinteressant for de som er godt vant til matematikk, fordi det "egentlig ikke står noe her" – det er først og

fremst ment som et eksempel på hvordan en kan lage seg notasjoner, og notasjon i seg selv er innholdsløst inntil en bruker den til noe.

Notasjon

Først: Sørg for at du tilegner deg stoffet under slik at det blir ditt eget, for *jeg garanterer på ingen måte at det ikke forekommer stygge feil*. Det er obligteksten som skal følges. Dersom du bruker notasjonen under i besvarelsen kan du bare referere til dette notatet og trenger ikke definere alt på nytt. Det kan jo også hende at du synes en annen notasjon er naturlig og det er greit så lenge en forklarer det – i matematikk finnes det ikke gal notasjon, bare gale resonnementer (og forsåvidt mer eller mindre *praktisk* notasjon).

Først har en problemgrafen bestånde av byene en kan reise til, som jeg vil bruke standard notasjon på: $G = (V, E)$ der V er nodene og E er kantene. E er kanskje ikke så interessant, for i denne oppgaven er grafen komplett (alle noder er forbundet med alle andre noder). Videre er grafen også urettet (samme avstand begge veier mellom to noder). Noder i grafen vil jeg bruke vanlige symboler for: $u \in V$.

Algoritmen en så bruker for å løse problemet består fundamentalt av å sjekke alle kombinasjoner av veier, men forhåpentligvis sjekke de i en smart rekkefølge og gjøre mange avskjæringer underveis. En bruker derfor en tilstandsgraf. Denne består av mulige tilstander en er i i algoritmen, og har størrelse som er eksponentiell i forhold til problemgrafen. Mengden av nodene i tilstandsgrafen kaller jeg T. Hver tilstand i T er en foreløpig vei gjennom problemgrafen som beskrevet i oppgaven, og jeg vil notere en slik vei med vektornotasjon: $\mathbf{p} \in T$.

Det er viktig å holde tunga rett i munnen her: Hver eneste node i T representerer en unik delvis travel handelsreisende-vei gjennom G. (Om en skulle tegne det opp kunne en tegnet opp en litt stor sirkel for hver node i T, og så tegnet opp hele G med en vei oppå inne i hver sirkel.)

Tilstandsgrafen er ikke komplett, en kan egentlig nå ganske få andre tilstander fra hver tilstand. Videre er tilstandsgrafen rettet: Selv om en kan komme seg til $\mathbf{r}$ fra $\mathbf{p}$ og $\mathbf{r}$ derfor er en nabo til $\mathbf{p}$ kan en ikke komme seg fra $\mathbf{p}$ til $\mathbf{r}$ så $\mathbf{p}$ er ingen nabo av $\mathbf{r}$ (mer under om lovlige overganger).

Som sagt kan en betrakte en tilstand som verken mer eller mindre enn en foreløpig vei gjennom grafen (som oppfyller kravene til en foreløpig travelling salesman-vei), så en $\mathbf{p} \in T$ representerer da både en tilstand og et "vei-objekt". En kan dermed snakke både om første og siste node av $\mathbf{p}$ ($\mathbf{p}$-som-vei, og en tenker i G) og av tilstander en kan nå fra $\mathbf{p}$ ($\mathbf{p}$-som-tilstand, og en tenker i T). Videre vil en slik vei bestå av en mengde noder,

så av og til vil jeg bruke $\mathbf{p}$ som bare mengder av noder i veien, slik: $\mathbf{p} \subset V$. Det vil gå fram av sammenhengen hva en mener. (Om en synes dette er rotete kan en evt. bare innføre funksjoner: $m(\mathbf{p})$ er mengden av noder som veien $v(\mathbf{p})$ til tilstanden $\mathbf{p}$ består av.)

Definerer så noen funksjoner:

- $|\mathbf{p}|$ er lengden på veien i antall noder
- $n(\mathbf{p})$ er antallet steg igjen på rundturen:
$$n(\mathbf{p}) = |V| - |\mathbf{p}| + 1$$
- $\ell(\mathbf{p})$ er siste node i veien $\mathbf{p}$. (ℓ skrives `\ell` i LaTeX).
- En heuristikk blir $h(\mathbf{p})$, og tar altså inn tilstander, $\mathbf{p} \in T$. Dvs
$$h : T \to \mathbb{R}$$
- La $\mathbf{p} \in T$. Andre lovlige tilstander i T en kan komme til fra $\mathbf{p}$ kaller en da naboer til $\mathbf{p}$; og disse er: Dersom $n(\mathbf{p}) = 0$ er en i en måltilstand og en har ingen naboer. Dersom $n(\mathbf{p}) = 1$ er eneste nabotilstand den som framkommer om en legger til s på slutten av veien $\mathbf{p}$. Dersom $n(\mathbf{p}) > 1$ er nabotilstander de som framkommer om en tar veien $\mathbf{p}$ og legger til en vilkårlig $v \in V \setminus \mathbf{p}$.
- Dersom $\mathbf{r}, \mathbf{p} \in T$ og $\mathbf{r}$ er en nabo-tilstand til $\mathbf{p}$ kan en snakke om kostnaden det har å gå fra $\mathbf{p}$ til $\mathbf{r}$ (ikke motsatt vei!), denne kaller vi $c(\mathbf{p}, \mathbf{r})$ og når den er definert vil den ha verdi
$$c(\mathbf{p}, \mathbf{r}) = d(\ell(\mathbf{p}), \ell(\mathbf{r}))$$

Henger en med til hit burde det være forholdsvis greit å løse oppgaven matematisk. Så til oppgavedefinisjonen: Finn ut hvorvidt heuristikken gitt i hver tilfelle har monotonitetsegenskapen, det vil si at dersom $\mathbf{p}, \mathbf{r} \in T$ og $\mathbf{r}$ kan nåes fra $\mathbf{p}$ så skal en finne ut om
$$h(\mathbf{p}) \leq h(\mathbf{r}) + c(\mathbf{p}, \mathbf{r})$$
alltid holder.

Heuristikk 1 Definerer først
$$I(\mathbf{p}) = \mathbf{p} \setminus \{s, \ell(\mathbf{p})\}.$$
Altså gir $I(\mathbf{p})$ mengden av det indre av $\mathbf{p}$ (mengden som framkommer om en fjerner første og siste element i $\mathbf{p}$). Så får vi:

$h_1(\mathbf{p}) =$ lengde av korteste vei fra $\ell(\mathbf{p})$ til s i G som ikke er innom noder i $I(\mathbf{p})$.

Heuristikk 2

$$\begin{aligned} Q(\mathbf{p}) &= \{d(u,v) : u \in V, v \in V \setminus \mathbf{p}\} \\ q(\mathbf{p}) &= \min Q(\mathbf{p}) \\ h_2(\mathbf{p}) &= n(\mathbf{p}) \cdot q(\mathbf{p}) \end{aligned}$$

Den første linja er altså en funksjon som gitt en tilstand returnerer en *mengde* med reelle tall, og er definert for seg fordi...vel, det kan jo være greit å ha...

Heuristikk 3 Definerer først (gjør bruk av I fra heuristikk 1):

$$U(\mathbf{p}) = V \setminus I(\mathbf{p})$$

Så får vi

$$h_3(\mathbf{p}) = \sum_{u \in U(\mathbf{p})} \min_{v \in V \setminus \{u\}} \{d(u,v)\}$$