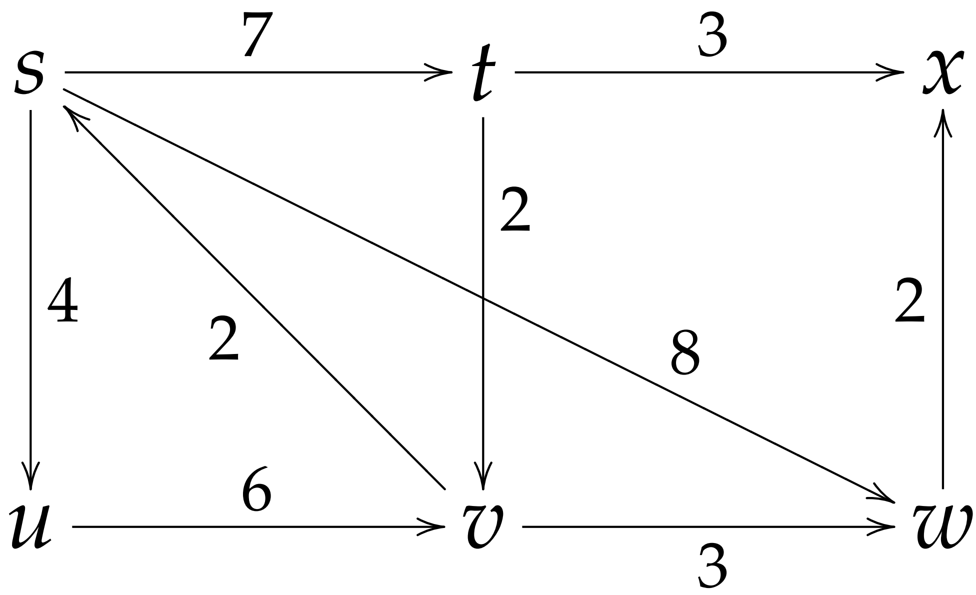
Teoriøving 7 + litt om Ford-Fulkerson

Magnus Lie Hetland

Oppgave 1 a



Bruk DIJKSTRA eller BELLMAN-FORD og finn minste avstand fra s til de andre nodene.

Svar/utregning (DIJKSTRA):

s	t	u	v	w	x
0	∞	∞	∞	∞	∞
–	7	**4**	∞	8	∞
–	**7**	–	10	8	∞
–	–	–	9	**8**	10
–	–	–	**9**	–	10
–	–	–	–	–	**10**
–	–	–	–	–	–

Oppgave 1 b

I hvilken rekkefølge vil nodene bli plukket ut dersom man bruker Dijkstras algoritme?

Svar: s, u, t, w, v, x

Oppgave 1 c

$$\begin{tikzcd}
s \arrow[r, "7"] \arrow[d, "4"] \arrow[drr, "8"] & t \arrow[r, "3"] \arrow[d, "2"] & x \\
u \arrow[r, "6"] & v \arrow[r, "-3"'] \arrow[ul, "-8"] & w \arrow[u, "-4"']
\end{tikzcd}$$

Bruk en algoritme og finn ut minste avstand fra s til de andre nodene.

Bruk f.eks. BELLMAN-FORD, som takler negative vekter. Grafen har ingen negative sykler.

Svar: $d = [0, 7, 4, 9, 6, 2]$

Oppgave 2 a

Hvilket problem løser DIIKSTRA og BELLMAN-FORD slik de blir presentert i pensum? løser hvilket problem?

Svar: Korteste vei fra én startnode til alle andre.

Oppgave 2 b

Hvilket problem løser FLOYD-WARSHALL slik den blir presentert i pensum?

Svar: Korteste vei fra alle noder til alle andre.

Oppgave 2 c

hvilken algoritme bruker man da for å finne korteste vei fra en node til en annen hvis grafe inneholder negative sykler?

Dijkstra, BELLMAN-FORD, FLOYD-WARSHALL eller ingen av delene?

Svar: I utgangspunktet er ikke problemet veldefinert når vi har negative sykler. Dermed kan ingen av algoritmene gi oss noe svar.

Livekel: Hvis vi kun er interessert i veien fra én node til en annen vil vi i enkelte tilfeller kunne finne en slik vei, dersom det er umulig å gå via den negative løkken. (Vi kan bruke BELLMAN-FORD og håpe det beste…)

Oppgave 2 d

Hvilken kjøretid har Dijkstras algoritme (Q implementert som en lineær array)?

Svar: $O(V^2)$

For hver node vi tar ut av Q er kjøretiden $O(|Q|)$ for å finne minimum. Dette gir $\sum_{i=1}^{V} i = O(V^2)$. Vi må i tillegg kjøre RELAX på alle kanter, så svaret blir $O(V^2 + E) = O(V^2)$.

Oppgave 2 e

Hvilken kjøretid har BELLMAN-FORD?

Svar: $O(VE)$

Vi kjører RELAX på alle kanter $|V| - 1$ ganger for å sikre at alle korteste veier (maks. $|V| - 1$ kanter) har fått kjørt RELAX på alle sine kanter i riktig rekkefølge.

Oppgave 2 f

Hva gjør INITIALIZE-SINGLE-SOURCE(G, S)?

Svar: Den sørger for at estimatet d er et gyldig overestimat og at foreldre-treet (π) er tomt. Med andre ord: $d[v] = \infty, \pi[v] = \text{nil}, d[s] = 0$

Oppgave 3 a

Hva skjer dersom man bytter ut **while** $Q \neq \emptyset$ med **while** $|Q| > 1$ i linje 4 i DIJKSTRA?

INITIALIZE-SINGLE-SOURCE(G, s)
$S \leftarrow \emptyset$
$Q \leftarrow V[G]$
while $Q \neq \emptyset$
  $u \leftarrow$ EXTRACT-MIN(Q)
  $S \leftarrow S \cup \{u\}$
  foreach $v \in Adj[u]$
    RELAX(u, v, w)

Svar: DIJKSTRA vil gi rett svar da siste element i køen umulig kan være forbedre den foreløpige løsningen. (Se f.eks. utregning i 1a.)

Oppgave 4 a

Hva gjør DIJKSTRA dersom man forandrer RELAX til det følgende?

$$
\begin{aligned}
&\textbf{if } d[v] > \max\{d[u], w(u,v)\} \\
&\qquad d[v] \leftarrow \max\{d[u], w(u,v)\} \\
&\qquad \pi[v] \leftarrow u
\end{aligned}
$$

Svar: Finner veien fra en node til en annen der maksimalavstanden mellom to noder er minimal.

Med denne nye RELAX vil $d[u]$ ta vare på den maksimale vekten mellom to noder, og sørge for at denne er så liten som mulig.

Kan man bruke det samme korrekthetsbeviset her?

Oppgave 4 b

Hva er kjøretiden for algoritmen i **a** hvis Q implementeres som en binær heap? (Du kan her anta at det er mulig å finne en vei fra source til alle noder i grafen.)

Svar: $O(E \log V)$

Nå tar det $O(\log V)$ tid å hente ut minimum, så dette koster $O(V \log V)$. I tillegg tar RELAX nå $O(\log V)$ tid, og gjøre én gang per kant. Totalt blir kjøretiden altså $O((E + V) \log V)$ som er $O(E \log V)$ hvis alle noder kan nås fra start-noden.

Oppgave 4 c

Hva er forholdet mellom kjøretidene A: $\Theta(E \log V)$ og B: $\Theta(V^2)$

Svar: A vil gå fortere for enkle grafer uten mange forbindelser.

Grunnen til dette er at $|E|$ da vil være nærmere $|V|$ enn $|V|^2$.

Flyt-regler

Kapasitet: Flyten i en kant kan ikke overgå kapasiteten.

Symmetri: Flyt i motsatt retning har motsatt fortegn.

Konservering: Ingen noder (unntatt kilde og sluk) konsumerer eller produserer flyt; inn = ut.

Symmetrien er et kinking punkt. For å sitere Bart Simpson:

> I didn't think it was physically possible,
> but this sucks *and* blows!

Maks-flyt

En opplagt måte å finne maksimal flyt på er følgende:

1. Start uten flyt

2. Finn en vei fra kilde til sluk der du kan dytte igjennom mer flyt

3. Hvis en slik vei ikke finnes så er vi ferdige; ellers, dytt flyten igjennom og gå til punkt 2.

Dette er den generelle FORD-FULKERSON-metoden. Hvis vi bruker bredde-først-søk for å finne veien fra kilde til sluk får vi en algoritme med polynomisk kjøretid (EDMONDS-KARP).

Hvordan finne en vei

Det er her vi kommer borti symmetrien ("this sucks *and* blows"). Den flytforøkende veien kan gå **mot pilenes retning**!

$$u \xrightarrow{4/10} v$$

Flyten fra u til v er positiv, og kan økes med 6 eller reduseres med 4.

Flyten fra v til u er negativ, og kan økes med 4 eller reduseres med 6.

Vi vil kun *øke* flyten, men ved å øke den negative flyten reduserer vi den positive (og omvendt)!

Vi leter etter en vei fra s til t der vi kan øke flyten i alle kantene, i **veiens** retning, ikke nødvendigvis **kantenes**.

Et eksempel

Er dette en flytforøkende vei (vi ser bort fra resten av grafen)? Hvor mye kan flyten økes med?

$$s \xrightarrow{1/6} u \xleftarrow{4/7} v \xrightarrow{2/8} t$$

Ja, det er en flytforøkende vei; flyten kan økes med 5 fra s til u, med 4 fra u til v, og med 6 fra v til t. Totalt altså med 4.

Hvordan kan dette være riktig? Hva kan vi si om flyt i andre kanter tilknyttet u og v? Hvis vi øker flyten fra s til u og flyten fra u til v med samme mengde, hva skjer med flyt-konserveringen i u? Vil det samme skje enten kanten går fra u til v eller fra v til u?

Tips

- Glem residualnettverk (i første omgang); de er ment som et pedagogisk virkemiddel, men kan virke forvirrende

- Ikke få angst av at flyten går "motstrøms"

- Lær deg flyt-reglene

- Tenk igjennom hvordan du finner en flytforøkende vei (med f.eks. DFS eller BFS) uten å bruke residualnettverk

- Tenk igjennom hvordan du vil oppdatere flyten når du har funnet en flytforøkende vei