Werkcollege Datastructuren, Maart 27, 2007
(Uitwerkingen van twee niet behandelde opgaven)

Clemens Grabmayer (mailto:clemens@phil.uu.nl), 2 april 2007.

R-13.6. *Bob loves foreign languages and wants to plan his course schedule for the following year. He is interested in the following nine language courses: LA15, LA16, LA22, LA31, LA32, LA126, LA127, LA141, LA169. The course prerequisites are:*

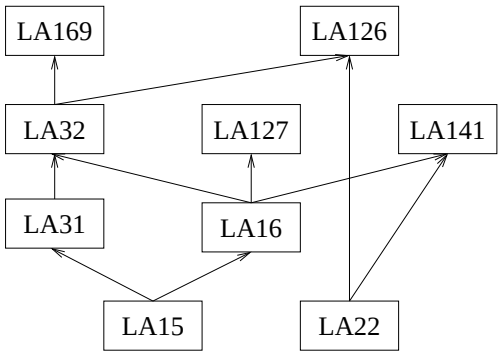
- *LA15: (none)*
- *LA16: LA15*
- *LA22: (none)*
- *LA31: LA15*
- *LA32: LA16, LA31*
- *LA126: LA22, LA32*
- *LA127: LA16*
- *LA141: LA22, LA16*
- *LA169: LA32*

Find a sequence of courses that allows Bob to satisfy all the prerequisites.

De 9 cursussen vormen, voor $V = \{\text{LA15}, \text{LA16}, \ldots, \text{LA169}\}$, en samen met de relatie

$$\alpha \to \beta \iff_{\text{def}} \text{ cursus } \alpha \text{ is een vereiste voor cursus } \beta \qquad (\text{voor alle } \alpha, \beta \in V\,),$$

een gerichte graaf $G = (V, \to)$ die als volgt kan worden getekend:



Figuur 1: De graaf G met knopen LA15, LA16, ..., LA169, en met takken die de vereiste-relatie $\to$ beschrijven.

We zoeken een rij $c_1, c_2, \ldots, c_9$ van cursussen $c_i \in V$, voor $i \in \{1, 2, \ldots, 9\}$, die paarsgewijs verschillen (d.w.z. voor alle $i, j \in \{1, 2, \ldots, 9\}$ geldt $c_i \neq c_j$ als $i \neq j$), en met de eigenschap

$$c_i \to c_j \implies i < j \qquad (\text{voor alle } i, j \in \{1, 2, \ldots, 9\}) \,. \tag{1}$$

Met andere woorden: we zoeken een totale strikte ordening $<$ op de verzameling V zodat

$$u \to v \implies u < v \qquad (\text{voor alle } u, v \in V) \,. \tag{2}$$

(Dat is om twee redenen: Ten eerste definieert iedere totale strikte ordening $<$ op V met de eigenschap (2) een rij $c_1, c_2, \ldots, c_9$ van paarsgewijs verschillende cursussen met de eigenschap (1). En ten tweede induceert andersom iedere rij $c_1, c_2, \ldots, c_9$ van paarsgewijs verschillende cursussen met de eigenschap (1) een totale strikte ordening op V met (2).) Zo een ordening is een *topologische ordening* op de gerichte graaf G in de zin van de definitie op blz. 616 in Goodrich and Tamassia's book "Data Structures and Algorithms in Java".

We berekenen een topologische ordening $<$ op G m.b.v. het topologische sorteeralgorime TopologicalSort op blz. 617 in Goodrich/Tamassia (waarbij stapsgewijs knopen zonder inkomende takken worden verwijderd, en altijd ook de takken vanuit een verwijderde knoop worden weggehaald). Op deze manier vinden we bijvoorbeeld de volgende topologische ordening op G:

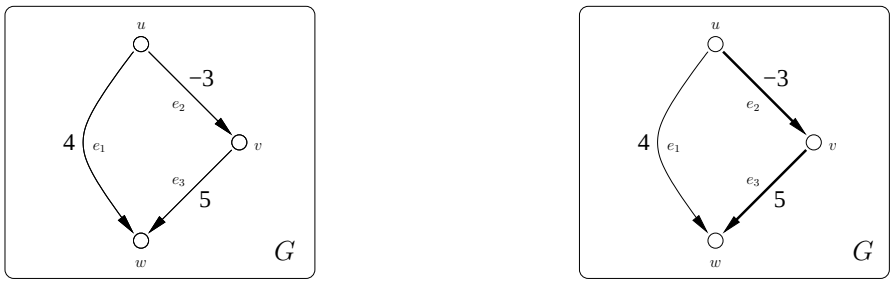
$$\text{LA15} < \text{LA22} < \text{LA31} < \text{LA16} < \text{LA32} < \text{LA127} < \text{LA141} < \text{LA169} < \text{LA126}$$

We concluderen dus uiteindelijk dat *één van de mogelijkheden* voor Bob om de taalcursussen te volgen daarin bestaat om ze in de volgorde LA15, LA22, LA31, LA16, LA32, LA127, LA141, LA169, LA126 te doorlopen.

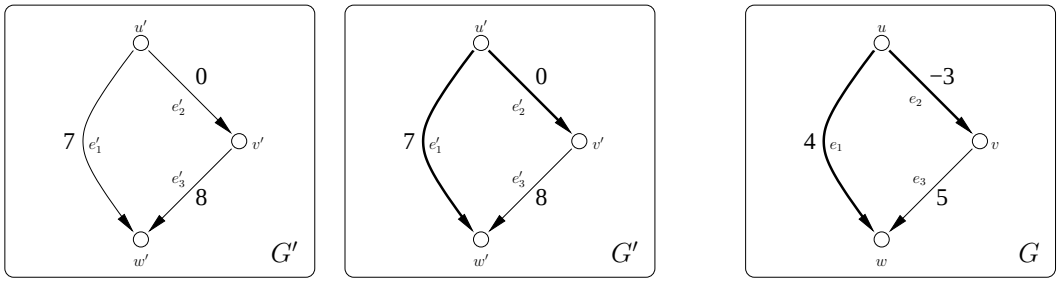
Extra Vraag. *Dijkstra's algoritme werkt alleen voor takken met niet-negatieve gewichten. Een naief idee zou kunnen zijn om als de kleinste negatieve tak de waarde $-n$ heeft, bij alle takken n op te tellen, dan Dijkstra's algoritme uit te voeren, en nadien overal n weer af te trekken. Laat zien dat dat niet werkt.*

We bekijken de gewogen graaf $G = (V, E, w)$ links in Figuur 2 met verzameling knopen $V = \{u, v, w\}$, verzameling takken $E = \{e_1, e_2, e_3\}$ en takkengewichten $w(e_1) = 4$, $w(e_2) = -3$, en $w(e_3) = 5$. De dikkere takken in de tekening van G rechts in Figuur 2 geven aan de kortste paden in G vanuit knoop u naar de knopen v en w. (Deze kortste paden zijn hier bepaald zonder Dijkstra's algoritme te hebben gebruikt.) In het bijzonder zien we dat het kortste pad vanuit u naar w via v loopt.

Door bij de gewichten van G de absolute waarde van het kleinste negatieve gewicht van een tak in G op te tellen (hier is dat de absolute waarde van -3), verkrijgen we de gewogen graaf $G' = (V', E', w')$ links in Figuur 3 met verzameling knopen $V' = \{u', v', w'\}$, verzameling takken $E' = \{e'_1, e'_2, e'_3\}$ en takkengewichten $w(e'_1) = 7$, $w(e'_2) = 0$, en $w(e'_3) = 8$. Als we nu Dijkstra's algoritme uitvoeren op de gewogen graaf G', vinden we de kortste-paden *spanning tree* in G' die in het midden van Figuur 3 is afgebeeld. In het bijzonder zien we dat het kortste pad vanuit u' naar w' via de directe tak vanuit u' naar w' loopt (en dus niet via de takken naar en vanuit de knoop v'). Als we deze kortste-paden-oplossing op G' naief zouden terugvertalen naar G (zie rechts in Figuur 3), dan zouden we krijgen dat ook op G het kortste pad vanuit u naar w via de directe tak van u naar w loopt.



Figuur 2: De graaf G zonder (links) en met (rechts) benadrukte kortste paden vanuit knoop u.



Figuur 3: De graaf G' zonder (links) en met (midden) benadrukte kortste paden vanuit knoop u'. Rechts een 'terugvertaling' naar de graaf G van de kortste-paden-oplossing in de graaf G'.

Maar dat is natuurlijk niet het geval (zoals we eerder hebben opgemerkt). En daarom is de in de vraag beschreven naieve idee om Dijkstra's algoritme toepasbaar te maken voor grafen die ook negatieve gewichten mogen bevatten, niet goed.

Opmerking: Wat hier mis gaat, is het volgende: De kortste paden in een gerichte, gewogen graaf blijven —in het algemeen— juist *niet* behouden als men bij het gewicht van een iedere tak dezelfde getal optelt. De reden daarvoor is dat paden die over vele takken lopen, door de opteloperatie worden benadeeld t.o.v. paden die maar over enkele takken lopen. Deze observatie geldt ook voor grafen die alleen maar positieve gewichten aan hun takken toekennen.