

## **Grafentheorie voor Telematica (151068)**

hoorcollegedocent: Hajo Broersma  
TW B116, tel. 3443

werkcollegedocenten: Kern  
Ruizenaar  
Vos

dictaat: nr. 556

schriftelijk tentamen: do 22 maart 2001

## Grove indeling colleges

Hfdst.	Onderwerp	# hc	# wc
1	Modellering	$\frac{1}{2}$	1
2	Basisbegrippen	1	$1\frac{1}{2}$
3	Samenhang	$2\frac{1}{2}$	$1\frac{1}{2}$
4	Meer samenhang	1	2
5	Stromen	$1\frac{1}{2}$	1
6	Puntkleuringen	2	2
7	Matchings	$1\frac{1}{2}$	2
8	Lijnkleuringen	1	1
	Vragenuur	1	
	Totaal # coll.	12	12

# Modellering m.b.v. grafen

- verzameling objecten
- relaties tussen paren objecten
- abstract model: graaf  $G = (V, E)$
- $V$ : punt per object
- $E$ : lijn per relatie

## Definities.

Een *graaf*  $G$  is een paar  $(V(G), E(G))$  waarbij  $V(G)$  een eindige niet-lege verzameling is (de *punten* van  $G$ ) en  $E(G)$  een verzameling van ongeordende paren  $\{u, v\}$  met  $u, v \in V(G)$  en  $u \neq v$  (de *lijnen* van  $G$ ). Als geen misverstand dreigt, schrijven we  $uv$  i.p.v.  $\{u, v\}$ . Als duidelijk is welke graaf  $G$  we beschouwen, gebruiken we  $V$  en  $E$  i.p.v.  $V(G)$  en  $E(G)$ .

Zij  $G = (V, E)$  een graaf. Twee punten uit  $V$  die als paar in  $E$  voorkomen noemen we *buren*. Een lijn is *incident* met een punt als het punt één van de elementen van het betreffende paar is. Het punt heet dan ook incident met de lijn. Twee lijnen zijn *buurlijnen* als de betreffende paren precies één element gemeen hebben. De punten  $u$  en  $v$  noemen we de *eindpunten* van de lijn  $uv$ .

De *buurverzameling* van een punt  $v \in V$ , notatie  $N(v)$ , is de verzameling buren van  $v$ , d.w.z.  $N(v) = \{u \in V \mid uv \in E\}$ .

De *graad* van een punt  $v \in V$ , notatie  $d(v)$ , is het aantal buren van  $v$ , d.w.z.  $d(v) = |N(v)|$ .

Met  $\delta(G)$  of  $\delta$  duiden we de kleinste graad aan die in  $G$  voorkomt, met  $\Delta(G)$  of  $\Delta$  de grootste graad.

Een punt met graad 0 noemen we een *geïsoleerd punt*, een punt met graad 1 een *eindpunt* (van  $G$ ).

**Propositie.** Voor elk punt  $v \in V$  geldt

$$0 \leq \delta(G) \leq d(v) \leq \Delta(G) \leq |V| - 1.$$

**Stelling.**  $\sum_{v \in V} d(v) = 2|E|$ .

Een graaf heet *k-regulier* als de graad van alle punten  $k$  is. Een graaf heet *regulier* als er een getal  $k$  bestaat zo dat hij  $k$ -regulier is.

Een *deelgraaf*  $G' = (V', E')$  van  $G = (V, E)$  is een graaf met  $V' \subset V$  en  $E' \subset E$ . Als  $G'$  een deelgraaf is van  $G$ , dan heet  $G$  een *supergraaf* van  $G'$ . Een deelgraaf  $G' = (V', E')$  van  $G = (V, E)$  is een *opspannende deelgraaf* als  $V' = V$ .

Zij  $S$  een deelverzameling van  $V$  of  $E$ . De *door  $S$  geïnduceerde deelgraaf* van  $G$ , notatie  $G[S]$ , wordt als volgt gedefinieerd:

- (i) als  $S \subset V$ : de deelgraaf met puntenverzameling  $S$  en lijnenverzameling  $\{uv \in E \mid u, v \in S\}$ ;
- (ii) als  $S \subset E$ : de deelgraaf met lijnenverzameling  $S$  en puntenverzameling  $\{v \in V \mid v \text{ is incident met een lijn uit } S\}$ ;

De graaf  $H$  is een *geïnduceerde deelgraaf* van  $G$  als  $H = G[S]$  voor zekere  $S \subset V$ .

Zij  $G$  een graaf,  $e \in E(G)$  en  $v \in V(G)$ .

Dan is  $G - e$  de graaf die uit  $G$  ontstaat door  $e$  te verwijderen;

$G - v$  ontstaat uit  $G$  door  $v$  én alle lijnen van  $G$  incident met  $v$  te verwijderen.

Op dezelfde manier definiëren we  $G - S$  als  $S \subseteq E(G)$  of  $S \subseteq V(G)$  door verwijdering van resp. alle lijnen van  $S$  uit  $G$ , of alle punten van  $S$  en alle lijnen incident met punten van  $S$ .

Als  $u, v \in V(G)$  en  $e = uv \notin E(G)$ , dan is  $G + e$  de graaf die uit  $G$  ontstaat door de lijn  $e$  toe te voegen.

Als  $v \notin V(G)$ , dan is  $G * v$  de graaf die uit  $G$  ontstaat door punt  $v$  toe te voegen plus alle lijnen tussen  $v$  en de punten van  $G$ .

In het volgende is  $G = (V, E)$  een graaf en  $u, v \in V$ .

Een  $(u, v)$ -wandeling of een wandeling van  $u$  naar  $v$  is een eindige rij  $v_0e_1v_1e_2 \dots v_{k-1}e_kv_k$  bestaande uit afwisselend punten en lijnen van  $G$  zo dat  $v_0 = u$ ,  $v_k = v$  en  $e_i = v_{i-1}v_i$  voor  $i = 1, \dots, k$ . Het getal  $k$  is de *lengte* van de wandeling. Een  $(u, v)$ -wandeling heet *gesloten* als  $u = v$ .

### **Opmerkingen.**

Aangezien de rij  $v_0e_1v_1e_2 \dots v_{k-1}e_kv_k$  die in de definitie voorkomt, eenduidig is bepaald door de rij  $v_0v_1 \dots v_k$ , volstaat men meestal met alleen deze rij punten te geven. De lengte van een wandeling is het aantal lijnen daarin, waarbij elke lijn even vaak geteld wordt als hij in de wandeling voorkomt. Het aantal punten, mits ook met multipliciteiten geteld, is altijd 1 groter dan de lengte.

Een wandeling heet een *route* als alle lijnen verschillend zijn.

Een wandeling heet een *pad* als alle punten (en dus alle lijnen) verschillend zijn.

Een gesloten wandeling  $v_0v_1 \dots v_k$  (met  $v_0 = v_k$  dus) heet een *cykel* als  $k \geq 3$  en  $v_1, \dots, v_k$  verschillend zijn.

**Stelling.** Een graaf waarin alle graden 2 of groter zijn, bevat een cykel.

De punten  $u$  en  $v$  zijn *verbonden* in  $G$  als er een  $(u, v)$ -wandeling bestaat.

**Stelling.** Als  $u$  en  $v$  verbonden zijn, bestaat er een  $(u, v)$ -pad.

Het *pad*  $P_n$  ( $n \geq 1$ ) is de graaf op  $n$  punten waarvan alle punten en alle lijnen tot één pad behoren.

De *cykel*  $C_n$  ( $n \geq 3$ ) is de graaf op  $n$  punten waarvan alle punten en lijnen tot één cykel behoren. Een cykel heet *even* of *oneven* al naar gelang het aantal punten even of oneven is.

Het *wiel*  $W_n$  ( $n \geq 4$ ) is de graaf die men kan verkrijgen uit de cykel  $C_{n-1}$  door een nieuw punt toe te voegen evenals de  $n - 1$  lijnen van het nieuwe punt naar alle punten van de cykel.

De *volledige graaf*  $K_n$  ( $n \geq 1$ ) bestaat uit  $n$  punten en daartussen alle lijnen.

$K_1$  heet ook wel de *triviale graaf*.

Een *lege graaf* is een graaf zonder lijnen.

Een *bipartiete graaf* is een graaf waarvan de puntenverzameling in twee disjuncte verzamelingen  $V_1$  en  $V_2$  is te splitsen, zo dat iedere lijn van de graaf een eindpunt in  $V_1$  en een eindpunt in  $V_2$  heeft.

De *volledige bipartiete graaf*  $K_{m,n}$  ( $m, n \geq 1$ ) heeft zo'n splitsing met  $|V_1| = m$ ,  $|V_2| = n$ , terwijl de lijnenverzameling bestaat uit *alle* lijnen van de punten in  $V_1$  naar de punten in  $V_2$ .

**Stelling.** Een graaf is bipartiet dan en slechts dan als alle cykels even zijn.

De (*d-dimensionale*) *kubusgraaf*  $Q_d$  ( $d \geq 1$ ) is de graaf met

$$V = \{ (e_1, \dots, e_d) \mid e_i \in \{0, 1\} \ (i = 1, \dots, d) \},$$

waarin twee punten buren zijn als de betreffende rijen op precies één plaats verschillen.

## Bomen

**Definities.** Een *bos* is een graaf zonder cyclen.

Een *boom* is een bos waarin ieder tweetal punten verbonden is.

**Stelling.** Een boom op  $n$  punten heeft  $n - 1$  lijnen.

**Stelling.** Tussen ieder tweetal punten van een boom bestaat een uniek pad.

**Stelling.** Als men uit een boom  $T$  met minstens één lijn een lijn  $e$  weglaat, ontstaat een bos van twee bomen.

**Stelling.** Als we aan een boom  $T$  een lijn toevoegen tussen twee niet-buren, ontstaat een graaf met precies één cykel.

**Definitie.** Een opspannende deelgraaf van  $G$  die een boom is, heet een *opspannende boom* van  $G$ .

**Stelling.** Een graaf  $G$  waarin ieder tweetal punten verbonden is, heeft een opspannende boom, en omgekeerd.

## Samenhang

**Definitie.** Een graaf is *samenhangend* als ieder tweetal punten verbonden is, d.w.z. als er tussen elk tweetal punten een pad bestaat.

**Definitie.** Een *component* van  $G$  is een maximale samenhangende deelgraaf van  $G$ .

**Opmerking.** Punten waartussen een wandeling (pad) bestaat, behoren dus tot dezelfde component. Die component is te bepalen door bij één punt te beginnen, alle (eventuele) burens en de tussenliggende lijnen toe te voegen, daarna de (eventuele) burens van die nieuwe punten plus alle tussenliggende lijnen, etcetera, totdat er geen nieuwe toe te voegen punten of lijnen meer zijn. Dit stopt met de component van de graaf waarin het beginpunt ligt. Dit is de hele graaf dan en slechts dan als die samenhangend is.

**Stelling.** Als in een graaf alle graden 2 of kleiner zijn, zijn alle componenten cykels of paden.

**Definities.** Als  $u$  en  $v$  verbonden zijn, dan is de *afstand* van  $u$  en  $v$ , notatie  $d(u, v)$ , de lengte van een kortste pad van  $u$  naar  $v$ . Als  $u$  en  $v$  niet verbonden zijn, definiëren we  $d(u, v) = \infty$ .

Als  $G$  samenhangend is, dan is de *diameter* van  $G$ , notatie  $d(G)$ , de maximale waarde van  $d(u, v)$  over alle paren  $u, v \in V$ .

**Stelling.** Voor een samenhangende bipartiete graaf zijn de verzamelingen  $V_1$  en  $V_2$  uit de definitie eenduidig bepaald (op naamgeving na).

De eenduidigheid gaat verloren voor een niet-samenhangende graaf.

## Kortste paden

**Kortste paden in gewone grafen.** Het bepalen van een kortste pad tussen twee punten  $u$  en  $v$  in een samenhangende graaf  $G$ , d.w.z. met lengte  $d(u, v)$ , is eenvoudig. Begin bij  $u$ . Als  $v = u$ , dan zijn we klaar; de lengte van het kortste pad is  $d(u, v) = 0$ . Als  $v \neq u$ , bekijk dan alle punten op afstand 1 van  $u$ , d.w.z. alle buren van  $u$ , ofwel de verzameling  $N(u)$ . Als  $v \in N(u)$ , dan is er een pad met lengte  $d(u, v) = 1$ . Als  $v \notin N(u)$ , bekijk dan de punten op afstand 2 van  $u$ , d.w.z. alle punten (uitgezonderd  $u$ ) die buren zijn van de buren van  $u$ , maar zelf geen buren zijn van  $u$ . Als  $v$  hierbij zit, dan is  $d(u, v) = 2$ ; zo niet, dan gaan we verder met punten op afstand 3, etcetera. Het is duidelijk dat dit na een eindig aantal stappen stopt en dat een kortste pad tussen  $u$  en  $v$  gevonden wordt.

**Opmerking.** In een meer realistische toepassing speelt niet de afstand tussen twee punten in de graaf een grote rol, maar juist de werkelijke afstand in het netwerk. Men wil in die situaties de informatie over de werkelijke afstanden niet verliezen in het wiskundig model. We kunnen dit ondervangen door de definitie van graaf in die situaties uit te breiden.

**Definitie.** Een *gewogen graaf* is een graaf waarvan de lijnen van reële getallen zijn voorzien, de zogenoemde *gewichten* van de lijnen.

Een 'gewone' graaf is op te vatten als een gewogen graaf waarin alle lijnen gewicht 1 hebben.

**Probleemstelling.** In een computernetwerk dat  $n$  computers met elkaar verbindt, willen we een kortste route tussen twee gegeven computers bepalen, in het geval de lengtes van de directe verbindingen gegeven zijn.

**Modellering.** We kunnen dit voorstellen door een gewogen graaf  $G$ . Een punt van  $G$  representeert een computer, een lijn de verbinding tussen twee computers en het gewicht van de lijn de lengte van die verbinding.

**Definities.** Het gewicht van een lijn  $e$  noteren we als  $w(e)$ . Voor een deelgraaf  $H$  van  $G$  definiëren we  $w(H) = \sum_{e \in E(H)} w(e)$ . Onder de *lengte* van een pad verstaan we het gewicht van dat pad. De *afstand* tussen twee punten  $u$  en  $v$  is de minimale lengte van een  $(u, v)$ -pad. We spreken af dat  $w(uv) = \infty$  als  $uv \notin E(G)$ .

We beschrijven een algoritme om de afstand tussen een vast punt  $u_1$  en een willekeurig ander punt te berekenen. Daartoe kennen we labels toe aan de punten van  $G$ .

1. Maak  $S = \{u_1\}$ ,  $l(u_1) = 0$  en  $l(v) = w(u_1v)$  voor  $v \neq u_1$ .
2. (*Toekenning definitief label*).  
Bereken  $\min\{l(v) \mid v \in \bar{S}\}$ , waarbij  $\bar{S} = V(G) \setminus S$ , en zij  $u$  een punt waarvoor dit minimum wordt aangenomen. Vervang  $S$  door  $S \cup \{u\}$ . Stop als  $S = V(G)$ . Ga in het andere geval naar stap 3.
3. (*Herziening voorlopige labels*).  
Vervang voor elk punt  $v \in \bar{S}$   $l(v)$  door  $\min\{l(v), l(u) + w(uv)\}$ . Ga naar stap 2.

## **Stelling.**

Laat  $S_i, u_i$  en  $l_i(v)$  de 'waarden' van  $S, u$  en  $l(v)$  zijn nadat stap 2 van de algoritme  $i - 1$  maal is uitgevoerd ( $v \in V(G)$ ,  $i = 1, \dots, n$ ).

Dan geldt voor alle  $v \in S_i$  dat  $d(u_1, v) = l_i(v)$ .

**Gevolg.** Voor  $i = n$  impliceert de stelling de correctheid van de algoritme van Dijkstra via de constatering dat  $S_n = V(G)$ .

We willen een methode om algoritmen met elkaar te vergelijken wat rekensnelheid betreft.

**Definitie.** Een *algoritme* is een precieze en ondubbelzinnige rij instructies die in een eindig aantal stappen leidt tot de oplossing van een probleem.

### **Voorbeeld (faculteits-algoritme).**

1. Maak  $i = 1$  en  $fac = 1$ .
2. Vervang  $fac$  door  $fac \cdot i$ .
3. Stop als  $i = n$ . Vervang in het andere geval  $i$  door  $i + 1$  en ga terug naar stap 2.

De algoritme gebruikt  $t(n) = 3n - 1$  'operaties'. We noemen in dit geval  $n$  de *probleemgrootte* en  $t(n)$  de *complexiteit* van de algoritme.

Als we de prestaties van twee algoritmen willen vergelijken, zullen we vnl. geïnteresseerd zijn in het gedrag voor grote waarden van de probleemgrootte: een algoritme is beter dan een andere als het voor steeds groter wordende inputs minder rekenstappen gebruikt. De volgende definitie maakt dit hard.

**Definitie.** Als  $f : N^* \rightarrow R_+$  en  $g : N^* \rightarrow R_+$  (complexiteits)functies zijn, dan zeggen we dat  $f$  *asymptotisch gedomineerd* wordt door  $g$ ,

notatie  $f(n) = O(g(n))$ ,

als er constanten  $c \in R_+$  en  $k \in N^*$  bestaan zo dat  $f(n) \leq cg(n)$  voor alle  $n \geq k$ .

We spreken  $f = O(g)$  uit als: “ $f$  is grote o van  $g$ ”.

**Voorbeelden.** In de 'faculteits-algoritme' geldt  $t(n) = O(n)$ ; we zeggen dan dat de algoritme een *lineaire* complexiteit heeft.

Zij  $f(n) = 100n^2$  en  $g(n) = 2^n$ .

Dan geldt  $f(n) = O(g(n))$ .

Er geldt niet  $g(n) = O(f(n))$ .

Omdat  $f(n) = O(g(n))$  en  $g(n) \neq O(f(n))$  zullen we een algoritme met complexiteit  $f(n)$  (*kwadratische* complexiteit  $O(n^2)$ ) prefereren boven een met complexiteit  $g(n)$  (*exponentiële* complexiteit  $O(2^n)$ ). Dat  $f(n)$  voor kleine waarden van  $n$  veel groter is dan  $g(n)$ , is van beperkt belang. Vergelijk echter eens  $f(1000)$  met  $g(1000)$ :  $g(1000) = 2^{1000} = (2^4)^{250} > 10^{250} = 10^{242} \cdot 10^8 = 10^{242} \cdot f(1000)$ .

**Definitie.** Beschouw een algoritme met complexiteit  $t(n)$ , waarbij  $n$  de probleemgrootte is.

De algoritme heet *goed* of *polynomiaal* als  $t(n)$  naar boven begrensd is door een polynoom in  $n$ , m.a.w. als  $t(n) = O(n^k)$  voor zekere vaste  $k \in \mathbb{N}$ .

**Voorbeelden.** De ‘faculteits-algoritme’ is goed. Een algoritme met complexiteit  $f(n) = 100n^2$  is goed, maar een algoritme met complexiteit  $g(n) = 2^n$  niet.

We behandelen algoritmen voor problemen die m.b.v. grafen gemodelleerd kunnen worden. Onder de probleemgrootte zullen we dan verstaan het aantal punten van de inputgraaf. Het benodigde aantal operaties hangt echter doorgaans niet alleen af van het aantal punten van de inputgraaf, maar ook van de structuur van deze graaf. We beperken onze analyses tot de zogenaamde *worst-case-complexiteit*, d.i. het maximale aantal benodigde operaties over alle inputgrafen met eenzelfde aantal punten.

**Complexiteit van de algoritme van Dijkstra.** De executie van stap 2 kost  $n - 2$  vergelijkingen ter bepaling van het minimum, de tweede  $n - 3$ , enz. Stap 2 kost dus in totaal  $\sum_{i=1}^{n-2} i = \frac{1}{2}(n - 1)(n - 2)$  vergelijkingen. De eerste executie van stap 3 kost  $n - 2$  vergelijkingen en  $n - 2$  optellingen, de tweede  $n - 3$  vergelijkingen en  $n - 3$  optellingen, enz. In totaal kost stap 3 dus  $\frac{1}{2}(n - 1)(n - 2)$  vergelijkingen en eenzelfde aantal optellingen. De algoritme heeft dus complexiteit  $O(n^2)$ . We hebben o.a. het nagaan of een punt al dan niet tot  $S$  behoort, buiten beschouwing gelaten. Calculeert men de hiervoor benodigde operaties in, dan blijkt de grootte-orde van de complexiteit niet beïnvloed te worden. In het bijzonder is de algoritme van Dijkstra dus een goede algoritme.