-SAMENVATTING-
Fundamenten voor de Informatica

Samenvatting van Hoofdstuk 4.4: Analyse van algoritmen
tot Hoofdstuk 5: Netwerkmodellen

Maxwell Szymanski
Bachelor Informatica
KULeuven

2016 - 2017

Chapter 1

Analyse van Algoritmen

1.1 Tijdscomplexiteit

Aan elementaire bewerkingen wordt een tijdsduur van 1 toegekend. De functie $tijd_A(n) : \mathbb{N} \to \mathbb{N}$ geeft voor een gegeven invoergrootte n het maximum aantal elementaire bewerkingen (in het **slechtste geval**) die door het algoritme A uitgevoerd zullen worden.

Definitie: O-notatie

Indien f en g functies van $\mathbb{N}$ naar $\mathbb{R}^+$ zijn, dan is $f(n)$ van orde $g(n)$, of $O(g(n))$ als:

$$\exists c \in \mathbb{R}_0^+, \exists N \in \mathbb{N}, \forall n \in \mathbb{N} : n \geq N \Rightarrow f(n) \leq cg(n)$$

Definitie: Θ-notatie/Asymptotische equivalentie

Twee functies f en g worden asymptotisch equivalent genoemd als f is $O(g)$ en g is $O(f)$. We noteren dit als f is $\Theta(g)$ en dus ook g is $\Theta(f)$.

Een algoritme A wordt beter dan B genoemd indien $tijd_A(n) = O(tijd_B(n)$ maar $tijd_B(n) \neq O(tijd_A(n))$. Dit wilt niet altijd zeggen dat A altijd boven B verkozen moet worden: indien A een grote constante factor c heeft, en men weet dat n klein blijft, is B efficiënter. Hieronder de meest voorkomende functies en hun asymptotisch gedrag:

$$\ln(n) \leq n \leq n.\ln(n) \leq n^2 \leq \ldots \leq n^k \leq \ldots \leq 2^n \leq n!$$

Definitie: $\sim$-notatie/Tilde-notatie

Er geldt dat $f \sim g$ of $f(n) \sim g(n)$ als:

$$\forall \epsilon \in \mathbb{R}_0^+, \exists N \in \mathbb{N}, \forall n \geq N\text{: } \mid \tfrac{f(n)}{g(n)} - 1 \mid < \epsilon$$

of als limiet geschreven:

$$\lim_{n\to\infty} \tfrac{f(n)}{g(n)} = 1$$

Eveneens een asymptotische equivalentie, maar strenger dan de Θ-notatie: f en g moeten even snel stijgen, op een constante factor na, die voor een grote n verwaarloosbaar is.

Definitie: Polynomiale tijd

Een algoritme is van polynomiale tijd a.s.a. zijn tijdscomplexiteit $O(n^k)$ is voor een $k \in \mathbb{N}$

Algoritmen van polynomiale tijd zijn gewenst, maar toch blijven er veel met een tijdscomplexiteit van $O(c^n)$ voor een $c > 1$.

Definitie: Exponentiële tijd

Een algoritme is van exponentië tijd a.s.a. zijn tijdscomplexiteit $O(\exp(n^k))$ is voor een $k \in \mathbb{N}$

Een expolentiële tijd is een bovengrens, en algoritmes van polynomiale tijd zijn een deelverzameling van algoritmen met een exponentiële tijd. Algoritmen met tijdscomplexiteit $O(n!)$, $O(n^n)$ en $O(n.\exp(n))$ zijn van exponentiële tijd, en $O(\exp(\exp(n)))$ niet.
Bij problemen met een exponentiële tijdscomplexiteit botst men tegen een 'muur', waar de complexiteit snel begint te stijgen. Snellere computers kunnen deze 'muur' enkel verschuiven, maar niet wegwerken.

Voorbeeld Voor $tijd_A(n) = n^5$ en $tijd_B(n) = 2^n$ geldt, bij een invoer van slechts 60 op een computer met 10 mln. berekeningen/s dat:

- A: $\frac{60^5}{10.000.000/s} = 77,76\ s$
- B: $\frac{2^{60}}{10.000.000/s} = 3655,9$ jaar

1.1.1 (On)beheersbare problemen

Problemen die niet herleidt kunnen worden tot een polynomiale tijdscomplexiteit, noemt men **onhandelbare** (Eng. *intractable*) problemen. Een voorbeeld is het handelsreizigersprobleem:

Handelsreizigersprobleem Gegeven zijn n steden $c_1, c_2, ..., c_n$, en tussen elk paar steden c_i en c_j kent men ook de afstand $d_{i,j}$. De handelsreiziger vertrekt in een stad c_k, en probeert elke stad precies 1 keer te bezoeken om vervolgens terug in c_k te geraken. Welke weg moet er afgelegd worden opdat deze de kortste is.
Men kan alle mogelijke permutaties afgaan, wat voor $(n-1)!$ stappen zorgt.

1.1.2 Differentie-/ recursiebetrekkingen

Eigenschap: Recursiebetrekking 1

De oplossing van de recursievergelijking van de vorm

$$T(n+1) = T(n) + f(n)$$

met $f(n)$ van graad $k \geq 0$, is van de graad $k+1$.

```
for (int i=1; i ¡=n; i++) {
    for (int j = 1; j ¡= i*i; j++) {som++;}}
```

Neem voor dit voorbeeld $T(n)$ gelijk aan het aantal keer dat som uitgevoerd wordt. neem de substitutie $n = n+1$, dan wordt bij $i = n+1$ de bewerking $(n+1)^2$ keer uitgevoerd. Dan geldt:

$$T(n+1) = T(n) + (n+1)^2 \text{ en geldt bovendien dat } T(1) = 1$$

Hierbij is $(n+1)^2 = f(n)$, en zal de oplossing van graad 3 zijn.

Eigenschap: Recursiebetrekking 2

De oplossing van de recursievergelijking van de vorm

$$T(n) = 1 + T(\lceil \tfrac{n}{2} \rceil) \text{ met } n \geq 2$$

met $T(0)$ is $T(n) = \lceil log_2(n) \rceil$.

1.2 Complexiteitsklassen van beslissingsproblemen

Het gaat niet langer over de complexiteit van een algoritme, maar over de complexiteit van een probleem.

⇒ Gegeven een probleem van grootte n, dan gaat het over hoeveel stappen het **best mogelijke** algoritme nodig heeft in het **slechts mogelijke** geval.

Turingmachines en de klasse P

Definitie: Polynomoale beslisbaarheid

> Een taal L wordt door een Turingmachine M in polynomiale tijd beslist indien er een $k \in \mathbb{N}$ bestaat zodat M voor elke invoerstring s beslist of $s \in L$, in een aantal stappen dat $O(n^k)$ met $n = |s|$ de lengte van de invoerstring. Een taal L is **polynomiaal beslisbaar** als er een TM bestaat die de taal in polynomiale tijd beslist.

Definitie: De klasse P

> Met **P** duidt men de klasse aan van alle talen waarvoor geldt dat er een TM bestaat die die taal in polynomiale tijd beslist.

Gevolg Alle reguliere talen worden door een eindige automaat beslist in een lineaire tijd (er worden steeds evenveel stappen uitgevoerd als de invoer lang is). Een eindige automaat A kan tot een TM M omgevormd worden zodat $L(A) = L(M)$, zodat M ook een lineaire tijdscomplexiteit heeft. Alle reguliere talen behoren bijgevolg tot **P**.

De taal $L = \{0^n 1^n \mid n \in \mathbb{N}\}$ is een niet-reguliere taal die toch tot **P** behoort.
Ook al is **P** gedefiniëerd als een klasse van talen, kunnen we dit doortrekken tot een klasse van beslissingsproblemen door het probleem tot een ja-nee-vraag te herleiden. De TM stopt dan bij een 'ja'. De omvorming van het TM is echter klein, waardoor geldt dat als M een polynomiale tijdscomplexteit heeft, M' dat ook heeft.

Niet-Deterministische Turingmachines en de klasse NP

Bij een NDTM zeggen we dat een string s in n stappen aanvaard wordt als n het kleinste aantal stappen is dat tot een aanvaardbare eindtoestand kan leiden. De tijdscomplexiteit van een NDTM M voor strings van lengte n wordt dan gedefinieerd als het maximum, over alle $s \in L(M)$ met $|s| = n$ van het aantal stappen nodig om s te aanvaarden, of 0 als $L(M)$ geen strings van lengte aanvaard.

Definitie: NDTM van polynomiale tijd

> Een NDTM M heet van polynomiale tijd a.s.a. $tijd_M(n) = O(n^k)$, voor een $k \in \mathbb{N}$. De strings moeten dus in polynomiale tijd aanvaard worden, over strings $\notin$ taal L wordt niks gezegd.

Definitie: De klasse NP

> Met **NP** duidt men de klasse aan van alle talen die beslisbaar en niet-deterministisch polynomiaal herkenbaar zijn.

> Omdat elke TM aanzien kan worden als een NDTM, geldt: **P** $\subseteq$ **NP**.

Polynomiale verifieerbaarheid

Bij het TSP geldt $O((n-1!))$ om het op te lossen. Maar om te zeggen of een voorgesteld parcours een oplossing is, kan in $O(n)$ gebeuren. We kunnen zeggen dat het TSP **polynomiaal verifieerbaar** is.

Definitie: Polynomiaal verifieerbaar

Een taal L is polynomiaal verifieerbaar a.s.a. er een TM M bestaat, zodat voor elke string s een andere string c bestaat (een 'certificaat'), zodat geldt: M aanvaardt de string (c, s) in een tijd polynomiaal in $|s|$, a.s.a. $s \in L$.

Men kan M zien als een machine die een NDTM nabootst, en c de rol van een orakel speelt, dat telkens aangeeft welke keuzes er gemaakt moeten worden.

Definitie: De klasse NP (Alternatieve definitie)

Met **NP** duidt men de klasse aan van alle talen die polynomiaal verifieerbaar zijn.

De klasse NP-Compleet

We zeggen dat een taal L_1 (dus ook probleem) niet moeilijker te beslissen is als een andere taal L_2, als er een polynomiaal algoritme bestaat om eender welk woord uit L_1 te vertalen in een woord uit L_2.

Definitie: Polynomiale transformatie van een taal

Gegeven, op een alfabet T_1 en T_2, twee talen $L_1 \subseteq T_1^*$ en $L_2 \subseteq T_2^*$. Men zegt dat L_1 **polynomiaal transformeert** in L_2 indien er een afbeelding $f : T_1^* \to T_2^*$ bestaat waarvoor geldt:

1. $\forall x \in T_1^* : x \in L_1 \Leftrightarrow f(x) \in L_2$
2. Er bestaat een (deterministische) TM die f in polynomiale tijd berekent.

Dit noteren we als $L_1 \propto L_2$.

Als een taal $L_1 \subseteq T_1^*$ polynomiaal transformeert in $L_2 \subseteq T_2^*$ door de functie f die berekend wordt door een TM M met een tijdscomplexiteit $O(n^k), k \subseteq \mathbb{N}_0$, dan $\exists c \in \mathbb{R}^+$ zodat:
$|f(x)| \leq c \mid x \mid^k$ voor alle niet lege $x \in T_1^*$.

Stelling:

Indien $L_1 \propto L_2$ en $L_2 \in \mathbf{P}$ dan is ook $L_1 \in \mathbf{P}$.

We beschouwen twee talen als equivalent (d.w.z. $L_1 \sim L_2$), indien de ene taal polynomiaal getransformeerd kan worden in de andere, en omgekeerd (als $L_1 \propto L_2$ en $L_2 \propto L_1$).
De relatie $\propto$ is transitief, en de relatie $\sim$ is een equivalentierelatie.

Door de equivalentierelatie bestaat de klasse **P** uit 3 equivalentieklassen, en is de klasse **P** de klasse van eenvoudige talen binnen een grotere (of gelijke?) verzameling **NP**. De moeilijkste talen (problemen) binnen **NP** noemt men **NP-Complete** talen (problemen).

Definitie: De klasse NP-Compleet (NPC)

De taal L is **NP-Compleet** a.s.a.:

1. $L \in \mathbf{NP}$
2. Voor elke andere taal $L' \in \mathbf{NP}$ geldt: $L' \propto L$

M.a.w. is een **NPC** taal een taal die zo complex is dat elke andere taal in **NP** er in polynomiale tijd naar vertaald kan worden.

OF

Een probleem is **NP-compleet** als het een oplossing heeft (algoritme) zodanig dat elk ander **NP** probleem een oplossing heeft die door een polynomiale transformatie van de **NPC** af te leiden is.

Stelling:

NPC is een equivalentieklasse voor de relatie $\sim$ (polynomiale equivalentie).

Stelling:

Indien $\mathbf{NPC} \bigcap \mathbf{P} \neq \emptyset$, dan is $\mathbf{NP} = \mathbf{P}$.

Satisfiability Problem (SAT) Gegeven een eindige verzameling U van Booleaanse variabelen, en een verzameling C van formules over U. Kan C voldaan worden? SAT is **NP**-compleet. Als men bv. n Booleaanse veranderlijken heeft en men moet nagaan of de formules kloppen, kan men een NDTM gebruiken die voor n veranderlijken een waarde raadt en dan controleert. Dit kan gebeuren in polynomiale tijd, en SAT behoort tot **NP**.

Stelling:

Voor twee talen geldt: als $L_1 \propto L_2$ en $L_1 \in \mathbf{NPC}$, dan is ook $L_2 \in \mathbf{NPC}$.

Chapter 2

Grafen en vlakke grafen

- **Graaf:** verzameling elementen waartussen verbindingen bestaan.
  - Bestaat uit **knopen** en **bogen**.
- **Multiverzameling** Verzameling waarin een element meerdere keren *kan* voorkomen.
- **Multipaar** Multiverzameling met twee elementen (bv. $\{1,1\},\{2,3\},\ldots$).
  - Volgorde niet van belang, (v_1, v_2) en (v_2, v_1) duiden op hetzelfde multipaar.
- $\mathcal{M}_2(V)$ Verzameling van alle multiparen die elementen uit V bevatten.
  - Lijkt op $V^2 = V \times V$, alleen is de volgorde niet van belang.

Definitie: Graaf

Een 3-tal $G = (V, E, \phi)$, met:
- **V:** verzameling van knopen.
- **E:** verzameling van bogen.
- ϕ**:** functie die met elke boog twee knopen associeert.

$$\phi : E \to \mathcal{M}_2(V)$$

Een alternatieve notatie voor de graaf is $G = (V, E)$, waarbij E uit multiparen bestaat. Een boog e met $\phi(e) = (v, w)$ noteren we als (v, w). Moeilijker om een specifieke boog aan te duiden in een graaf met parallelle bogen.

2.1 Begrippen

- Boog e **valt in** op knoop v als $v \in \phi(e)$.
- Knoop v en w zijn **adjecent** $\Leftrightarrow \exists e \in E : \phi(e) = (v, w)$.
- Boog e is een **lus** $\Leftrightarrow \exists v \in V : \phi(e) = (v, v)$.
- Bogen e_1 en e_2 zijn **parallel** $\Leftrightarrow \phi(e_1) = \phi(e_2)$.

Definitie: Enkelvoudige graaf

Een graaf zonder lussen en parallelle bogen, m.a.w:

- $\forall e \in E : \phi(e) = (v, w) \Rightarrow v \neq w$.
- $\forall e_1, e_2 \in E : e_1 \neq e_2 \Rightarrow \phi(e_1) \neq \phi(e_2)$.

Definitie: Graad

De **graad** van een knoop v, genoteerd als $\sigma(v)$, is het aantal bogen dat op de knoop invallen.

Eigenschappen: Graad

- Het aantal bogen in een graaf is gelijk aan $\frac{1}{2}\sum\sigma(v_i)$.
- De som van de graden van alle knopen is altijd even.
- Het aantal knopen met een oneven graad in een graaf is altijd even.

Definitie: Pad

Een **pad** in een graaf $G(V, E)$ is een rij bogen van de vorm $\{(v_0, v_1), (v_1, v_2), \ldots, (v_{n-1}, v_n)\}$ waarbij $\forall i : (v_i, v_{i+1}) \in E$.

- **Lengte:** aantal bogen in een pad.
- **Enkelvoudig pad:** pad waarvan alle knopen verschillend zijn.
  - $\forall i, j : i \neq j \Rightarrow v_i \neq v_j$
- **Kring:** een pad waarbij alle bogen verschillend zijn en $v_0 = v_n$.
  - Tussen twee knopen v, w die deel uitmaken van dezelfde kring, bestaan steeds minstens 2 verschillende paden.
  - Als tussen twee knopen twee verschillende paden lopen, bevat G een kring.
- **Enkelvoudige kring:** Een kring waarbij alle knopen verschillend zijn, behalve $v_0 = v_n$.
- **Samenhangende graaf:** een graaf waarbij tussen elke twee knopen een pad bestaat.
- **Hamiltoniaans pad:** pad waarbij elke knoop van G precies één keer voorkomt.
- **Hamiltoniaanse kring:** enkelvoudige kring waarbij elke knoop van G precies één keer voorkomt.
- **Euleriaans pad:** pad waarin elke boog precies één keer voorkomt en elke knoop minstens één keer.
- **Euleriaanse kring:** Euleriaans pad dat ook een kring is.

Eigenschappen: Euleriaanse paden/kringen

- Een graaf G heeft een *Euleriaans pad* a.s.a. de graaf samenhangend is en hoogstens twee knopen een oneven graad hebben.
- Een graag G heeft een *Euleriaanse kring* a.s.a. de graaf samenhangend is en elke knoop een even graad heeft.

2.1.1 Deelgrafen

Definitie: Deelgraaf

Een graaf $G'(V', E')$ is een deelgraaf van $G = (V, E)$, genoteerd $G' \subseteq G$, a.s.a. $V' \subseteq V$ en $E' \subseteq E$.

Een deelgraaf kan alleen maar bogen hebben tussen knopen die ook in de deelgraaf zitten: $E' \subseteq E$ en ook $E' \subseteq \mathcal{M}_2(V') \Rightarrow E' \subseteq E \bigcap \mathcal{M}_2(V')$

Definitie: Geinduceerde deelgraaf

Een deelgraaf G' met alle bogen in G tussen de knopen $V' : E' = E \bigcap \mathcal{M}_2(V')$
oftewel $E' = \{(v, w) \in E \mid v, w \in V'\}$

Is de grootst mogelijke deelgraaf van G' die enkel knopen V' bevat.

Definitie: Component

Een graaf $G' = (V', E')$ is een component van graaf G a.s.a.:

- $G' \subseteq G$
- G' is niet leeg.
- G' is samenhangend.
- $\nexists$ samenhangende graaf $G'' : G' \subset G'' \subseteq G$.

Definitie: Partitie

Voor een partitie van G, gevormd door componenten, geldt dat elke $e \in E$ en $v \in V$ in precies één component voorkomen. Voor n partities geldt:

- $\bigcup_i V_i = V$
- $\bigcap_i V_i = \emptyset$
- $\bigcup_i E_i = E$
- $\bigcap_i E_i = \emptyset$

Definitie: Gerichte graaf

Een koppel $G = (V, E)$, met V een verzameling van knopen en $E \subseteq V^2$ een verzameling koppels.

Bij *koppels* maakt de volgorde van de knopen van een boog wel uit .

- **Gericht pad:** een pad $(v_0, v_1, \ldots, v_n)$ in een gerichte graaf, met $\forall i : (v_i, v_{i+1}) \in E$.
- **Ongericht pad:** een pad $(v_0, v_1, \ldots, v_n)$ in een gerichte graaf, met $\forall i : (v_i, v_{i+1}) \in E \vee (v_{i+1}, v_i) \in E$.
- Een gerichte graaf is **samenhangend** a.s.a. er tussen elke twee knopen een ongericht pad bestaat.

2.2 Isomorfisme

Definitie: Graaf-Isomorfisme

Een graaf $G = (V, E)$ is isomorf met graaf $G' = (V', E')$ a.s.a. er een bijectie $h : V \to V'$ bestaat zodat $\{h(v) \mid v \in V\} = V'$ en $\{(h(v_1), h(v_2)) \mid (v_1, v_2) \in E\} = E'$

De functie h stelt een hernoeming voor (beeldt elk element van V af op V').

- Twee grafen met **dezelfde matrixvoorstelling zijn isomorf**, MAAR isomorfe grafen hebben niet altijd dezelfde matrixvoorstelling (wel dezelfde *verzameling* van matrices, rij- en kolomvolgorde kan verschillen).

2.3 Bijzondere klassen van grafen

Definitie: Kliek

Een enkelvoudige graaf waarin elke knoop rechtstreeks verbonden is met elke andere knoop, genoteerd als K_n: een kliek met n knopen.

Definitie: Tweeledige graaf

Een enkelvoudige graaf dat in twee deelverzamelingen V_1 en V_2 opgedeeld kan worden zodat er enkel bogen bestaan tussen V_1 en V_2, maar geen binnen V_1 of V_2. Dus geldt:

- $V_1 \bigcap V_2 = V$
- $V_1 \bigcup V_2 = \emptyset$
- $E \subseteq \{(v_1, v_2) \mid v_1 \in V_1, v_2 \in V_2\}$

Definitie: Volledig verbonden tweeledige graaf

Een tweeledige graaf waarbij elke knoop uit V_1 verbonden is met elke knoop uit V_2, oftewel $E = \{(v_1, v_2) \mid v_1 \in V_1, v_2 \in V_2\}$.
Genoteerd als $K_{n,m}$, met n knopen in V_1 en m knopen in V_2.

2.4 Vlakke grafen

- **Vlakke graaf:** graaf die getekend kan worden zonder snijdende bogen.
- **Zijvlak:** vlak omgeven door een kring. Buitenste gebied is ook een zijvlak!
- Twee punten liggen in verschillende snijvlakken a.s.a. er geen lijn tussen de twee punten getrokken kan worden zonder een boog te snijden.
- Zij Z de verzameling met alle zijvlakken van een graaf als elementen, dan is $\beta(z)$ het aantal bogen waardoor $z \in Z$ begrensd is.
- De som B is gedefinieerd als: $B = \sum_{z \in Z} \beta(z)$.

Eigenschappen: Vlakke grafen

- Elke deelgraaf van een vlakke graaf is ook vlak.
- Elke kringvrije graaf is vlak, en heeft 1 zijvlak (nl. het buitenste).
- In een kringvrije graaf met minstens 1 boog bestaat er steeds een knoop $v \in V$ waarvoor geldt: $\sigma(v) = 1$.
- Voor elke vlakke graaf geldt: $B \leq 2e$.
- Voor elke enkelvoudige vlakke graaf met $f \geq 2$ geldt: $B \geq 3f$.
- Voor elke enkelvoudige vlakke graaf met $f \geq 2$ geldt: $2e \geq 3f$ of $f \leq \frac{2}{3}e$.
- Voor elke vlakke graaf met $e \geq 2$ geldt: $e \leq 3v - 6$.
- In elke vlakke, enkelvoudige graaf bestaat er minstens 1 knoop v zodat $\sigma(v) \leq 5$

Definitie: Formule van Euler

Zij G een samenhangende vlakke graaf met $v \geq 1$ knopen, e bogen en f zijvlakken, dan geldt:

$$v - e + f = 2$$

Definitie: Homeomorfisme

Een graaf G' is homeomorf met G a.s.a. G' uit G bekomen kan worden door een of meerdere keren een knoop v van graad 2 te nemen, deze en de bogen (u, v) en (v, w) weg te nemen, en te vervangen door (u, w) (**rijreductie**).

Als G homeomorf is met G', geldt: G is vlak $\Leftrightarrow$ G' is vlak.

Definitie: Stelling van Kuratowski

Een graaf is vlak a.s.a. de graaf geen deelgraaf bevat die homeomorf is met K_5 of $K_{3,3}$.

2.4.1 Veelvlakken

- **Veelvlak:** ruimtelijk lichaam begrensd door een aantal zijvlakken.
- De knopen worden vaak **hoeken** genoemd, en de bogen **ribben**.
- **Platonische lichamen (**regelmatige veelvlakken**):** elk zijvlak is door evenveel ribben omgeven, en elke hoek verbindt evenveel ribben.
  - Elke knoop heeft dezelfde graad.
  - Elk zijvlak (ook het buitenste) heeft evenveel aangrenzende bogen.
  - Er zijn maar 5 platonische lichamen.

2.4.2 Duale grafen

- Zij G een graaf met v knopen. De duale graaf G' heeft 1 knoop voor elk zijvlak ($v' = f$ knopen). Voor elke boog e tussen de oorspronkelijke zijvlakken is er een boog e' van G' die e snijdt, door de twee knopen in de aanliggende zijvlakken grenzend aan e te verbinden.
- Kan gebruikt worden om het kleurenprobleem van kaarten op te lossen, waarbij men de duale graaf van een kaart neemt en het kleuringsprobleem hierop toepast.
- Een duale graaf van een vlakke graaf is ook vlak.

2.4.3 Kleuring van grafen

- Het toekennen van een kleur aan elke $v \in V$ zodanig dat de kleur van v en w verschilt indien $(v, w) \in E$.
- **n-kleuring:** kleuring met n of minder verschillende kleuren.
- **Minimale kleuring:** n-kleuring met een minimale n.
- **4-kleurenprobleem:** heeft elke vlakke graaf een 4-kleuring? Voorlopig nog enkel computergewijs aangetoond, maar nog niet formeel bewezen.
- Elke enkelvoudige vlakke graaf heeft een 5-kleuring.

Chapter 3

(Gewortelde) Bomen

3.1 Bomen

Definitie: Boom

Een graaf is een boom als het de volgende 3 eigenschappen bezit:

- Samenhangend.
- Kringvrij.
- Bevat n knopen en $n-1$ bogen.

Een graaf met 2 van de bovenstaande eigenschappen heeft automatisch ook de 3^e eigenschap.

Definitie: Opspannende boom

$T(V_T, E_T)$ is een opspannende boom voor $G = (V, E)$ a.s.a:

- T een boom is.
- $V_T = V$ en $E_T \subseteq E$.

Eigenschappen: Opspannende boom

- Enkel een samenhangende graaf kan een opspannende boom hebben.
- Elke samenhangende graaf heeft een opspannende boom.
- Als T een maximale kringvrije deelgraaf is van een samenhangende graaf G, dan is T een opspannende boom van G.
- Als T een kringvrije deelgraaf is van een samenhangende graaf G, en T heeft $n-1$ bogen, dan is T een opspannende boom voor G.

Het construeren van opspannende bomen wordt gebruikt in zoekproblemen, dat vaak herleid kan worden tot het doorlopen van een graaf op zoek naar een bepaalde knoop dat de oplossing voorstelt. We willen dan grafen doorlopen zonder dat we een knoop meerdere keren tegenkomen, maar waarbij we wel zeker zijn dat we geen knopen overslaan. Dit komt neer op het opstellen van een opspannende boom.

3.1.1 Minimaal opspannende boom

Definitie: Minimaal opspannende boom

Zij G een gewogen graaf. Dan is de boom T een minimaal opspannende boom van G als er geen andere opspannende boom een kleiner gewicht heeft.

Algoritme van Prim: bouwt een MOB door met een willekeurige knoop te beginne, en van daaruit een boom op te bouwen door herhaaldelijk een boog met het laagste gewicht toe te voegen aan de reeds opgebouwde boom.

Algoritme: Algoritme van Prim

Gegeven: een samenhangende, gewogen graaf $G(V, E)$ met n knopen.
Resultaat: T, een MOB voor G.
$T := (\{v_0\}, \emptyset)$
zolang T minder dan $n - 1$ bogen heeft:
kies $(u, v) \in E$ zodat $u \in T, v \notin T$ en $w(u, v) = min\{w(x, y) \mid x \in T, y \notin T\}$
voeg (u, v) toe aan T

Algoritme van Kruskal: werkt op een gelijkaardige manier aan die van Prim, maar de deelgraaf S hoeft tijdens het algoritme niet per sé samenhangend, en dus een boom te zijn.

Algoritme: Algoritme van Kruskal

Gegeven: een samenhangende, gewogen graaf $G(V, E)$ met n knopen.
Resultaat: S, een MOB voor G.
$S := (\emptyset, \emptyset)$
zolang S minder dan $n - 1$ bogen heeft:
$(u, v) :=$ de boog met het kleinste gewicht die geen kring in S maakt
voeg (u, v) toe aan S

3.2 Gewortelde bomen

3.2.1 Gewortelde boom

Definitie: Gewortelde boom

Een tupel $(V, E, w$, met $T = (V, E)$ een boom en $w \in V$ de **wortel**.

Meestal wordt een gewortelde boom genoteerd als T, waaruit V, E en w er impliciet uit afgeleid kunnen worden.

- In een gewortelde boom is elke knoop v verbonden met de wortel w door een uniek pad $P_v = \{v_0 = w, v_1, v_2, \ldots, v_h = v\}$.
- De lengte van het pad noemen we de **hoogte** h van v, genoteerd $h(v)$.
- Voor elke i, j met $i < j$ geldt:
  - v_i is een **ouder** van v_{i+1}
  - v_{i+1} is een **kind** van v_i
  - v_i is een **voorouder** van v_j
  - v_j is een **afstammeling** van v_i
- Als v en w kinderen zijn van dezelfde ouder (op dezelfde hoogte h zitten), zijn ze **siblings**.
- Een knoop die geen kinderen heeft, noemt een **blad**.

- Een knoop die geen blad is, noemt men een **inwendige knoop**.
- Een **deelboom** van T met wortel v is een gewortelde boom die v en al zijn afstammelingen bevat, met de bogen ertussen.
- De **hoogte van een gewortelde boom** is de maximale hoogte van zijn knopen: $h(T) = max\{h(v) \mid v \in T\}$
- Een boom wordt gewoonlijk met de wortel bovenaan getekend, waardoor de hoogte $h(T)$ vaak de **diepte** genoemd wordt.

3.2.2 Binaire boom

Definitie: Binaire boom

In een binaire boom heeft elke $v \in V$, behalve de wortel w, een ouder, en alle inwendige knopen minstens één kind en hoogstens 2 kinderen.

Definitie: Volledige binaire boom

Een volledig binaire boom is een binaire boom waarbij elke inwendige knoop precies twee kinderen heeft.

In een binaire boom wordt een onderscheid gemaakt tussen het linker- en rechterkind. Een boom waarin x_1 het linkerkind is van x, en x_2 het rechterkind, is niet hetzelfde als een boom waarbij het omgekeerd is. Daarom wordt de volgende formele definitie gegeven van een binaire boom:

Een **binaire boom** is een kwadrupel (V, E, w, λ) met $E \subseteq V \times V$, $w \in V$ en λ de partiele functie is die voor elke knoop aangeeft welke het linker- en rechterkind is. $\lambda : V \times \{l, r\} \to E$

Eigenschappen: Binaire boom

- Elke volledige binaire boom met i inwendige knopen heeft $i + 1$ bladere en $2i + 1$ knopen in totaal.
- In een binaire boom bet t bladeren en hoogte h geldt: $t \leq 2^h$.
- In een binaire boom met n knopen en hoogte h geldt: $n + 1 \leq 2^{h+1}$.

3.2.3 Binaire zoekboom (BST)

Definitie: Binaire zoekboom

Een binaire boom waarin aan elke knoop een waarde $w(v)$ is geassocieerd (bv. een getal), zodanig dat als l behoort tot de linker- en r tot de rechterdeelboom van v, dan is $w(l) < w(v) < w(r)$.

Algoritme: Volledig doorlopen van een gewortelde boom

$S := \{w\}$
$R := \emptyset$
zolang $S \neq \emptyset$:
    kies een willekeurige knoop $a \in S$
    $S := S\ \{a\} \cup \{k \mid k \in kinderen(a)\}$
    $R := R \cup \{a\}$
    behandel a

Dit algoritme heeft als belangrijke eigenschap dat de boom volledig wordt doorlopen én geen enkele knoop twee keer wordt doorlopen. Hieruit volgt dus ook dat het algoritme altijd eindigt

(n knopen behandelt). Naast het willekeurig doorlopen van knopen, zijn er ook systematischere aanpakken. Bij **diepte-eerst** nemen we steeds een knoop met maximale diepte, en bij **breedte-eerst** een knoop met minimale diepte.

Algoritme: Diepte-eerst doorlopen van een gewortelde boom

$S := (w, 0)$
$R := \emptyset$
zolang $S \neq \emptyset$:
    kies een willekeurige (a, n) uit S met maximale n
    $S := S \setminus \{(a, n)\} \cup \{(v, n + 1) \mid v \in kinderen(a)\}$
    $R := R \cup \{a\}$
    behandel a

Algoritme: Breedte-eerst doorlopen van een gewortelde boom

$S := (w, 0)$
$R := \emptyset$
zolang $S \neq \emptyset$:
    kies een willekeurige (a, n) uit S met *minimale* n
    $S := S \setminus \{(a, n)\} \cup \{(v, n + 1) \mid v \in kinderen(a)\}$
    $R := R \cup \{a\}$
    behandel a

Comparatieve studie De algoritmen kunnen vergeleken worden door naar de maximale grootte van S te kijken tijdens het algoritme. Bij *diepte-eerst* is S maximaal net voordat we de eerste keer een blad behandelen: $\mid S \mid = d + 1$. Bij *breedte-eerst* is dat wanneer alle knopen op diepte $d - 1$ behandeld zijn, S bevat dan alle bladeren: $\mid S \mid = 2^d$

⇒ Diepte-eerst heeft een plaatsvoordeel.

Maar vaak is een eindige oplossingsverzameling van een probleem te vinden in een oneindige verzameling van "bijna"-oplossingen (op een eindige diepte), gestructureerd als boom (bv. spel-bomen). Diepte-eerst kan het sneller vinden enkel en alleen als het de juiste keuze maakt, anders loopt het verloren.

⇒ Breedte-eerst heeft een voordeel bij oneindige verzamelingen.

3.2.4 Prefix-, infix- en postfix-orde

Er zijn verschillende manieren (orde's) om bomen te doorlopen.

- **Prefix-orde:** De wortel wordt afgedrukt, en daarna wordt de linker-, daarna de rechterdeelboom afgedrukt. (- / 1 2 * 3 + 4 5)
- **Infix-orde:** De linkerdeelboom wordt eerst behandelt, dan de wortel en dan de rechterdeelboom. (1 / 2 - 3 * 4 + 5)
- **Postfix-orde:** Eerst de linkerdeelboom wordt behandelt, dan de rechterdeelboom, en dan pas de wortel. (1 2 / 3 4 5 + * -)

3.2.5 Spelbomen

Als voorbeeld nemen we □ voor de eigen speler en ◯ voor de tegenstander. Als we een spelboom uittekenen met alle mogelijke situaties, duiden we een blad aan met 1 als het een winnende situatie is (voor □), en 0 als het ene verliezende situatie is (een winnende situatie voor ◯).

Minmax-procedure We kunnen nu de kinderen *propageren* naar de ouders, waarbij een □ het maximum krijgt van de kinderen, en ◯ het minimum van de kinderen.

Indien uit de minmax-procedure blijkt dat de wortel een 1 krijgt, wilt dit zeggen dat indien de beginner telkens de juiste zet doet, hij kan winnen (een beginner-wint-altijd spel).

Soms is het moeilijk tot onmogelijk om een hele spelboom tot de volledige diepte op te bouwen, en moeten we stoppen bij een bepaalde diepte d. Het is dan niet mogelijk om te weten of een bepaalde positie winnend is of niet. Daarvoor gebruikt men een evaluatiefunctie.

Evaluatiefunctie E E kent aan elke knoop op diepte d een getal toe die groter is naarmate de positie beter is voor $\square$. De evaluatiefunctie is meestal niet onfeilbaar.

Definitie: α-cutoff

> Komt voor bij een $\square$-knoop w als een kleinkind m een waarde heeft kleiner of gelijk aan de α**-waarde** van w (tot nu toe grootste waarde van een kleinkind/grootste ondergrens). Men mag dan de deelboom met wortel van ouder m verwijderen uit het spelboom (d.w.z. van die deelboom moeten er geen labels meer berekend worden).

Definitie: β-cutoff

> Komt voor bij een $\bigcirc$-knoop w als een kleinkind m een waarde heeft groter of gelijk aan de β**-waarde** van w (tot nu toe kleinste waarde van een kleinkind/kleinste bovengrens). Men mag dan de deelboom met wortel van ouder m verwijderen uit het spelboom.

Met deze definities kunnen we nu een α-β-algoritme definieren:

α-β**-algoritme** De spelboom wordt diepte-eerst doorlopen (van links naar rechts). TODO

Chapter 4

Meerkeuzevragen

Talen & Automaten

- Een string heeft steeds een eindige lengte.
  ◦ **Juist:** *Definitie 4.2.*

- Een taal bevat steeds een eindig aantal strings.
  ◦ **Fout:** *Hoewel strings altijd een eindige lengte hebben, kunnen er oneindig veel strings gevormd worden uit een alfabet. De taal $\sum^*$ is bv. oneindig lang.*

- De verzameling van alle strings die gevormd kunnen worden met bepaald alfabet, is een taal over dat alfabet.
  ◦ **Juist:**

- Voor eender welke twee talen S en T, regulier of niet, is $TS * T$ een reguliere taal.
  ◦ **Fout:**

- $\{a\}^*$ bevat oneindig veel strings.
  ◦ **Juist:** *De Kleene-sluiting kan men oneindig vaak toepassen.*

- Niet-deterministische eindige automaten kunnen dezelfde talen herkennen als deterministische, maar som efficinter in tijd (d.w.z. strings worden soms herkend met minder toestandstransities).
  ◦ **Fout:**

- Niet-deterministische Turingmachines kunnen meer talen herkennen dan deterministische.
  ◦ **Fout:** *Voor elke NDTM bestaat er een DTM (zie stelling).*

- Een eindige taal is steeds regulier.
  ◦ **Juist:**

- Een reguliere taal bevat steeds de lege string.
  ◦ **Fout:**

- Voor elke reguliere taal L en voor elke string s kan in tijd $O(n)$ beslist worden of $s \in L$, met n de lengte van s.
  ◦ **Juist:**

- Elke eindige automaat A beslist of $s \in L(A)$ in een aantal stappen $O(m)$, met m het aantal toestanden van A.
  ◦ **Fout:**

- De verzameling $\{a^n b^n \mid n \in \mathbb{N}\}$ is een reguliere taal over $\{a, b, c, 0, ..., 9\}$.
  ◦ **Fout:** *Voor elke reguliere taal bestaat er een eindige automaat, en we hebben bewezen dat er voor deze taal geen automaat bestaat.*

- De verzameling $\{a^n b^n \mid n \in \mathbb{N},\ 2n \leq 20\}$ is een reguliere taal over $\{a, b, c, 0, ...9\}$.
  ◦ **Juist:**

- Voor elke functie $f : \mathbb{N} \to \{0, 1\}$ bestaat er een Turingmachine die f berekent.
  ◦ **Fout:** *Analoog met stelling 4.6.*

- Voor elke functie $f : \{0, 1\}^2 \to \{0, 1\}$ (d.w.z., voor elke binaire booleaanse operator) bestaat er een Turingmachine die f berekent.
  ◦ **Juist:**

- Voor elke binaire booleaanse operator f bestaat er een eindige automaat die f berekent.
  ◦ **Juist:**

- Elke eindige taal is regulier.
  ◦ **Juist:**

- Elke reguliere taal is eindig.
  ◦ **Fout:**

- Elke reguliere taal bevat de lege string.
  ◦ **Fout:**

- De lege taal is een reguliere taal.
  ◦ **Juist:**

- Voor elke reguliere taal L geldt: als $x, y \in L \Rightarrow xy \in L$.
  ◦ **Fout:**

- Voor elke reguliere taal bestaat er een eindige automaat die die taal herkent.
  ◦ **Juist:** *Stelling 4.2.*

- De verzameling van $\{(ab)^n \mid n \in \mathbb{N}\}$ is een reguliere taal.
  ◦ **Juist:** *Een eenvoudige eindige automaat kan deze taal voorstellen.*

- De verzameling van $\{a^m b^n \mid n, m \in \mathbb{N}\}$ is een reguliere taal.
  ◦ **Juist:** *Een eenvoudige eindige automaat kan deze taal voorstellen.*

- De verzameling van $\{a^n b^n \mid n \in \mathbb{N}\}$ is een reguliere taal.
  ◦ **Fout:** *Zie bewijs pagina 34.*

- De verzameling van alle wiskundige uitdrukkingen gevormd met +, -, /, x en haakjes is een reguliere taal.
  ◦ **Fout:** *Er kan geen eindige automaat worden gemaakt die een geheugen heeft om het aantal open haakjes te tellen, en te controleren of er evenveel gesloten haakjes zijn.*

- De klasse van talen herkend door eindige toestandsautomaten is dezelfde als de klasse herkend door niet-determinitische eindige automaten.
  ◦ **Juist:** *Men kan elke NDFA herleiden tot een DFA,*

- Voor elk java-programma bestaat er een Turingmachine die voor eender welke input dezelfde output geeft als dat programma.
  ◦ **Juist:**

- Voor elke binaire booleaanse operator $f : \mathbb{B} \times \mathbb{B} \to$, met $\mathbb{B} = \{0, 1\}$ bestaat er een eindige automaat die f berekent.
  ◦ **Juist:**

- Voor elke functie $f : \mathbb{N} \to \mathbb{N}$ bestaat er een eindige automaat die f berekent.
  ◦ **Fout:** *Stelling 4.6*

- Voor elke functie $f : \mathbb{N} \to \mathbb{N}$ bestaat er een TM die f berekent.
  ◦ **Fout:** *Stelling 4.6*

Complexiteitsklassen

- We weten zeker dat **P** een deelverzameling is van **NPC** en dat **NPC** een deelverzameling is van **NP**.
  ◦ **Fout:** *We weten enkel dat* $\boldsymbol{P \subseteq NP}$.

- Bepalen of een gewogen graaf een Hamiltoniaanse kring met gewicht kleiner dan 100 heeft is zeker in **P**.
  ◦ **Fout:** *Het algoritme moet back-tracken indien het een foute beslissing maakt, waardoor het geen polynomiaal algoritme is. Zie link*

- Bepalen of een graaf een Euleriaanse kring heeft, is zeker in **NP**.
  ◦ **Juist:** *Greedy algoritme mogelijk, zit in* $\boldsymbol{P}$.

- Bepalen of het kortste pad tussen twee gegeven knopen in een gewogen graaf en een gewicht kleiner dan 100, is zeker **NP**-compleet.
  ◦ **Fout:**

- Uit de stelling van Cook volgt SAT is een element van **NPC**.
  ◦ **Juist:** *Stelling 4.10*

- Uit de stelling van Cook volgt SAT is geen een element van **P**.
  ◦ **Fout:** *We weten niet of elementen in* $\boldsymbol{NPC}$ *geen elementen zijn in* $\boldsymbol{P}$.

- Als $L_1 \propto L_2$ en $L_1 \in \mathbf{P}$, dan geldt steeds $L_2 \in \mathbf{P}$.
  ◦ **Fout:** *Moet zijn: Als* $L_1 \propto L_2$ *en* $L_2 \in \boldsymbol{P}$*, dan geldt steeds* $L_1 \in \boldsymbol{P}$

- We weten zeker dat $\mathbf{P} \subseteq \mathbf{NP} \subseteq \mathbf{NPC}$.
  ◦ **Fout:** *We weten enkel dat* $\boldsymbol{P \subseteq NP}$.

Grafen & Bomen

- Elke vlakke graaf voldoet aan $v - e + f = 2$ met v het aantal knopen, e het aantal bogen en f het aantal zijvlakken.
  ◦ **Fout:** *De graaf moet ook samenhangend zijn.*

- In een samenhangende graaf bestaat er een pad tussen elke twee knopen.
  ◦ **Juist:** *Definitie samenhangende graaf.*

- $K_{2,2}$ is homeomorf met K_3.
  ◦ **Juist:** *Vervang in elke subset 1 knoop en de twee geassocieerde bogen, door een boog. We hebben dan de graaf* $K_{2,2}$ *(definitie homeomorfisme).*

- Als twee grafen isomorf zijn, hebben ze evenveel knopen.
  ◦ **Juist:** *Definitie isomorfisme: enkel een hernoeming van de knopen is toegelaten.*

- Als twee grafen homeomorf zijn, hebben ze evenveel bogen.
  ◦ **Fout:** *Men kan een graaf* G' *uit* G *bekomen door knopen van graad 2 te vervangen door een boog.* G' *en* G *zijn dan homeomorf, maar bevatten een verschillend aantal knopen en bogen.*

- Een boom met n knopen heeft steeds $n - 1$ bogen.
  ◦ **Juist:** *Zie definitie boom.*

- Elke samenhangende graaf heeft precies 1 opspannende boom.
  ◦ **Fout:** *Bij niet-unieke gewichten kunnen meerdere bomen voorkomen.*

- Elke boom heeft een twee-kleuring.
  ◦ **Juist:** *De kinderen hebben telkens een andere kleur dan hun ouder.*

- Een graaf met een Euleriaanse kring heeft nooit knopen met oneven graad.
  ◦ **Juist:** *Stelling 5.4.*

- Elke samenhangende vlakke graaf heeft een Euleriaanse kring.
  ◦ **Fout:** *Een samenhangende vlakke graaf kan een knoop met een oneven graad bevatten. Dan bevat het geen Euleriaanse kring meer.*

- De tweeledige grafen $K_{3,4}$ en $K_{4,3}$ zijn isomorf.
  ◦ **Juist:** *De twee grafen zijn gewoon gespiegeld.*

- Elke tweeledige graaf heeft een tweekleuring.
  ◦ **Juist:** *De knopen van de eerste subset hebben een bepaalde kleur, en de andere subset een andere kleur. Bijgevolg zijn alle knopen gekleurd.*

- Elke tweeledige graaf is vlak.
  ◦ **Fout:** *Tweeledige grafen kunnen kringen bevatten die de graaf niet vlak maken, bv. $K_{3,3}$.*

- Elke vlakke graaf heeft een vierkleuring.
  ◦ **Juist:**

- $K_{3,3}$ is vlak.
  ◦ **Fout:** *Intuitie, of formeel bewijs: link*

- $K_{3,3}$ heeft een vierkleuring.
  ◦ **Juist:** *Elke tweeledige graaf heeft een minimale kleuring van 2. Een graaf met 6 knopen kan dus een 2, 3, 4, 5 of 6-kleuring hebben.*

- Elke niet-vlakke graaf heeft minstens een kring.
  ◦ **Juist:** *Elke kringvrije is vlak.*

- Uit de stelling van Kuratowski volgt dat elke graaf die geen subgraaf bevat die isomorf is met K_5 of $K_{3,3}$ vlak is.
  ◦ **Fout:** *Moet homeomorf zijn i.p.v. isomorf.*

- Van elke vlakke graaf kan een boom gemaakt worden door k bogen weg te halen met k het aantal kringen.
  ◦ **Fout:**

- Om een opspannende boom te hebben moet een graaf samenhangend en vlak zijn.
  ◦ **Fout:** *Een graaf moet niet vlak zijn om een MOB te hebben.*

- Elke graaf met enkel knopen met even graad heeft een Euleriaanse kring.
  ◦ **Fout:** *De graaf moet ook samenhangend zijn.*

- Elke graaf die K_5 bevat heeft zeker geen 4-kleuring.
  ◦ **Juist:** *Elke knoop is verbonden met 4 andere knopen. Dus er zijn al minstens 5 kleuren nodig om de graaf te kleuren.*

- Elke graaf met minder dan n componenten heeft een n-kleuring.
  ◦ **Fout:** *Men kan geen n kleuren gebruiken in een graaf met minder dan n knopen.*

- Elke vlakke graaf voldoet aan $v - e + f = 2$ (met v het aantal knopen, e het aantal bogen, en f het aantal zijvlakken).
  ◦ **Fout:** *De graaf moet vlak en samenhangend zijn.*

Prim, Kruskal & Dijkstra

- Het algoritme van Kruskal en het algoritme van Prim zijn twee verschillende manieren om hetzelfde probleem op te lossen.
  ◦ **Juist:** *Ze zoeken altijd de MOB van een gegeven graaf.*

- Het algoritme van Kruskal en het algoritme van Prim geven steeds dezelfde uitkomst.
  ◦ **Fout:** *Bij grafen met niet-unieke gewichten kunnen er meerdere MOB voorkomen. Een ander algoritme kan dus een andere MOB vinden.*
- De tijdscomplexiteit van het algoritme van Prim is lineair in het aantal knopen.
  ◦ **Fout:**
- De tijdscomplexiteit van het algoritme van Prim is lineair in het aantal bogen.
  ◦ **Fout:**
- Het algoritme van Dijkstra heeft complexiteit $0(n^3)$ (met n het aantal knopen).
  ◦ **Juist:**
- Het algoritme van Dijkstra is asymptotisch equivalent met n^3. (met n het aantal knopen)
  ◦ **Fout:**
- Het algoritme van Dijkstra kan door een deterministische TM uitgevoerd worden.
  ◦ **Juist:**

Andere universiteiten

Dit zijn enkele meerkeuzevragen van Fundamenten die op examens van andere universiteiten gesteld werden.

Talen & Automaten

- If M is a Turing machine and $w \in \sum^*$ is a string, the TM M either accepts or rejects w.
  ◦ **Fout:** *A TM can loop on w.*
- If A is a regular language, then A is finite.
  ◦ **Fout:** *The language a^* is regular but infinite.*
- The language $\{a^n b^n \mid n \geq 0\}$ has regular expression a^*b^*.
  ◦ **Fout:** *a^*b^* generates the string $abb \notin \{a^n b^n \mid n \geq 0\}$.*
- If $A \subseteq B$ and B is a regular language, then A is a regular language.
  ◦ **Fout:** *For example, let $A = \{a^n b^n \mid n \geq 0\}$ and $B = (a \cup b)^*$. Then $A \subseteq B$, B is regular, but A is nonregular.*
- Every nonregular language is infinite.
  ◦ **Juist:** *Suppose A is nonregular and finite. But each finite language is regular, which is a contradiction.*
- Every infinite language is nonregular.
  ◦ **Fout:** *$\{0, 1\}^*$ is a regular language that is infinite*
- The regular expressions $(a \mid b)^*$ and $(b^*a^*)^*$ generate the same language.
  ◦ **Juist:**
- If a language A is regular, then A^* must be regular.
  ◦ **Juist:**
- We know of a problem that is both in the class **NP** and in the class **P**.
  ◦ **Juist:** *Any problem in **NP** is also in **P**.*

Grafentheorie

- If a graph on n vertices does not contain a cycle, then it must have exactly $n-1$ edges.
  - **Fout:** *It can very well be an empty graph.*
- Suppose G and H are isomorphic graphs. If G is a tree, then H is a tree.
  - **Juist:**
- In een enkelvoudige graaf met met meer dan 1 knoop, zijn er minstens 2 knopen met dezelfde graad.
  - **Juist:**
- Let T be the shortest path tree rooted at x for some graph G. Then for any vertex y, the shortest path from x to y in G is the same as the path from x to y in T.
  - **Juist:**
- Number of odd degree vertices is even in an undirected graph.
  - **Juist:**
- Sum of degrees of all vertices is even in an undirected graph.
  - **Juist:**

Externe links:

- Link voor ShareLatex (om eventueel vragen toe te voegen/oplossingen aan te vullen): link
- Andere examens met oplossingen: link
- **P**, **NP**en **NPC**: link