HO
GENT

Probleemoplossend Denken II

Lesnota's

Stijn Lievens

Professionele Bachelor in de Toegepaste Informatica

Inhoudsopgave

Voorwoord

Dit is de cursus voor het opleidingsonderdeel Probleemoplossend Denken II, uit het tweede jaar professionele bachelor informatica.

Dit opleidingsonderdeel bestaat uit drie grote onderdelen, namelijk: "kansrekening", "bomen en grafen" en "operationeel onderzoek". Dit zijn meteen ook de drie delen van deze cursustekst.

Het opleidingsonderdeel Probleemoplossend Denken II is relatief wiskundig en theoretisch. In deze tekst wordt de nadruk echter niet gelegd op de wiskundige bewijzen. Enkel wanneer deze kort zijn en helpen om de eigenschap beter te begrijpen (en te onthouden) worden deze vermeld. Wanneer de bewijzen een meer wiskundige of theoretische inslag hebben worden ze in deze grootte afgedrukt. Het belangrijkste is echter dat je de eigenschappen kan *toepassen*. Hiertoe zijn een heel aantal oefeningen verwerkt in deze tekst. In de lessen zal je de kans krijgen om een aantal van deze oefeningen samen met je medestudenten en met hulp van de lesgever op te lossen. Het is *zeer belangrijk* dat je zelf de overige oefeningen en opgaven tracht te maken; dat is de beste manier om voldoende vertrouwd te geraken met de leerstof.

Om te demonstreren dat de besproken algoritmen ook in de praktijk werken voorzien we een aantal oefeningensessies op PC. Hiervoor zullen er aparte opgaven worden opgesteld.

Ik dank oud-collega Roland Maerivoet voor het ter beschikking stellen van zijn cursusmateriaal. Een groot deel van het eerste deel van deze cursus is hierop gebaseerd.

Dit is het achtste jaar dat (een versie van) de eerste twee delen van deze tekst worden gebruikt. Het derde deel wordt dit jaar voor de derde maal gebruikt en is bijgevolg nog niet zo hard "getest" als de eerste twee delen. Net zoals in de voorgaande jaren ben ik elke student die opmerkingen of

aanmerkingen ter verbetering (bv. tikfouten, spellingsfouten, onduidelijkheden, enz.) aanbrengt ten zeerste dankbaar. En belangrijker: je opvolgers zullen je er ook dankbaar voor zijn. Voor het aangeven en bijhouden van errata zal er op Chamilo een forum worden aangemaakt.

Tenslotte wens ik jullie veel lees-, leer- en oefenplezier.

Stijn Lievens

Deel I

Kansrekening

HOOFDSTUK 1

Gebeurtenissen en hun kansen

In dit eerste hoofdstuk introduceren we het begrip GEBEURTENIS als deelverzameling van een verzameling van alle mogelijke uitkomsten van een experiment, het UNIVERSUM. Aan deze gebeurtenissen wordt een KANS of WAARSCHIJNLIJKHEID gehecht; de drie regels (axioma's) waaraan kansen moeten voldoen worden besproken en hieruit worden een aantal bijkomende eigenschappen van kansen afgeleid. Het begrip VOORWAARDELIJKE KANS wordt geïntroduceerd als een verhouding van kansen; hieruit worden de begrippen (ON)AFHANKELIJKE gebeurtenissen afgeleid. Met de REGEL VAN BAYES kunnen we de waarschijnlijkheid bepalen van mogelijke oorzaken gegeven een effect.

1.1 Inleiding

De KANSREKENING (ook waarschijnlijkheidsleer of probabiliteitsleer genoemd) houdt zich bezig met de studie van *gebeurtenissen* of *toevalsverschijnselen*. Dit zijn verschijnselen waarvan de individuele uitkomsten onzeker zijn, maar waar bij een groot aantal herhalingen een regelmatige verdeling van relatieve frequenties ontstaat.

De kansrekening ontstond bij de studie van kansspelen en dat is ook de invalshoek van waaruit wij dit onderwerp in eerste instantie zullen bekijken. Toch dient meteen gezegd dat de kansrekening haar grootste successen boekt(e) op andere terreinen, zoals

- vergelijkende studies van afmetingen in de biologie en de sociologie;

- de genetica;
- het verzekeringswezen;
- beslissingsproblemen;
- studie van verkeersstromen (inclusief telefoonverkeer);
- het wegfilteren van ruis uit communicatiesystemen;
- kwaliteitsstudie bij productieprocessen;
- studie van verspreiding van epidemieën;
- data-mining;
- de quantummechanica.

Voorbeeld 1.1 We starten met een gokkersprobleem dat de kansrekening als wetenschappelijke theorie lanceerde. Chevalier de Méré was een Franse schrijver in de 17de eeuw die wel hield van een gokje.

Bij een eerste spel gokte hij steeds opnieuw dat er minstens één zes zou gegooid worden wanneer een dobbelsteen vier keer werd gegooid. Bij dit spel won hij systematisch.

Om het spel uitdagender te maken werd een nieuw spel bedacht om dubbelzes te gooien bij een worp met twee dobbelstenen. Chevalier de Méré redeneerde (correct) dat de kans op dubbelzes slechts $1/6$ is van de kans op een zes met één dobbelsteen. Hij redeneerde (incorrect) dat als men dus 6 keer meer gooide (i.e. 24 keer i.p.v. 4 keer) hij ook (systematisch) zou winnen. Helaas voor hem verloor hij zwaar.

Hij vroeg toen Blaise Pascal om hulp die daarop samen met Pierre de Fermat de grondslag legde voor de waarschijnlijkheidsleer. ∎

Voorbeeld 1.2 Welke kans is volgens jou het grootst?

- De kans dat iemand op een bepaalde dag jarig is;
- de kans dat twee mensen dezelfde verjaardag hebben. ∎

Het fysisch begrip *kans* lijkt intuïtief voldoende duidelijk. Toch moeten we goed opletten en zullen we een en ander nauwkeuriger moeten specifiëren.

Als we bv. zeggen "Tien tegen één dat het regent vandaag.", dan drukken we een kans op een niet wetenschappelijke manier uit. Kansen worden steeds voorgesteld door een getal tussen nul en één. Een kans gelijk aan nul correspondeert met een onmogelijkheid en een kans gelijk aan 1 met een zekerheid. Een gebeurtenis met als kans 0,000001 is zeer onwaarschijnlijk.

We zullen een goede maat nodig hebben voor onzekere *gebeurtenissen* en bijgevolg hebben we een *fysische* theorie nodig. Daarbij kunnen we niet alleen kansen berekenen, maar kunnen we onze resultaten op een operationele manier vergelijken met de realiteit. Anders komt men al te gemakkelijk tot paradoxen.

We moeten opletten met uitspraken als: "Negen kansen op tien dat het vandaag regent." Hoe moeten we dat immers controleren?[1] Als we echter zeggen: "Als ik een evenwichtig muntstuk tos, dan is de kans op munt 0,50", dan kan dat wel geverifieerd worden aan de hand van een (groot aantal) *experimenten*. Het is zo dat de "kans" overeenkomt met de relatieve frequentie *op lange termijn*. Verder heeft de beschrijving van het opgooien van een muntstuk twee onderdelen:

1. het opstellen van een lijst van mogelijke uitkomsten;
2. het bepalen van de kans van elke uitkomst.

Dit soort beschrijving vormt meteen de basis voor alle kansmodellen. De verzamelingenleer speelt hierbij een grote rol.

1.2 Universum of uitkomstenruimte

Definitie 1.3 Het UNIVERSUM of de UITKOMSTENRUIMTE van een experiment is de verzameling van alle mogelijke uitkomsten van dit experiment en wordt genoteerd met Ω. ∎

Opmerking 1.4 Het is van belang dat de uitkomstenruimte *volledig* is: elke mogelijke uitkomst van een experiment moet tot Ω behoren. Bovendien moet elke uitkomst van een experiment overeenkomen met *juist één* element van Ω.

[1] De uitspraak is niet zinloos!

Samengevat moet het na het uitvoeren van een experiment mogelijk zijn om eenduidig aan te geven welk element van Ω zich heeft voorgedaan. ∎

Voorbeeld 1.5 We bekijken de uitkomstenruimte van enkele eenvoudige experimenten:

- bij het opgooien van een muntstuk is $\Omega = \{\text{Kop}, \text{Munt}\}$;
- bij het opgooien van 1 dobbelsteen is $\Omega = \{1, 2, 3, 4, 5, 6\}$;
- bij het opgooien van 2 dobbelstenen wordt dit

$$\Omega = \{1, 2, 3, 4, 5, 6\} \times \{1, 2, 3, 4, 5, 6\} = \{(1, 1), (1, 2), \ldots, (6, 6)\};$$

- wanneer men willekeurig een punt kiest in de eenheidscirkel dan is

$$\Omega = \{(x, y) \in \mathbb{R}^2 \mid x^2 + y^2 \leq 1\};$$

- bij het trekken van één kaart uit een (klassiek) kaartspel is

$$\begin{aligned}\Omega = \{&\diamondsuit 1, \diamondsuit 2, \ldots, \diamondsuit 13, \clubsuit 1, \clubsuit 2, \ldots, \clubsuit 13,\\ &\heartsuit 1, \heartsuit 2, \ldots, \heartsuit 13, \spadesuit 1, \spadesuit 2 \ldots, \spadesuit 13\}.\end{aligned}$$

∎

1.3 Gebeurtenissen

De uitkomstenruimte is een opsomming van alle mogelijke uitkomsten van een experiment, maar vaak zijn we niet geïnteresseerd in een specifieke uitkomst. Als je bv. hebt ingezet op 'rood' bij roulette, dan is het enige wat je interesseert of de uitkomst rood is of niet.

Voorbeeld 1.6 Wanneer een munstuk éénmaal wordt opgegooid, dan is $\Omega = \{\text{K}, \text{M}\}$. We kunnen dan geïnteresseerd zijn in de volgende gebeurtenissen:

1. de uitkomst is kop;
2. de uitkomst is kop of munt;
3. de uitkomst is tegelijkertijd kop en munt (het is hoogst onwaarschijnlijk dat deze gebeurtenis zich voordoet!);

4. de uitkomst is geen kop.

Al deze gebeurtenissen kunnen als een deelverzameling van Ω beschreven worden. We vinden respectievelijk:

1. $A_1 = \{\mathrm{K}\}$;
2. $A_2 = \{\mathrm{K}\} \cup \{\mathrm{M}\}$;
3. $A_3 = \{\mathrm{K}\} \cap \{\mathrm{M}\}$;
4. $A_4 = \overline{\{\mathrm{K}\}} = \Omega \setminus \{K\}$. ∎

Voorbeeld 1.7 Bij het éénmaal opgooien van een dobbelsteen is $\Omega = \{1, 2, 3, 4, 5, 6\}$. We kunnen spreken over de volgende gebeurtenissen:

1. de uitkomst is het getal 1, of $A_1 = \{1\}$;
2. de uitkomst is een even getal, of $A_2 = \{2, 4, 6\}$;
3. de uitkomst is even en is niet groter dan 3, of $A_3 = \{2, 4, 6\} \cap \overline{\{4, 5, 6\}}$;
4. de uitkomst is niet even, of $A_4 = \overline{\{2, 4, 6\}}$. ∎

Definitie 1.8 Een GEBEURTENIS is een deelverzameling van de uitkomstenruimte. Een ENKELVOUDIGE of ELEMENTAIRE gebeurtenis is een singleton; een SAMENGESTELDE GEBEURTENIS heeft cardinaliteit groter dan 1. ∎

Gebeurtenissen die geen gemeenschappelijke uitkomsten hebben noemt men DISJUNCT. Disjuncte gebeurtenissen kunnen dus nooit samen voorkomen. Wanneer de gebeurtenissen A en B disjunct zijn dan geldt $A \cap B = \emptyset$.

Voorbeeld 1.9 Bij het gooien van een dobbelsteen geldt:

1. de gebeurtenis $A_1 = \{1\}$ is enkelvoudig;
2. de gebeurtenis $A_2 = \{2, 4, 6\}$ is samengesteld;
3. de gebeurtenissen A_1 en A_2 zijn disjunct;
4. de gebeurtenissen A_2 en A_4 zijn disjunct. ∎

Zoals uit de voorbeelden blijkt kan men startend met de gebeurtenissen A en B de volgende gebeurtenissen vormen:

1. A **of** B, of wiskundig genoteerd $A \cup B$;
2. A **en** B, of wiskundig genoteerd $A \cap B$;
3. **niet** A, of wiskundig genoteerd $\overline{A}$.

Opmerking 1.10 Uit het feit dat voor twee gebeurtenissen A_1 en A_2 geldt dat $A_1 \cup A_2$ eveneens een gebeurtenis is, leidt men gemakkelijk (door inductie) af dat de unie van n (met $n \in \mathbb{N}$) gebeurtenissen A_1 t.e.m. A_n eveneens een gebeurtenis is. Hetzelfde geldt natuurlijk voor de doorsnede van gebeurtenissen. ∎

Voorbeeld 1.11 (**Aftelbaar oneindige uitkomstenruimte**) Beschouw een experiment waarbij men herhaaldelijk een muntstuk opwerpt totdat kop wordt bekomen. In dit geval is

$$\Omega = \{\omega_1, \omega_2, \omega_3, \ldots\},$$

waarbij ω_i de gebeurtenis voorstelt dat de eerste $i-1$ worpen munt zijn en de i-de worp kop. We kunnen dan bv. geïnteresseerd zijn of een *even* aantal worpen nodig was voor het gooien van de eerste kop, i.e.:

$$A = \{\omega_2, \omega_4, \omega_6, \ldots\}.$$

Dit is een *(aftelbaar) oneindige* unie van elementen van Ω. We zouden dit ook graag als een gebeurtenis aanzien. ∎

1.3.1 Oefeningen

1. Een dobbelsteen wordt éénmaal opgegooid. Geef de volgende gebeurtenissen door opsomming:
   a) er werd een even getal gegooid;
   b) er werd een priemgetal gegooid;
   c) er werd een even getal of een priemgetal gegooid;
   d) er werd een even getal en een priemgetal gegooid.

2. Er wordt tweemaal gegooid met een dobbelsteen. Omschrijf de volgende gebeurtenissen:

   a) er wordt 1 gegooid bij de eerste worp;

   b) de twee worpen vertonen hetzelfde aantal ogen;

   c) het minimum van de worpen is drie.

3. Geef de volgende gebeurtenissen voor het experiment in Voorbeeld 1.11:

   a) er zijn juist vijf worpen nodig;

   b) er zijn hoogstens vijf worpen nodig;

   c) er zijn minstens vijf worpen nodig.

1.4 Kansen en kansruimte

We wensen nu aan elke gebeurtenis A een getal te koppelen dat uitdrukt hoe waarschijnlijk het is dat deze gebeurtenis voorkomt bij het uitvoeren van een experiment. We noemen dit getal de KANS of WAARSCHIJNLIJKHEID van A, en we noteren deze kans als $\mathbb{P}(A)$.

Definitie 1.12 Het toekennen van kansen aan gebeurtenissen dient aan de volgende drie regels te voldoen.

1. Kansen zijn steeds positief: voor elke gebeurtenis A geldt dat

$$\mathbb{P}(A) \geq 0.$$

2. De uitkomstenruimte heeft kans 1:

$$\mathbb{P}(\Omega) = 1.$$

3. Wanneer A en B *disjuncte* gebeurtenissen zijn dan is

$$\mathbb{P}(A \cup B) = \mathbb{P}(A) + \mathbb{P}(B).$$

   Dit noemt men de SOMREGEL.

Wanneer de functie $\mathbb{P}$ aan de bovenstaande eigenschappen (axioma's) voldoet dan noemt men het drietal $(\Omega, \mathcal{P}(\Omega), \mathbb{P})$ een KANSRUIMTE[2]. ∎

Opmerking 1.13 De somregel moet als volgt uitgebreid worden wanneer we ook willen werken met oneindige uitkomstenruimtes. Wanneer $A_1, A_2, \ldots$ een collectie is van twee aan twee disjuncte gebeurtenissen (i.e. $A_i \cap A_j = \emptyset$ als $i \neq j$), dan moet gelden:

$$\mathbb{P}(\bigcup_{i=1}^{\infty} A_i) = \sum_{i=1}^{\infty} \mathbb{P}(A_i).$$

∎

Opmerking 1.14 Merk op dat de drie regels in Definitie 1.12 volledig in overeenstemming zijn met onze intuïtie. Inderdaad, kansen zijn positief en aangezien de gebeurtenis Ω steeds voorkomt moet de kans dat Ω voorkomt 100% zijn. Als we de kans van een gebeurtenis A zien als verhouding van het aantal voorkomens $N(A)$ van A tot het totaal aantal uitvoeringen van een experiment dan is het duidelijk, als A en B nooit samen kunnen optreden, dat

$$N(A \cup B) = N(A) + N(B)$$

waaruit de somregel volgt. ∎

Hoewel de drie regels in Definitie 1.12 zeer eenvoudig zijn kan men ze toch gebruiken om (alle) andere eigenschappen van kansen af te leiden. We bewijzen er hier enkele van.

Stelling 1.15 Kansen voldoen aan de volgende eigenschappen:

1. Voor elke gebeurtenis A geldt dat

$$\mathbb{P}(\overline{A}) = 1 - \mathbb{P}(A).$$

2. De onmogelijke gebeurtenis heeft kans nul:

$$\mathbb{P}(\emptyset) = 0.$$

[2]In deze cursus laten we enkele technische details m.b.t. de kansruimte buiten beschouwing.

3. Als $A \subseteq B$, dan is $\mathbb{P}(A) \leq \mathbb{P}(B)$; in het bijzonder geldt

$$\mathbb{P}(A) = \mathbb{P}(B) - \mathbb{P}(B \setminus A) \quad \text{als } A \subseteq B.$$

Dit laatste kan ook gezien worden als een *verschilregel*.

4. De UITGEBREIDE SOMREGEL is:

$$\mathbb{P}(A \cup B) = \mathbb{P}(A) + \mathbb{P}(B) - \mathbb{P}(A \cap B).$$ ∎

Bewijs 1. We weten dat A en $\overline{A}$ disjuncte gebeurtenissen zijn. Bovendien geldt $\Omega = A \cup \overline{A}$, en dus vinden we

$$\begin{aligned} 1 &= \mathbb{P}(\Omega) && \text{(axioma 2 voor kansen)} \\ &= \mathbb{P}(A \cup \overline{A}) && \text{(definitie complement)} \\ &= \mathbb{P}(A) + \mathbb{P}(\overline{A}) && \text{(somregel).} \end{aligned}$$

Dit is equivalent met het gestelde.

2. We weten dat $\mathbb{P}(\Omega) = 1$ en $\overline{\Omega} = \emptyset$, zodat het gestelde onmiddellijk volgt uit de voorgaande eigenschap.

3. Als $A \subseteq B$, dan geldt $B = A \cup (B \setminus A)$ waarbij A en $B \setminus A$ disjuncte gebeurtenissen zijn. Dus geldt

$$\begin{aligned} \mathbb{P}(B) &= \mathbb{P}\big(A \cup (B \setminus A)\big) && \text{(herschrijven } B\text{)} \\ &= \mathbb{P}(A) + \mathbb{P}(B \setminus A) && \text{(somregel)} \\ &\geq \mathbb{P}(A) && \text{(axioma 1 voor kansen).} \end{aligned}$$

We zien dus in het bijzonder dat:

$$\mathbb{P}(A) = \mathbb{P}(B) - \mathbb{P}(B \setminus A) \quad \text{als } A \subseteq B.$$

4. Er geldt $A \cup B = A \cup (B \setminus A)$. Dit is een unie van twee disjuncte gebeurtenissen. Bovendien geldt $B \setminus A = B \setminus (A \cap B)$, met $A \cap B \subseteq B$. M.b.v. de vorige eigenschap volgt hieruit dat:

$$\begin{aligned} \mathbb{P}(A \cup B) &= \mathbb{P}\big(A \cup (B \setminus A)\big) && \text{(herschrijven } A \cup B\text{)} \\ &= \mathbb{P}(A) + \mathbb{P}(B \setminus A) && \text{(somregel)} \\ &= \mathbb{P}(A) + \mathbb{P}\big(B \setminus (A \cap B)\big) && \text{(herschrijven } B \setminus A\text{)} \\ &= \mathbb{P}(A) + \mathbb{P}(B) - \mathbb{P}(A \cap B) && \text{(eigenschap 3).} \quad \diamondsuit \end{aligned}$$

1.4.1 Eindig universum

Als de uitkomstenruimte eindig is volgt uit de axioma's in Definitie 1.12 dat de som van de kansen van alle elementaire gebeurtenissen gelijk moet zijn aan één. Met andere woorden als $\Omega = \{\omega_1, \omega_2, \omega_3, \ldots, \omega_n\}$, dan geldt

$$\sum_{i=1}^{n} \mathbb{P}(\omega_i) = 1.$$

Wanneer de elementaire gebeurtenissen allen *even waarschijnlijk* zijn, i.e. $\mathbb{P}(\omega_i) = 1/n$, dan bekomt men de FORMULE VAN LAPLACE:

$$\mathbb{P}(A) = \frac{\#(A)}{\#(\Omega)}.$$

Opmerking 1.16 Vroeger nam men dit als kansdefinitie, maar dat blijkt tot problemen te leiden: voor de definitie van kans wordt gesteund op "even waarschijnlijk zijn" of "dezelfde kans hebben". Dit is een voorbeeld van een cirkelredenering. In het gewone taalgebruik zijn definities vaak circulair maar in de wiskunde is dit onaanvaardbaar. Verder ontbreekt het de formule ook aan algemeenheid. ∎

Voorbeeld 1.17 Men trekt willekeurig een kaart uit een klassiek kaartspel. Wat is de kans dat men een aas trekt?

De gebeurtenis "men trekt een aas" wordt voorgesteld door de volgende verzameling

$$A = \{\diamondsuit 1, \spadesuit 1, \heartsuit 1, \clubsuit 1\}.$$

De kardinaliteit van het universum Ω is uiteraard 52. Volgens de formule van Laplace krijgen we dus

$$\mathbb{P}(A) = \frac{\#(A)}{\#(\Omega)} = \frac{4}{52} = \frac{1}{13}.$$ ∎

1.4.2 Oefeningen

1. Men gooit éénmaal met een eerlijke dobbelsteen.
   a) Wat is de kans op een even getal?
   b) Wat is de kans op een oneven getal?

c) Wat is de kans op een priemgetal?

2. Wat is de kans om met twee dobbelstenen meer dan acht te gooien?

3. Een duistere urne bevat 6 blauwe en 4 rode bollen die verder identiek zijn. Men trekt één na één, twee bollen uit de urne met teruglegging van de getrokken bol. Wat is de kans dat men twee blauwe bollen heeft getrokken?

4. Wat is de kans om bij de Belgische lotto zes juiste cijfers te hebben? Er zitten 45 ballen in de trommel en er worden er zes getrokken zonder teruglegging.

5. Als huishoudens groepen mensen zijn die samenleven, dan vindt men voor de kansen op het aantal leden van een huishouden respectievelijk:

Omvang	1	2	3	4	5	6	7 of meer
Kans	0,236	0,320	0,181	0,156	0,069	0,024	0,014

Het kleine aantal huishoudens met meer dan 7 leden werd samengevoegd in de laatste categorie. Wat is de kans dat er meer dan twee mensen in een huishouden zijn?

6. Een toegangscode tot een (amateurs) computersysteem bestaat uit drie letters die door een randomgenerator worden toegewezen.

a) Wat is de kans dat een code wordt toegewezen zonder de letter x?

b) Wat is de kans dat een code wordt toegewezen waarvan alle letters verschillend zijn?

7. Bij een vervalst muntstuk is $\mathbb{P}(\mathrm{K}) = 6/10$. Wat is dan de kans op "munt"?

8. Veronderstel dat
$$\Omega = \{\omega_1, \omega_2, \omega_3, \omega_4, \omega_5\}$$
met
$$\mathbb{P}(\omega_1) = \frac{3}{10},\ \mathbb{P}(\omega_2) = \frac{2}{10},\ \mathbb{P}(\omega_3) = \frac{1}{10},\ \mathbb{P}(\omega_4) = \frac{1}{10} \text{ en } \mathbb{P}(\omega_5) = \frac{3}{10}.$$
Wat is $\mathbb{P}(\{\omega_2, \omega_4, \omega_5\})$?

9. Twee zijden van een dobbelsteen dragen het cijfer 1, drie dragen het cijfer 2 en één zijde het cijfer 3. Wat is de kans op een oneven getal als men aanneemt dat elke zijde even waarschijnlijk is?

10. Men vervalst een dobbelsteen zodanig dat de kans op een bepaald cijfer evenredig is met dit cijfer.

    a) Geef de elementaire kansen.

    b) Wat is de kans op een even getal?

    c) Wat is de kans op een oneven getal?

    d) Wat is de kans op een priemgetal?

    e) Wat is de kans op een even getal of een priemgetal?

    f) Wat is de kans op een oneven priemgetal?

11. Een doos bevat 10 lampen. Bij vergissing worden 4 afgekeurde lampen mee verpakt. Men neemt lukraak drie lampen uit de doos. Wat is $\mathbb{P}(\omega_i)$ waarbij ω_i de elementaire gebeurtenis voorstelt dat exact i met $0 \leq i \leq 3$ van de drie gekozen lampen defect zijn?

12. Hoe luidt de algemene uitgebreide somregel voor drie gebeurtenissen? Geef m.a.w. een uitdrukking voor

$$\mathbb{P}(A_1 \cup A_2 \cup A_3).$$

    Kun je de formule veralgemenen tot n gebeurtenissen?

13. Aan de Hogeschool Gent mogen (moeten?) 500 studenten 3 keuzevakken kiezen. Van de 500 studenten volgen er 320 cursus A, 190 volgen cursus B en 280 volgen cursus C. Er zijn 80 studenten die A en B volgen, 200 die A en C volgen terwijl 60 studenten B en C volgen.

    Bereken de kans dat een willekeurig gekozen student

    a) de drie cursussen volgt;

    b) A volgt maar niet B;

    c) C volgt maar niet B;

    d) A of C volgt maar niet B;

    e) A volgt maar B noch C.

1.5 Voorwaardelijke kansen en (on)afhankelijkheid van gebeurtenissen

Uitdrukkingen zijn vaak van de vorm

> als B voorkomt, dan is de waarschijnlijkheid dat A voorkomt gelijk aan p.

Hierbij kunnen B en A respectievelijk gebeurtenissen zijn zoals "het alarm gaat af" en "er is een inbraak aan de gang".

Om een formule te vinden voor p is het handig om weer eventjes frequentistisch te denken. Wanneer we een experiment herhaaldelijk uitvoeren dan kunnen we telkens het al dan niet voorkomen van A en/of B observeren. Veronderstel nu dat we enkel geïnteresseerd zijn in dié experimenten waarvoor B voorkomt. De verhouding van het aantal keer dat A voorkomt voor deze experimenten is $N(A \cap B)/N(B)$, aangezien B voor deze experimenten steeds voorkomt. Er geldt nu echter dat

$$\frac{N(A \cap B)}{N(B)} = \frac{N(A \cap B)/N}{N(B)/N}.$$

Als we deze verhoudingen nu als een waarschijnlijkheid beschouwen dan zien we dat het getal p redelijkerwijs moet gedefinieerd worden als

$$p = \frac{\mathbb{P}(A \cap B)}{\mathbb{P}(B)}.$$

Deze kans wordt genoteerd als $\mathbb{P}(A \mid B)$.

Definitie 1.18 Als $\mathbb{P}(B) > 0$, dan is de VOORWAARDELIJKE KANS dat A voorkomt als gegeven is dat B voorkomt gedefinieerd als:

$$\mathbb{P}(A \mid B) = \frac{\mathbb{P}(A \cap B)}{\mathbb{P}(B)}.$$

Opmerking 1.19 We spreken $\mathbb{P}(A \mid B)$ uit als "de kans op A gegeven B".

Voorbeeld 1.20 Er wordt tweemaal gegooid met een eerlijke dobbelsteen. Wat is de waarschijnlijkheid dat de som van beide ogen 7 of meer is als gegeven is dat de eerste worp een twee opleverde?

Intuïtitief is het duidelijk dat de gevraagde kans gelijk is aan $2/6$ aangezien we bij de tweede worp 5 of 6 moeten gooien.

Laat ons het voorbeeld echter nauwkeuriger uitwerken. We weten dat

$$\Omega = \{1,2,3,4,5,6\}^2,$$

met voor elke gebeurtenis A

$$\mathbb{P}(A) = \frac{\#(A)}{36}.$$

Veronderstel nu dat B de gebeurtenis is dat de eerste worp een twee toont, terwijl A de gebeurtenis is dat de som 7 of meer is. We hebben:

$$B = \{(2,b)\colon 1 \leq b \leq 6\} \quad \text{en} \quad A = \{(a,b) \in \Omega\colon a+b \geq 7\}.$$

Men gaat dan eenvoudig na dat

$$A \cap B = \{(2,5),(2,6)\},$$

en dus is volgens de definitie van voorwaardelijke kans

$$\mathbb{P}(A \mid B) = \frac{\mathbb{P}(A \cap B)}{\mathbb{P}(B)} = \frac{2/36}{6/36} = \frac{2}{6},$$

in overeenstemming met het intuïtief bekomen antwoord. ∎

In het algemeen is het zo dat het voorkomen van een gebeurtenis B de waarschijnlijkheid dat een andere gebeurtenis A voorkomt wijzigt. Zo zal bv. het feit dat "het alarm afgaat" (gebeurtenis B) een invloed hebben op de waarschijnlijkheid dat er een "inbraak aan de gang is" (gebeurtenis A), i.e. de waarschijnlijkheid $\mathbb{P}(A)$ wordt vervangen door de waarschijnlijkheid $\mathbb{P}(A \mid B)$. Wanneer de waarschijnlijkhheid echter *niet* verandert dan zeggen we dat A en B onafhankelijk zijn. Dit is echter slechts gedefinieerd wanneer $\mathbb{P}(B) > 0$. Een formele definitie van onafhankelijkheid wordt hieronder gegeven.

Definitie 1.21 Twee gebeurtenissen A en B zijn ONAFHANKELIJK als

$$\mathbb{P}(A \cap B) = \mathbb{P}(A)\,\mathbb{P}(B).$$

∎

Opmerking 1.22 Intuïtief zijn gebeurtenissen onafhankelijk als "ze niets met elkaar te maken hebben". Het al of niet voorkomen van de ene gebeurtenis levert je m.a.w. geen extra informatie op m.b.t. de waarschijnlijkheid van het voorkomen van de andere gebeurtenis.

We kunnen dit ook zien a.d.h.v. de definities. Veronderstel dat $\mathbb{P}(A) > 0$ en $\mathbb{P}(B) > 0$ en dat A en B onafhankelijk zijn volgens Definitie 1.21. Dan geldt

$$\begin{aligned} \mathbb{P}(A)\,\mathbb{P}(B) &= \mathbb{P}(A \cap B) && (A \text{ en } B \text{ zijn onafhankelijk}) \\ &= \mathbb{P}(B)\,\mathbb{P}(A\,|B) && (\text{definitie voorwaardelijke kans}) \end{aligned}$$

en dus is

$$\mathbb{P}(A) = \mathbb{P}(A \mid B).$$

∎

Opmerking 1.23 (Onafhankelijke versus disjuncte gebeurtenissen)
Verwar *onafhankelijke* gebeurtenissen niet met *disjuncte* gebeurtenissen. Twee disjuncte gebeurtenissen A en B met $\mathbb{P}(A) > 0$ en $\mathbb{P}(B) > 0$ zijn steeds *afhankelijk*, want

$$\mathbb{P}(A \cap B) = \mathbb{P}(\emptyset) = 0 \neq \mathbb{P}(A)\,\mathbb{P}(B).$$

Dit is ook logisch; als je bv. weet dat A voorkomt dan weet je zeker dat B niet voorkomt. De kennis over het al of niet voorkomen van A heeft dus zeker een invloed op je kennis van het al dan niet voorkomen van B. ∎

Voorbeeld 1.24 Beschouw een standaard kaartspel waaruit willekeurig een kaart wordt getrokken. We stellen door A de gebeurtenis voor dat de getrokken kaart een $\diamondsuit$ is, terwijl B de gebeurtenis voorstelt dat de getrokken kaart een aas is. We gaan nu na of A en B onafhankelijk zijn. Intuïtief lijkt dit zo te moeten zijn omdat het feit dat je weet dat de kaart een $\diamondsuit$ is geen invloed heeft op het al dan niet een aas zijn aangezien alle 4 de kaarttypes juist één aas hebben. We gaan dit nu formeel na. We vinden

$$\mathbb{P}(A) = \frac{13}{52} = \frac{1}{4} \quad \text{en} \quad \mathbb{P}(B) = \frac{4}{52},$$

terwijl, aangezien er juist één kaart is die $\diamondsuit 1$ is:

$$\mathbb{P}(A \cap B) = \frac{1}{52} = \mathbb{P}(A)\,\mathbb{P}(B).$$

De gebeurtenissen A en B zijn dus inderdaad onafhankelijk. ∎

Voorbeeld 1.25 (Vervolg Voorbeeld 1.20) De gebeurtenis A was "de som is zeven of meer", terwijl B de gebeurtenis is "er werd twee gegooid met de eerste dobbelsteen".

We weten reeds dat

$$\mathbb{P}(A \cap B) = \frac{2}{36}.$$

We kunnen ook eenvoudig nagaan dat

$$\mathbb{P}(A) = \frac{21}{36} = \frac{7}{12} \quad \text{en} \quad \mathbb{P}(B) = \frac{6}{36} = \frac{1}{6}.$$

Nu is

$$\mathbb{P}(A)\,\mathbb{P}(B) = \frac{7}{72} \neq \frac{2}{36} = \mathbb{P}(A \cap B).$$

De gebeurtenissen A en B zijn dus afhankelijk. ∎

Voorbeeld 1.26 Beschouw een familie met 3 kinderen waarbij elk kind dezelfde kans heeft om een jongen of een meisje te zijn, onafhankelijk van het geslacht van de andere kinderen. Beschouw de volgende gebeurtenissen:

$$A = \{\text{alle kinderen hebben hetzelfde geslacht}\},$$
$$B = \{\text{er is hoogstens één jongen}\},$$
$$C = \{\text{er is minstens één jongen en minstens één meisje}\}.$$

We tonen aan dat A en B onafhankelijk zijn. Als we alle acht de (even waarschijnlijke) mogelijkheden opsommen dan vinden we dat in 2 ervan alle kinderen hetzelfde geslacht hebben (i.e. A komt voor), en dat in 4 van de gevallen er hoogstens één jongen is (i.e. B komt voor). Hieruit volgt dat

$$\mathbb{P}(A) = \frac{2}{8} = \frac{1}{4} \quad \text{en} \quad \mathbb{P}(B) = \frac{4}{8} = \frac{1}{2}.$$

Er is slechts één geval waarbij A en B terzelfdertijd voorkomen (als alle drie de kinderen meisjes zijn), en dus is

$$\mathbb{P}(A \cap B) = \frac{1}{8} = \mathbb{P}(A)\,\mathbb{P}(B).$$

De gebeurtenissen A en B zijn dus onafhankelijk. Op dezelfde manier toont men aan dat B en C onafhankelijk zijn.

Wat met A en C? Men vindt dat $\mathbb{P}(C) = 3/4$, en

$$\mathbb{P}(A \cap C) = 0 \neq \mathbb{P}(A)\,\mathbb{P}(C),$$

en dus zijn de gebeurtenissen A en C *niet* onafhankelijk. ∎

Stelling 1.27 (Kettingregel) Wanneer A_1 t.e.m. A_n n gebeurtenissen zijn waarvoor

$$\mathbb{P}(A_1 \cap A_2 \cap \cdots \cap A_{n-1}) > 0$$

dan geldt

$$\begin{aligned}&\mathbb{P}(A_1 \cap A_2 \cap \cdots \cap A_n) = \\ &\quad \mathbb{P}(A_1)\,\mathbb{P}(A_2 \mid A_1)\,\mathbb{P}(A_3 \mid A_1 \cap A_2) \cdots \mathbb{P}(A_n \mid A_1 \cap A_2 \cap \cdots \cap A_{n-1}).\end{aligned}$$

De waarschijnlijkheid van de doorsnede kan dus geschreven worden als een product van voorwaardelijke waarschijnlijkheden. Dit noemt men ook de KETTINGREGEL of PRODUCTREGEL. ∎

Bewijs Uit de definitie van voorwaardelijke kans volgt dat

$$\mathbb{P}(B \cap A) = \mathbb{P}(B)\,\mathbb{P}(A \mid B)$$

als $\mathbb{P}(B) > 0$. We passen dit nu éénmaal toe:

$$\begin{aligned}&\mathbb{P}(A_1 \cap A_2 \cap \cdots \cap A_{n-1} \cap A_n) \\ &\quad = \mathbb{P}\big((A_1 \cap A_2 \cap \cdots \cap A_{n-1}) \cap A_n\big) \\ &\quad = \mathbb{P}(A_1 \cap A_2 \cap \cdots \cap A_{n-1})\,\mathbb{P}(A_n \mid A_1 \cap A_2 \cap \cdots \cap A_{n-1}).\end{aligned}$$

Dit proces kunnen we ook toepassen op $\mathbb{P}(A_1 \cap A_2 \cap \cdots \cap A_{n-1})$:

$$\begin{aligned}&\mathbb{P}(A_1 \cap A_2 \cap \cdots \cap A_{n-1}) \\ &\quad = \mathbb{P}(A_1 \cap A_2 \cap \cdots \cap A_{n-2})\,\mathbb{P}(A_{n-1} \mid A_1 \cap A_2 \cap \cdots \cap A_{n-2}).\end{aligned}$$

Men kan dit blijven herhalen om de te bewijzen formule te bekomen. ♢

Voorbeeld 1.28 (Verjaardagsparadox) Veronderstel dat er k personen in een kamer aanwezig zijn. Veronderstel verder dat de verjaardagen van de personen onafhankelijk zijn van elkaar en dat elke dag van de 365 dagen van het jaar dezelfde waarschijnlijkheid heeft om iemands verjaardag te zijn.

Vind een formule om te berekenen wat de kans is dat alle personen een verschillende verjaardag hebben. Wat is de eerste waarde voor k waarvoor deze kans minder dan 50% is?

We kunnen de verjaardagen van de personen voorstellen m.b.v. een k-tal $(x_1, x_2, \ldots, x_k)$. We beschouwen nu een aantal gebeurtenissen A_1, A_2, t.e.m. A_k waarbij A_j de gebeurtenis voorstelt dat x_j verschillend is van alle

elementen uit de verzameling $\{x_1, x_2, \ldots, x_{j-1}\}$. De kans dat alle personen een verschillende verjaardag hebben is dus

$$\mathbb{P}(A_1 \cap A_2 \cap A_3 \cap \cdots \cap A_k).$$

M.b.v. de productregel schrijven we dit als:

$$\mathbb{P}(A_1)\,\mathbb{P}(A_2 \mid A_1)\,\mathbb{P}(A_3 \mid A_1 \cap A_2) \cdots \mathbb{P}(A_k \mid A_1 \cap A_2 \cap \cdots \cap A_{k-1}).$$

Wat is nu

$$\mathbb{P}(A_j \mid A_1 \cap A_2 \cap \cdots \cap A_{j-1})?$$

Dit is de kans dat de verjaardag van persoon j verschillend is van de verjaardagen van personen 1 t.e.m. $j-1$ *als gegeven is dat personen* 1 *t.e.m.* $j-1$ *allemaal een verschillende verjaardag hebben.* Aangezien er dus reeds $j-1$ dagen ingenomen zijn, zijn er nog $365-(j-1) = 366-j$ "vrije"dagen over voor de verjaardag van persoon j. We vinden dus

$$\mathbb{P}(A_j \mid A_1 \cap A_2 \cap \cdots \cap A_{j-1}) = \frac{366-j}{365}.$$

De kans p dat alle k personen een verschillende verjaardag hebben is dus

$$\prod_{j=1}^{k} \frac{366-j}{365}.$$

Dit is de gevraagde formule.

In tabelvorm vinden we (voor enkele waarden van k)

k	5	10	15	20	21	22	23	24	30	40
p	0.973	0.883	0.747	0.589	0.556	0.524	0.493	0.462	0.294	0.109

Hieruit zien we dat wanneer er 23 personen in de kamer zijn het voor de eerste keer zo is dat de kans dat ze allemaal een verschillende verjaardag hebben kleiner is dan 50%.

Intuïtief lijkt dit aantal klein maar er zijn natuurlijk reeds

$$\binom{23}{2} = \frac{23 \times 22}{2} = 253$$

paren van personen aanwezig in een kamer met 23 personen. ∎

1.5.1 Oefeningen

1. Een vaas bevat 20 witte en 30 zwarte bollen. Men trekt willekeurig drie bollen uit de vaas. Bereken de kans om 3 witte bollen te trekken

   a) met teruglegging;

   b) zonder teruglegging.

2. De kans dat van een koppel de man nog leeft over 10 jaar is $1/4$. De kans dat de vrouw dan nog leeft is $1/3$. (De gegevens komen uit zogenaamde *sterftetafels*, een veel gebruikt instrument in het gebied van de (levens)verzekeringen.) Bereken de kans dat

   a) beiden nog leven over 10 jaar;

   b) ten minste één van de twee nog leeft over 10 jaar;

   c) geen van beiden nog leeft;

   d) alleen de vrouw nog leeft.

   Welke veronderstelling heb je gemaakt tijdens je berekeningen?

3. Doos A bevat 8 onderdelen, waarvan er 3 defect zijn. Doos B bevat 5 onderdelen, waarvan er 2 stuk zijn. Men neemt lukraak een onderdeel uit elke doos.

   a) Wat is de kans dat beide onderdelen intact zijn?

   b) Wat is de kans dat juist één van de twee onderdelen defect is?

   c) Als juist één van de twee onderdelen defect is, wat is dan de kans dat het defecte stuk uit doos A komt?

4. De kansen dat 3 boogschutters een bepaald doelwit raken zijn respectievelijk $1/6, 1/4$ en $1/3$. Iedere boogschutter schiet één keer naar het doel.

   a) Wat is de kans dat juist 1 van de 3 het doel raakt?

   b) Als juist 1 schutter het doel raakte, wat is dan de kans dat het de eerste schutter was?

5. Ga het volgende formeel na: als A en B onafhankelijke gebeurtenissen zijn, dan zijn ook $\overline{A}$ en B onafhankelijke gebeurtenissen.

   Hieruit volgt dan ook onmiddellijk dat A en $\overline{B}$ onafhankelijk zijn als A en B onafhankelijk zijn. Hetzelfde geldt dan uiteraard voor $\overline{A}$ en $\overline{B}$.

6. Veronderstel dat A en B gebeurtenissen zijn waarvoor $\mathbb{P}(A) = 1/4$, $\mathbb{P}(A \cap B) = 1/6$ en $\mathbb{P}(B) = p$.

   a) Bereken p als men ervan uitgaat dat A en B onafhankelijk zijn.

   b) Bereken p als $B \subset A$.

7. Veronderstel dat A en B onafhankelijke gebeurtenissen zijn met $\mathbb{P}(A) = 1/2$ en $\mathbb{P}(A \cap B) = 1/3$. Bereken dan

   a) $\mathbb{P}(B)$;

   b) $\mathbb{P}(A \mid B)$;

   c) $\mathbb{P}(\overline{B} \mid A)$.

8. Een boekhouder die het beleid nakijkt van een bedrijf op het vlak van verkopen op krediet stelt vast dat van de 500 klanten er 20% een balans hebben van meer dan 750 EUR, de anderen hebben 750 EUR of minder. Qua duurtijd van de rekeningen hebben 30% van de rekeningen meer dan 3 maanden open gestaan, de andere minder. Van degene die meer dan drie maanden open stonden waren er 20% met een balans van meer dan 750 EUR. We kunnen dit in tabelvorm weergeven:

		A	A′	
		$>$ 750 EUR	$\leq$ 750 EUR	Totaal
B	$\leq$ 3 maand	70	280	350
B′	$>$ 3 maand	30	120	150
	Totaal	100	400	500

   a) Hoeveel elementen bevat Ω?

   b) Wat is de kans dat een willekeurig gekozen balans minder is dan 750 EUR en meer dan 3 maanden open stond? Bepaal deze kans, gebruikmakend van voorwaardelijke kansen.

   c) Ga formeel na dat A en B' onafhankelijke gebeurtenissen zijn.

   d) Bereken $\mathbb{P}(A \cup B)$ op twee manieren, eerst uit de tabelwaarden en dan met de algemene somregel.

1.5.2 Regel van Bayes

Voorbeeld 1.29 Een virusscanner ontdekt een bepaald virus. De virusscanner is 95% betrouwbaar voor besmette computers, en voor een niet-besmette computer geeft de virusscanner in 1% van de gevallen aan dat deze toch besmet zou zijn. Laat B de gebeurtenis voorstellen dat de computer besmet is terwijl V de gebeurtenis is dat de virusscanner het virus ontdekt (en dus zegt dat het virus aanwezig is). Bovenstaande gegevens worden dus wiskundig als volgt uitgedrukt:

$$\mathbb{P}(V \mid B) = 95\% \quad \text{en} \quad \mathbb{P}(V \mid \overline{B}) = 1\%.$$

Wanneer nu de virusscanner aangeeft dat de computer besmet is met het virus dan willen we uiteraard weten wat de kans is dat de computer daadwerkelijk besmet is, m.a.w. we wensen de waarde van

$$\mathbb{P}(B \mid V)$$

te kennen. Deze voorwaardelijke waarschijnlijkheid werd echter niet gegeven.

Aangezien $B \cap V = V \cap B$ volgt uit Definitie 1.18 dat

$$\mathbb{P}(B \mid V)\,\mathbb{P}(V) = \mathbb{P}(V \mid B)\,\mathbb{P}(B),$$

waaruit we de gewenste waarschijnlijkheid kunnen bepalen als

$$\mathbb{P}(B \mid V) = \frac{\mathbb{P}(V \mid B)\,\mathbb{P}(B)}{\mathbb{P}(V)}.$$

Op dit moment kunnen we de gevraagde waarschijnlijkheid nog steeds *niet* berekenen omdat we niet weten wat $\mathbb{P}(B)$ en $\mathbb{P}(V)$ zijn.

Veronderstel nu dat men weet dat wereldwijd 4% van de computers besmet zijn met het virus, dus $\mathbb{P}(B) = 4\%$. De waarschijnlijkheid $\mathbb{P}(V)$ kan nu ook bepaald worden uit de gegevens: inderdaad er geldt steeds dat

$$V = V \cap \Omega = V \cap (B \cup \overline{B}) = (V \cap B) \cup (V \cap \overline{B}).$$

We hebben V dus geschreven als de unie van twee disjuncte gebeurtenissen, en dus kunnen we de somregel toepassen:

$$\begin{aligned}\mathbb{P}(V) &= \mathbb{P}(V \cap B) + \mathbb{P}(V \cap \overline{B}) \\ &= \mathbb{P}(B)\,\mathbb{P}(V \mid B) + \mathbb{P}(\overline{B})\,\mathbb{P}(V \mid \overline{B}).\end{aligned}$$

Als we hier nu het cijfervoorbeeld verder uitwerken dan vinden we

$$\begin{aligned}\mathbb{P}(B \mid V) &= \frac{\mathbb{P}(V \mid B)\,\mathbb{P}(B)}{\mathbb{P}(V)} \\ &= \frac{\mathbb{P}(V \mid B)\,\mathbb{P}(B)}{\mathbb{P}(B)\,\mathbb{P}(V \mid B) + \mathbb{P}(\overline{B})\,\mathbb{P}(V \mid \overline{B})} \\ &= \frac{4\% \times 95\%}{4\% \times 95\% + (1 - 4\%) \times 1\%} \\ &= 0.7983\end{aligned}$$

De waarschijnlijkheid dat de computer besmet is als de virusscanner dit aangeeft is dus minder dan 80%! ∎

We veralgemenen de situatie uit het voorbeeld. Veronderstel dat een gebeurtenis A n verschillende en elkaar wederzijds uitsluitende oorzaken $B_1, B_2, \ldots, B_n$ heeft, dan is

$$\Omega = B_1 \cup B_2 \cup \cdots \cup B_n.$$

Men heeft dan ook

$$\begin{aligned}A &= A \cap \Omega \\ &= A \cap (B_1 \cup B_2 \cup \cdots \cup B_n) \\ &= (A \cap B_1) \cup (A \cap B_2) \cup \cdots \cup (A \cap B_n),\end{aligned}$$

waaruit (aangezien de gebeurtenissen $A \cap B_i$ en $A \cap B_j$ disjunct zijn als $i \neq j$) de WET VAN DE TOTALE WAARSCHIJNLIJKHEID volgt:

$$\mathbb{P}(A) = \sum_{i=1}^{n} \mathbb{P}(A \cap B_i).$$

Verder heeft men

$$\mathbb{P}(B_j \mid A) = \frac{\mathbb{P}(A \cap B_j)}{\mathbb{P}(A)}$$

zodat

$$\mathbb{P}(B_j \mid A) = \frac{\mathbb{P}(A \cap B_j)}{\sum_{i=1}^{n} \mathbb{P}(A \cap B_i)}$$

Nu is $\mathbb{P}(A \cap B_j) = \mathbb{P}(B_j)\,\mathbb{P}(A \mid B_j)$, zodat men vindt

$$\mathbb{P}(B_j \mid A) = \frac{\mathbb{P}(B_j)\,\mathbb{P}(A \mid B_j)}{\sum_{i=1}^{n} \mathbb{P}(B_i)\,\mathbb{P}(A \mid B_i)}$$

en dit is de REGEL VAN BAYES.

Stelling 1.30 (Regel van Bayes) Gegeven een gebeurtenis A met n elkaar wederzijds uitsluitende oorzaken B_i, i.e. $B_i \cap B_j = \emptyset$ als $i \neq j$ en $\Omega = \cup_{i=1}^{n} B_i$, dan geldt voor elke j dat

$$\mathbb{P}(B_j \mid A) = \frac{\mathbb{P}(B_j)\,\mathbb{P}(A \mid B_j)}{\mathbb{P}(A)} = \frac{\mathbb{P}(B_j)\,\mathbb{P}(A \mid B_j)}{\sum_{i=1}^{n} \mathbb{P}(B_i)\,\mathbb{P}(A \mid B_i)}.$$ ∎

Bewijs Dit werd zonet geleverd. ♢

Opmerking 1.31 Dit resultaat is belangrijk voor het opstellen van zogenaamde *boomdiagrammen of beslissingsbomen* in het kader van opeenvolgende beslissingen. ∎

Voorbeeld 1.32 Drie machines M1, M2 en M3 worden ingezet voor het vervaardigen van computer chips. De machines verschillen in ouderdom en de ervaring leert dat van de chips die afkomstig zijn van de oudste machine M1 (die instaat voor 20% van de totale productie) 8% niet voldoet. Voor M2 (die 30% voor haar rekening neemt) is dit 5% en voor M3 (die de overige 50% vervaardigt) is dit nog 2%. Wat is de kans dat, als een defecte chip wordt geconstateerd, deze afkomstig is van M1?

Wanneer de gebeurtenis "de chip is defect" zich voordoet dan werd deze veroorzaakt door één van de drie gebeurtenissen M_i ($i \in \{1,2,3\}$) dat de chip werd geproduceerd door de eerste, tweede of derde machine. M.b.v. de regel van Bayes vinden we dus

$$\mathbb{P}(M_1 \mid D) = \frac{\mathbb{P}(M_1)\,\mathbb{P}(D \mid M_1)}{\sum_{i=1}^{3} \mathbb{P}(M_i)\,\mathbb{P}(D \mid M_i)}.$$

Uit de opgave weten we

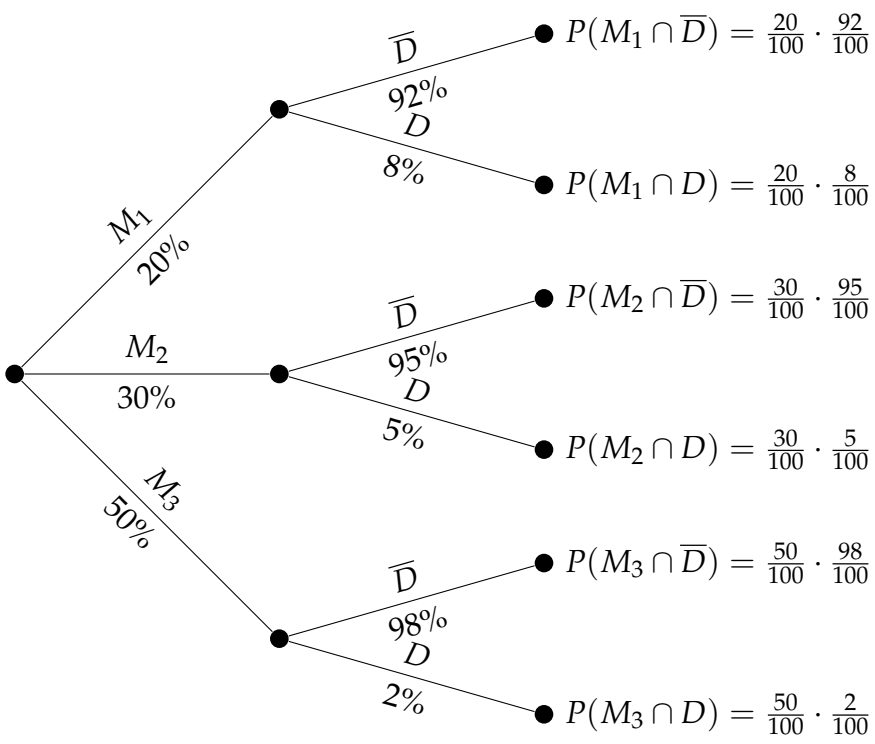
$$\mathbb{P}(M_1) = \frac{2}{10}, \quad \mathbb{P}(M_2) = \frac{3}{10} \quad \text{en} \quad \mathbb{P}(M_3) = \frac{5}{10}.$$

We kennen ook de volgende voorwaardelijke kansen:

$$\mathbb{P}(D \mid M_1) = \frac{8}{100}, \quad \mathbb{P}(D \mid M_2) = \frac{5}{100} \quad \text{en} \quad \mathbb{P}(D \mid M_3) = \frac{2}{100}.$$

Hieruit berekenen we nu dat

$$\mathbb{P}(M_1 \mid D) = \frac{20 \times 8}{20 \times 8 + 30 \times 5 + 50 \times 2} = \frac{16}{41} \approx 39\%.$$



Figuur 1.1: Een beslissingsboom voor het probleem in Voorbeeld 1.32.

Men leest dit ook gemakkelijk af uit de beslissingsboom in Figuur 1.1.

Welk percentage van de totale productie voldoet niet?

We berekenen de waarschijnlijkheid op een defecte chip, er rekening mee houdend dat een chip door één van de drie machines geproduceerd is.

$$\begin{aligned}\mathbb{P}(D) &= \mathbb{P}(D \cap M_1) + \mathbb{P}(D \cap M_2) + \mathbb{P}(D \cap M_3) \\ &= \sum_{i=1}^{3} \mathbb{P}(M_i)\,\mathbb{P}(D \mid M_i) = \frac{41}{1000}.\end{aligned}$$

Het percentage dat niet voldoet is dus 4.1%. ∎

1.5.3 Oefeningen

1. In het domein "De Visput" heeft 28% van de wandelaars juist één hond mee. Van deze wandelaars-met-één-hond zijn 45% mannen en 55% vrouwen. Verder blijken mannen de voorkeur te geven aan een

grote (macho) hond: 70% van de mannen gaat op stap met een grote hond en slechts 30% heeft een kleinere hond bij. Vrouwen verkiezen een kleinere hond: 75% van de vrouwen gaat op stap met een kleine hond en slechts 25% holt achter een grote hond aan. Als je één kleine hond tegenkomt, bepaal dan de kans dat zijn baas een vrouw is.

2. Veronderstel dat op een multiple choice examen de volgende situatie zich voordoet. Ofwel weet een student het antwoord met 100% zekerheid, ofwel weet hij het antwoord helemaal niet, en dan gokt hij. Veronderstel dat de kans dat een student het antwoord weet gelijk is aan p, en dat er m mogelijke antwoorden zijn per vraag.

   a) Wat is de waarschijnlijkheid dat indien het antwoord op een bepaalde vraag correct is, de student het antwoord ook daadwerkelijk wist?

   b) Wat wordt deze waarschijnlijkheid als het aantal antwoorden oneindig wordt?

3. Een computer programma bestaat uit twee modules. De eerste module bevat een fout met waarschijnlijkheid 0.2; de tweede module is complexer en de waarschijnlijkheid dat deze een fout bevat is 0.4. Fouten in de twee modules zijn onafhankelijk van elkaar. Als er enkel een fout zit in de eerste module dan crasht het programma met waarschijnlijkheid 0.5. Voor de tweede module is dit 0.8. Wanneer beide modules een fout bevatten, dan crasht het programma met 90% waarschijnlijkheid.

   Bij uitvoering crasht het programma. Wat is de waarschijnlijkheid dat beide modules een fout bevatten?

HOOFDSTUK **2**

Kans- of toevalsvariabelen

In dit hoofdstuk bekijken we de definitie van een TOEVALSVERANDERLIJKE; dit is eigenlijk niets anders dan een afbeelding van Ω naar $\mathbb{R}$. Wanneer er kansen gehecht zijn aan de verschillende elementen van Ω, dan is het eenvoudig om ook te spreken over de kans dat de toevalsveranderlijke X een waarde aanneemt die hoogstens x_i is; dit wordt weergegeven door de KANSVERDELINGSFUNCTIE. Voor DISCRETE TOEVALSVERANDERLIJKEN kan men m.b.v. de KANSFUNCTIE ook spreken over de kans dat X exact de waarde x_i aanneemt. Voor CONTINUE TOEVALSVERANDERLIJKEN kan men enkel spreken over de KANSDICHTHEID, die aangeeft welke regio's meer waarschijnlijk zijn dan andere. We geven ook definities voor de belangrijkste centraliteits- en spreidingsmaten zoals de VERWACHTINGSWAARDE, VARIANTIE en STANDAARDAFWIJKING samen met enkele van hun eigenschappen.

2.1 Inleiding

Het kan gebeuren dat we bij toevalsverschijnselen niet zozeer in de kansen van de verschillende uitkomsten geïnteresseerd zijn, maar aan de verschillende gebeurtenissen getalwaarden hechten.

Voorbeeld 2.1 Bij een kansspel trekt een speler een kaart uit een spel kaarten. Er wordt afgesproken dat hij 1 EUR krijgt bij het trekken van een boer, 2 EUR bij het trekken van een dame en 3 EUR bij het trekken van een heer. In alle andere gevallen krijgt hij niets.

We kunnen dit expliciet maken door een afbeelding X van Ω naar $\mathbb{R}$ te de-

finiëren:

$$X\colon \Omega \to \mathbb{R} : \omega \mapsto X(\omega) = \begin{cases} 1 & \text{als } \omega \in \{\diamond 11, \clubsuit 11, \heartsuit 11, \spadesuit 11\} \\ 2 & \text{als } \omega \in \{\diamond 12, \clubsuit 12, \heartsuit 12, \spadesuit 12\} \\ 3 & \text{als } \omega \in \{\diamond 13, \clubsuit 13, \heartsuit 13, \spadesuit 13\} \\ 0 & \text{anders.} \end{cases}$$

Voorbeeld 2.2 Wanneer men een muntstuk tweemaal opgooit dan is

$$\Omega = \{\text{KK}, \text{KM}, \text{MK}, \text{MM}\}.$$

Als $X(\omega)$ het aantal keer kop is voor $\omega \in \Omega$, dan hebben we:

$$X(\text{KK}) = 2, \quad X(\text{KM}) = 1, \quad X(\text{MK}) = 1 \quad \text{en} \quad X(\text{MM}) = 0.$$

Voorbeeld 2.3 Als men de wijzer van een rad-van-fortuin vanuit een bepaalde beginpositie (bijvoorbeeld Noord) een draai geeft, dan zal deze bij stilstand een willekeurige hoek aangeven[1], dus $\Omega = [0, 2\pi[$. We kunnen dan een functie H beschouwen die met elke ω de hoek zelf associeert, dus

$$H\colon \Omega \to \mathbb{R}\colon \omega \mapsto H(\omega) = \omega.$$

We kunnen eveneens een tweede functie K beschouwen die met elke ω het *kwadrant* associeert waartoe ω behoort, dus

$$K\colon \Omega \to \mathbb{R}\colon \omega \mapsto K(\omega) = \begin{cases} 1 & \text{als } \omega \in [0, \pi/2[\\ 2 & \text{als } \omega \in [\pi/2, \pi[\\ 3 & \text{als } \omega \in [\pi, 3\pi/2[\\ 4 & \text{als } \omega \in [3\pi/2, 2\pi[. \end{cases}$$

In de voorgaande voorbeelden werden enkele kansvariabelen (zoals H en K) gegeven. We geven nu een definitie.

Definitie 2.4 Een KANSVARIABELE X is een afbeelding van Ω naar $\mathbb{R}$. Deze afbeelding associeert met elke mogelijke uitkomst van een kansexperiment dus een reëel getal[2].

[1] We gebruiken hier *radialen* om de hoek voor te stellen.

[2] In een meer theoretische benadering zijn er nog (technische) restricties voor deze afbeelding, maar dan gaan we hier nu niet op in.

Men maakt onderscheid tussen discrete en continue kansvariabelen. De hiermee geassocieerde kansmodellen vormen de basis van de inferentiële statistiek[3].

2.2 Discrete kansvariabelen

Definitie 2.5 Een kansvariabele X is DISCREET wanneer X slechts een eindig of aftelbaar oneindig aantal waarden aanneemt. Dit wil zeggen dat

$$\text{bld}(X) = \{x_1, x_2, \ldots, x_n\} \quad \text{of} \quad \text{bld}(X) = \{x_1, x_2, \ldots\}.$$ ∎

Voorbeeld 2.6 De kansvariabele K uit Voorbeeld 2.3 neemt als waarden 1, 2, 3 en 4 aan, of

$$\text{bld}(K) = \{1, 2, 3, 4\}.$$

Deze kansvariabele is dus discreet. ∎

Als men nu een kansruimte heeft met universum Ω, waarbij de waarschijnlijkheid van een gebeurtenis A gegeven wordt door $\mathbb{P}(A)$, dan kunnen we met een discrete kansvariabele X op een natuurlijke manier een kansmodel associëren.

Men zegt

$$f_X(x_i) = \mathbb{P}(X = x_i) = \mathbb{P}\big(A(x_i)\big),$$

waarbij $A(x_i)$ de gebeurtenis voorstelt dat X op x_i wordt afgebeeld, i.e.

$$A(x_i) = \{\omega \in \Omega \mid X(\omega) = x_i\}.$$

Men neemt m.a.w. alle elementen van Ω die door X op de waarde x_i worden afgebeeld en van deze verzameling (gebeurtenis) bepaalt men de kans. Deze kans is de definitie van $\mathbb{P}(X = x_i)$.

[3] Zie cursus onderzoekstechnieken.

Voorbeeld 2.7 Voor de kansvariabele in Voorbeeld 2.1 geldt, gebruikmakend van de regel van Laplace:

$$f_X(3) = \mathbb{P}(X = 3) = \mathbb{P}(\{\diamond 13, \clubsuit 13, \heartsuit 13, \spadesuit 13\}) = \frac{4}{52} = \frac{1}{13},$$
$$f_X(2) = \mathbb{P}(X = 2) = \mathbb{P}(\{\diamond 12, \clubsuit 12, \heartsuit 12, \spadesuit 12\}) = \frac{4}{52} = \frac{1}{13},$$
$$f_X(1) = \mathbb{P}(X = 1) = \mathbb{P}(\{\diamond 11, \clubsuit 11, \heartsuit 11, \spadesuit 11\}) = \frac{4}{52} = \frac{1}{13},$$
$$f_X(0) = \mathbb{P}(X = 0) = \mathbb{P}(\{\diamond 1, \diamond 2, \ldots, \spadesuit 9, \spadesuit 10\}) = \frac{40}{52} = \frac{10}{13}.$$ ∎

Definitie 2.8 De functie f_X

$$f_X : \mathbb{R} \to [0,1] : x \mapsto f_X(x) = \mathbb{P}(X = x)$$

noemt men de KANSFUNCTIE van de discrete toevalsveranderlijke X. ∎

Opmerking 2.9 Wanneer de kansvariabele X duidelijk is uit de context (of irrelevant is), dan schrijven we kortweg f i.p.v. f_X. ∎

Eigenschap 2.10 Een kansfunctie f voor een discrete toevalsveranderlijke voldoet aan de volgende eigenschappen:

1. Een kansfunctie is steeds positief: $0 \leq f(x)$.
2. Een kansfunctie is hoogstens één: $f(x) \leq 1$.
3. De som van de waarden $f(x)$ voor alle mogelijke uitkomsten x is één:

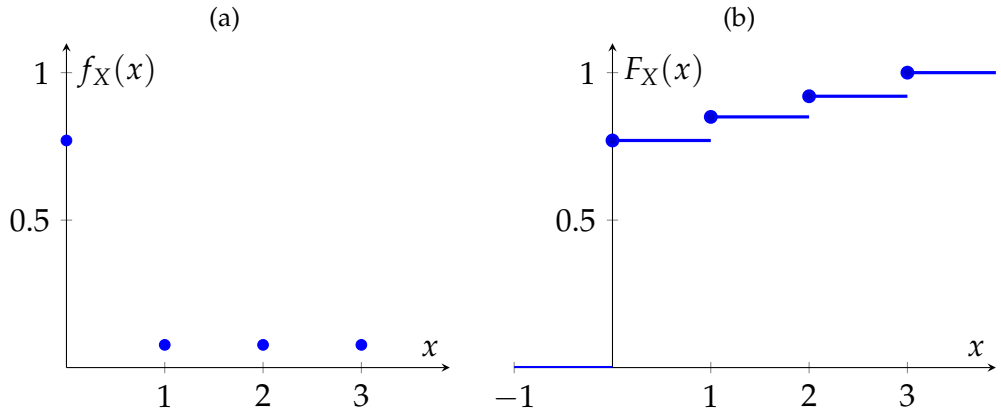
$$\sum_{x \in \text{bld}(X)} f(x) = 1.$$ ∎

Bewijs De eerste twee eigenschappen volgen onmiddellijk uit het feit dat $f(x)$ een *kans* voorstelt.

We bewijzen nu de derde eigenschap. Laat $A(x_i)$ opnieuw de gebeurtenis voorstellen dat X op x_i wordt afgebeeld:

$$A(x_i) = \{\omega \in \Omega \mid X(\omega) = x_i\}.$$

Dan is het duidelijk dat $A(x_i)$ en $A(x_j)$ disjuncte gebeurtenissen zijn als x_i en x_j verschillend zijn (want dezelfde ω kan niet tegelijkertijd op x_i en x_j



Figuur 2.1: Links zie je de grafiek van $f_X(x)$ horend bij de kansvariabele X uit Voorbeeld 2.1. Rechts zie je de kansverdelingsfunctie $F_X(x)$ voor dezelfde toevalsveranderlijke.

worden afgebeeld). Bovendien wordt (bij definitie) *elke* ω op *een* x_i afgebeeld. Anders gezegd:

$$\Omega = \bigcup_{x \in \text{bld}(X)} A(x).$$

Uit de somregel volgt dan dat

$$1 = \sum_{x \in \text{bld}(X)} \mathbb{P}\big(A(x)\big) = \sum_{x \in \text{bld}(X)} f(x).$$

♢

De kansfunctie f kan uiteraard ook steeds grafisch voorgesteld worden.

Voorbeeld 2.11 Bij het kaartspelletje van Voorbeeld 2.1 vindt men

x_i	0	1	2	3
$f(x_i)$	10/13	1/13	1/13	1/13

De bijhorende grafiek ziet men in Figuur 2.1. Men gaat gemakkelijk na dat

$$f(0) + f(1) + f(2) + f(3) = 1.$$

∎

Definitie 2.12 Men definieert ook nog een *cumulatieve* kansfunctie, de KANSVERDELINGSFUNCTIE. Deze wordt gedefinieerd als

$$F_X : \mathbb{R} \to [0,1] : x \mapsto F_X(x) = \mathbb{P}(X \leq x) = \mathbb{P}(\{\omega \in \Omega \mid X(\omega) \leq x\}).$$

∎

Voorbeeld 2.13 We berekenen enkele waarden en bepalen de grafiek van de kansverdelingsfunctie F_X voor de kansvariabele X uit Voorbeeld 2.1.

x	-2	0	$1/4$	$1/2$	1	$\sqrt{2}$	2	3	3.1
$F_X(x)$	0	$10/13$	$10/13$	$10/13$	$11/13$	$11/13$	$12/13$	1	1

Wanneer men F_X volledig specificeert dan krijgt men

$$F_X(x) = \begin{cases} 0 & \text{als } x < 0 \\ 10/13 & \text{als } 0 \leq x < 1 \\ 11/13 & \text{als } 1 \leq x < 2 \\ 12/13 & \text{als } 2 \leq x < 3 \\ 1 & \text{als } x \geq 3. \end{cases}$$

De bijhorende grafiek ziet men in Figuur 2.1. Merk op dat deze grafiek stuksgewijs constant is, i.e. de functiewaarde blijft steeds constant in een bepaald interval. Er zijn discontinuïteiten (sprongen) in de punten x_i die behoren tot het beeld van X. ∎

2.2.1 Oefeningen

1. Bij een dobbelspel met één dobbelsteen krijgt men het drievoud uitbetaald van het aantal ogen. Dit geeft aanleiding tot een kansvariabele X. Geef de uitkomstenverzameling Ω, f_X en F_X. Geef de grafiek van zowel f_X als F_X. Men spreekt van de DISCRETE UNIFORME VERDELING.

2. Men gooit een evenwichtig muntstuk op tot voor de eerste keer "munt" verschijnt. We noemen X het daartoe benodigde aantal worpen. Wat is het beeld van X? Geef $f_X(x)$ en $F_X(x)$. Schets hun grafiek. Hoe groot is de kans dat het aantal worpen minstens 3 en ten hoogste 6 bedraagt?

3. Bij een dobbelspel met twee eerlijke dobbelstenen noteert men X als de som van het aantal ogen. Onderzoek f_X en F_X.

4. Veronderstel dat het munstuk in Voorbeeld 2.2 zodanig vervalst is dat $\mathbb{P}(K) = 6/10$. Bepaal f_X en F_X.

2.3 Continue kansvariabelen

Als we terugkijken naar de kansvariabele H uit Voorbeeld 2.3, dan kan men intuïtief aanvoelen dat de kans dat de wijzer *exact* stopt op een hoek $1/2$ (radiaal) gelijk is aan nul. Er zijn immers (overaftelbaar) oneindig veel mogelijkheden waar de wijzer kan stoppen, en elk heeft evenveel "kans". Het is echter wel zinvol om na te gaan wat de kans is dat de wijzer stopt *tussen* twee hoeken a en b. De fundamentele grootheid bij continue kansvariabelen is dan ook de kansverdelingsfunctie.

Definitie 2.14 De KANSVERDELINGSFUNCTIE F_X van een toevalsveranderlijke X is de functie van $\mathbb{R}$ naar het het interval $[0,1]$ gegeven door:

$$F_X\colon \mathbb{R} \to [0,1]\colon x \mapsto F_X(x) = \mathbb{P}(X \leq x) = \mathbb{P}(\{\omega \in \Omega \mid X(\omega) \leq x\}).$$ ∎

Definitie 2.15 Een toevalsveranderlijke X is CONTINU als er een functie f_X van $\mathbb{R}$ naar $\mathbb{R}^+$ bestaat zodanig dat

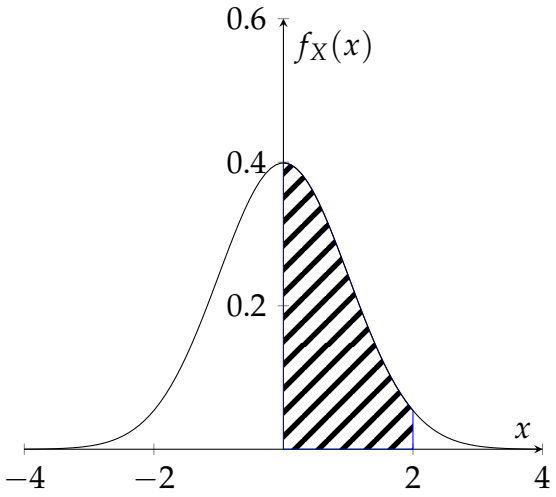
$$F_X(x) = \int_{-\infty}^{x} f_X(y)\,\mathrm{d}y.$$

De functie f_X wordt de KANSDICHTHEID genoemd. ∎

Een kansmodel voor X werd dus bekomen door aan een uitkomst A (een interval of unie van intervallen) een kans toe te kennen die gelijk is aan de oppervlakte boven A en onder de grafiek van de kromme f_X. Figuur 2.2 geeft een illustratie. We maken dit expliciet m.b.v. de volgende eigenschap.

Eigenschap 2.16 De functie f_X voldoet aan de volgende eigenschappen.

1. De functie f_X is nergens negatief, i.e. $f_X(x) \geq 0$.
2. Als men f_X integreert over $\mathbb{R}$ dan bekomt men als uitkomst 1:
$$\int_{-\infty}^{+\infty} f_X(x)\,\mathrm{d}x = 1.$$
3. De kans dat X behoort tot een interval $[a,b]$ wordt uitgedrukt d.m.v. de volgende integraal:
$$\mathbb{P}(a \leq X \leq b) = \int_a^b f_X(x)\,\mathrm{d}x.$$ ∎



Figuur 2.2: Als de figuur de grafiek voorstelt van de kansdichtheid $f_X(x)$, dan geeft de integraal $\int_0^2 f_X(x)\,\mathrm{d}x$ de oppervlakte aan van het gearceerde gedeelte van de figuur.

Bewijs

1. Dit volgt onmiddellijk uit de definitie van f_X.
2. We gebruiken Definitie 2.15 en we vinden

$$\int_{-\infty}^{+\infty} f_X(x)\,\mathrm{d}x = F_X(+\infty) = \mathbb{P}(X \leq +\infty) = 1.$$

   De laatste ongelijkheid geldt omdat voor elke $\omega \in \Omega$ geldt dat $X(\omega) \leq +\infty$.
3. M.b.v. eigenschappen van integralen vinden we:

$$\begin{aligned}\mathbb{P}(a \leq X \leq b) &= F_X(b) - F_X(a)\\ &= \int_{-\infty}^{b} f_X(x)\,\mathrm{d}x - \int_{-\infty}^{a} f_X(x)\,\mathrm{d}x\\ &= \int_{-\infty}^{b} f_X(x)\,\mathrm{d}x + \int_{a}^{-\infty} f_X(x)\,\mathrm{d}x\\ &= \int_{a}^{b} f_X(x)\,\mathrm{d}x.\end{aligned}$$

♢

Opmerking 2.17 Uit puntje (3) van Eigenschap 2.16 volgt dat de kans dat X gelegen is tussen a en b is gelijk aan het gedeelte van de oppervlakte boven $[a, b]$ en onder f_X. Dit volgt uit de meetkundige interpretatie van integralen voor niet-negatieve functies. Verder zien we ook uit puntje (2) dat de oppervlakte onder de kromme is gelijk aan 1. ∎

Opmerking 2.18 De eigenschap vermeldt *niet* dat $f_X(x)$ steeds hoogstens 1 is. Dit is in het algemeen ook niet geldig; er bestaan continue toevalsveranderlijk waarvoor $f_X(x) > 1$ voor bepaalde reële getallen x. Het is dan ook *fout* om $f_X(a)$ te zien als de kans dat de toevalsveranderlijke X de waarde a aanneemt. Deze waarschijnlijkheid is immers steeds nul:

$$\mathbb{P}(X = a) = \mathbb{P}(a \leq X \leq a) = \int_a^a f_X(x)\,\mathrm{d}x = 0.$$

Dit betekent ook dat voor een continue toevalsveranderlijke X steeds geldt dat

$$\mathbb{P}(a \leq X \leq b) = \mathbb{P}(a \leq X < b) = \mathbb{P}(a < X \leq b) = \mathbb{P}(a < X < b).$$

Deze laatste eigenschap is *niet* geldig voor discrete toevalsveranderlijken. ∎

Opmerking 2.19 Uit Definitie 2.15 volgt dat F_X de primitieve functie van is f_X. ∎

Voorbeeld 2.20 Voor de kansvariabele H uit Voorbeeld 2.3 geldt door de inherente symmetrie van het experiment dat de kans dat de hoek ω tot het interval $[a, b]$ (met $a < b$) behoort recht evenredig is met de lengte van dit interval, i.e.

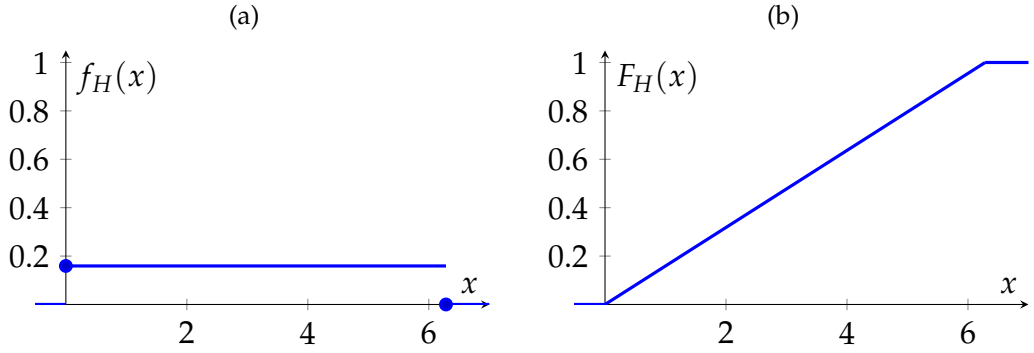
$$\mathbb{P}([a, b]) = \frac{b - a}{2\pi},$$

waarbij de constante $1/2\pi$ zo gekozen is dat $\mathbb{P}(\Omega) = 1$. We leiden hieruit het functievoorschrift van $F_H(x)$ af. Veronderstel dat $0 \leq x < 2\pi$. We hebben dan

$$\begin{aligned} F_H(x) &= \mathbb{P}(H \leq x) \\ &= \mathbb{P}(0 \leq H \leq x) \\ &= \mathbb{P}(\{\omega \in \Omega \mid 0 \leq H(\omega) \leq x\}) \\ &= \mathbb{P}(\{\omega \in \Omega \mid 0 \leq \omega \leq x\}) \\ &= \mathbb{P}([0, x]) \\ &= \frac{x}{2\pi}. \end{aligned}$$

Samengevat vinden we

$$F_H(x) = \begin{cases} 0 & \text{als } x < 0 \\ \dfrac{x}{2\pi} & \text{als } 0 \leq x < 2\pi \\ 1 & \text{als } x \geq 2\pi. \end{cases}$$



Figuur 2.3: Grafiek van de kansdichtheid en de kansverdelingsfunctie van de toevalsveranderlijke H uit Voorbeeld 2.3.

Hieruit volgt dan dat

$$f_H(x) = \begin{cases} \frac{1}{2\pi} & \text{als } 0 \leq x < 2\pi \\ 0 & \text{anders.} \end{cases}$$

Figuur 2.3 geeft de grafiek van de kansdichtheid en de kansverdelingsfunctie van de toevalsveranderlijke H. ∎

Eigenschap 2.21 Zowel voor discrete als voor continue kansvariabelen geldt:

1. F is gedefinieerd in geheel $\mathbb{R}$, i.e. $\text{dom}(F) = \mathbb{R}$.
2. F is niet dalend, i.e.
$$x < y \implies F(x) \leq F(y).$$
3. De X-as is een horizontale asymptoot op min oneindig, i.e.
$$\lim_{x \to -\infty} F(x) = 0.$$
4. De rechte met vergelijking $y = 1$ is een horizontale asymptoot op plus oneindig, i.e.
$$\lim_{x \to +\infty} F(x) = 1.$$

∎

Bewijs Het bewijs is een eenvoudige oefening. ♢

2.4 Verwachtingswaarde en variantie

Een kansverdeling van een toevalsvariabele is een geïdealiseerde verdeling van relatieve frequenties op lange termijn. De grafieken van de kansfuncties en kansverdelingsfuncties doen natuurlijk denken aan de staafdiagrammen en histogrammen die men gebruikt voor het voorstellen van data[4]. Het ligt nu voor de hand ook voor de toevalsvariabelen numerieke maatgetallen als gemiddelde en standaardafwijking te definiëren.

De manier waarop gemiddelde en standaardafwijking gedefinieerd worden verschilt naargelang de kansvariabele discreet dan wel continu is.

2.4.1 Discrete kansvariabele

Veronderstel dat een toevalsveranderlijke X geassocieerd werd met de uitkomstensruimte Ω van een bepaald experiment, i.e. met elke $\omega \in \Omega$ werd een reëel getal $X(\omega)$ geassocieerd. Men voert nu een groot aantal, N, onafhankelijke herhalingen van het experiment uit en men bekomt dus een reeks van reële getallen $X(\omega_i)$ waarbij ω_i de uitkomst van het i-de experiment voorstelt. We kunnen nu het (rekenkundig) gemiddelde van deze reële getallen berekenen:

$$\frac{1}{N}\big(X(\omega_1) + X(\omega_2) + \cdots + X(\omega_N)\big).$$

Wanneer de toevalsveranderlijke X discreet is en slechts n mogelijke waarden x_1 t.e.m. x_n aanneemt, dan kunnen we dit rekenkundig gemiddelde ook berekenen door alle uitkomsten x_i te gaan groeperen. We doen dit als volgt, waarbij $\#x_i$ het aantal keer voorstelt dat de waarde x_i voorkomt in de rij van N getallen:

$$\begin{aligned}\frac{1}{N}\big(x_1 \cdot \#x_1 + x_2 \cdot \#x_2 + \cdots + x_n \cdot \#x_n\big)\\ = x_1 \cdot \frac{\#x_1}{N} + x_2 \cdot \frac{\#x_2}{N} + \cdots + x_n \cdot \frac{\#x_n}{N}.\end{aligned}$$

Wanneer men de limiet neemt waarbij N oneindig groot wordt, dan zullen de verhoudingen $\#x_i/N$ naderen naar $\mathbb{P}(X = x_i)$. Hieruit leidt men dan de definitie voor de verwachtingswaarde van een discrete toevalsveranderlijke af.

[4] Zie cursus Onderzoekstechnieken.

Definitie 2.22 De VERWACHTINGSWAARDE van een discrete toevalsveranderlijke X wordt genoteerd als μ_X of $\mathrm{E}(X)$, en is een gewogen gemiddelde van de waarden x_i die X kan aannemen met de respectievelijke kansen als gewichten. In formulevorm:

$$\mathrm{E}(X) = \sum_i x_i \, \mathbb{P}(X = x_i) = \sum_i x_i f_X(x_i). \qquad \blacksquare$$

Voorbeeld 2.23 We berekenen de verwachte opbrengst bij het kaartspel uit Voorbeeld 2.1:

$$\begin{aligned} \mu_X &= \sum_i x_i f_X(x_i) \\ &= 0 \cdot f_X(0) + 1 \cdot f_X(1) + 2 \cdot f_X(2) + 3 \cdot f_X(3) \\ &= 0 \cdot \frac{10}{13} + 1 \cdot \frac{1}{13} + 2 \cdot \frac{1}{13} + 3 \cdot \frac{1}{13} = \frac{6}{13}. \end{aligned}$$

Meestal wordt bij dergelijke spelen een inzet gevraagd. Per definitie noemen we een spel EERLIJK als de verwachte opbrengst gelijk is aan die inzet. Wat zou je hier een acceptabele inzet vinden? $\blacksquare$

De verwachtingswaarde geeft aan rond welke waarde de kansvariabele ligt; het is een CENTRALITEITSMAAT. Vaak is het echter ook interessant om te weten wat de spreiding is t.o.v. dit gemiddelde, i.e. we zouden graag m.b.v. een getal uitdrukken hoe "gespreid" de grafiek van de kansfunctie van de kansvariabele X is.

Definitie 2.24 De VARIANTIE van een discrete toevalsveranderlijke X, genoteerd σ_X^2, is de gewogen gemiddelde kwadratische afwijking t.o.v. zijn verwachtingswaarde. In symbolen:

$$\sigma_X^2 = \sum_i (x_i - \mu_X)^2 \, \mathbb{P}(X = x_i) = \sum_i (x_i - \mu_X)^2 f_X(x_i). \qquad \blacksquare$$

Opmerking 2.25 De *eenheid* van variantie is het kwadraat van de eenheid van de oorspronkelijke toevalsveranderlijke (en de verwachtingswaarde). Vaak is het *eenvoudiger* om te werken met een spreidingsmaat waarvan de eenheid dezelfde is als die van de verwachtingswaarde. Zo'n spreidingsmaat kunnen we eenvoudig construeren door de vierkantswortel te nemen van de variantie, en wordt de STANDAARDAFWIJKING genoemd. De standaardafwijking wordt genoteerd als σ_X. In formulevorm vinden we

$$\sigma_X = \sqrt{\sigma_X^2}. \qquad \blacksquare$$

Voorbeeld 2.26 We berekenen de variantie σ_X^2 en de standaardafwijking σ_X voor de toevalsveranderlijke X uit Voorbeeld 2.1. We weten reeds dat $\mu_X = 6/13$, en we vinden dus

$$\begin{aligned}\sigma_X^2 &= \sum_i (x_i - \mu_X)^2 f_X(x_i) \\ &= (0 - \frac{6}{13})^2 \cdot f_X(0) + (1 - \frac{6}{13})^2 \cdot f_X(1) \\ &\quad + (2 - \frac{6}{13})^2 \cdot f_X(2) + (3 - \frac{6}{13})^2 \cdot f_X(3) \\ &= \frac{36}{169} \cdot \frac{10}{13} + \frac{49}{169} \cdot \frac{1}{13} + \frac{400}{169} \cdot \frac{1}{13} + \frac{1089}{169} \cdot \frac{1}{13} \\ &= \frac{146}{169}.\end{aligned}$$

Voor de standaardafwijking vinden we dus:

$$\sigma_X = \sqrt{\sigma_X^2} = \sqrt{\frac{146}{169}} \approx 0.9295\,.$$

∎

Voorbeeld 2.27 (Intuïtie variantie) Beschouw twee toevalsveranderlijken X en Y. De toevalsveranderlijke X stelt het aantal ontvangen emails per dag voor van een eerste gebruiker. Deze gebruiker ontvangt elke dag ofwel 48 ofwel 52 emails, elk met 50% kans. De toevalsveranderlijke Y stelt het aantal ontvangen emails voor van een tweede gebruiker. Deze tweede gebruiker ontvangt met 50% kans geen enkele email, en met 50% kans 100 emails.

Men vindt gemakkelijk dat de kansfuncties van de toevalsveranderlijken X en Y gegeven worden door:

x_i	48	52
$f_X(x_i)$	1/2	1/2

en

y_i	0	100
$f_Y(y_i)$	1/2	1/2

De verwachtingswaarde van beide toevalsveranderlijken is 50; elke gebruiker ontvangt gemiddeld 50 emails per dag. Toch lijkt het intuïtief duidelijk dat de tweede gebruiker een grotere *variabiliteit* ervaart in het aantal emails dat hij ontvangt in zijn inbox.

We berekenen de varianties van de toevalsveranderlijken:

$$\sigma_X^2 = \frac{1}{2}(48-50)^2 + \frac{1}{2}(52-50)^2 = \frac{4}{2} + \frac{4}{2} = 4,$$

en

$$\sigma_Y^2 = \frac{1}{2}(0-50)^2 + \frac{1}{2}(100-50)^2 = \frac{50^2}{2} + \frac{50^2}{2} = 2500.$$

Men ziet dus ook dat σ_Y^2 veel groter is dan σ_X^2. ∎

2.4.2 Continue kansvariabele

Op analoge manier als bij de discrete kansvariabelen heeft men de volgende definities.

Definitie 2.28 De verwachtingswaarde $\mathrm{E}(X)$ (of μ_X) van een continue toevalsveranderlijke X wordt gedefinieerd als:

$$\mu_X = \int_{-\infty}^{+\infty} x f_X(x)\,\mathrm{d}x,$$

terwijl de variantie σ_X^2 wordt gegeven door

$$\sigma_X^2 = \int_{-\infty}^{+\infty} (x-\mu_X)^2 f_X(x)\,\mathrm{d}x.$$ ∎

Opmerking 2.29 Men bekomt de formules voor continue kansvariabelen door in de overeenkomstige formule voor discrete kansvariabelen het sommatieteken te vervangen door een integratiesymbool. ∎

Opmerking 2.30 De standaardafwijking is nog steeds de vierkantswortel uit de variantie. ∎

Voorbeeld 2.31 We berekenen μ_H voor de kansvariabele H uit Voorbeeld 2.3. We gebruiken het functievoorschrift voor de kansdichtheid f_H uit Voorbeeld 2.20. We vinden:

$$\begin{aligned}
\mu_H &= \int_{-\infty}^{+\infty} x f_H(x)\,\mathrm{d}x \\
&= \int_{-\infty}^{0} x f_H(x)\,\mathrm{d}x + \int_{0}^{2\pi} x f_H(x)\,\mathrm{d}x + \int_{2\pi}^{+\infty} x f_H(x)\,\mathrm{d}x \\
&= \int_{-\infty}^{0} x\cdot 0\,\mathrm{d}x + \int_{0}^{2\pi} x\frac{1}{2\pi}\,\mathrm{d}x + \int_{2\pi}^{+\infty} x\cdot 0\,\mathrm{d}x \\
&= \frac{1}{2\pi}\int_{0}^{2\pi} x\,\mathrm{d}x \\
&= \pi.
\end{aligned}$$

Voor het berekenen van de laatste integraal moeten we de oppervlakte berekenen van een driehoek met basis 2π en hoogte 2π. Deze oppervlakte is $2\pi^2$, waaruit het gestelde volgt. Merk op dat de berekende verwachtingswaarde volkomen in overeenstemming is met onze intuïtie betreffende de betekenis van gemiddelde waarde of verwachtingswaarde.

Voor de variantie vinden we:

$$\begin{aligned}\sigma_X^2 &= \int_{-\infty}^{+\infty} (x-\mu_X)^2 f_H(x)\,\mathrm{d}x \\ &= \frac{1}{2\pi}\int_0^{2\pi} (x-\pi)^2\,\mathrm{d}x \\ &= \frac{1}{2\pi}\left[\frac{(x-\pi)^3}{3}\right]_0^{2\pi} \\ &= \frac{1}{2\pi}\left(\frac{\pi^3}{3} - \frac{-\pi^3}{3}\right) \\ &= \frac{\pi^2}{3}.\end{aligned}$$

Hierbij werd gebruikgemaakt van het feit dat

$$\int x^n\,\mathrm{d}x = \frac{x^{n+1}}{n+1} + C, \quad \text{als} \quad n \neq -1.$$

De standaardafwijking is dus:

$$\sigma_X = \frac{\pi}{\sqrt{3}}.$$

∎

2.4.3 Eigenschappen van verwachtingswaarde en variantie

Alvorens over te gaan op de eigenschappen van verwachtingswaarde en variantie bekijken we het concept van een *functie van een toevalsveranderlijke*.

Veronderstel dat X een toevalsveranderlijke is, i.e. $X\colon \Omega \to \mathbb{R}$ en dat g een reële functie is, i.e. $g\colon \mathbb{R} \to \mathbb{R}$, dan is $g \circ X$ ook een functie van Ω naar $\mathbb{R}$ en dus ook een toevalsveranderlijke. Meer specifiek geldt:

$$g \circ X\colon \Omega \to \mathbb{R}\colon \omega \mapsto g(X(\omega)).$$

We kunnen voor deze nieuwe toevalsveranderlijke $Y = g(X)$ de verwachtingswaarde $\mathrm{E}(Y)$ berekenen *zonder* dat we expliciet de kansfunctie f_Y moeten opstellen!

Eigenschap 2.32 (Law of the unconscious statistician) Als X een discrete toevalsveranderlijke is en g is een functie van $\mathbb{R}$ naar $\mathbb{R}$ dan geldt[5]:

$$\mathrm{E}(Y) = \mathrm{E}(g(X)) = \sum_{x \in \mathrm{bld}(X)} g(x) f_X(x) = \sum_{x \in \mathrm{bld}(X)} g(x)\, \mathbb{P}(X = x).$$

Merk op dat in deze formule de kansfunctie van X wordt gebruikt en *niet* die van Y. ∎

Bewijs Zonder bewijs. ♢

Opmerking 2.33 Deze eigenschap is ook geldig voor continue toevalsveranderlijken, al moet men dan natuurlijk met de kansdichtheid werken. ∎

Voorbeeld 2.34 (Illustratie van Eigenschap 2.32) Veronderstel dat X de waarden -2, -1, 1 en 3 aanneemt met waarschijnlijkheid $\frac{1}{4}$, $\frac{1}{8}$, $\frac{1}{4}$ en $\frac{3}{8}$ respectievelijk. Veronderstel dat g de functie is die x afbeeldt op x^2.

We berekenen eerst $\mathrm{E}(Y)$ aan de hand van f_Y. We gaan gemakkelijk na dat $Y = g(X) = X^2$ de waarden 1, 4 en 9 aanneemt met waarschijnlijkheid $\frac{1}{8} + \frac{1}{4} = \frac{3}{8}$, $\frac{1}{4}$ en $\frac{3}{8}$ respectievelijk. We kunnen dan $E(Y)$ berekenen met de definitie van waarschijnlijkheid:

$$\mathrm{E}(Y) = \sum_{y \in \mathrm{bld}(Y)} y\, \mathbb{P}(Y = y) = 1 \times \frac{3}{8} + 4 \times \frac{1}{4} + 9 \times \frac{3}{8} = \frac{19}{4}.$$

We kunnen dit echter ook berekenen m.b.v. Eigenschap 2.32:

$$\begin{aligned} \mathrm{E}(Y) = \mathrm{E}(X^2) &= \sum_{x \in \mathrm{bld}(X)} x^2\, \mathbb{P}(X = x) \\ &= (-2)^2 \times \frac{1}{4} + (-1)^2 \times \frac{1}{8} + (1)^2 \times \frac{1}{4} + (3)^2 \times \frac{3}{8} = \frac{19}{4}. \end{aligned}$$ ∎

Eigenschap 2.35 De verwachtingswaarde en variantie van een (discrete of continue) toevalsveranderlijke X voldoen aan de volgende eigenschappen:

1. Als X constant is, i.e. $X(\omega) = k$ voor alle elementen ω van Ω, dan is $\mathrm{E}(X) = k$.

[5] We laten opnieuw enkel technische beperkingen buiten beschouwing.

2. Als $a \in \mathbb{R}$ een constante is, dan geldt:

$$\mathrm{E}(X + a) = \mathrm{E}(X) + a,$$

waaruit in het bijzonder volgt dat

$$\mathrm{E}(X - \mu_X) = 0.$$

3. Als $a \in \mathbb{R}$ een constante is, dan geldt:

$$\mathrm{E}(aX) = a\,\mathrm{E}(X).$$

4. Er geldt steeds dat

$$\sigma_X^2 = \mathrm{E}(X^2) - \mu_X^2.$$

Deze formule geeft de mogelijkheid om de variantie efficiënter te berekenen dan rechtstreeks via de definitie.

5. De variantie wijzigt niet als we de toevalsveranderlijke "verschuiven". Er geldt voor elke constante $a \in \mathbb{R}$ dat

$$\sigma_{(X+a)}^2 = \sigma_X^2.$$

6. Vermenigvuldigen met een constante wijzigt de variantie op een kwadratische manier:

$$\sigma_{aX}^2 = a^2\sigma_X^2.$$ ∎

Bewijs We laten het bewijs als oefening. ♢

Voorbeeld 2.36 We berekenen de variantie van de kansvariabele uit Voorbeeld 2.1 op een andere manier. We starten met de berekening van $\mathrm{E}(X^2)$ gebruikmakend van Eigenschap 2.32:

$$\mathrm{E}(X^2) = \sum_i x_i^2 f_X(x_i) = 0^2 \times \frac{10}{13} + 1^2 \times \frac{1}{13} + 2^2 \times \frac{1}{13} + 3^2 \times \frac{1}{13} = \frac{14}{13}.$$

Dus wordt de variantie volgens Eigenschap 2.35 gegeven door

$$\sigma_X^2 = \mathrm{E}(X^2) - \mathrm{E}(X)^2 = \frac{14}{13} - \frac{6^2}{13^2} = \frac{146}{169}.$$

Dit is in overeenstemming met hetgeen in Voorbeeld 2.26 werd bekomen. ∎

Voorbeeld 2.37 Veronderstel dat we in Voorbeeld 2.1 de opbrengst van het kaartspel *verdubbelen*, i.e. we creëren een toevalsveranderlijke $Y = 2X$. Expliciet wordt Y dus gegeven door:

$$Y\colon \Omega \to \mathbb{R} : \omega \mapsto Y(\omega) = \begin{cases} 2 & \text{als } \omega \in \{\diamond 11, \clubsuit 11, \heartsuit 11, \spadesuit 11\} \\ 4 & \text{als } \omega \in \{\diamond 12, \clubsuit 12, \heartsuit 12, \spadesuit 12\} \\ 6 & \text{als } \omega \in \{\diamond 13, \clubsuit 13, \heartsuit 13, \spadesuit 13\} \\ 0 & \text{anders.} \end{cases}$$

Als we nu $\mathrm{E}(Y)$ berekenen, dan vinden we:

$$\begin{aligned} \mu_Y &= \sum_i x_i f_Y(x_i) \\ &= 0 \cdot f_Y(0) + 2 \cdot f_Y(2) + 4 \cdot f_Y(4) + 6 \cdot f_Y(6) \\ &= 0 \cdot \frac{10}{13} + 2 \cdot \frac{1}{13} + 4 \cdot \frac{1}{13} + 6 \cdot \frac{1}{13} = \frac{12}{13}. \end{aligned}$$

Merk op dat dus inderdaad $\mathrm{E}(Y) = \mathrm{E}(2X) = 2\,\mathrm{E}(X)$.

We berekenen nu nog de variantie σ_Y^2 van Y:

$$\begin{aligned} \sigma_Y^2 &= \sum_i (x_i - \mu_Y)^2 f_Y(x_i) \\ &= (0 - \frac{12}{13})^2 \times f_Y(0) + (2 - \frac{12}{13})^2 \times f_Y(2) \\ &\quad + (4 - \frac{12}{13})^2 \times f_Y(4) + (6 - \frac{12}{13})^2 \times f_Y(6) \\ &= \frac{584}{169}. \end{aligned}$$

We vinden dus dat $\sigma_Y^2 = \sigma_{2X}^2 = 4\sigma_X^2$, in overeenstemming met Eigenschap 2.35. ∎

Men kan voor elke toevalsveranderlijke X steeds overgaan op een "gestandaardiseerde" toevalsveranderlijke met verwachtingswaarde nul en variantie (en standaardafwijking) één.

Gevolg 2.38 (Standaardiseren van een toevalsveranderlijke) Voor elke toevalsveranderlijke X waarvoor $\sigma_X > 0$ geldt dat de toevalsveranderlijke Z die gedefinieerd is als

$$Z = \frac{X - \mu_X}{\sigma_X}$$

gemiddelde nul en variantie één heeft, i.e.

$$\mu_Z = 0 \quad \text{en} \quad \sigma_Z^2 = 1. \qquad \blacksquare$$

Bewijs Dit volgt onmiddellijk uit de eigenschappen van verwachtingswaarde en variantie. ♢

2.4.4 Oefeningen

1. Bij een kleine verzekeringsmaatschappij heeft men ondervonden dat – gemiddeld – 1 op 10000 polissen resulteert in een schadeclaim van 200000 EUR, 1 op 1000 in een claim van 50000 EUR, 1 op 50 in een claim van 2000 EUR en de rest in een schadeclaim van 0 EUR. Men wenst nu te weten (teneinde een prijsaanpassing door te voeren) wat men gemiddeld uitbetaalt per polis.

2. Een kioskhouder verkoopt elke zaterdag een aantal exemplaren van een bepaalde krant. Het aantal exemplaren dat op een willekeurige zaterdag wordt *gevraagd* kan beschouwd worden als een kansvariabele. Nadat hij de verkoop een tijdlang heeft opgevolgd bekomt hij volgende resultaten

Aantal kranten	Kans
30	0.10
35	0.35
40	0.30
45	0.20
50	0.05

   a) De kioskhouder koopt elke zaterdag 40 kranten in. Hoe groot is de kans dat hij niet aan de vraag kan voldoen?

   b) Wat is de verwachtingswaarde van de vraag op een willekeurige zaterdag?

   c) Wat is de variantie van de vraag?

d) De kranten kosten de kioskhouder 0.60 EUR, terwijl de verkoopprijs 1.15 EUR bedraagt. Kranten die niet verkocht werden moeten worden weggegooid. Bereken de verwachte winst van de kioskhouder op een willekeurige zaterdag, als hij 40 kranten inkoopt.

3. Een fabriek heeft twee afdelingen. Tot nu toe werden reparaties uitgevoerd in twee afzonderlijke werkplaatsen. Het bedrijf overweegt nu een nieuwe gemeenschappelijke werkplaats in te richten. Het machinepark werd gedurende 100 dagen bestudeerd en men vond volgende resultaten:

#Defecten per dag	0	1	2
Afdeling A	30	50	20
Afdeling B	20	30	50

Men neemt aan dat de defecten onafhankelijk zijn. Hoeveel defecten mag men per dag verwachten in de gemeenschappelijke werkplaats? Neem (voor deze oefening) aan dat uit de tabel de *exacte* kansfunctie kan afgeleid worden voor de twee relevante toevalsveranderlijken.

4. Veronderstel dat men in een ziekenhuis bloedtesten uitvoert voor een bepaalde ziekte waaraan ongeveer 1 op 100 mensen lijden. De mensen komen naar het ziekenhuis in groepen van 50 (bijvoorbeeld scholen) en de directeur vraagt zich af of – in plaats van individueel te testen – het niet beter zou zijn de 50 personen samen te testen. Als ze allemaal negatief zijn kan hij de hele groep gezond verklaren, zoniet zou hij overgaan tot individuele testen.

   a) Wat is het verwachte aantal testen als individueel wordt gescreend?

   b) Wat is het verwachte aantal testen als in groep wordt gescreend?

   c) Wat zou jij nu beslissen als directeur?

5. Bij het roulettespel kan men een inzet doen op 37 nummers, genummerd van 0 tot 36. Er zijn 18 rode vakjes, 18 zwarte vakjes en het nummer 0 is wit. Als men een bedrag inzet op een bepaald nummer en dit nummer wint, dan krijgt men 36 keer het oorspronkelijke bedrag uitbetaald. Is dit een eerlijk spel?

HOOFDSTUK 3

Kansverdelingen

Nu bekijken we de definities en eigenschappen van enkele belangrijke discrete en continue kansverdelingen. We starten met de BERNOULLIVERDELING, het prototype van een experiment met slechts twee mogelijke uitkomsten (succes/mislukking). De BINOMIALE VERDELING telt hoeveel keer men succes had bij het uitvoeren van een *vast aantal* Bernoulli-experimenten. De GEOMETRISCHE VERDELING telt hoe vaak men moet proberen alvorens het eerste succes voorkomt wanneer een bepaald Bernoulli-experiment herhaaldelijk uitgevoerd wordt. De POISSON VERDELING kan men zien als een speciaal geval van de binomiale verdeling waarbij het aantal experimenten zeer groot is en tegelijkertijd de slaagkans van een individueel experiment zeer klein is. De EXPONENTIËLE VERDELING is een continue verdeling die de wachttijd weergeeft tot het eerste voorkomen van een bepaald verschijnsel wanneer deze verschijnsels voorkomen volgens een Poisson-proces. De UNIFORME VERDELING is dan weer de meest eenvoudige continue verdeling met een kansdichtheid die constant is binnen een bepaald interval (en nul daarbuiten).

3.1 Inleiding

Bij een gewoon dobbelspel wíl men natuurlijk geen voorspelbaarheid. In andere gevallen zou men maar wat graag over een betrouwbaar model beschikken. We denken bv. aan

- het optreden van aardbevingen of vulkaanuitbarstingen in dichtbe-

volkte gebieden;

- het maken van allerhande voorspellingen in sociologie en economie.

In sommige gevallen zal men een STATISTISCH MODEL maken voor de data, wat meestal neerkomt op het vinden van een kansverdeling (een wiskundige formule die de waarschijnlijkheden beschrijft voor het optreden van de verschillende waarden) of een meer ingewikkeld stochastisch proces (een wiskundig systeem dat het fysisch proces modelleert), of iets tussen de twee.

Om voorspellingen te doen omtrent een populatie of proces (het doel van de inferentiële statistiek[1]) kan men steunen op historische of experimentele gegevens, waarmee men dan gemiddelden, spreiding, helling van kleinste kwadraten rechte etc. berekent. Het kan gebeuren dat men niet over voldoende gegevens beschikt om een accurate voorspelling te maken, maar zelfs bij een overvloed aan gegevens zal een theoretische verdeling dikwijls een *betere beschrijving* geven van de situatie.

Beschouw bijvoorbeeld volgende empirische verdeling voor de uitkomsten bij het 60000 keer gooien van een dobbelsteen:

Ogen	h_i
1	9978
2	10021
3	10013
4	9903
5	9986
6	10099

Het is duidelijk dat de (theoretische) uniforme verdeling hier beter is dan de experimentele.

Bovendien is het werken met theoretische verdelingen ook veel *praktischer*. Het is wel essentieel dat het gemiddelde en de spreiding van de theoretische

[1] Zie cursus onderzoekstechnieken.

verdeling overeenkomen met gemiddelde en spreiding van de experimentele of historische verdeling. Daarnaast moeten beide verdelingen dezelfde vorm (scheefheid en curtosis) hebben. Veel in de praktijk voorkomende kansmodellen kunnen worden benaderd door één van de theoretische verdelingen die we hierna zullen bestuderen.

We kunnen dan niet alleen kansen berekenen aan de hand van een formule maar ook verwachtingswaarde en spreiding.

3.2 Discrete kansverdelingen

3.2.1 Bernoulliverdeling

Veel eenvoudige experimenten hebben slechts twee mogelijke uitkomsten. Men noemt ze BERNOULLI-EXPERIMENTEN [2].

Voorbeeld 3.1 De volgende experimenten zijn voorbeelden van Bernoulli-experimenten:

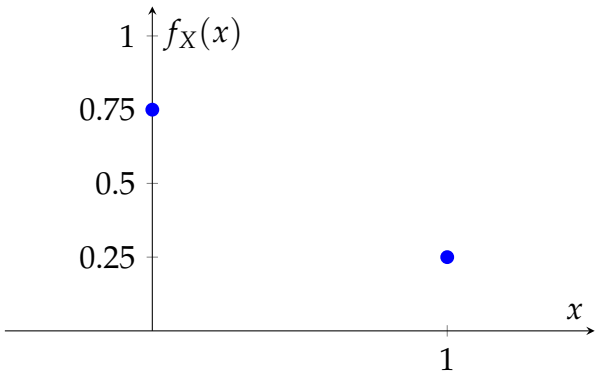
- het opgooien van een muntstuk: kop of munt;
- een dobbelsteen gooien: zes of geen zes;
- de geboorte van een kind bij een bepaald ouderpaar: meisje of jongen;
- een bal uit een vaas met witte en zwarte ballen trekken: wit of zwart. ∎

We noemen de ene uitkomst "succes" en stellen ze voor door 1, de andere noemen we "mislukking" en stellen we voor door 0. De kans op succes noteren we als p; de kans op mislukking is dan $1 - p$ en dit noteren we met q. De verdeling is volledig bepaald van zodra men de waarde van p kent.

Opmerking 3.2 Merk op dat de benamingen "succes" en "mislukking" volledig willekeurig zijn. Het kan best zijn dat voor een bepaald experiment een "succesvolle" uitkomst helemaal niet zo gunstig is. ∎

De Bernoulliverdeling heeft de volgende kansfunctie:

[2] Jacob Bernoulli, 1645-1705



Figuur 3.1: Grafiek van de kansfunctie van een toevalsveranderlijke die een Bernoulliverdeling volgt met parameter $p = 1/4$.

x_i	0	1
$\mathbb{P}(X = x_i)$	q	p

De kansfunctie wordt grafisch voorgesteld in Figuur 3.1.

Eigenschap 3.3 (Verwachtingswaarde en variantie) Als X een toevalsveranderlijke is die een Bernoulliverdeling volgt met kans op succes p, dan geldt

$$\mu_X = p$$

en

$$\sigma_X^2 = p(1-p).$$ ∎

Bewijs We passen de definitie van verwachtingswaarde toe:

$$\mu_X = x_0\,\mathbb{P}(X = x_0) + x_1\,\mathbb{P}(X = x_1) = 0 \times q + 1 \times p = p.$$

Voor de variantie berekenen we

$$\mathrm{E}(X^2) = x_0^2\,\mathbb{P}(X = x_0) + x_1^2\,\mathbb{P}(X = x_1) = 0^2 \times q + 1^2 \times p = p,$$

zodat we vinden

$$\sigma_X^2 = \mathrm{E}(X^2) - \mathrm{E}(X)^2 = p - p^2 = p(1-p).$$ ♢

3.2.2 Binomiale verdeling: aantal successen

De binomiale verdeling is een zeer belangrijke kansverdeling. Men kan, vertrekkend van Bernoulli-experimenten, nieuwe kansexperimenten opzetten; namelijk door n onafhankelijke uitvoeringen van eenzelfde Bernoulli-experiment te beschouwen.

De toevalsveranderlijke X die telt hoeveel successen er voorkomen wanneer men n onafhankelijke Bernoulli-experimenten uitvoert volgt een BINOMIALE VERDELING. Zo'n verdeling wordt gekenmerkt door twee parameters, namelijk n, het aantal maal dat het Bernoulli-experiment wordt herhaald en p, de kans op succes van een individueel Bernoulli-experiment. We noteren dit als

$$X \sim B(n, p).$$

Voorbeeld 3.4 De volgende toevalsveranderlijken volgen een binomiale verdeling:

- De toevalsveranderlijke die het aantal keer kop telt wanneer men 5 keer met een muntstuk gooit.
- De toevalsveranderlijke die het aantal winnende loten telt onder 10 gekochte lotjes.
- De toevalsveranderlijke die het aantal jongens telt onder 4 geboorten bij een bepaald ouderpaar.
- De toevalsveranderlijke die het aantal witte ballen telt wanneer men drie ballen uit een vaas met witte en zwarte ballen trekt. De getrokken bal wordt teruggelegd. ∎

Eigenschap 3.5 (Kansfunctie) Veronderstel dat $X \sim B(n, p)$, dan wordt de kansfunctie van X gegeven door:

$$f_X(k) = \mathbb{P}(X = k) = C_n^k p^k q^{n-k} = \binom{n}{k} p^k q^{n-k}, \quad \text{als } 0 \leq k \leq n. \qquad \blacksquare$$

Bewijs Het universum van een binomiaal experiment is de verzameling van alle herhalingsvariaties van 2 elementen (nl. S en M) n aan n , i.e. elk element van Ω bestaat uit een bepaalde opeenvolging van successen en mislukkingen, n in totaal. Het is duidelijk dat $\#\Omega = 2^n$.

We beschouwen nu het element ω_k van Ω, waarbij de eerste k Bernoulli-experimenten een succes waren, en de laatste $n-k$ een mislukking. In formulevorm:

$$\omega_k = (\underbrace{S, S, \ldots, S}_{k}, \underbrace{M, M, \ldots, M}_{n-k})$$

Aangezien de Bernoulli-experimenten onafhankelijk zijn, worden de kansen van de elementen van Ω toegekend volgens de productregel van kansen. We vinden m.a.w.

$$\mathbb{P}(\omega_k) = \underbrace{p \cdot p \cdot \cdots \cdot p}_{k} \cdot \underbrace{q \cdot q \cdot \cdots \cdot q}_{n-k} = p^k \cdot q^{n-k}.$$

Maar ook alle elementen die bekomen worden door onderlinge permutaties van de S-en en de M-en hebben dezelfde waarschijnlijkheid als ω_k. Dit aantal is natuurlijk precies het aantal deelverzamelingen met k elementen uit een verzameling met n elementen, of dus C_n^k. Dit leidt tot

$$\mathbb{P}(X = k) = C_n^k p^k q^{n-k}. \qquad \diamondsuit$$

Opmerking 3.6 Dat met deze definitie voor het berekenen van de kansen een kansmodel wordt opgebouwd volgt uit het binomium van Newton

$$\sum_{k=0}^{n} \mathbb{P}(X = k) = \sum_{k=0}^{n} C_n^k p^k q^{n-k} = (p+q)^n = 1^n = 1,$$

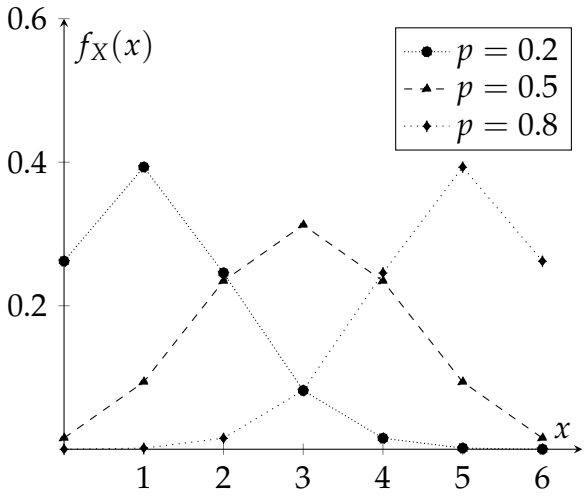
omdat natuurlijk $p + q = 1$. ∎

In Figuur 3.2 worden de grafieken van enkele binomiale kansfuncties weergegeven.

De kansverdelingsfunctie F_X geeft dan de kans op *ten hoogste* k successen: $F_X(k) = \mathbb{P}(X \leq k)$. Met de complementregel vindt men dan gemakkelijk

$$\mathbb{P}(X \geq k) = 1 - \mathbb{P}(X < k) = 1 - \mathbb{P}(X \leq k-1) = 1 - F(k-1).$$

Hierbij hebben we gebruikgemaakt van het feit dat de binomiale verdeling een *discrete* verdeling is. Dit kan soms helpen om het rekenwerk te verkorten.



Figuur 3.2: De grafieken van de kansfunctie voor een binomiaal verdeelde toevalsveranderlijke waarbij n steeds gelijk is aan 6, i.e. $X \sim B(6, p)$. Merk op dat voor $p = 0.5$ de kansfunctie symmetrisch is. **Opmerking:** de kansfunctie is *enkel gedefinieerd* voor gehele waarden van x.
De lijnen tussen de symbolen zijn enkel een visueel hulpmiddel!

Eigenschap 3.7 (Verwachtingswaarde en variantie) Als $X \sim B(n, p)$ dan geldt

$$\mu_X = np$$

en

$$\sigma_X^2 = np(1-p). \qquad \blacksquare$$

Bewijs We gebruiken de definitie van verwachtingswaarde en met behulp van (technische) sommanipulaties bekomen we:

$$\begin{aligned}
\mu_X &= \sum_{k=0}^{n} k\,\mathbb{P}(X=k) = \sum_{k=0}^{n} kC_n^k p^k q^{n-k} \\
&= \sum_{k=1}^{n} kC_n^k p^k q^{n-k} = \sum_{k=1}^{n} k\frac{n!}{k!(n-k)!} p^k q^{n-k} \\
&= \sum_{k=1}^{n} \frac{n!}{(k-1)!(n-k)!} p^k q^{n-k} = np \sum_{k=1}^{n} \frac{(n-1)!}{(k-1)!(n-k)!} p^{k-1} q^{n-k} \\
&= np \sum_{k=0}^{n-1} \frac{(n-1)!}{k!(n-1-k)!} p^k q^{n-1-k} = np(p+q)^{n-1} \\
&= np.
\end{aligned}$$

De formule voor σ_X^2 vindt men op een gelijkaardige manier door eerst $\mathrm{E}(X^2)$ te berekenen, en er rekening mee te houden dat $k^2 = k(k-1) + k$. $\diamondsuit$

Opmerking 3.8 Merk op dat de formules in Eigenschap 3.7 logische uitbreidingen zijn van de overeenkomstige formules voor de Bernoulliverdeling. Bovendien is zeker de formule voor de verwachtingswaarde intuïtief duidelijk. Als men bv. 10 keer (n) een eerlijk ($p = 1/2$) muntstuk opgooit, dan verwacht men gemiddeld $5 = n \times p$ keer kop te zien. ∎

3.2.3 Geometrische verdeling: wachten op het eerste succes

Veronderstel dat we onafhankelijke Bernoulli-experimenten uitvoeren met kans op succes p. Na hoeveel uitvoeringen bekomt men voor het eerst een succes? De kansveranderlijke X die het *rangnummer* aangeeft van het eerste succes is verdeeld volgens een GEOMETRISCHE verdeling. Zo'n geometrische verdeling wordt volledig bepaald door de parameter p die de kans op succes aangeeft voor één enkel Bernoulli-experiment.

Voorbeeld 3.9 De volgende kansveranderlijken zijn geometrisch verdeeld:

- De kansveranderlijke die telt hoe vaak men met een muntstuk moet werpen alvorens voor de eerste maal kop te bekomen.
- De kansveranderlijke die aangeeft hoeveel bloedafnames men moet doen alvorens men de bloedgroep AB aantreft.
- Het aantal keer dat men op de roulette moet spelen alvorens men voor de eerste maal wint. ∎

Eigenschap 3.10 (Kansfunctie) Als X geometrisch verdeeld is met parameter p, dan wordt de kansfunctie f_X gegeven door

$$f_X(k) = \mathbb{P}(X = k) = q^{k-1}p, \quad \text{voor } k \geq 1.$$ ∎

Bewijs Men voert onafhankelijke Bernoulli-experimenten uit tot men de eerste maal een succes bekomt. Laat ω_k de gebeurtenis voorstellen dat dit eerste succes voorkwam bij de k-de poging. Zoals we reeds zagen in Voorbeeld 1.11 geldt dus

$$\Omega = \{\omega_1, \omega_2, \omega_3, \ldots\}.$$

De gebeurtenis ω_k kunnen we schematisch voorstellen als

$$\omega_k = \underbrace{MM \cdots M}_{k-1} S.$$

Uit de productregel voor kansen volgt nu gemakkelijk dat

$$\mathbb{P}(\omega_k) = \underbrace{q \cdot q \cdot \cdots \cdot q}_{k-1} \cdot p = q^{k-1}p.$$

Dit bewijst het gestelde aangezien ω_k de *enige* gebeurtenis is waarvoor X de waarde k aanneemt. ♢

Opmerking 3.11 Uit de reekssom van de meetkundige (ofte geometrische) reeks met reden q, i.e.

$$1 + q + q^2 + q^3 + \cdots = \sum_{i=0}^{\infty} q^i = \frac{1}{1-q},$$

volgt onmiddellijk dat

$$\sum_{k=1}^{\infty} q^{k-1}p = p \sum_{k=1}^{\infty} q^{k-1} = p\frac{1}{1-q} = 1.$$

We hebben dus inderdaad te maken met een kansfunctie. ∎

Een typisch verloop van de geometrische kansfunctie wordt gegeven in Figuur 3.3.

Eigenschap 3.12 (Verwachtingswaarde en variantie) Als X geometrisch verdeeld is met parameter p dan geldt:

$$\mu_X = \frac{1}{p}$$

en

$$\sigma_X^2 = \frac{q}{p^2}.$$

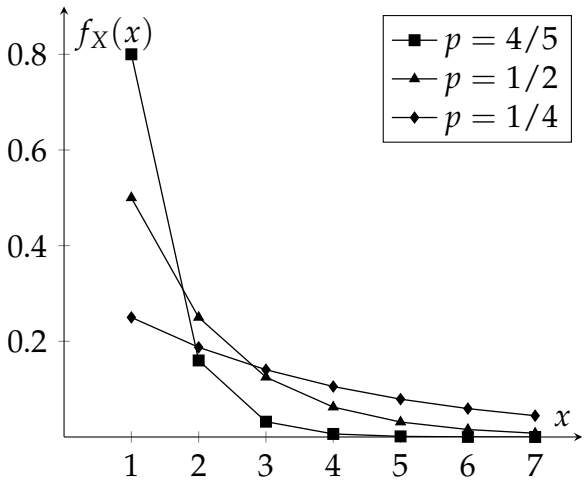
∎

Bewijs We passen de definitie van verwachtingswaarde toe:

$$\begin{aligned}\mu_X &= \sum_{k=1}^{\infty} k f_X(k) \\ &= \sum_{k=1}^{\infty} k q^{k-1} p \\ &= p + 2qp + 3q^2p + 4q^3p + \cdots\end{aligned}$$

Vermenigvuldigen we linker- en rechterlid met q dan vinden we

$$q\mu_X = pq + 2q^2p + 3q^3p + 4q^4p + \cdots$$



Figuur 3.3: Illustratie van de kansfunctie van geometrisch verdeelde toevalsveranderlijken voor verschillende waarden van p. **Opmerking:** de kansfunctie is *enkel gedefinieerd* voor gehele waarden van x. De lijnen tussen de symbolen zijn enkel een visueel hulpmiddel!

Als we nu beide uitdrukkingen van elkaar aftrekken dan vinden we

$$(1-q)\mu_X = p + pq + pq^2 + pq^3 + \cdots = p(1 + q + q^2 + q^3 + \cdots) = \frac{p}{1-q} = 1.$$

Hieruit volgt dus inderdaad dat $\mu_X = 1/p$.

De berekening voor de variantie is iets ingewikkelder. ♢

Opmerking 3.13 De geometrische verdeling bezit de *Markov-eigenschap*; ze heeft geen geheugen. Intuïtief is dit duidelijk: ook al had men eerst een rijtje van 20 mislukkingen, dan zal de kans op het eerstvolgende succes daar niet door beïnvloed worden. Meer formeel:

$$\mathbb{P}(X > m + n \mid X > m) = \mathbb{P}(X > n).$$

We kunnen dit ook gemakkelijk nagaan. De toevalsveranderlijke X is strikt groter dan k als er geen enkel succes voorkwam in de eerste k experimenten. We vinden dus

$$\mathbb{P}(X > k) = q^k.$$

M.b.v. de definitie van voorwaardelijke kans vinden we:

$$\begin{aligned}\mathbb{P}(X > m+n \mid X > m) &= \frac{\mathbb{P}\big((X > m+n) \cap (X > m)\big)}{\mathbb{P}(X > m)} \\ &= \frac{\mathbb{P}(X > m+n)}{\mathbb{P}(X > m)} \\ &= \frac{q^{m+n}}{q^m} \\ &= q^n \\ &= \mathbb{P}(X > n).\end{aligned}$$

Tenslotte kan men via de geometrische verdeling bepalen hoeveel trekkingen (met teruglegging) men moet uitvoeren om uit een populatie van n verschillende elementen er juist r verschillende te verzamelen. Het zal duidelijk zijn dat dit erg belangrijk is bij het bepalen van de grootte van steekproeven[3].

Als we met X_k het aantal trekkingen tussen het bekomen van de eerste k elementen en het $(k+1)$-ste verschillende element aanduiden, dan heeft X_k een geometrische verdeling met parameter $(n-k)/n$.

Het aantal benodigde trekkingen om de eerste r verschillende elementen te bekomen is dan te schrijven als

$$1 + X_1 + X_2 + \cdots + X_{r-1}.$$

Voorbeeld 3.14 We bepalen de gemiddelde wachttijd wanneer we uit 6 verschillende elementen er 3 verschillende wensen te bekomen. Het aantal benodigde trekkingen is verdeeld volgens

$$1 + X_1 + X_2,$$

waarbij X_i geometrisch verdeeld is met parameter $(6-i)/6$ voor $i \in \{1,2\}$. We hebben dus

$$\mathrm{E}(X_i) = \frac{6}{6-i},$$

en dus is

$$\mathrm{E}(1 + X_1 + X_2) = 1 + \mathrm{E}(X_1) + \mathrm{E}(X_2) = 1 + \frac{6}{5} + \frac{6}{4} = \frac{37}{10}.$$

[3]Zie cursus onderzoekstechnieken

Men moet dus gemiddeld 3.7 trekkingen doen alvorens 3 verschillende elementen werden bekomen bij trekking met teruglegging uit een populatie van 6 elementen.

Een analoge berekening toont aan dat de gemiddelde wachttijd totdat men alle 6 de elementen heeft getrokken 14.7 is. ∎

3.2.4 Poisson verdeling: zeldzame gebeurtenissen

In veel toepassingen heeft men te maken met een zeer groot aantal, genoteerd n, onafhankelijke uitvoeringen van eenzelfde Bernoulli-experiment, waarbij de kans op succes p zeer klein is en het gemiddeld aantal successen $n \cdot p$ ongeveer constant is. Deze constante wordt genoteerd met λ. Het aantal successen voldoet dan aan wat men noemt een POISSON-verdeling met parameter λ.

Voorbeeld 3.15 We geven een aantal voorbeelden van toevalsveranderlijken waarvoor de Poisson-verdeling een uitstekend model is.

1. Het aantal auto's dat per dag stopt aan een bepaald benzinestation om te tanken. In dit geval is n het aantal auto's dat per dag aan dat benzinestation passeert, terwijl p de kans is dat een auto stopt om te tanken.

2. Het aantal foutieve transmissietekens per seconde bij satellietcommunicatie. Hier is n het aantal tekens dat per seconde overgeseind wordt via de satelliet en p is de kans dat een fout optreedt.

3. Het aantal complicaties per dag bij injectie van een bepaald medicijn. Hier is n het aantal injecties met een bepaald medicijn per dag terwijl p de kans is dat een bepaalde schadelijke nevenreactie optreedt. ∎

Eigenschap 3.16 (Kansfunctie) Als X Poisson-verdeeld is met parameter λ, dan wordt zijn kansfunctie gegeven door

$$f_X(k) = \mathbb{P}(X = k) = e^{-\lambda}\frac{\lambda^k}{k!}, \quad \text{met} \quad k \geq 0.$$ ∎

Bewijs Uit de definitie volgt dat de Poissonverdeling een benadering is van de binomiale verdeling onder de voorwaarden $n \gg 0$, $p \approx 0$ en $n \cdot p = \lambda$ met $\lambda \in \mathbb{R}$.

We starten met de kansfunctie van de binomiale verdeling van een binomiaal verdeelde veranderlijke Y:

$$\begin{aligned}\mathbb{P}(Y=k) &= \frac{n\cdot(n-1)\cdot\cdots\cdot(n-k+1)p^k\cdot q^{n-k}}{k!}\\ &= \frac{n\cdot(n-1)\cdot\cdots\cdot(n-k+1)}{n^k}\cdot\frac{(np)^k}{q^k k!}\cdot q^n\end{aligned}$$

Nu geldt:

$$\lim_{n\to+\infty}\frac{n\cdot(n-1)\cdot\cdots\cdot(n-k+1)}{n^k}=1$$

en

$$(n\cdot p)^k=\lambda^k \quad \text{en} \quad q^k\approx 1.$$

De moeilijkste limietovergang[4] is die voor q^n. Men heeft

$$\lim_{n\to+\infty}q^n=\lim_{n\to+\infty}(1-p)^n=\lim_{n\to+\infty}(1-\frac{n\cdot p}{n})^n=\lim_{n\to+\infty}(1-\frac{\lambda}{n})^n=e^{-\lambda}.$$

Hieruit vindt men de formule voor de kansfunctie van f. ♢

Voorbeeld 3.17 We verifiëren dit wiskundig bewijs aan de hand van een experiment dat wordt samengevat in Tabel 3.1. Het aantal optredens van het koppel $(7,7)$ tussen 100 koppels van willekeurige getallen $(i,j)\in\mathbb{N}\times\mathbb{N}$ waarbij i en j begrensd zijn door 1 en 10 volgt een binomiale verdeling met $n=100$ en $p=0,01$.

We vergelijken de waarden van de kansfuncties van de binomiale en de Poisson-verdeling alsook de empirische frequenties die bekomen werden door een computerexperiment uit te voeren.

We zien dat de Poisson-verdeling inderdaad een goede benadering is voor de binomiale verdeling. ∎

Eigenschap 3.18 (Verwachtingswaarde en variantie) Als de toevalsveranderlijke X Poisson-verdeeld is met parameter λ, dan geldt

$$\mu_X=\lambda$$

en

$$\sigma_X^2=\lambda.$$ ∎

[4] Maak gebruik van $\lim_{n\to+\infty}(1+\frac{x}{n})^n=e^x$.

k	binomiaal	Poisson	empirisch
0	0,366032	0,367879	0,41
1	0,369730	0,367879	0,34
2	0,184865	0,183940	0,16
3	0,060999	0,061313	0,08
4	0,014942	0,015328	0,00
5	0,002898	0,003066	0,01
6	0,000463	0,000511	0,00
7	0,000063	0,000073	0,00
8	0,000007	0,000009	0,00
9	0,000001	0,000001	0,00

Tabel 3.1: De tweede kolom geeft de kansfunctie van een binomiale verdeling met $n = 100$ en $p = 0,01$; de derde kolom geeft de kansfunctie van een Poisson-verdeling met $\lambda = 1$; de laatste kolom geeft de relatieve frequenties van de uitkomsten van een computerexperiment. In dit experiment werden 100 keer 100 koppels getallen getrokken. De eerste rij wijst erop dat in 41 gevallen het koppel $(7,7)$ niet aanwezig was onder de 100 getrokken koppels, in 34 gevallen was het koppel $(7,7)$ juist één keer aanwezig onder de 100 getrokken koppels, ...

Bewijs Rechtstreeks toepassen van de definities en (eenvoudige) reeksmanipulaties leveren het gestelde. ♢

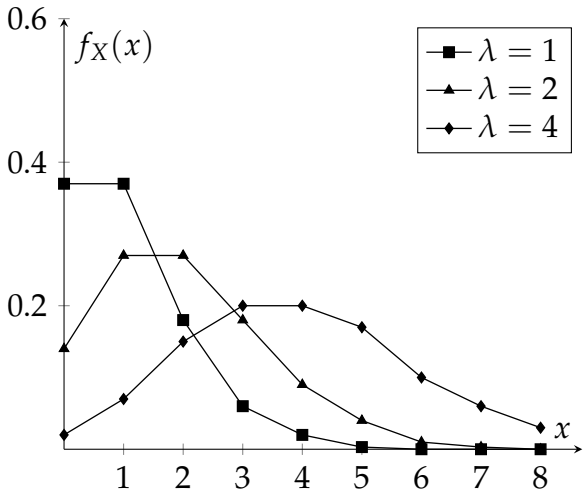
Opmerking 3.19 Dit is in overeenstemming met de gekende resultaten voor de binomiale verdeling! Laat $n \to \infty$ en $p \to 0$ met $n \cdot p = \lambda$. ∎

Voor kleine waarden van λ is de verdeling scheef naar links, voor grotere waarden wordt ze meer en meer symmetrisch. Dit wordt geïllustreerd in Figuur 3.4.

We vermelden nog dat de Poisson-verdeling een grote rol speelt bij de zogenaamde *Poisson-processen*. Daarbij wordt het eenheidstijdsinterval vervangen door een willekeurig tijdsinterval $[0, t]$. Dit leidt tot de kans

$$\mathbb{P}(X = k) = e^{-\lambda t} \cdot \frac{(\lambda t)^k}{k!}$$

voor het vinden van precies k successen in het interval $[0, t]$.



Figuur 3.4: Illustratie van de kansfunctie van Poisson-verdeelde toevalsveranderlijken voor verschillende waarden van λ. **Opmerking:** de kansfunctie is *enkel gedefinieerd* voor gehele waarden van x. De lijnen tussen de symbolen zijn enkel een visueel hulpmiddel!

De kans op geen enkel succes in het interval $[0, t]$ is dan

$$\mathbb{P}(X = 0) = e^{-\lambda t}$$

en de kans op 1 of meer successen

$$\mathbb{P}(X \geq 1) = 1 - \mathbb{P}(X = 0) = 1 - e^{-\lambda t}.$$

De parameter λ is een fysische constante die de dichtheid van het aantal successen aangeeft.

3.2.5 Oefeningen

1. Een verzekeringsmaatschappij sluit met 10 personen een levensverzekering af. De kans dat een individuele persoon nog in leven is na 30 jaar is 60%. Bereken de kans dat na 30 jaar

   a) alle verzekerden nog leven;

   b) minstens 3 personen nog in leven zijn;

   c) exact 4 personen in leven zijn.

2. Om de verkoop van een bepaald artikel te stimuleren heeft een fabrikant in een kwart van de pakjes een geschenkje verpakt. Iemand koopt twee pakjes.

   a) Bereken de kans op twee geschenken.

   b) Wat is het gemiddeld aantal geschenken dat zo'n persoon mag verwachten, i.e. wat is μ?

   c) Bepaal de variantie σ^2.

3. Een wijnhandelaar blijkt zijn flessen enigszins onnauwkeurig te vullen. Hierdoor voldoet 10% van de afgeleverde flessen niet aan de inhoudsnorm van het etiket.

   a) Een consument koopt 12 flessen wijn. Hoe groot is de kans dat precies 2 flessen niet aan de norm voldoen?

   b) Iemand koopt 144 flessen. Hoe groot is de kans dat hoogstens 10 flessen niet aan de norm voldoen?

4. Bij een injectie met een bepaald medicijn is de kans op een schadelijke nevenreactie 0,001. Bereken de kans dat bij 2000 inspuitingen

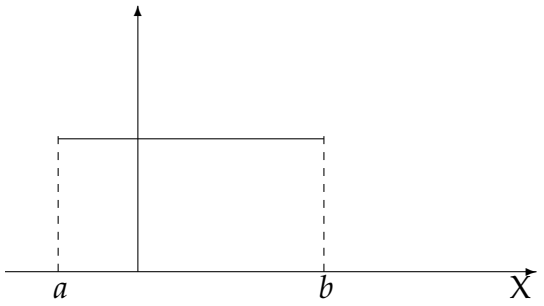
   a) 3 patiënten deze reactie vertonen;

   b) meer dan 5 patiënten de reactie vertonen.

5. Het aantal klanten dat per minuut een postkantoor binnenkomt, mag worden beschouwd als een kansvariabele met een Poisson-verdeling. De kans dat binnen een minuut niemand binnenkomt, bedraagt 0,6065. Bepaal de kans dat in een periode van 10 minuten er meer dan 10 klanten binnenkomen.

6. Bij een telefooncentrale komen gemiddeld 180 oproepen per uur binnen. Het aantal oproepen per uur mag beschouwd worden als een toevalsveranderlijke met een Poisson-verdeling. In een minuut kunnen hoogstens 6 gesprekken verwerkt worden. Bereken de kans dat in een bepaalde minuut overbelasting optreedt.

7. Op een bepaalde verkeersweg tussen twee steden gebeuren per jaar gemiddeld 10 verkeersongevallen. Bereken de kans dat er tijdens het komende jaar 8 of meer ongevallen gebeuren in de veronderstelling dat het aantal verkeersongevallen per jaar een Poisson-verdeling volgt.



Figuur 3.5: Kansdichtheid voor een continue uniforme verdeling met grenzen a en b.

8. Een webrobot zoekt naar een bepaald sleutelwoord in een reeks van onafhankelijk gekozen webpagina's. Men gaat ervan uit dat 15% van de webpagina's dit sleutelwoord bevatten.

   a) Wat is de kans dat de webrobot een webpagina vindt die het sleutelwoord bevat in de eerste vijf bezochte websites?

   b) Hoeveel webpagina's moet de webrobot gemiddeld bezoeken om een webpagina te vinden die het sleutelwoord bevat?

3.3 Continue kansverdelingen

We bekijken nu enkele continue kansverdelingen.

3.3.1 Uniforme verdeling

Dit is de meest eenvoudige continue verdeling. Boven een bepaalde *ondergrens* a en onder een bepaalde *bovengrens* b is de kansdichtheid constant (en verschillend van nul). Intuïtief, maar wiskundig verkeerd(!), betekent dit dat alle waarden tussen a en b een even grote kans hebben om voor te komen. Dit is bv. het geval bij het rad-der-fortuin experiment uit Voorbeeld 2.3. Men heeft dus een kansdichtheid die eruit ziet als in Figuur 3.5.

Eigenschap 3.20 (Kansdichtheid) Als X een continue UNIFORME VERDELING volgt met grenzen a en b, dan wordt de kansdichtheid van X gegeven door

$$f_X(x) = \begin{cases} \dfrac{1}{b-a} & \text{als } a \leq x \leq b, \\ 0 & \text{anders.} \end{cases}$$

Bewijs Men dient enkel te controleren dat de totale oppervlakte boven de x-as en onder de grafiek van $f_X(x)$ gelijk is aan 1. Dit is onmiddellijk duidelijk. ♢

Eigenschap 3.21 (Verwachtingswaarde en variantie) Als X een continue uniforme verdeling volgt met grenzen a en b dan geldt

$$\mu_X = \frac{(a+b)}{2},$$

en

$$\sigma_X^2 = \frac{(b-a)^2}{12}.$$ ∎

Bewijs Het bewijs is een rechtstreekse toepassing van de definitie van μ_X en σ_X^2. De integralen zijn eenvoudig te berekenen. In Voorbeeld 2.31 wordt de berekening gemaakt met $a = 0$ en $b = 2\pi$. ♢

Opmerking 3.22 De formule voor de verwachtingswaarde is intuïtief duidelijk aangezien de uniforme verdeling *symmetrisch* is rond het punt $(a + b)/2$. Verder zien we ook dat de variantie stijgt als de afstand tussen b en a groter wordt. ∎

De uniforme verdeling wordt soms gebruikt als weinig gegevens beschikbaar zijn over een experimentele of historische verdeling, maar waarbij men wel een schatting kan doen over onder- en bovengrens.

Verder wordt de uniforme verdeling bij *simulaties* gebruikt voor het genereren van random-getallen, waarmee dan weer trekkingen kunnen gebeuren uit andere verdelingen.

3.3.2 De exponentiële verdeling

Veronderstel dat we te maken hebben met een Poisson-proces met parameter λ. We kunnen ons dan afvragen wat de verdeling is van de *tijd* T die nodig is voordat het eerste succes voorkomt. We zeggen dat T een EXPONENTIËLE VERDELING heeft.

Eigenschap 3.23 (Kansdichtheid en kansverdelingsfunctie)

De kansdichtheid en kansverdelingsfunctie van een exponentieel verdeelde veranderlijke T met parameter λ worden respectievelijk gegeven door:

$$f_T(t) = \begin{cases} 0 & \text{als } t < 0 \\ \lambda e^{-\lambda t} & \text{als } t \geq 0, \end{cases}$$

en

$$F_T(t) = \begin{cases} 0 & \text{als } t < 0 \\ 1 - e^{-\lambda t} & \text{als } t \geq 0. \end{cases}$$

∎

Bewijs Het is eenvoudig in te zien dat de $T > t$ a.s.a. er nul voorkomens zijn in het interval $[0, t]$:

$$\mathbb{P}(T > t) = \mathbb{P}(\text{nul voorkomens in}[0, t]) = e^{-\lambda t}.$$

Uit de complementswet volgt dan dat

$$F_T(t) = \mathbb{P}(T \leq t) = 1 - e^{-\lambda t}.$$

Door het berekenen van de afgeleide functie (naar t) vindt men onmiddellijk de formule voor $f_T(t)$. ♢

In Figuur 3.6 ziet men een grafiek van de kansdichtheid van een exponentiële verdeling.

Eigenschap 3.24 (Verwachtingswaarde en variantie) Als T een toevalsveranderlijke is die exponentieel verdeeld is met parameter λ dan geldt

$$\mu_T = \frac{1}{\lambda}$$

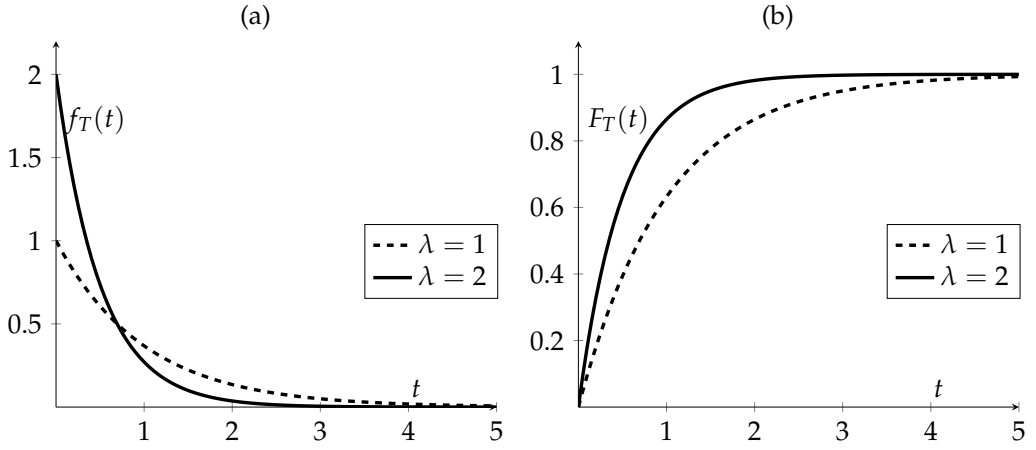
en

$$\sigma_T^2 = \frac{1}{\lambda^2}.$$

∎

Bewijs Rechtstreeks toepassen van de definitie van gemiddelde en verwachtingswaarde voor een continue toevalsveranderlijke levert het gestelde. ♢

Opmerking 3.25 Hoe groter de dichtheid van de voorkomens, i.e. hoe groter λ, hoe kleiner de gemiddelde wachttijd. Het resultaat is inderdaad intuïtief logisch: als er bv. (gemiddeld) 5 gebeurtenissen per minuut zijn, i.e. $\lambda = 5/\text{minuut}$, dan duurt het (gemiddeld) $1/5$ minuut voor een gebeurtenis zich voordoet. ∎



Figuur 3.6: Grafieken van de kansdichtheden van een exponentiële verdeling met $\lambda = 1$ en $\lambda = 2$. Rechts worden de grafieken van de corresponderende kansverdelingsfuncties getoond.

Eigenschap 3.26 De exponentiële verdeling voldoet aan de MARKOV eigenschap; ze bezit namelijk geen geheugen. Wiskundig betekent dit dat voor elke grootheid Δt geldt dat

$$\mathbb{P}(T > t + \Delta t \,|\, T > \Delta t) = \mathbb{P}(T > t). \qquad \blacksquare$$

Bewijs Het bewijs is een rechtstreekse toepassing van de formule voor voorwaardelijke kans:

$$\begin{aligned}
\mathbb{P}(T > t + \Delta t \,|\, T > \Delta t) &= \frac{\mathbb{P}\big((T > t + \Delta t) \cap (T > \Delta t)\big)}{\mathbb{P}(T > \Delta t)} \\
&= \frac{\mathbb{P}(T > t + \Delta t)}{\mathbb{P}(T > \Delta t)} \\
&= \frac{e^{-\lambda(t+\Delta t)}}{e^{-\lambda \Delta t}} \\
&= e^{-\lambda t} \\
&= \mathbb{P}(T > t).
\end{aligned}$$ ♢

Opmerking 3.27 Wanneer T dus de tijd voorstelt tussen twee voorkomens van een bepaalde gebeurtenis, dan zien we dat de tijd tot het volgende voorkomen onafhankelijk is van het tijdstip van het laatste voorkomen. ∎

3.3.3 Oefeningen

1. Jobs worden naar een printer gezonden aan een gemiddeld tempo van 3 jobs per uur.

   a) Wat is de verwachte tijd tussen jobs?

   b) Wat is de waarschijnlijkheid dat de volgende job wordt gezonden binnen de volgende 5 minuten?

2. De tijd nodig door een technicus om een bepaald soort machine te herstellen heeft een exponentiële verdeling met een gemiddelde van 4 uur. Er bestaat speciaal gereedschap dat dit gemiddelde doet dalen naar 2 uur. Wanneer de technicus erin slaagt om een machine te herstellen in minder dan 2 uur, dan krijgt hij 100 EUR, anders krijgt hij 80 EUR. Bepaal de verwachte meeropbrengst per machine wanneer de technicus het speciale gereedschap gebruikt.

3. Een programma is verdeeld in 3 modules die parallel worden gecompileerd op 3 verschillende computers. De compilatietijd van elk blok is exponentieel verdeeld met een gemiddelde van 5 minuten, onafhankelijk van de andere blokken. Het programma is gecompileerd wanneer alle blokken gecompileerd zijn.

   a) Wat is de waarschijnlijkheid dat het programma volledig gecompileerd is in minder dan 5 minuten?

   b) Wat is de verwachte compilatietijd? Maak gebruik van het feit dat je de verwachtingswaarde van een toevalsveranderlijke X die enkel niet-negatieve waarden aanneemt ook als volgt kan berekenen:
   $$\mu_X = \int_0^\infty (1 - F_X(x))\,\mathrm{d}x.$$

Deel II

Bomen en Grafen

HOOFDSTUK 4

Bomen

We starten met de recursieve definitie van GEWORTELDE BOMEN samen met hun basisbegrippen. We bespreken kort twee DATASTRUCTUREN om zo'n bomen voor te stellen in het computergeheugen. Aangezien gewortelde bomen op een natuurlijke wijze recursief gedefinieerd zijn volgt hieruit dat een aantal bewerkingen en berekeningen ook gemakkelijk recursief kunnen geïmplementeerd worden. Vervolgens geven we de definitie van een BINAIRE BOOM samen met een aantal basisbewerkingen. BINAIRE ZOEKBOMEN laten, dankzij hun ordeningseigenschap, toe om (meestal) snel elementen op te zoeken, toe te voegen en te verwijderen. BINAIRE HOPEN worden gebruikt om op een efficiënte en eenvoudige manier een PRIORITEITSWACHTRIJ te implementeren.

4.1 Terminologie m.b.t. bomen

Er zijn veel situaties waarin informatie geordend is volgens één of andere hiërarchische structuur. Denken we bv. maar aan bestandsystemen, de structuur van XML-bestanden, familiestambomen, organisatorische structuren enzovoort. De abstractie die zulke situaties modelleert is een boom, een fundamenteel begrip in de informatica.

Aangezien bomen een recursieve structuur vertonen, is een *recursieve definitie* aangewezen.

Definitie 4.1 Een GEWORTELDE BOOM T is een verzameling van TOPPEN die aan de volgende eigenschappen voldoet:

1. Er is één speciale top t die de WORTEL van de boom wordt genoemd.
2. De andere toppen zijn verdeeld in $m \geq 0$ disjuncte verzamelingen $T_1, \ldots, T_m$ die op hun beurt elk weer een gewortelde boom zijn. ∎

Opmerking 4.2 Merk op dat één enkele top ook steeds een boom is aangezien het toegelaten is dat het aantal verzamelingen m nul is. Dit is het eindgeval van de recursie. ∎

We zeggen dat de bomen T_1 t.e.m. T_m de DEELBOMEN zijn van T. De wortels $t_1, \ldots, t_m$ van de deelbomen T_1 t.e.m. T_m worden de KINDEREN van de wortel t genoemd. Omgekeerd is t de OUDER van $t_1, \ldots, t_m$. De toppen $t_1, \ldots, t_m$ zijn BROERS van elkaar. De termen AFSTAMMELING en VOOROUDER zijn logische uitbreidingen van de kind/ouder terminologie. De afstammelingen van een top t vormen een verzameling die bestaat uit de kinderen van t, en alle afstammelingen van de kinderen. Een top t is een voorouder van een top t' als en slechts als t' een afstammeling is van t.

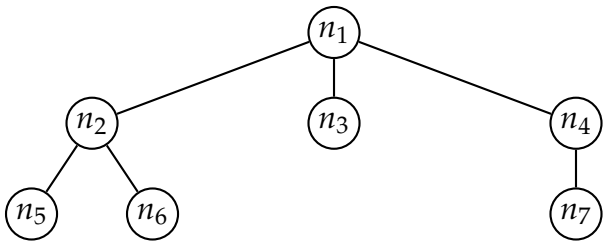
Uit de recursieve definitie van een gewortelde boom volgt dat elke top in de boom uiteindelijk de wortel is van een deelboom die bevat is in de boom. Het aantal deelbomen van een top wordt de GRAAD van die top genoemd. Een BLAD is een top met graad nul. Een top die geen blad is wordt INTERN genoemd. De GRAAD van een *boom* wordt gedefinieerd als het maximum van de graden van zijn toppen.

Bomen hebben ook een grafische voorstelling. De toppen worden meestal als kleine cirkeltjes getekend, eventueel met een label erin om duidelijk te maken over welke top het gaat. De wortel van een boom wordt bovenaan getekend, en we tekenen een lijn tussen de wortel en zijn kinderen. Figuur 4.1 geeft een voorbeeld van een boom.

Voorbeeld 4.3 De wortel van de boom T in Figuur 4.1 is de top met label n_1. De andere toppen zijn verdeeld in 3 disjuncte verzamelingen, nl.

$$T_1 = \{n_2, n_5, n_6\}, \quad T_2 = \{n_3\} \quad \text{en} \quad T_3 = \{n_4, n_7\}.$$

De toppen n_2, n_3 en n_4, zijn de kinderen van de top n_1. Omgekeerd is n_1 dus de ouder van n_2, n_3 en n_4. De toppen n_2, n_3 en n_4 zijn broers van elkaar want ze hebben dezelfde ouder.



Figuur 4.1: Een eerste voorbeeld van een boom.

De verzameling T_1 is op zijn beurt weer een boom met de top n_2 als wortel. De toppen n_5 en n_6 zijn de kinderen van n_2. Omgekeerd is de top n_2 de ouder van n_5 en n_6. De toppen n_5 en n_6 zijn broers.

Aangezien n_5 en n_6 kinderen zijn van n_2, die op zijn beurt weer een kind is van n_1, zijn n_5 en n_6 afstammelingen van n_1. In dit voorbeeld bestaat de verzameling van de afstammelingen van n_1 uit alle toppen van T behalve n_1.

Uit de figuur lezen we ook gemakkelijk de graad van de verschillende toppen af. De graad van n_1 is 3, want n_1 heeft 3 kinderen. De graad van n_2 is 2, want deze top heeft 2 kinderen, enzovoort.

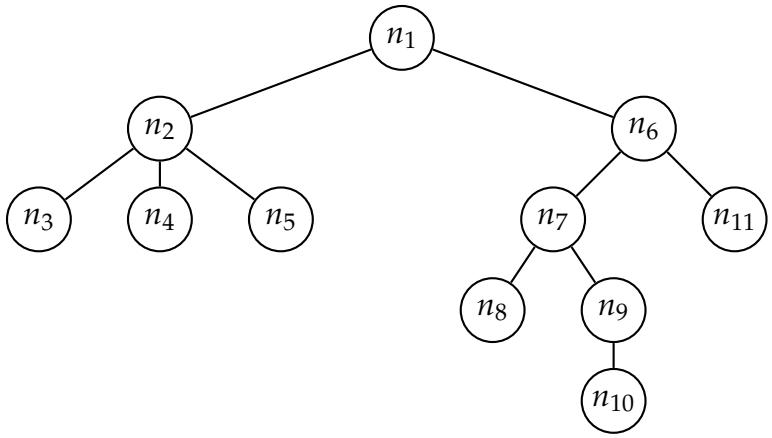
De bladeren van de boom zijn de toppen n_5, n_6, n_3 en n_7, want deze toppen hebben allen graad nul.

De graad van de boom is 3 want dit is het maximum van de graden van zijn toppen. Merk op dat in dit geval de graad van de boom gelijk is aan de graad van zijn wortel maar dat is zeker niet altijd het geval. ∎

De DIEPTE van een top n m.b.t. een boom T wordt als volgt (recursief) gedefinieerd: de diepte van de wortel van T is nul terwijl de diepte van elke andere top één meer is dan de diepte van zijn ouder. Op deze manier krijgt elke top een unieke diepte. De diepte van de *boom* T is de maximale diepte van zijn toppen. De HOOGTE van een top is de diepte van de deelboom met die top als wortel. De diepte van een top geeft m.a.w. de 'afstand' tot de wortel terwijl de hoogte de 'afstand' geeft tot het verste blad. De hoogte van een *boom* wordt gedefinieerd als de hoogte van zijn wortel. Het is eenvoudig in te zien dat de hoogte en de diepte van een *boom* steeds gelijk zijn aan elkaar.

top	diepte	hoogte
n_1	0	2
n_2	1	1
n_3	1	0
n_4	1	1
n_5	2	0
n_6	2	0
n_7	2	0

Tabel 4.1: Diepte en hoogte van de toppen van de boom in Figuur 4.1.



Figuur 4.2: Een voorbeeldboom.

Voorbeeld 4.4 In Tabel 4.1 staat voor elke top van de boom in Figuur 4.1 de diepte en de hoogte opgesomd. Men kan deze tabel gemakkelijk zelf verifiëren. Start met de wortel n_1 de diepte 0 toe te kennen. Zijn kinderen krijgen dan diepte 1. De kinderen van de kinderen krijgen dan diepte 2.

De diepte van de boom T is 2 aangezien dit de maximale diepte is van de toppen in de boom. De hoogte van de top n_1 is dus 2.

Als we de diepte van de boom T_1 bekijken, dan zien we dat daar de maximale diepte gelijk is aan 1, en dus is de hoogte van de top n_2 (de wortel van T_1) gelijk aan 1. ∎

4.1.1 Oefeningen

1. Bekijk de boom in Figuur 4.2. Beantwoord de volgende vragen:

   a) Geef de wortel van de boom.

   b) Geef de verzamelingen T_1 t.e.m. T_m volgens Definitie 4.1.

   c) Geef de kinderen van elke top in de boom.

   d) Geef de graad van elke top in de boom.

   e) Geef de graad van de boom T.

   f) Welke toppen zijn broers van elkaar?

   g) Geef de bladeren van de boom.

   h) Geef de afstammelingen van n_6.

   i) Geef de voorouders van n_{10}.

   j) Maak een tabel waarin voor elke top zijn hoogte en diepte wordt gegeven.

4.2 Datastructuren voor bomen

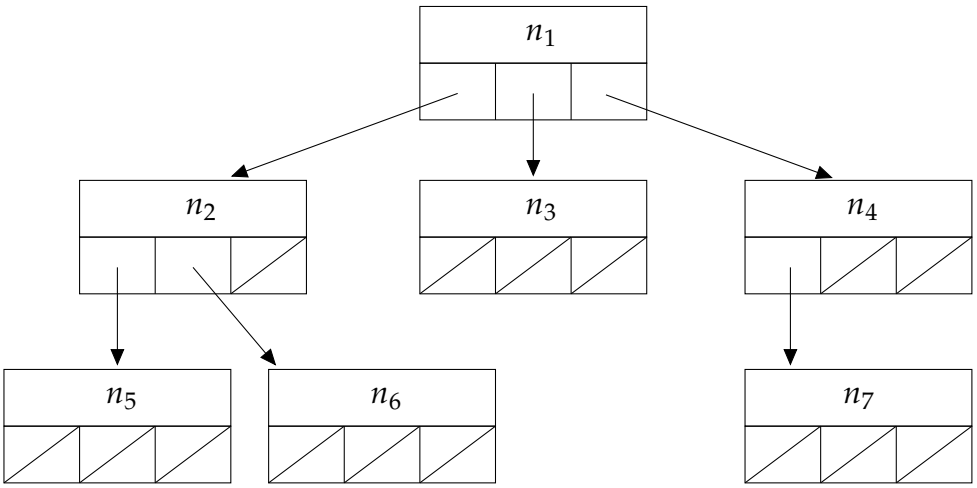
Er zijn verschillende manieren om bomen voor te stellen in het geheugen van een computer. Wij bespreken hier twee voorstellingen.

Opmerking 4.5 Wanneer men bomen voorstelt in het computergeheugen dan legt men vaak een (willekeurige) volgorde op aan de deelbomen, daar waar dat bij de wiskundige definitie niet het geval is. ∎

4.2.1 Array-van-kinderen voorstelling

De eenvoudigste manier om een boom voor te stellen is door rechtstreeks de vader-kind relatie te implementeren. Dit betekent dat we een structuur `Top` definiëren die een veld heeft om de data van de top bij te houden, alsook een array van referenties naar de kinderen van die top. Wanneer een kind niet bestaat dan wordt dit voorgesteld door de referentie `null`. De boom zelf bestaat uit een referentie naar zijn wortel.

Voorbeeld 4.6 In Figuur 4.3 staat een voorstelling getekend van hoe de boom in Figuur 4.1 zou kunnen opgeslaan worden in het computergeheugen. Aangezien de graad van de boom 3 is, zijn er in elke structuur voor



Figuur 4.3: Array-van-kinderen voorstelling voor de boom in Figuur 4.1. De referentie naar de wortel wordt niet explictiet getoond.

een top drie posities voorzien die een referentie naar een andere top kunnen bijhouden. Wanneer er geen kind is om naar te verwijzen dan wordt dit aangeduid door een schuine streep. ∎

We kunnen in Figuur 4.3 zien dat er veel referenties niet gebruikt worden. Inderdaad, we hebben geheugenruimte voorzien voor $7 \times 3 = 21$ referenties, waarvan er slechts 6 effectief gebruikt worden. Merk op dat het aantal gebruikte referenties juist één minder is dan het aantal toppen van de boom.

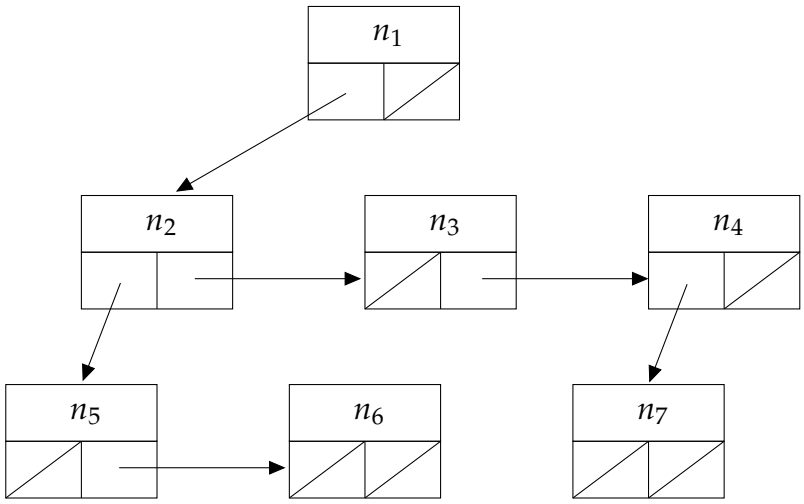
We kunnen de observatie uit de figuur gemakkelijk veralgemenen: inderdaad, stel dat n het aantal toppen van de boom T is en dat de graad van T gelijk is aan k. Wanneer we de array-van-kinderen voorstelling gebruiken dan zijn er in totaal $n \times k$ referenties. Echter, van deze referenties zijn er slechts $n - 1$ verschillend van `null` (want elke top, behalve de wortel, is kind van juist één andere top). Dit betekent dat slechts een fractie

$$\frac{n-1}{nk} \approx \frac{1}{k}$$

van de referenties verschillend is van `null` en dus effectief gebruikt wordt.

Wanneer bv. de graad van de boom gelijk is aan 3 wordt dus slechts ongeveer $1/3$ van de referenties gebruikt.

Een nadeel van de array-van-kinderen voorstelling is dus het inefficiënt gebruik van geheugen, zeker wanneer de graad van de boom groot is.



Figuur 4.4: De eerste-kind-volgende-broer voorstelling van de boom in Figuur 4.1.

4.2.2 Eerste-kind-volgende-broer voorstelling

We kunnen een boom voorstellen op een manier die efficiënter met het geheugen omgaat. In plaats van in elke top referenties naar al zijn kinderen op te slaan, houden we altijd juist twee referenties bij: een referentie naar zijn eerste kind, en een referentie naar zijn volgende broer.

Voorbeeld 4.7 In Figuur 4.4 is een voorstelling getekend van de eerste-kind-volgende-broer voorstelling van de voorbeeldboom in Figuur 4.1. De eerste referentie in de structuur die een top voorstelt is een referentie naar het "eerste" kind. De tweede referentie is er een naar de "volgende" broer van de top. ∎

Uit Figuur 4.4 leiden we af dat er nu heel wat minder `null` referenties zijn dan bij de array-van-kinderen voorstelling. Inderdaad, we hebben geheugenruimte voorzien voor $2 \times 7 = 14$ referenties waarvan er opnieuw 6 effectief gebruikt worden.

Met deze voorstelling is de referentie naar het eerste kind `null` als en slechts als de top een blad is. Wanneer de top geen blad is, dan zijn de kinderen van de top geschakeld in een lineaire lijst structuur. Dit betekent dat we hier dus geen rechtstreekse toegang hebben tot bv. het derde kind: om de kinderen van een bepaalde top te vinden moeten we in deze voorstel-

ling dus eerst de referentie volgen naar zijn eerste kind. Vervolgens moet er geïtereerd worden over de (gelinkte lijst) van volgende-broer referenties.

De eerste-kind-volgende-broer voorstelling is dus efficiënter qua geheugengebruik maar navigeren in de boom wordt iets moeilijker.

4.2.3 Oefeningen

1. Bereken hoeveel null-referenties er zullen zijn bij de array-van-kinderen voorstelling van de boom in Figuur 4.2. Wat is de verhouding van het aantal effectief gebruikte referenties tot het aantal voorziene referenties?

2. Teken de array-van-kinderen voorstelling van de boom in Figuur 4.2.

3. Bereken hoeveel null-referenties er zullen zijn bij eerste-kind-volgende-broer voorstelling van de boom in Figuur 4.2. Wat is de verhouding van het aantal effectief gebruikte referenties tot het aantal voorziene referenties?

4. Wat is de verhouding van het aantal effectief gebruikte referenties tot het aantal voorziene referenties voor een willekeurige gewortelde boom van graad k met n toppen.

5. Teken de eerste-kind-volgende-broer voorstelling van de boom in Figuur 4.2.

4.3 Recursie op bomen

4.3.1 Alle toppen van een boom bezoeken

Om alle toppen van een boom te bezoeken zijn er meerdere mogelijkheden. We bespreken er hier twee van. Een eerste mogelijkheid is om eerst de wortel van de boom te bezoeken en daarna (recursief en op dezelfde manier) de toppen van zijn deelbomen te doorlopen. De tweede mogelijkheid is om eerst (recursief) de toppen van de deelbomen te doorlopen en daarna pas de wortel van de boom te bezoeken. Beide mogelijkheden worden respectievelijk PREORDE en POSTORDE doorlopen van de boom genoemd.

Om een boom te doorlopen in preorde gaat men als volgt tewerk:

1. Bezoek de wortel van de boom.
2. Doorloop de deelbomen van de wortel in preorde.

Om een boom te doorlopen in postorde doet men het volgende:

1. Doorloop de deelbomen van de wortel in postorde.
2. Bezoek de wortel van de boom.

Dit proces zal eindigen want wanneer de boom slechts uit één top bestaat (m.a.w. als die top een blad is), dan moet er niets gedaan worden in de tweede stap wanneer preorde wordt gebruikt (of de eerste stap in het geval van postorde). Hier eindigt de recursie dus. Bovendien hebben de deelbomen steeds (strikt) minder toppen dan de originele boom, zodanig dat men in een eindig aantal stappen het basisgeval zal bereiken. Men komt dus niet in een oneindig diepe recursie terecht.

In Algoritme 4.1 wordt de pseudocode gegeven voor het in preorde doorlopen van een boom.

Algoritme 4.1 Doorlopen van een boom in preorde

Invoer Een gewortelde boom T, en een visit functie.
Uitvoer De visit functie is aangeroepen voor elke top van de boom.

```
1: function PREORDE(T,visit)
2:     PreOrdeRecursief(T.wortel, visit)          ▷ start met de wortel
3: end function
4: function PREORDERECURSIEF(v, visit)
5:     visit(v)
6:     for all w ∈ kinderen(v) do          ▷ implementatie-onafhankelijk
7:         PreOrdeRecursief(w, visit)
8:     end for
9: end function
```

Voorbeeld 4.8 Beschouw opnieuw de boom uit Figuur 4.1. Veronderstel nu dat we tijdens elk 'bezoek' aan een top enkel het label van die top afdrukken (print).

In Tabel 4.2 kunnen we zien in welke volgorde de verschillende methodes worden opgeroepen. Als we kijken in welke volgorde de print-opdrachten

```
PreOrdeRecursief(n1)
  print(n1)
  PreOrdeRecursief(n2)
    print(n2)
    PreOrdeRecursief(n5)
      print(n5)
    PreOrdeRecursief(n6)
      print(n6)
  PreOrdeRecursief(n3)
    print(n3)
  PreOrdeRecursief(n4)
    print(n4)
    PreOrdeRecursief(n7)
      print(n7)
```

```
PostOrdeRecursief(n1)
  PostOrdeRecursief(n2)
    PostOrdeRecursief(n5)
      print(n5)
    PostOrdeRecursief(n6)
      print(n6)
    print(n2)
  PostOrdeRecursief(n3)
    print(n3)
  PostOrdeRecursief(n4)
    PostOrdeRecursief(n7)
      print(n7)
    print(n4)
  print(n1)
```

Tabel 4.2: Traceren van de methode-oproepen wanneer de voorbeeldboom in Figuur 4.1 in preorde resp. postorde doorlopen wordt.

voorkomen dan zien we dat de labels in deze volgorde afgedrukt worden wanneer de boom in preorde doorlopen wordt:

$$n_1, n_2, n_5, n_6, n_3, n_4, n_7.$$

Wanneer de boom postorde doorlopen en de labels afdrukken dan krijgen we:

$$n_5, n_6, n_2, n_3, n_7, n_4, n_1.$$

Dit kan je eveneens aflezen uit Tabel 4.2. ∎

4.3.2 Eenvoudige berekeningen op bomen

Veel eenvoudige berekeningen op bomen kunnen op een natuurlijke manier recursief geformuleerd worden op een manier die doet denken aan het pre- of postorde doorlopen van de toppen. Vaak moet er voor en na de recursieve oproep ook nog wat werk worden gedaan, zoals het initialiseren en bijwerken van variabelen.

Veronderstel dat we het aantal toppen van een boom wensen te berekenen, maar dat de voorstelling van een boom dit niet rechtstreeks toelaat[1]. Uit

[1]Dit betekent dat we geen veld bijhouden voor de grootte.

de definitie van een boom kunnen we zien dat het aantal toppen $\#(T)$ in de boom T gelijk is aan

$$\#(T) = 1 + \#(T_1) + \#(T_2) + \cdots + \#(T_m). \tag{4.1}$$

Deze formule drukt uit dat het aantal toppen in een boom gelijk is aan de som van het aantal toppen in elk van de deelbomen van de wortel, vermeerderd met 1 (we mogen de wortel zelf niet vergeten). Formule (4.1) kan eenvoudig vertaald worden in een recursief algoritme, zie Algoritme 4.2.

Algoritme 4.2 Berekenen van aantal toppen van een boom.

Invoer Een gewortelde boom T
Uitvoer Het aantal toppen van de boom.

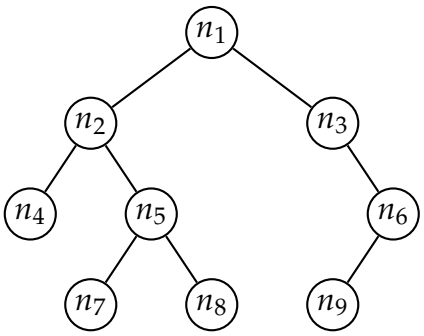
```
1:  function AANTAL(T)
2:      return AantalRecursief(T.wortel)
3:  end function
4:  function AANTALRECURSIEF(v)
5:      n ← 1                                   ▷ Top zelf niet vergeten
6:      for all w ∈ kinderen(v) do
7:          n ← n + AantalRecursief(w)
8:      end for
9:      return n
10: end function
```

Een andere berekening die eenvoudig kan geïmplementeerd worden is het berekenen van de hoogte van een boom. Inderdaad, de hoogte $h(T)$ van een boom is in het algemeen één meer dan het maximum van de hoogtes van zijn deelbomen:

$$h(T) = \begin{cases} 0 & \text{als } m = 0 \\ 1 + \max\big(h(T_1), h(T_2), \ldots, h(T_m)\big) & \text{als } m > 0. \end{cases} \tag{4.2}$$

4.3.3 Oefeningen

1. Geef de volgorde waarin de toppen worden bezocht wanneer de boom in Figuur 4.2 respectievelijk in preorde en postorde wordt doorlopen.
2. Geef code analoog aan Algoritme 4.1 om een gewortelde boom in postorde te doorlopen.
3. Geef code analoog aan Algoritme 4.2 om de hoogte van een gewortelde boom te berekenen. Baseer je op formule (4.2).



Figuur 4.5: Een binaire boom bestaande uit 9 toppen.

4.4 Binaire bomen

4.4.1 Definitie en eigenschappen

We beschouwen nu bomen die zeer vaak voorkomen in de praktijk, nl. binaire bomen. We geven opnieuw een recursieve definitie:

Definitie 4.9 Een BINAIRE BOOM is een verzameling toppen die

1. ofwel leeg is,
2. ofwel bestaat uit een *wortel* en twee disjuncte verzamelingen T_l en T_r, die op hun beurt ook een binaire boom zijn. We noemen T_l en T_r respectievelijk de *linker-* en *rechterdeelboom* van de wortel. ∎

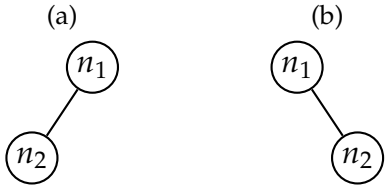
Voorbeeld 4.10 In Figuur 4.5 zien we een voorbeeld van een binaire boom T met

$$T = \{n_1, n_2, n_3, n_4, n_5, n_6, n_7, n_8, n_9\}.$$

De linker- en rechterdeelboom van de wortel n_1 zijn respectievelijk:

$$T_l = \{n_2, n_4, n_5, n_7, n_8\} \quad \text{en} \quad T_r = \{n_3, n_6, n_9\}.$$

De verzamelingen T_l en T_r zijn op hun beurt weer binaire bomen. De wortel van T_r is de top n_3. De linkerdeelboom van n_3 is leeg, terwijl de rechterdeelboom bestaat uit de toppen n_6 en n_9. Zoals je ziet worden lege bomen eenvoudigweg niet getekend. ∎



Figuur 4.6: Twee verschillende binaire bomen die gelijk zijn als "gewone" gewortelde bomen.

Opmerking 4.11 Een eerste verschil tussen de "gewone" gewortelde bomen uit Definitie 4.1 en binaire bomen is dat een binaire boom geen enkele top kan bevatten. Een tweede verschil is dat er bij binaire bomen altijd een ordening aan de deelbomen wordt opgelegd. Inderdaad, bekijk bv. de twee bomen in Figuur 4.6. Deze twee bomen zijn identiek als we ze beschouwen als "gewone" gewortelde bomen, maar ze zijn verschillend als binaire bomen. Inderdaad, bij de ene boom is de rechterdeelboom leeg, terwijl bij de andere boom de linkerdeelboom leeg is. ∎

In een binaire boom is het gemakkelijk om een bovengrens te geven voor het aantal toppen met een bepaalde diepte k. De diepte van een top in een binaire boom wordt trouwens op dezelfde manier gedefinieerd als in een "gewone" gewortelde boom.

Eigenschap 4.12 In een binaire boom is het aantal toppen met diepte k hoogstens 2^k. ∎

Bewijs In een niet-lege binaire boom is er steeds juist één top met diepte 0, nl. de wortel. Deze wortel heeft hoogstens twee niet-lege deelbomen, dus zijn er hoogstens 2 toppen met diepte 1. Elk van de toppen met diepte 1 heeft opnieuw hoogstens twee niet-lege deelbomen, dus het aantal toppen met diepte twee is hoogstens $2 \times 2 = 4$. Zo verdergaand zien we (door inductie) dat het aantal toppen met diepte k hoogstens 2^k is. ♢

Stel dat we weten dat de diepte van een binaire boom T gelijk is aan d. Dan kunnen we m.b.v. de vorige eigenschap een bovengrens geven voor het aantal toppen dat die boom kan bevatten. Een ondergrens volgt gemakkelijk uit het feit dat er minstens één top moet zijn op elke diepte.

Eigenschap 4.13 Voor een (niet-lege) binaire boom T met diepte $d \geq 0$ geldt dat:

$$d + 1 \leq \#(T) \leq 2^{d+1} - 1. \tag{4.3}$$

∎

Bewijs We tellen het aantal toppen van een boom T door het aantal toppen met een bepaalde diepte k te tellen en dit voor alle mogelijke dieptes. In formulevorm:

$$\begin{aligned}
\#(T) &= \sum_{k=0}^{d} \text{aantal toppen met diepte } k \\
&\leq \sum_{k=0}^{d} 2^k && \text{wegens Eigenschap 4.12} \\
&= 1 + 2 + 4 + \cdots + 2^d \\
&= 2^{d+1} - 1.
\end{aligned}$$

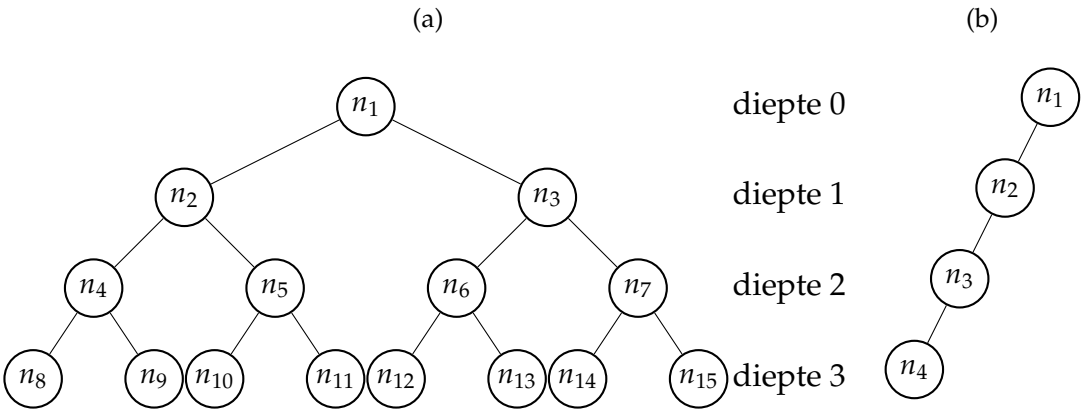
Dit bewijst de bovengrens voor $\#(T)$.

Een ondergrens voor het aantal toppen in een binaire boom met diepte d is ook gemakkelijk te bepalen. De boom zal het kleinste aantal toppen hebben wanneer er juist één top is op elke diepte. Het minimum aantal toppen in een binaire boom van diepte d is dus $d + 1$. ♢

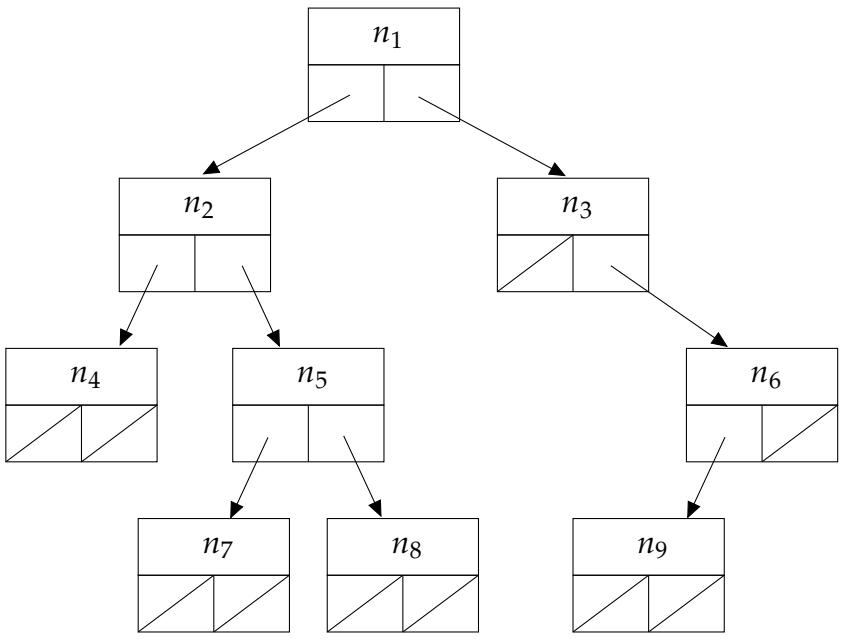
Voorbeeld 4.14 In Figuur 4.7 zie je aan de linkerkant de binaire boom van diepte 3 met het maximaal aantal toppen, nl. $2^{3+1} - 1 = 15$. Aan de rechterkant zie je een binaire boom van diepte 3 met het minimum aantal toppen. Merk op ook dat er slechts één binaire boom is met het maximaal aantal toppen voor een gegeven diepte, terwijl er *veel* verschillende binaire bomen zijn met het minimum aantal toppen voor een gegeven diepte. ∎

4.4.2 Voorstelling van een binaire boom

Elke top van een binaire boom wordt voorgesteld door een structuur die, behalve de eigenlijke data voor die top, ook twee referenties bijhoudt naar respectievelijk de linker- en rechterdeelboom. Wanneer de linker- of rechterdeelboom leeg is, dan is de overeenkomstige referentie `null`. De binaire boom zelf wordt voorgesteld door een referentie naar zijn wortel.



Figuur 4.7: De binaire boom met diepte 3 die het maximum aantal toppen bevat versus een binaire boom van diepte 3 die het minimum aantal toppen bevat.



Figuur 4.8: Interne voorstelling van de binaire boom in Figuur 4.5.

Voorbeeld 4.15 Figuur 4.8 geeft de interne voorstelling van de binaire boom in Figuur 4.5. De `null` referenties worden aangeduid met een schuine streep. ∎

4.4.3 Alle toppen van een binaire boom bezoeken

We hebben in Sectie 4.3 reeds gezien hoe we alle toppen van een gewortelde boom kunnen bezoeken, door de boom te doorlopen in ofwel pre- of postorde. Als we spreken over pre- of postorde doorlopen van binaire bomen dan is het (bijna) altijd zo dat de linkerdeelboom steeds doorlopen wordt voor de rechterdeelboom.

Om een binaire boom in PREORDE te doorlopen gaan we dus als volgt tewerk:

1. Bezoek de wortel van boom.
2. Als de linkerdeelboom niet leeg is, doorloop de linkerdeelboom dan recursief in preorde.
3. Als de rechterdeelboom niet leeg is, doorloop de rechterdeelboom dan recursief in preorde.

Om een binaire boom in POSTORDE te doorlopen gaan we dus als volgt tewerk:

1. Als de linkerdeelboom niet leeg is, doorloop de linkerdeelboom dan recursief in postorde.
2. Als de rechterdeelboom niet leeg is, doorloop de rechterdeelboom dan recursief in postorde.
3. Bezoek de wortel van boom.

Omdat elke top nu steeds twee deelbomen heeft (die eventueel leeg kunnen zijn), is er nog een derde natuurlijke mogelijkheid om een binaire boom te doorlopen. We kunnen namelijk de wortel bezoeken *tussen* het doorlopen van de linker- en rechterdeelboom. Bij bomen die meer dan twee kinderen kunnen hebben heeft deze optie weinig zin omdat er geen "natuurlijke" manier is om de wortel tussen de kinderen te plaatsen. Deze derde manier

om een binaire boom te doorlopen noemt men het INORDE doorlopen van de boom.

Om een binaire boom in INORDE te doorlopen gaan we dus als volgt tewerk:

1. Als de linkerdeelboom niet leeg is, doorloop de linkerdeelboom dan recursief in inorde.
2. Bezoek de wortel van boom.
3. Als de rechterdeelboom niet leeg is, doorloop de rechterdeelboom dan recursief in inorde.

Algoritme 4.3 Doorlopen van een binaire boom in inorde

Invoer Een binaire boom T en een visit functie.
Uitvoer De visit functie is aangeroepen voor elke top van T.

```
1:  function INORDE(T,visit)
2:      if T ≠ ∅ then                          ▷ controleer dat boom niet leeg is
3:          InOrdeRecursief(T.wortel, visit)   ▷ start met de wortel
4:      end if
5:  end function
6:  function INORDERECURSIEF(v, visit)
7:      if v.links ≠ ∅ then
8:          InOrdeRecursief(v.links, visit)
9:      end if
10:     visit(v)                   ▷ visit aanroepen tussen recursieve oproepen
11:     if v.rechts ≠ ∅ then
12:         InOrdeRecursief(v.rechts, visit)
13:     end if
14: end function
```

Voorbeeld 4.16 Tijdens een bezoek aan een top drukken we zijn label af. Wanneer we de toppen van de boom in Figuur 4.5 in preorde doorlopen dan worden de labels afgedrukt in de volgorde:

$$n_1, n_2, n_4, n_5, n_7, n_8, n_3, n_6, n_9.$$

Dit kunnen we afleiden uit Tabel 4.3 die de oproepen traceert voor de verschillende manieren om een binaire boom te doorlopen. Voor postorde lezen we af uit dezelfde tabel dat de volgorde van afdrukken de volgende is:

$$n_4, n_7, n_8, n_5, n_2, n_9, n_6, n_3, n_1.$$

preorde(n_1)
  print(n_1)
  preorde(n_2)
  print(n_2)
  preorde(n_4)
  print(n_4)
  preorde(n_5)
  print(n_5)
  preorde(n_7)
  print(n_7)
  preorde(n_8)
  print(n_8)
  preorde(n_3)
  print(n_3)
  preorde(n_6)
  print(n_6)
  preorde(n_9)
  print(n_9)
	postorde(n_1)
	  postorde(n_2)
	  postorde(n_4)
	  print(n_4)
	  postorde(n_5)
	  postorde(n_7)
	  print(n_7)
	  postorde(n_8)
	  print(n_8)
	  print(n_5)
	  print(n_2)
	  postorde(n_3)
	  postorde(n_6)
	  postorde(n_9)
	  print(n_9)
	  print(n_6)
	  print(n_3)
	  print(n_1)
		inorde(n_1)
		  inorde(n_2)
		  inorde(n_4)
		  print(n_4)
		  print(n_2)
		  inorde(n_5)
		  inorde(n_7)
		  print(n_7)
		  print(n_5)
		  inorde(n_8)
		  print(n_8)
		  print(n_1)
		  inorde(n_3)
		  print(n_3)
		  inorde(n_6)
		  inorde(n_9)
		  print(n_9)
		  print(n_6)

Tabel 4.3: Trace van de oproepen wanneer men de binaire boom in Figuur 4.5 in pre-, post- en inorde doorloopt.

Bij het in inorde doorlopen zullen de labels als volgt afgedrukt worden:

$$n_4, n_2, n_7, n_5, n_8, n_1, n_3, n_9, n_6.$$ ∎

4.4.4 Oefeningen

1. a) Teken de binaire boom met labels 0 t.e.m. 9 waarvoor de inorde sequentie

   $$9, 3, 1, 0, 4, 2, 7, 6, 8, 5$$

   is terwijl de postorde sequentie

   $$9, 1, 4, 0, 3, 6, 7, 5, 8, 2$$

   is.

   b) Doe nu hetzelfde voor de volgende sequenties, of leg uit waarom zo'n binaire boom niet bestaat:

   inorde: $9, 3, 1, 0, 4, 2, 7, 6, 8, 5$

en

postorde: $9, 1, 4, 0, 3, 6, 5, 7, 8, 2$

4.5 Binaire zoekbomen

Een vaak voorkomende activiteit in een computerprogramma is het bijhouden van een verzameling waarden waarbij we wensen:

1. gegevens toe te voegen aan de verzameling,
2. gegevens te verwijderen uit de verzameling, en
3. te controleren of een element aanwezig is in de verzameling.

Er zijn verschillende gegevensstructuren die kunnen gebruikt worden om deze functionaliteit te implementeren, zoals bv. (lineair gelinkte) lijsten en hash-tabellen.

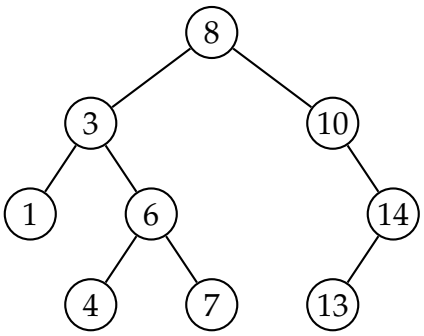
Een andere datastructuur die deze bewerkingen kan implementeren is de binaire zoekboom. Een noodzakelijke voorwaarde om een binaire zoekboom te kunnen gebruiken is dat de labels een *totaal geordende verzameling* vormen. Dit wil zeggen dat er voor elk paar (mogelijke) labels x en y geldt dat

$$x < y \quad \text{of} \quad y < x \quad \text{of} \quad x = y.$$

Anders gezegd: wanneer x en y verschillend zijn dan kunnen we zeggen welk element het kleinste is; er zijn geen onvergelijkbare elementen. De labels zouden bv. gehele getallen kunnen zijn, of karakterreeksen (strings) die lexicografisch (i.e. alfabetisch) vergeleken worden. De labels van een binaire zoekboom worden dikwijls SLEUTELS genoemd.

Definitie 4.17 Een BINAIRE ZOEKBOOM is een gelabelde binaire boom die aan een bijzondere voorwaarde, de *binaire zoekboomeigenschap*, voldoet. ∎

Definitie 4.18 De BINAIRE ZOEKBOOMEIGENSCHAP is de volgende: voor *elke* top x van de binaire zoekboom geldt dat *alle* toppen in de linkerdeelboom van x een label hebben dat kleiner is dan het label van x, terwijl voor *alle* toppen in de rechterdeelboom van x geldt dat hun label groter is dan het label van x. ∎



Figuur 4.9: Een binaire zoekboom met 9 toppen.

Voorbeeld 4.19 In Figuur 4.9 zien we een binaire boom waarvan de labels gehele getallen zijn. We controleren dat de binaire zoekboomeigenschap voldaan is. Hiertoe moeten we voor elke top nagaan dat alle labels in de linkerdeelboom (resp. rechterdeelboom) van die top kleiner (resp. groter) zijn dan het label van de top. We controleren dit m.b.v. onderstaande tabel:

sleutel top	sleutels links	sleutels rechts
8	$\{1,3,4,6,7\}$	$\{10,13,14\}$
3	$\{1\}$	$\{4,6,7\}$
1	$\emptyset$	$\emptyset$
6	$\{4\}$	$\{7\}$
4	$\emptyset$	$\emptyset$
7	$\emptyset$	$\emptyset$
10	$\emptyset$	$\{13,14\}$
14	$\{13\}$	$\emptyset$
13	$\emptyset$	$\emptyset$

Hieruit blijkt dat de binaire zoekboomeigenschap voor deze boom voldaan is. ∎

Opmerking 4.20 Wanneer we de boom in Figuur 4.9 in inorde doorlopen dan worden de toppen in de volgende volgorde bezocht:

$$1,3,4,6,7,8,10,13,14.$$

Merk op dat deze rij gesorteerd is van klein naar groot. Een binaire zoekboom houdt zijn waarden dus ook *gesorteerd* bij, al is dat op het eerste zicht misschien niet zo duidelijk als bij een gesorteerde array. ∎

4.5.1 Opzoeken van een sleutel in een binaire zoekboom

Wanneer we een sleutel x willen opzoeken in een binaire zoekboom dan kunnen we gebruik maken van de binaire zoekboomeigenschap om (snel) uit te maken of x aanwezig is of niet.

Inderdaad, wanneer de binaire zoekboom T leeg is, dan is x uiteraard niet aanwezig. Wanneer T niet leeg is, en x is kleiner dan het label van de wortel van T, dan moet x (indien aanwezig in T) zich in de linkerdeelboom van de wortel bevinden. Analoog moet x zich in de rechterdeelboom van de wortel bevinden wanneer x groter is dan de sleutel van de wortel. In het laatste geval (wanneer x gelijk is aan het label van de wortel) bevindt x zich in de wortel van de boom.

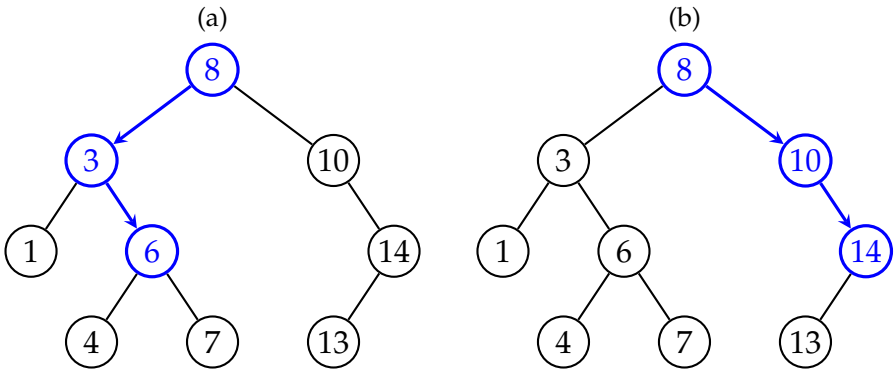
Opmerking 4.21 Merk op dat dit zoekproces lijkt op de manier waarop er binair gezocht wordt in een gesorteerde array. Alleen is het zo dat bij een binaire zoekboom het aantal te onderzoeken waarden niet steeds exact in twee wordt verdeeld zoals dat bij binair zoeken wel het geval is. ∎

Om dus (recursief) te zoeken naar een bepaalde waarde x in een binaire zoekboom T gaat men als volgt tewerk:

1. Wanneer de boom leeg is, geef dan 'niet gevonden' terug.
2. Vergelijk x met de sleutel van de wortel.
    a) Wanneer x kleiner is dan dit label, zoek dan (recursief) in de linkerdeelboom.
    b) Wanneer x groter is dan dit label, zoek dan (recursief) in de rechterdeelboom.
    c) Geef de wortel van de boom terug (x werd gevonden).

Voorbeeld 4.22 Veronderstel dat we op zoek gaan naar de sleutel 6 in de voorbeeldboom van Figuur 4.9.

We starten aan de wortel van de boom, en we zien dat de boom niet leeg is. We vergelijken 6 met de sleutel van de wortel (dit is 8) en we zien dat 6 kleiner is dan 8. We gaan nu verder in de linkerdeelboom met als wortel de top met label 3. De boom is dus opnieuw niet leeg, en we vergelijken 6 met het label 3 van deze top. We zien dat 6 groter is dan 3, en we gaan recursief



Figuur 4.10: In (a) wordt succesvol gezocht naar de sleutel 6 in de binaire zoekboom. In (b) zoeken we tevergeefs naar de sleutel 15.

verder in de rechterdeelboom. Wanneer we nu 6 vergelijken met de sleutel van de wortel van deze deelboom, dan zien we dat we de gewenste top gevonden hebben. ∎

Opmerking 4.23 Om de top 6 te vinden hebben we drie vergelijkingen uitgevoerd; dit is één meer dan de diepte van deze top. ∎

Voorbeeld 4.24 Veronderstel dat we op zoek gaan naar het element met de waarde 15 in de voorbeeldboom van Figuur 4.9.

We starten opnieuw aan de wortel van de boom, en we vergelijken 15 met 8 (de sleutel van de wortel). Aangezien 15 groter is, gaan we recursief verder in de rechterdeelboom. We vergelijken vervolgens 15 met het label 10, en we gaan opnieuw recursief verder in de rechterdeelboom. We zoeken dus opnieuw recursief in de rechterdeelboom, en we vergelijken het label 14 met 15. Aangezien 15 groter is gaan we opnieuw recursief verder in de rechterdeelboom. We zien nu dat deze boom leeg is, en dus kunnen we besluiten dat het element met de waarde 15 niet tot de boom behoort. ∎

Soms kan het ook nuttig zijn om de kleinste of grootste sleutel van een boom te kennen. Door de binaire zoekboomeigenschap weten we dat het kleinste element van een binaire zoekboom steeds tot de linkerdeelboom van de wortel behoort (wanneer die niet leeg is). Om het kleinste element van een (niet-lege) binaire zoekboom op te zoeken gaan we dus als volgt tewerk:

1. Wanneer de linkerdeelboom van de wortel leeg is, geef dan (de sleutel van) de wortel terug.

2. In het andere geval zoek je recursief naar het kleinste element van de linkerdeelboom.

Om het grootste element te vinden gaan we natuurlijk analoog tewerk.

Voorbeeld 4.25 We zoeken het kleinste element van de binaire zoekboom in Figuur 4.9.

Aangezien de linkerdeelboom van de wortel niet leeg is, gaan we recursief op zoek naar het kleinste element in de deelboom met als wortel de top 3. Hier is de linkerdeelboom opnieuw niet leeg, dus zoeken we naar het kleinste element van de deelboom met als wortel de top 1. Nu is de linkerdeelboom van de top met als label 1 leeg, en dus is het kleinste element van de binaire zoekboom gelijk aan 1. ∎

4.5.2 Toevoegen van een sleutel aan een binaire zoekboom

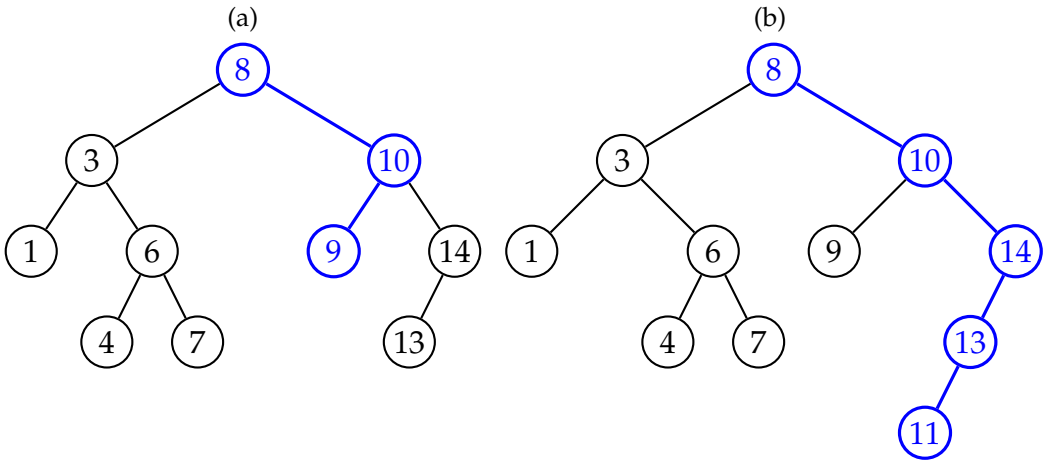
Wanneer we een sleutel x toevoegen aan een binaire zoekboom T, dan is het van belang dat de nieuwe boom T' eveneens een binaire zoekboom is, m.a.w. dat de nieuwe boom ook aan de binaire zoekboomeigenschap voldoet.

Toevoegen van een element aan een binaire zoekboom is eenvoudig en kan recursief worden geformuleerd. Wanneer we x wensen toe te voegen aan een lege boom T, dan vervangen we T eenvoudigweg door een boom bestaande uit één top met x als zijn sleutel. Wanneer de boom T niet leeg is en x bevat in de wortel dan moeten we niets doen (want x behoort reeds tot de boom). In het andere geval voegen we x toe aan de linker- of rechterdeelboom al naargelang het label van x kleiner of groter is dan het label van de wortel.

We kunnen het toevoegen van een sleutel x aan een (niet-lege) binaire zoekboom dan ook als volgt recursief formuleren:

1. Vergelijk x met het label van de wortel.

   a) Wanneer x kleiner is dan het label van de wortel, voeg dan x toe aan de linkerdeelboom wanneer die niet leeg is. Wanneer de lin-



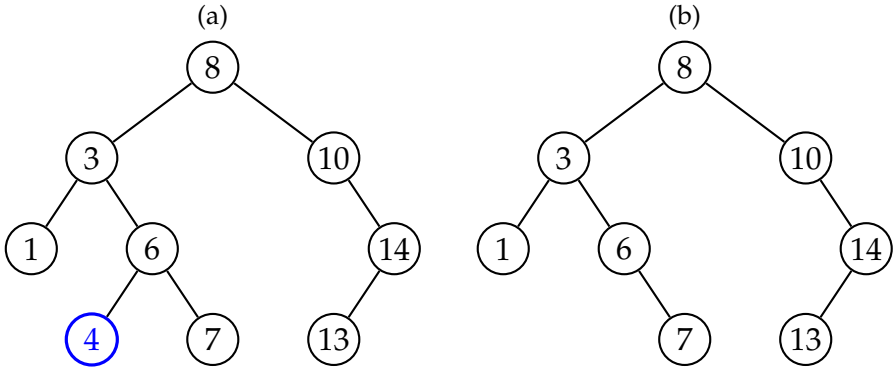
Figuur 4.11: In (a) zie je de resulterende boom na toevoegen van de sleutel 9 aan de boom in Figuur 4.9. In (b) wordt dan nog eens de sleutel 11 toegevoegd.

kerdeelboom leeg is vervang dan de (`null`)-referentie naar de linkerdeelboom door de referentie naar een nieuwe top met x als label.

b) Wanneer x groter is dan het label van de wortel, voeg dan x toe aan de rechterdeelboom wanneer die niet leeg is. Wanneer de rechterdeelboom leeg is vervang dan de (`null`)-referentie naar de rechterdeelboom door de referentie naar een nieuwe top met x als label.

c) Doe niets, want x behoort reeds tot de boom.

Voorbeeld 4.26 Veronderstel dat we de sleutel 9 willen toevoegen aan de binaire zoekboom in Figuur 4.9. We starten in de wortel en zien dat 9 groter is dan 8, het label van de wortel. We gaan dus verder in de rechterdeelboom. Aangezien 10 (het label van de wortel van de rechterdeelboom) groter is dan 9, moet het nieuwe element aan de linkerdeelboom van 10 toegevoegd worden. Aangezien deze linkerdeelboom leeg is, hebben we de juiste plaats waar 9 moet worden toegevoegd gevonden. We creëren een nieuwe top en maken deze de wortel van de linkerdeelboom van de top met als label 10. Je ziet de resulterende boom in Figuur 4.11(a). ∎



Figuur 4.12: Links zie je de originele zoekboom. Rechts zie je de binaire zoekboom na het verwijderen van de sleutel 4. De sleutel 4 bevindt zich in een blad en dus is verwijderen eenvoudig.

4.5.3 Verwijderen van een sleutel uit een binaire zoekboom

Het verwijderen van een sleutel uit een binaire zoekboom is iets moeilijker dan het opzoeken of het toevoegen van een element.

Het verwijderen van een sleutel x start met het opzoeken van deze sleutel in de boom. Er kunnen zich nu drie gevallen voordien:

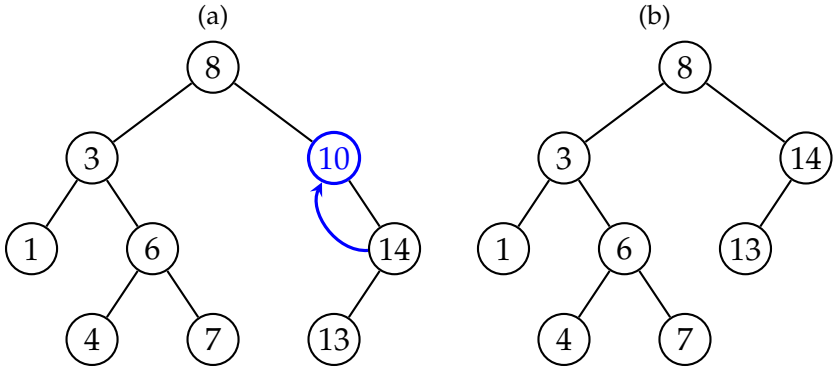
1. De sleutel x bevindt zich in een blad.
2. De sleutel x bevindt zich in een top met één kind.
3. De sleutel x bevindt zich in een top met twee kinderen.

We bekijken nu deze gevallen één voor één.

Sleutel bevindt zich in een blad
Wanneer x zich in een blad bevindt, dan kunnen we eenvoudigweg dit blad verwijderen.

Voorbeeld 4.27 In Figuur 4.12 zie je de resulterende zoekboom wanneer het element 4 uit de boom wordt verwijderd. Aangezien de top die het element 4 bevat een blad is, kan deze top zonder problemen uit de boom worden verwijderd. ∎

Sleutel bevindt zich in een top met één kind
Wanneer de top n die x bevat slechts één kind heeft, dan kunnen we n ver-



Figuur 4.13: Binaire zoekboom na het verwijderen van het element 10 uit de boom in Figuur 4.9.

vangen door dit ene kind. Anders gezegd: het nieuwe linker- of rechterkind van de ouder van n is zijn vroegere kleinkind[2]. De binaire zoekboomeigenschap zal in dit geval nog steeds geldig zijn.

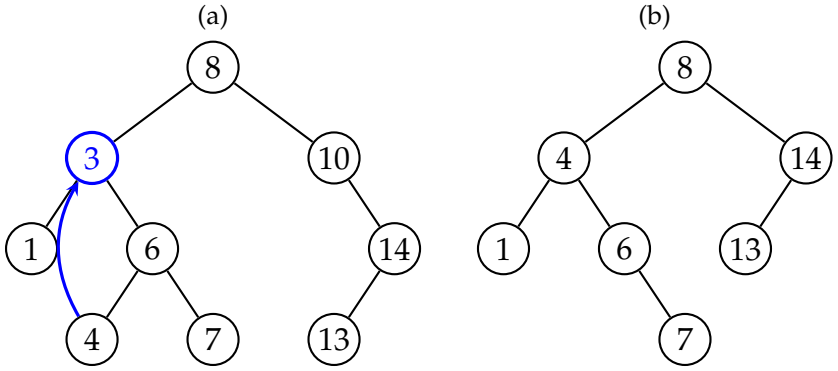
Voorbeeld 4.28 Wanneer we de sleutel 10 wensen te verwijderen uit de boom in Figuur 4.9 dan zien we dat 10 slechts één kind heeft (de lege linkerdeelboom wordt niet geteld als een kind). Dit betekent dat we in de boom de top met label 10 eenvoudigweg kunnen vervangen door de top met label 14. Dit komt neer op het vervangen van de rechterwijzer in de top met label 8. De resulterende boom is getoond in Figuur 4.13(b). ∎

Sleutel bevindt zich in een top met twee kinderen

Wanneer tenslotte de top n die x bevat twee kinderen heeft dan kunnen we eerst x vervangen door het kleinste element y van de rechterdeelboom. Dit element is de *opvolger* van x. In een volgende stap kunnen we dit element y gaan verwijderen uit de rechterdeelboom. Merk op dat de top m die dit element y bevat hoogstens één kind kan hebben. In het bijzonder moet de linkerdeelboom van m de lege boom zijn. Dit betekent dat de top m eenvoudig te verwijderen is. We hebben het probleem dus gereduceerd tot het verwijderen van een top met hoogstens één kind.

Voorbeeld 4.29 Wanneer we de sleutel 3 wensen te verwijderen uit de boom, dan zien we dat de top 3 twee kinderen heeft. We gaan dus eerst op zoek naar het kleinste element groter dan 3. Dit element bevindt zich

[2]Wanneer n de wortel van de boom is, dan krijgt de boom een nieuwe wortel.



Figuur 4.14: Binaire zoekboom na het verwijderen van het element 3 uit de boom in Figuur 4.9.

in de rechterdeelboom van 3 en zal steeds hoogstens 1 kind hebben, aangezien de linkerdeelboom leeg moet zijn. In dit geval is de opvolger van 3 de sleutel 4. We vervangen dus eerst in de boom het label 3 door het label 4. Vervolgens verwijderen we de top die 4 bevatte. De resulterende boom zie je in Figuur 4.14(b). ∎

4.5.4 Tijdscomplexiteit van de bewerkingen

Het is intuïtief duidelijk dat alle bewerkingen op een binaire zoekboom in het slechtste geval een uitvoeringstijd hebben die evenredig is met de diepte van de boom, i.e. de uitvoeringstijd van de bewerkingen is in het slechtste geval $\Theta(d)$.

Wanneer de boom n toppen bevat, weten we uit Eigenschap 4.13 dat

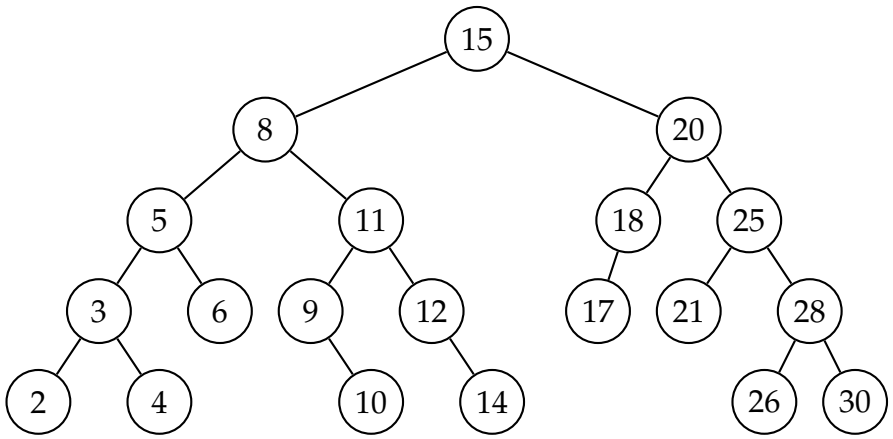
$$n \leq 2^{d+1} - 1.$$

Door van beide leden de logaritme met grondtal twee te nemen zien we dat voor een totaal gebalanceerde boom geldt dat $\Theta(d) = \Theta(\lg(n))$. Voor een totaal gebalanceerde boom hebben alle bewerkingen dus een tijdscomplexiteit $\Theta(\lg(n))$, met n het aantal toppen in de zoekboom.

Echter, uit Eigenschap 4.13 zien we ook dat voor een totaal ongebalanceerde boom

$$d + 1 \leq n,$$

zodat $\Theta(d) = \Theta(n)$.



Figuur 4.15: Een binaire zoekboom.

Men kan aantonen dat in het gemiddeld geval (wanneer elementen random worden toegevoegd aan een boom) de resulterende boom een diepte zal hebben die $\Theta(\lg(n))$ is.

4.5.5 Oefeningen

1. Geef de binaire zoekboom die opgebouwd wordt door de volgende sleutels

   $$4, 7, 5, 8, 11, 3, 2, 9, 10, 6$$

   één voor één aan de zoekboom toe te voegen in de gegeven volgorde.

2. Veronderstel dat men een binaire zoekboom opbouwt door sleutels één voor één toe te voegen aan een initieel lege boom.

   a) Geef een rij van lengte 7 die een binaire zoekboom van minimale diepte oplevert.

   b) Geef een rij van lengte 15 die een binaire zoekboom van minimale diepte oplevert.

   c) Geef een rij van lengte 5 die een binaire zoekboom van maximale diepte oplevert. Wanneer krijg je een over het algemeen een slecht gebalanceerde binaire zoekboom?

3. Beschouw de binaire zoekboom in Figuur 4.15. Voer de volgende opdrachten één na één uit.

a) Welke toppen worden er bezocht bij het zoeken naar de top 12?

b) Welke toppen worden er bezocht bij het zoeken naar de top 27?

c) Voeg een top met sleutelwaarde 23 toe aan de zoekboom. Teken de resulterende zoekboom.

d) Voeg vervolgens een top met sleutelwaarde 22 toe aan de zoekboom. Teken de resulterende zoekboom.

e) Verwijder de top met waarde 4 uit de zoekboom. Teken de resulterende zoekboom.

f) Verwijder vervolgens de top met waarde 18 uit de zoekboom. Teken de resulterende zoekboom.

g) Verwijder vervolgens de top met waarde 20 uit de zoekboom. Teken de resulterende zoekboom.

4.6 Binaire hopen

4.6.1 Prioriteitswachtrij

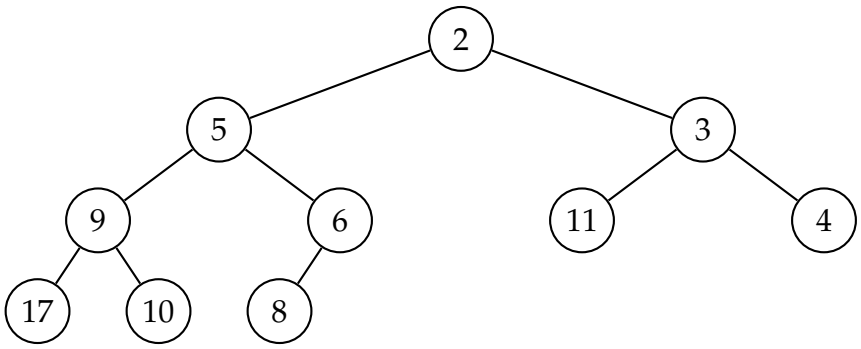
Binaire zoekbomen worden vaak gebruikt als implementatie van het abstract datatype "Verzameling". Een ander abstract datatype dat vaak wordt gebruikt in toepassingen is een PRIORITEITSWACHTRIJ. Dit is een uitbreiding van de "gewone" FIFO wachtrij.

De elementen van een wachtrij bestaan typisch uit twee delen: een sleutel en een waarde. De sleutel geeft in dit geval de prioriteit aan. Meestal is het zo dat een kleinere sleutel wijst op een grotere prioriteit.

Bij een prioriteitswachtrij kan men

1. het element met de kleinste sleutel opzoeken;
2. het element met de kleinste sleutel verwijderen;
3. een nieuw element toevoegen aan de wachtrij.

Opmerking 4.30 In een prioriteitswachtrij kan *enkel* het element met de kleinste sleutel efficiënt bereikt worden. Dit betekent dat een prioriteitswachtrij flexibel is wat betreft het toevoegen van elementen maar niet wat betreft het verwijderen. ∎



Figuur 4.16: Een binaire hoop.

4.6.2 Implementatie als binaire hoop

We bekijken nu hoe we een prioriteitswachtrij kunnen implementeren als een binaire hoop. In essentie is een binaire hoop een complete binaire boom die aan een extra ordeningseigenschap voldoet.

Definitie 4.31 Een COMPLETE BINAIRE BOOM is een binaire boom van diepte d waarbij het aantal toppen met diepte $k < d$ maximaal (dus 2^k, zie Eigenschap 4.12) is. De toppen met diepte d komen voor van "links naar rechts". ∎

Voorbeeld 4.32 In Figuur 4.16 ziet men een voorbeeld van een complete binaire boom. De niveaus met diepte 0, 1 en 2 zijn volledig gevuld. Het diepste niveau, met toppen van diepte 3, is gevuld van links naar rechts. ∎

Definitie 4.33 De ORDENINGSEIGENSCHAP VOOR BINAIRE HOPEN zegt dat de sleutel van *elke* top hoogstens gelijk is aan de kleinste sleutel van zijn kinderen. ∎

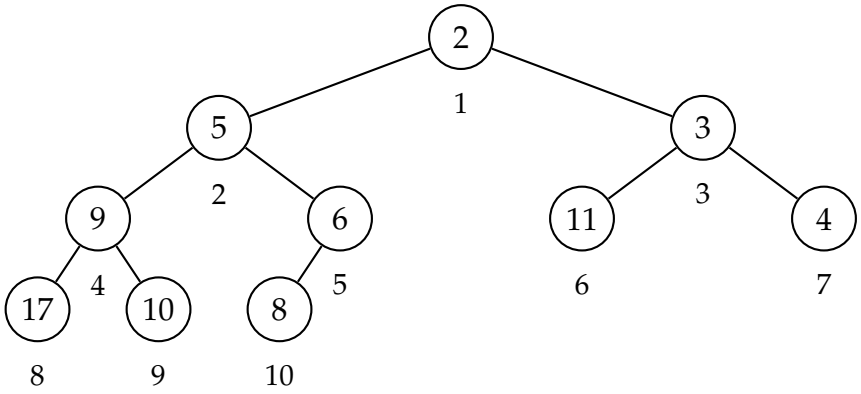
Voorbeeld 4.34 Men gaat gemakkelijk na dat de complete binaire boom in Figuur 4.16 aan de ordeningseigenschap voor binaire hopen voldoet. Dit zien we eenvoudig uit Tabel 4.4. ∎

4.6.3 Implementatie

Het is mogelijk om een binaire hoop te implementeren als een binaire boom, waarbij men voor elke top bijhoudt wat zijn linker- en rechterkinderen zijn.

Sleutel ouder	Sleutels kinderen
2	{5, 3}
5	{9, 6}
3	{11, 4}
9	{17, 10}
6	{8}
11	∅
4	∅
17	∅
10	∅
8	∅

Tabel 4.4: Voor elke top uit de boom in Figuur 4.16 wordt zijn sleutel en de sleutel van zijn kinderen gegeven. De sleutel van de ouder is steeds kleiner dan de sleutel van zijn kinderen.



Figuur 4.17: De binaire hoop uit Figuur 4.16 maar nu wordt voor elke top zijn rangnummer aangegeven.

Omdat de binaire boom echter steeds compleet is bestaat er een eenvoudigere (en snellere) manier die een gewone array gebruikt.

Veronderstel dat we de toppen van een complete binaire boom gaan nummeren op de volgende manier. De wortel van de boom heeft rangnummer 1. De toppen op diepte 1 worden consecutief genummerd van links naar rechts (en krijgen dus rangnummers 2 en 3). Daarna worden de toppen op diepte 2 genummerd (startend vanaf 4) van links naar rechts, enzovoort.

Voorbeeld 4.35 In Figuur 4.17 wordt de binaire hoop uit Figuur 4.16 her-

haald, maar nu wordt voor elke top zijn rangnummer aangegeven. ∎

Er is een eenvoudig verband tussen het rangnummer van een top en het rangnummer van zijn kinderen. Dit verband wordt gegeven in de volgende eigenschap.

Eigenschap 4.36 Wanneer een top rangnummer i heeft, dan hebben zijn linker- en rechterkind (als die bestaan) respectievelijk rangnummer $2i$ en $2i + 1$.

Omgekeerd geldt: wanneer een top rangnummer i heeft (en deze top is niet de wortel van de boom), dan heeft zijn ouder rangummer $\text{floor}(i/2)$. We kunnen dus stellen dat:

$$\begin{aligned} \text{left}(i) &= 2i, \\ \text{right}(i) &= 2i + 1, \\ \text{parent}(i) &= \text{floor}(i/2). \end{aligned}$$

∎

Dit betekent dat we een binaire hoop kunnen opslaan in een array, waarbij de ouder-kind relaties afgeleid worden uit Eigenschap 4.36.

Voorbeeld 4.37 Hieronder ziet men hoe de voorgestelde nummering eruit ziet voor de binaire hoop in Figuur 4.16.

Index	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
Sleutel		2	5	3	9	6	11	4	17	10	8				

Merk op dat positie 0 niet gebruikt wordt. De posities 11 t.e.m. 14 zijn nog beschikbaar voor nieuwe elementen. ∎

4.6.4 Opzoeken van het element met de kleinste sleutel

Uit de ordeningseigenschap voor binaire hopen volgt dat het element met de kleinste sleutel steeds de wortel van de boom is (want elke andere top heeft een ouder waarvoor de sleutelwaarde kleiner is dan de sleutelwaarde van die top). Het opzoeken van het element met de kleinste sleutel is dus een triviale bewerking, die de binaire hoop ongewijzigd laat. Deze bewerking kan uiteraard steeds uitgevoerd worden in constante tijd.

4.6.5 Toevoegen van een element

Om een nieuw element toe te voegen aan de binaire hoop gaan we als volgt tewerk:

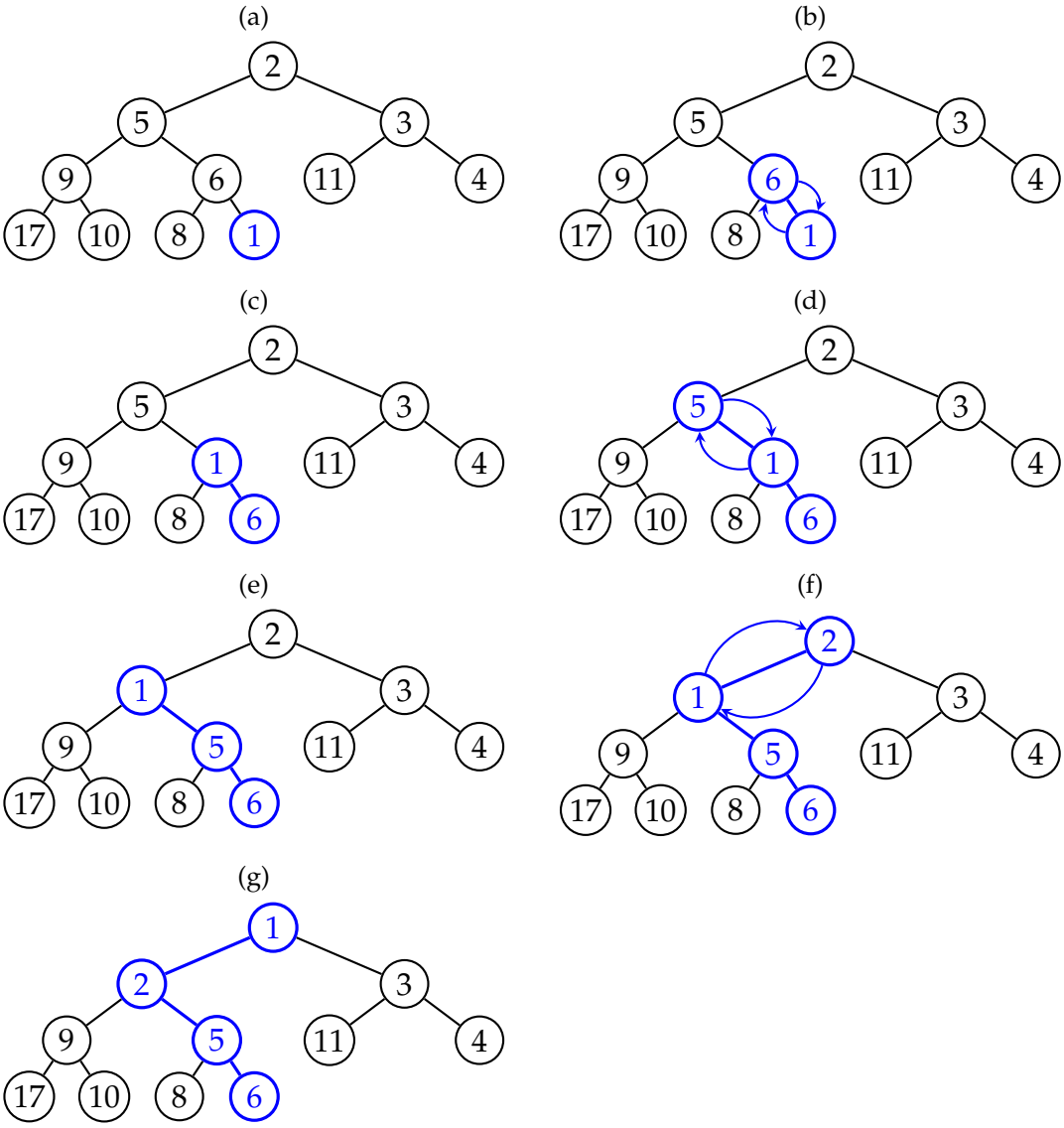
1. Creëer een nieuw element.
2. Voeg dit element toe op de eerste beschikbare plaats. Dit betekent dus als een nieuw blad, met rangnummer i, waarbij we er steeds voor zorgen dat het diepste niveau gevuld is van links naar rechts.
3. Op dit moment is het in het algemeen zo dat de ordeningseigenschap voor binaire hopen nu kan geschonden zijn tussen i en parent(i). Indien dit zo is, verwissel dan i en parent(i). Dit herstelt de ordeningseigenschap tussen i en zijn ouder. Eventueel is nu de ordeningseigenschap tussen parent(i) en parent(parent(i)) geschonden. Indien dit zo is wissel dan beide elementen. Ga zo verder tot de binaire hoop is hersteld.

Opmerking 4.38 Het proces van het herstellen van de binaire hoop van beneden naar boven, noemt men soms ook OMHOOG BUBBELEN, omdat het nieuwe element als het ware omhoog bubbelt in de hoop tot het op zijn correcte plaats staat. ∎

Voorbeeld 4.39 In Figuur 4.18 zien we hoe we tewerk gaan om het element met sleutel 1 toe te voegen.

We voegen een nieuwe top toe met sleutel 1 op de plaats waar het volgende blad moet komen. Op dit moment voldoet de binaire boom niet meer aan de ordeningseigenschap voor binaire hopen: de sleutel 1 is kleiner dan de sleutel van zijn ouder (sleutelwaarde 6). Merk ook op dat dit de *enige* plaats is waar de ordeningseigenschap van binaire hopen geschonden is. We kunnen dit herstellen door de sleutelwaarden 1 en 6 van plaats te verwisselen.

Op dit moment hebben we nog steeds geen binaire hoop, aangezien 1 kleiner is dan 5. Merk opnieuw op dat dit de enige plaats is in de binaire hoop waar de binaire ordeningseigenschap geschonden is. We wisselen 1 en 5 teneinde deze schending ongedaan te maken. Merk op dat in de nieuwe boom de top met sleutel 5 aan de ordeningseigenshap voldoet want 6 en 8 waren afstammelingen van 5 in de oorspronkelijke binaire hoop.



Figuur 4.18: Toevoegen van sleutel 1 aan een binaire hoop.

De binaire boom voldoet nog steeds niet aan de ordeningseigenschap, want 1 is kleiner dan de sleutel van zijn ouder die sleutelwaarde 2 heeft. Wisselen lost dit op, en nu hebben we terug een correcte binaire hoop. Merk op dat 2 kleiner is dan zijn kinderen: dit zal steeds zo zijn aangezien 9 en 5 afstammelingen waren van 2 in de oorspronkelijke binaire hoop.

In dit geval hebben we dus 3 verwisselingen moeten doorvoeren om de binaire hoop te herstellen. Dit is in deze binaire hoop ook het maximaal aantal verwisselingen dat nodig kan zijn. ∎

4.6.6 Verwijderen van het element met de kleinste sleutel

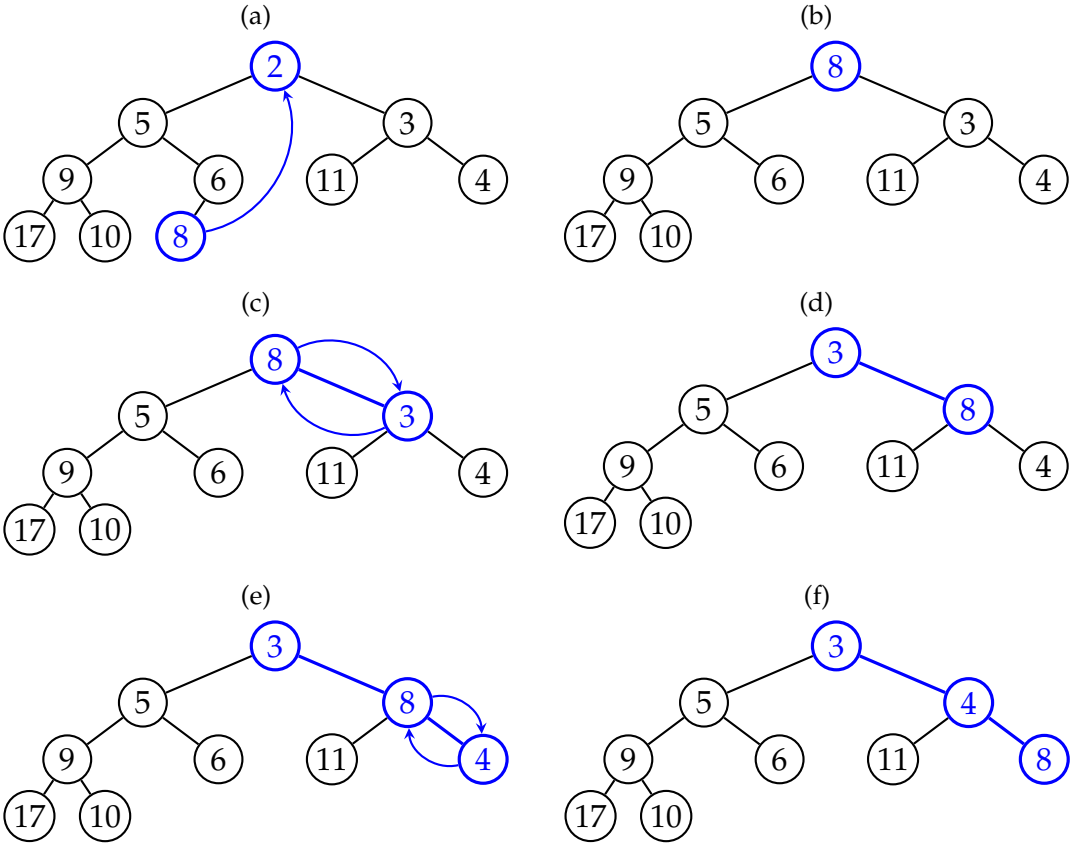
Wanneer we het element met de kleinste sleutel verwijderen, dan krijgen we een "gat" aan de wortel van de boom. We vullen dit gat op door het meest rechtse blad op het diepste niveau in dit "gat" te duwen (en het meest rechtse blad op het diepste niveau te verwijderen). Op dit moment is wellicht de ordeningseigenschap geschonden tussen de wortel en zijn kinderen. Herstel dit door de wortel om te wisselen met zijn *kleinste* kind. Ga zo verder tot de binaire hoop hersteld is.

Verwijderen van een element verloopt dus als volgt:

1. Verwissel de wortel met het meest rechtse blad met de grootste diepte.
2. Verwijder het meest rechtse blad; de binaire hoop heeft nu een element minder.
3. Indien de ordeningseigenschap geschonden is in de wortel, herstel deze dan door de wortel en zijn kleinste kind i van plaats te verwisselen. Indien de ordeningseigenschap nu geschonden is in i, herstel ze dan door i te verwisselen met de kleinste van zijn kinderen. Ga zo verder tot de binaire hoop hersteld is.

Opmerking 4.40 Het proces dat gebruikt wordt bij het verwijderen van een element noemt men soms ook OMLAAG BUBBELEN, omdat het laatste element van de hoop omlaag bubbelt doorheen de hoop tot het op zijn juiste plaats staat. ∎

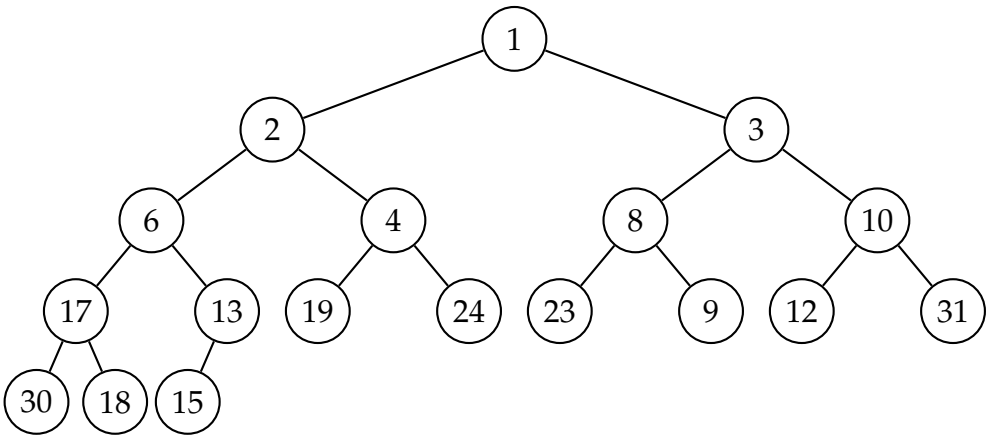
Voorbeeld 4.41 In Figuur 4.19 zien we hoe het element met de kleinste sleutelwaarde (2 in dit geval) uit de binaire hoop wordt verwijderd.



Figuur 4.19: Illustratie van de stappen die gezet worden bij het verwijderen van het kleinste element uit een binaire hoop.

In de eerste stap gaan we het "gat" dat achtergelaten wordt door 2, opvullen met de sleutelwaarde van het meest rechtste blad (8 in dit geval). Daarna wordt dit blad verwijderd. Op die manier hebben we nog steeds een complete binaire boom, maar de ordeningseigenschap is nu geschonden in de wortel want $8 > \min(5, 3)$. We wisselen dus 8 en 3 om deze schending van de ordeningseigenschap te herstellen.

Nu hebben we nog steeds geen binaire hoop, want de top met sleutelwaarde 8 voldoet nog steeds niet aan de ordeningseigenschap: $8 > \min(11, 4)$. De oplossing bestaat erin om de sleutelwaarden 8 en 4 van plaats te wisselen. Na deze verwisseling voldoet de binaire boom aan de ordeningseigenschap voor binaire hopen en is dus opnieuw een geldige binaire hoop. ∎



Figuur 4.20: Een binaire hoop.

4.6.7 Tijdscomplexiteit van de bewerkingen

Een binaire hoop met n toppen heeft diepte $\Theta\big(\lg(n)\big)$. Dit betekent onmiddellijk dat de tijdscomplexiteit van de bewerkingen toevoegen van een element en verwijderen van het kleinste element in het slechtste geval $\Theta\big(\lg(n)\big)$ is.

Het ophalen van het kleinste element is uiteraard een bewerking die in constante tijd kan uitgevoerd worden: $T(n) = \Theta(1)$.

4.6.8 Oefeningen

1. Start met een lege binaire hoop. Voeg achtereenvolgens de volgende elementen toe aan de binaire hoop:

   $$11, 13, 1, 15, 6, 5, 9, 16, 3, 10, 7, 4, 12, 14, 2.$$

   Teken de resulterende hoop na elke toevoeging.

2. Beschouw de binaire hoop in Figuur 4.20. Verwijder de drie kleinste elementen uit deze binaire hoop. Teken de binaire hoop na elke verwijdering.

3. Wat worden de relaties in Eigenschap 4.36 wanneer een binaire hoop wordt opgeslaan in een array met als eerste index 0?

4. Waar kan het maximale element zich bevinden in een binaire hoop, aannemende dat de binaire hoop bestaat uit verschillende elementen.

HOOFDSTUK 5

Graafalgoritmes

We starten dit hoofdstuk met de definitie van en begrippen omtrent GRAFEN. Daarna volgen, net zoals in het hoofdstuk m.b.t. bomen, twee DATASTRUCTUREN om grafen voor te stellen in het computergeheugen. Vervolgens bekijken we verschillende manieren om te ZOEKEN in een graaf. Het DIEPTE-EERST zoekproces vormt de basis voor een efficiënt algoritme voor TOPOLOGISCH SORTEREN, i.e. het plaatsen van taken in een volgorde zodanig dat de afhankelijkheden voldaan zijn. Voor het vinden van het KORTSTE PAD tussen twee knopen zien we voor gewogen grafen het ALGORITME VAN DIJKSTRA. Vervolgens zien we twee algoritmen voor het vinden van een MINIMALE KOST OPSPANNENDE BOOM, nl. de algoritmes van PRIM en KRUSKAL. Om af te sluiten bespreken we kort het HANDELSREIZIGERSPROBLEEM waarvoor er waarschijnlijk geen efficiënt algoritme bestaat. We zien één benaderingsalgoritme gebaseerd op minimale kost opspannende bomen.

5.1 Terminologie m.b.t. grafen

In heel wat praktische situaties heeft men te maken met de situatie waarin 'objecten' verbonden zijn door een bepaalde relatie: steden zijn met elkaar verbonden m.b.v. wegen, computers zijn verbonden m.b.v. netwerkkabels, luchthavens zijn met elkaar verbonden door directe vluchten. In al de opgesomde situaties is het vaak zinvol om een bepaalde 'kost' te associëren met een verbinding: de afstand in kilometer, de communicatiesnelheid van de verbinding, of de duur van de rechtstreekse vlucht. Al deze situaties kunnen gemodelleerd worden aan de hand van een graaf. In dit hoofdstuk be-

studeren we grafen, samen met enkele fundamentele algoritmen voor grafen.

Definitie 5.1 Een GRAAF G bestaat uit een verzameling KNOPEN V, en een verzameling BOGEN E. Elke boog verbindt twee knopen, en we noteren $e = (v, w)$. De graaf G wordt genoteerd als het koppel (V, E), dus $G = (V, E)$. ∎

Wanneer v en w verbonden zijn door een boog, i.e. wanneer $(v, w) \in E$, dan zijn v en w ADJACENT. We zeggen dan ook dat de knopen v en w INCIDENT zijn met e, en omgekeerd ook dat e incident is met v en w. De BUREN van een knoop v zijn alle knopen w die adjacent zijn met v, en we schrijven buren(v) om deze verzameling aan te duiden. Het aantal buren van een knoop v noemt men de GRAAD van deze knoop.

Het aantal knopen van de graaf, i.e. $\#(V)$, wordt de ORDE van de graaf genoemd, en wordt vaak als n genoteerd. Het aantal bogen, i.e. $\#(E)$, noemt men de GROOTTE van de graaf en wordt traditioneel aangeduid met m.

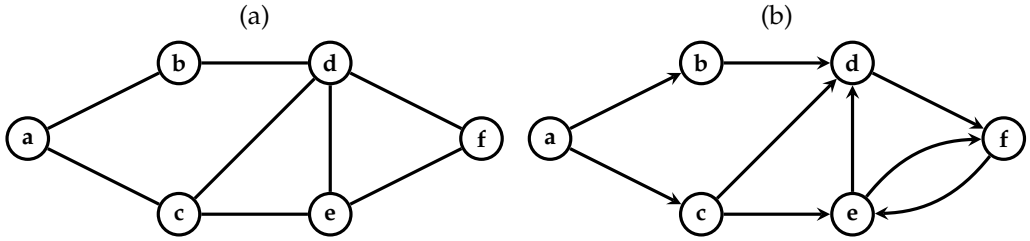
Opmerking 5.2 In plaats van "knoop" zullen we in deze tekst soms ook het synoniem "top" gebruiken om een element van V te benoemen. ∎

Opmerking 5.3 Wanneer de boogparen niet geordend zijn dan spreekt men van een ONGERICHTE graaf. Wanneer de boogparen wel geordend zijn dan heeft men een GERICHTE graaf. In een gerichte graaf heeft een boog (v, w) een STAART v en een KOP w. ∎

Voorbeeld 5.4 Een voorbeeld van een ongerichte graaf is de vriendschapsgraaf van Facebook. De knopen van deze graaf zijn de personen met een account op Facebook. Twee knopen zijn adjacent als de personen "vriend" zijn van elkaar op Facebook. Als v "vriend" is van w, dan is omgekeerd ook w "vriend" van v. De vriendschapgraaf van Facebook is dus een ongerichte graaf. ∎

Voorbeeld 5.5 De volgersgraaf van Twitter is een gerichte graaf. De knopen van deze graaf zijn de accounts op Twitter, en er is een boog van v naar w indien v het account w volgt. ∎

Grafen worden grafisch voorgesteld op een manier analoog aan bomen. De knopen van de graaf worden voorgesteld door een bolletje, eventueel voorzien van een label. Wanneer twee knopen adjacent zijn, dan worden de twee



Figuur 5.1: Een voorbeeld van een ongerichte en gerichte graaf.

knopen verbonden door een lijn: dit stelt de boog voor. Wanneer de boog geordend is (m.a.w. wanneer men te maken heeft met een gerichte graaf), dan wordt er een pijl toevoegd die de richting van de boog aangeeft. De pijl wijst van de staart naar de kop.

Voorbeeld 5.6 In Figuur 5.1(a) zien we een ongerichte graaf met als knopenverzameling

$$V = \{a, b, c, d, e, f\}.$$

Aangezien $\#(V) = 6$ is de orde van deze graaf 6. De bogenverzameling E is

$$E = \{(a, b), (a, c), (b, d), (c, d), (c, e), (d, e), (d, f), (e, f)\}.$$

Het aantal elementen van E is 8, en dus is de grootte van de graaf gelijk aan 8.

De buren van knoop c zijn de knopen a, d en e:

$$\text{buren}(c) = \{a, d, e\}.$$

De graad van de top c is dus 3.

In Figuur 5.1(b) zien we een gerichte graaf met dezelfde knopenverzameling

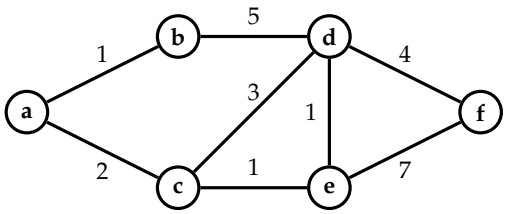
$$V = \{a, b, c, d, e, f\}.$$

Aangezien $\#(V) = 6$ is de orde van deze graaf eveneens 6. De bogenverzameling E is

$$E = \{(a, b), (a, c), (b, d), (c, d), (c, e), (d, f), (e, d), (e, f), (f, e)\}.$$

Het aantal elementen van E is 9, en dus is de grootte van de graaf 9. Merk op dat bogen voor een gerichte graaf steeds genoteerd worden van staart naar kop. De buren van knoop c in de gerichte graaf zijn de knopen d en e:

$$\text{buren}(c) = \{d, e\}.$$



Figuur 5.2: Een gewogen ongerichte graaf.

In de gerichte graaf is de graad van de top c dus 2. ∎

Soms kan het ook handig zijn om extra informatie te associëren met de bogen. Vaak neemt deze informatie de vorm aan van een getal. In dit geval spreekt men van een GEWOGEN graaf. Deze informatie kan bv. zijn: de afstand tussen twee steden in kilometer, de reistijd tussen twee luchthavens enzovoort.

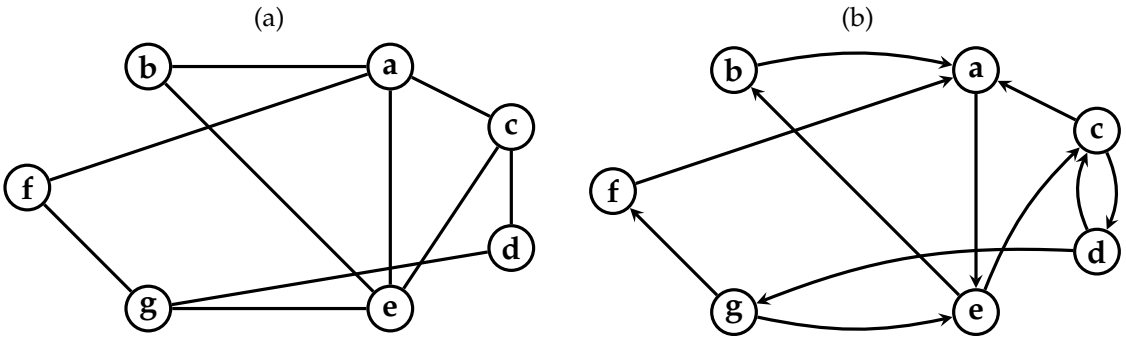
Definitie 5.7 Een PAD in een graaf G is een opsomming van toppen $(v_1, v_2, \ldots, v_k)$ zodanig dat er een boog bestaat tussen v_i en v_{i+1} voor $i \in \{1, 2, \ldots, k-1\}$. De LENGTE van dit pad is $k-1$, zijnde het aantal bogen op dit pad. ∎

Opmerking 5.8 Het triviale geval $k = 1$ is toegestaan. Elke top v is een pad met lengte nul van v naar v. ∎

Definitie 5.9 Een ENKELVOUDIGE CYKEL in een graaf is een pad waarvan de lengte strikt positief is en dat begint en eindigt in dezelfde knoop, waarbij alle knopen (behalve de start- en eindknoop) verschillend zijn en waarbij bovendien geen boog méér dan een keer wordt doorlopen. ∎

Voorbeeld 5.10 In de ongerichte graaf van Figuur 5.1(a) is (a, b, d, c, e, f) een pad van lengte 5 van a naar f. Dit is *geen* pad in de gerichte graaf van Figuur 5.1(b) omdat er geen boog is van d naar c (enkel van c naar d).

In beide grafen is het pad (d, f, e, d) een enkelvoudige cykel van lengte 3. In de gerichte graaf is (e, f, e) een enkelvoudige cykel van lengte 2, maar dit is *niet* zo in de ongerichte graaf omdat we de boog (e, f) twee keer zouden gebruiken in dit pad.



Figuur 5.3: Twee voorbeeldgrafen voor de oefeningen.

In de ongerichte graaf is (a, b, d, f, e, d, c, a) een pad van lengte 7 van a naar a. Dit pad is *geen* enkelvoudige cykel want de knoop d wordt twee keer gebruikt! ∎

5.1.1 Oefeningen

1. Beschouw de gerichte en ongerichte graaf in Figuur 5.3.

   a) Geef de bogenverzameling van deze twee grafen.

   b) Geef voor beide grafen de verzameling buren(e). Wat is de graad van de knoop e in beide gevallen?

   c) Vind het kortste pad (i.e. het pad met de kleinste lengte) van b naar d in beide grafen.

   d) Vind in beide grafen de langste enkelvoudige cykel die d bevat.

5.2 Datastructuren voor grafen

5.2.1 De adjacentiematrix

We veronderstellen dat de knopen van de graaf $G = (V, E)$ genummerd zijn van 1 t.e.m. n. Wanneer we te maken hebben met een (ongewogen) graaf dan kunnen we deze voorstellen door een ADJACENTIEMATRIX A. Voor deze adjacentiematrix geldt:

$$A_{i,j} = \begin{cases} 1 & \text{als } (i,j) \in E \\ 0 & \text{anders.} \end{cases}$$

Opmerking 5.11 Wanneer de graaf G een ongerichte graaf is, dan is de adjacentiematrix A een symmetrische matrix: wanneer er een boog gaat van i naar j dan is er (bij definitie) ook een boog van j naar i. ∎

Voorbeeld 5.12 Hieronder zien we de adjacentiematrix van de graaf in Figuur 5.1(a):

$$A = \begin{pmatrix} 0 & 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 1 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 1 & 0 \end{pmatrix}.$$

Aangezien de graaf ongericht is, hebben we inderdaad dat A een symmetrische matrix is: $A^T = A$. In dit voorbeeld werd er stilzwijgend verondersteld dat de knoop met label a rangnummer 1 heeft, de knoop met label b rangnummer 2 enzovoort. ∎

Voorbeeld 5.13 Bekijken we de gerichte graaf in Figuur 5.1(b), dan is zijn adjacentiematrix:

$$A = \begin{pmatrix} 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 \end{pmatrix}.$$

Merk op dat deze matrix niet symmetrisch is. ∎

De adjacentiematrix kan uitgebreid worden om ook de gewichten van de bogen op te slaan. In dit geval bevat $A_{i,j}$ het gewicht van de boog (i, j). Men dient er dan wel op te letten dat men een "speciale" waarde voorziet om aan te geven dat i en j niet adjacent zijn. Wanneer men zeker weet dat de gewichten positief zijn (bv. omdat ze een afstand voorstellen), dan kan men de waarde -1 nemen voor deze speciale waarde.

De geheugenruimte die een adjacentiematrix in beslag neemt is steeds $\Theta(n^2)$, onafhankelijk van het aantal bogen in de graaf. Wanneer het aantal bogen klein is in vergelijking met het maximale aantal bogen (n^2 in een gerichte graaf), dan verspilt deze voorstellingswijze heel wat geheugenruimte.

Een graaf waarvoor het aantal bogen "klein" is t.o.v. het aantal mogelijke bogen noemt men een IJLE graaf. Merk op dat er niet exact wordt gespecifieerd wat er bedoeld wordt met "klein".

Het bepalen of twee knopen adjacent zijn of niet gebeurt in constante tijd $\Theta(1)$ aangezien dit neerkomt op het ophalen van een element uit een tweedimensionale array.

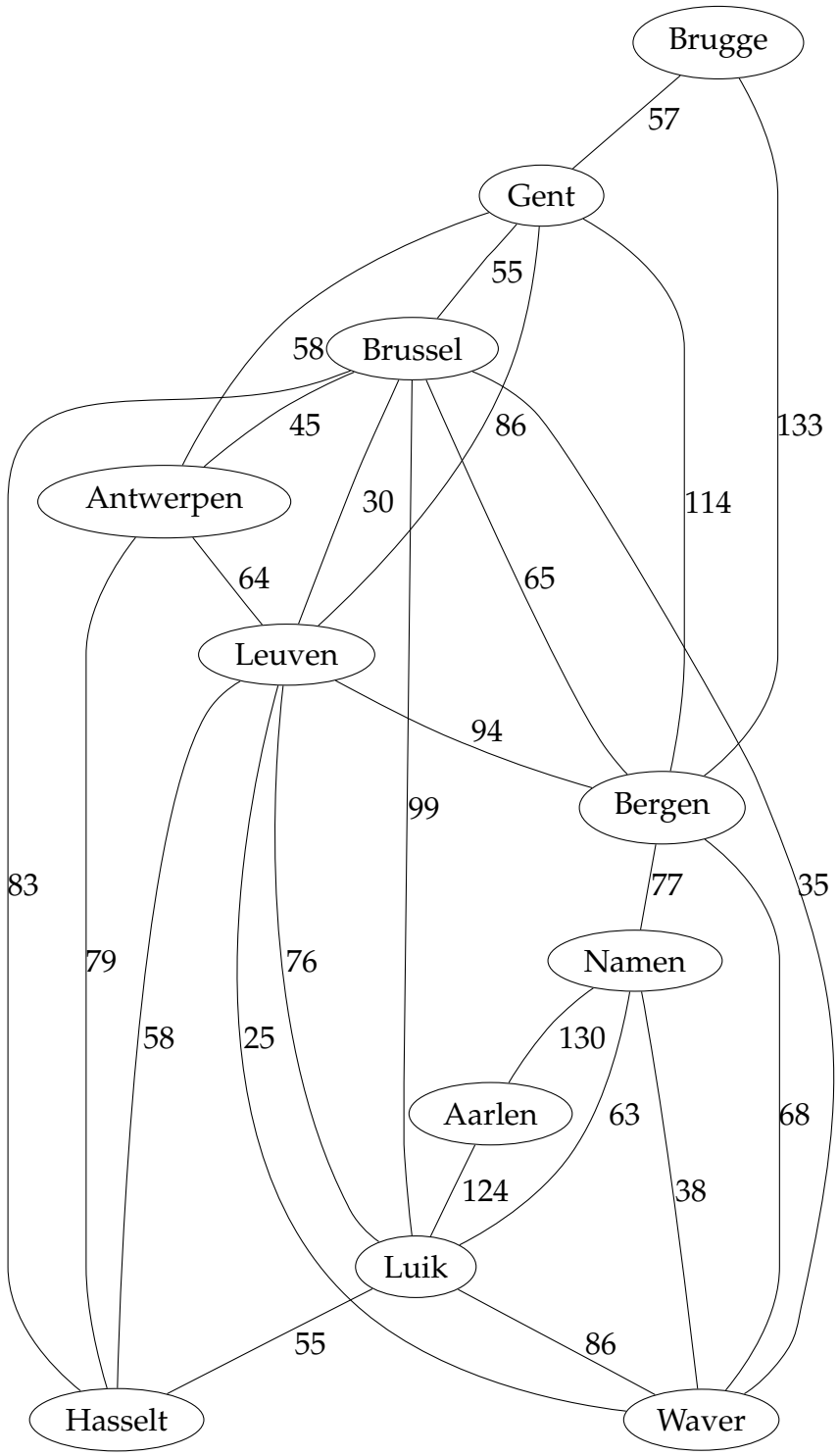
Het overlopen van alle buren van een knoop is een bewerking die $\Theta(n)$ tijd in beslag neemt, onafhankelijk van de graad van de knoop. Men moet immers steeds de volledige rij corresponderend met de knoop overlopen.

Opmerking 5.14 Wanneer de toppen niet genummerd zijn van 1 t.e.m. n maar bv. gelabeld zijn met karakterstrings, dan kan men een "woordenboek" gebruiken om elk label om te zetten naar een geheel getal tussen 1 en n. Een mogelijke implementatie van zo'n woordenboek is m.b.v. een hashmap. ∎

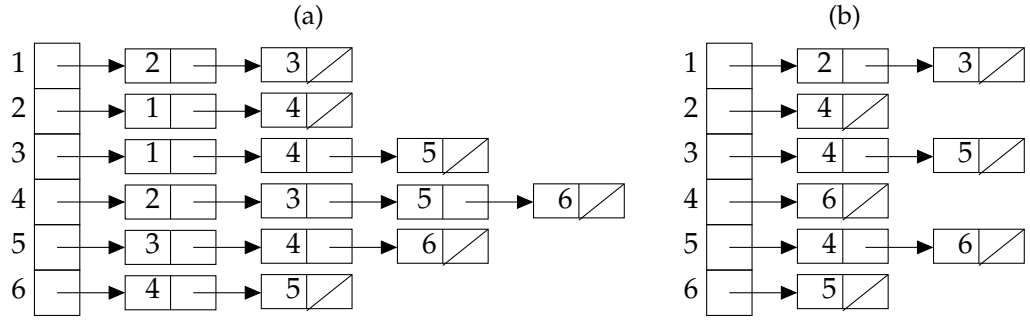
Voorbeeld 5.15 In Figuur 5.4 zien we een gewogen gelabelde graaf. Hieronder zien we het gebruikte woordenboek samen met de corresponderende adjacentiematrix.

Brugge	$\to 1$	∞	57	∞	∞	∞	133	∞	∞	∞	∞	∞
Gent	$\to 2$	57	∞	55	58	86	114	∞	∞	∞	∞	∞
Brussel	$\to 3$	∞	55	∞	45	30	65	∞	∞	99	35	83
Antwerpen	$\to 4$	∞	58	45	∞	64	∞	∞	∞	∞	∞	79
Leuven	$\to 5$	∞	86	30	64	∞	94	∞	∞	76	25	58
Bergen	$\to 6$	133	114	65	∞	94	∞	77	∞	∞	68	∞
Namen	$\to 7$	∞	∞	∞	∞	∞	77	∞	130	63	38	∞
Aarlen	$\to 8$	∞	∞	∞	∞	∞	∞	130	∞	124	∞	∞
Luik	$\to 9$	∞	∞	99	∞	76	∞	63	124	∞	86	55
Waver	$\to 10$	∞	∞	35	∞	25	68	38	∞	86	∞	∞
Hasselt	$\to 11$	∞	∞	83	79	58	∞	∞	∞	55	∞	∞

Merk op dat we als speciale waarde hier ∞ hebben gebruikt. We duiden aan dat de graaf geen lussen heeft door op de diagonaal eveneens ∞ te plaatsen. ∎



Figuur 5.4: Graaf met Belgische provinciehoofsteden (en Brussel). Het gewicht van elke boog geeft de afstand tussen de twee betrokken steden weer zoals aangegeven door Google Maps. Gewichten staan steeds rechts van een boog.



Figuur 5.5: Adjacentielijsten van de grafen in Figuur 5.1.

5.2.2 De adjacentielijst-voorstelling

De adjacentielijst-voorstelling lost het probleem van verspilling van geheugenruimte in de adjacentiematrix-voorstelling voor ijle grafen op.

De ADJACENTIELIJST-VOORSTELLING van een graaf G bestaat uit een array van toppen, genummerd 1 t.e.m. n. Op de plaats i van deze array worden, in een lineair gelinkte lijst, de buren van top i bijgehouden.

De adjacentielijst-voorstelling gebruikt $\Theta(n + m)$ geheugenruimte. Er wordt een array van n toppen bijgehouden, en elk van de m bogen van de graaf geeft aanleiding tot één (of twee) knopen[1] in de geschakelde lijst.

Het bepalen of twee knopen adjacent zijn kan nu *niet* langer in constante tijd gebeuren. Om te weten of i en j adjacent zijn moeten we immers de gelinkte lijst horend bij i overlopen om na te gaan of j in deze lijst aanwezig is.

Als we alle buren van een knoop i willen overlopen dan gebeurt dit nu in een tijd die lineair is in het aantal buren van de knoop i. Dit is theoretisch de best mogelijke uitvoeringstijd.

Voorbeeld 5.16 In Figuur 5.5 ziet men de adjacentielijst-voorstelling voor de grafen in Figuur 5.1. Merk op dat voor de ongerichte graaf elke boog *twee* keer voorkomt in de adjacentielijst-voorstelling. ∎

Opmerking 5.17 Wanneer in een adjacentielijst-voorstelling de knopen niet gelabeld zijn van 1 t.e.m. n dan kan men opnieuw een woordenboek gebruiken. De sleutels zijn in dit geval de labels van de knopen, terwijl de bijhorende waarden de gelinkte lijsten van buren zijn. ∎

[1] Dit zijn dus *niet* de knopen zoals in de definitie van een graaf, maar bv. `NodeList` componenten van een lineair geschakelde lijst.

5.2.3 Oefeningen

1. Beschouw de gerichte en ongerichte graaf in Figuur 5.3.
   a) Geef voor beide grafen de adjacentiematrix. Je mag veronderstellen dat a rangnummer 1 heeft, b rangnummer 2 enzovoort.
   b) Geef voor beide grafen de adjacentielijst-voorstelling. Je mag veronderstellen dat a rangnummer 1 heeft, b rangnummer 2 enzovoort.
2. Hoe berekent men de graad van een top i van een graaf G wanneer de adjacentiematrix A van de graaf gegeven is? Geef een algoritme. Wat is de tijdscomplexiteit van deze methode?
3. Hoe berekent men de graad van een top i van een graaf G wanneer de adjacentielijst-voorstelling van de graaf gegeven is? Geef een algoritme. Wat is de tijdscomplexiteit van deze methode?

5.3 Zoeken in Grafen

Veronderstel dat een graaf $G = (V, E)$ gegeven is. Dan kunnen we geïnteresseerd zijn om algoritmes te vinden die de volgende vragen kunnen beantwoorden:

1. Welke knopen kunnen we bereiken vanuit een knoop v?
2. Bestaat er een pad van v naar een specifieke knoop w?
3. Wat is het kortste pad van v naar w?

5.3.1 Generiek Zoeken

Het doel van een zoekalgoritme bestaat erin om in een graaf G vanaf een gegeven knoop v alle knopen te vinden die bereikbaar zijn vanaf v, i.e. alle knopen w waarvoor er een pad bestaat van v naar w. Bovendien zouden we graag hebben dat dit efficiënt gebeurt, i.e. dat de uitvoeringstijd $\Theta(n + m)$ is.

Een generiek zoekalgoritme voor een gegeven graaf G en startend vanuit een knoop s gaat als volgt: We markeren de knoop s als ontdekt, i.e. initieel

bestaat het *ontdekte gebied* enkel uit de knoop s. In elke stap breiden we het ontdekte gebied uit door een boog (u, v) te kiezen waarbij u reeds ontdekt is en v nog niet ontdekt is, i.e. de boog (u, v) kruist als het ware de grens tussen het ontdekte en onontdekte gebied. Op deze manier breiden we het ontdekte gebied uit met de knoop v. We herhalen dit proces tot er geen bogen meer zijn die de grens tussen ontdekt en onontdekt gebied kruisen. Het algoritme wordt gegeven in Algoritme 5.1.

Algoritme 5.1 Generiek zoeken in een graaf.

Invoer Een gerichte of ongerichte graaf $G = (V, E)$ met orde $n > 0$. Een knoop s waarvan het zoeken vertrekt. De knopen zijn genummerd van 1 tot n, i.e. $V = \{1, 2, \ldots, n\}$.

Uitvoer Een array D met $D[v] =$ **true** als en slechts als er een pad bestaat van s naar v.

```
1: function ZoekGeneriek(G, s)
2:     D ← [false, false, ..., false]          ▷ n keer false
3:     D[s] ← true                              ▷ markeer s
4:     while ∃(u, v): D[u] = true ∧ D[v] = false do
5:         kies een boog (u, v) met D[u] = true ∧ D[v] = false
6:         D[v] ← true                          ▷ markeer v
7:     end while
8:     return D
9: end function
```

Opmerking 5.18 Dit algoritme is *ondergespecifiëerd* in die zin dat het niet duidelijk is *hoe* de keuze van de boog (u, v) gebeurt. We zullen dit in de volgende secties doen. ∎

Eigenschap 5.19 Wanneer het algoritme voor generiek zoeken eindigt dan geldt voor elke knoop v van G dat v gemarkeerd is als "ontdekt" (i.e. $D[v] =$ **true**) als en slechts als er een pad bestaat van s naar v in G. ∎

Bewijs **We bewijzen het ⇒ deel eerst.**
Veronderstel dat het algoritme de top v markeert als ontdekt. We willen nu aantonen dat er een pad bestaat van s naar v. Het gestelde is duidelijk waar voor de eerste ontdekte knoop, want dit is de knoop s. Veronderstel nu dat het gestelde waar is voor knopen die vóór de m-de knoop ontdekt zijn, en dat v als m-de knoop werd ontdekt, met (u, v) als gekozen boog. Uit de inductiehypothese volgt dat er een pad bestaat van s naar u, dat via de boog (u, v) uitgebreid wordt tot een pad van s naar v.

Nu tonen we het ⇐ gedeelte aan.
Omgekeerd bewijzen we nu dat het algoritme alle knopen ontdekt waarvoor een pad

bestaat vanaf s. Veronderstel dat dit niet zo is, dus veronderstel dat er een knoop v is waarvoor er een pad bestaat van s naar v maar waarvoor het algoritme eindigt met $D[v] =$ **false**. Beschouw een pad van s naar v en noem u de *eerste* knoop op dit pad waarvoor $D[u] =$ **false** (merk op dat $u \neq s$). Voor de voorganger van u op dit pad, noem deze w, geldt dus $D[w] =$ **true**. Maar dan zou het algoritme nog niet gestopt zijn want op regel 4 voldoet de boog (w, u) aan de voorwaarde, zodanig dat de lus nog minstens één keer zou uitgevoerd worden. M.a.w. wanneer het algoritme stopt zijn alle knopen op het pad van s naar v ontdekt. In het bijzonder is ook v ontdekt. ♢

Voorbeeld 5.20 In Figuur 5.6 ziet men een *mogelijk* verloop van het algoritme voor generiek zoeken. We gebruiken de ongerichte graaf uit Figuur 5.1(a) maar we hebben de knopen expliciet hernoemd als 1 t.e.m. 6.

De startknoop is de knoop 1. In blauwe stippellijn zijn telkens de bogen aangeduid die voldoen aan de voorwaarde op regel 4 van het algoritme. In de rechterkolom ziet men dan telkens de graaf nadat een (random) keuze werd gemaakt uit de blauwe bogen.

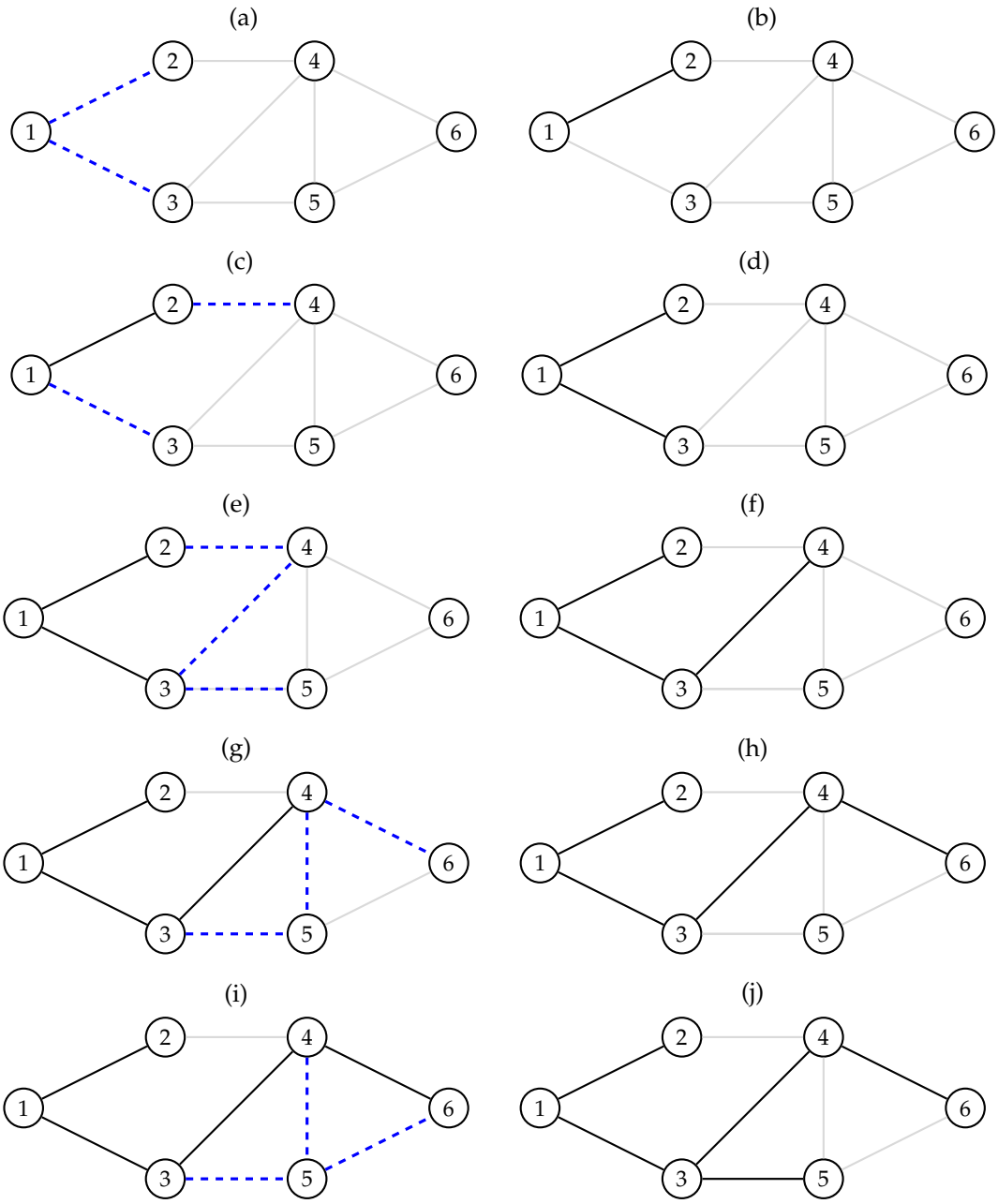
We sommen hieronder het verloop van het algoritme op:

gemarkeerde knopen	mogelijke bogen	gekozen boog
$\{1\}$	$(1,2), (1,3)$	$(1,2)$
$\{1,2\}$	$(1,3), (2,4)$	$(1,3)$
$\{1,2,3\}$	$(2,4), (3,4), (3,5)$	$(3,4)$
$\{1,2,3,4\}$	$(3,5), (4,5), (4,6)$	$(4,6)$
$\{1,2,3,4,6\}$	$(3,5), (4,5), (5,6)$	$(3,5)$
$\{1,2,3,4,5,6\}$	geen	geen

Merk op dat dit algoritme de bogen *niet* op een systematische manier afloopt. ∎

5.3.2 Breedte-Eerst Zoeken

Het algoritme voor breedte-eerst zoeken is een instantie van het algoritme voor generiek zoeken. Met dit algoritme gaan we de knopen van een graaf G bezoeken in "lagen", i.e. eerst bezoeken we de knoop s zelf, dan de knopen die juist één boog verwijderd zijn van s, dan de knopen die juist twee bogen verwijderd zijn van s, enzovoort.



Figuur 5.6: Voorbeeld van generiek zoeken.

De datastructuur die gebruikt wordt om breedte-eerst zoeken te implementeren is een FIFO-datastructuur, i.e. een wachtrij. Meer specifiek zal de wachtrij dié knopen v bevatten die reeds zijn ontdekt, maar waarvoor er mogelijks nog buren u zijn die nog niet ontdekt zijn.

De pseudocode voor breedte-eerst zoeken wordt gegeven in Algoritme 5.2.

Algoritme 5.2 Breedte-eerst zoeken in een graaf.

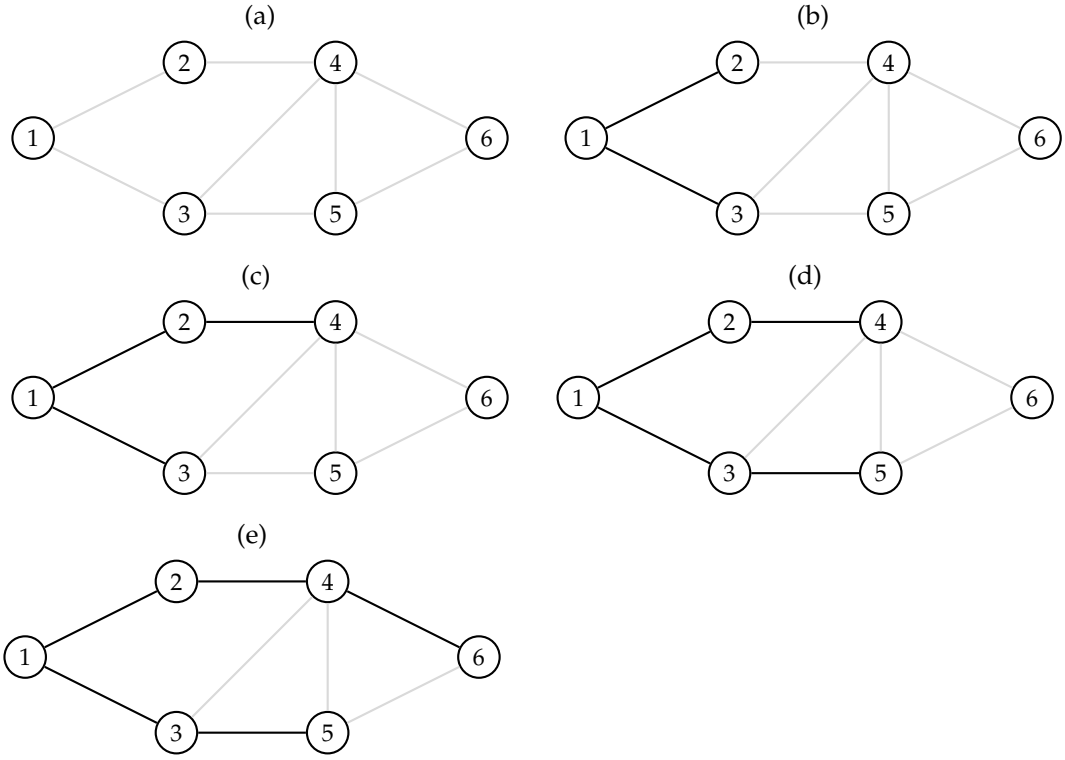
Invoer Een gerichte of ongerichte graaf $G = (V, E)$ met orde $n > 0$. Een knoop s waarvan het zoeken vertrekt. De knopen zijn genummerd van 1 tot n, i.e. $V = \{1, 2, \ldots, n\}$.

Uitvoer Een array D met $D[v] =$ **true** als en slechts als er een pad bestaat van s naar v.

```
 1: function BREEDTEEERST(G, s)
 2:     D ← [false, false, ..., false]          ▷ n keer false
 3:     D[s] ← true                              ▷ markeer s
 4:     Q.init()                                 ▷ wachtrij van knopen
 5:     Q.enqueue(s)
 6:     while Q ≠ ∅ do
 7:         v ← Q.dequeue()
 8:         for all w ∈ buren(v) do
 9:             if D[w] = false then             ▷ w nog niet ontdekt
10:                 D[w] ← true
11:                 Q.enqueue(w)
12:             end if
13:         end for
14:     end while
15:     return D
16: end function
```

Voorbeeld 5.21 We voeren breedte-eerst zoeken uit op de ongerichte voorbeeldgraaf in Figuur 5.1(a), maar we noemen de knopen nu expliciet 1 t.e.m. 6.

We duiden op de Figuur 5.7 aan welke *bogen* doorlopen werden door het algoritme, i.e. welke bogen ervoor zorgden dat een knoop werd toegevoegd aan de wachtrij. In de tabel hieronder zie je expliciet de inhoud van de wachtrij telkens regel 6 wordt uitgevoerd. We hebben er hier voor gekozen om de buren steeds te doorlopen in volgorde van hun label, i.e. alsof hun adjacentielijst-voorstelling die is in Figuur 5.5.



Figuur 5.7: Stap voor stap uitvoering van breedte-eerst zoeken. Elke figuur toont de bogen die gekozen zijn wanneer de test op regel 6 van Algoritme 5.2 wordt uitgevoerd.

iteratie	Q	D
1	[1]	[T,F,F,F,F,F]
2	[2,3]	[T,T,T,F,F,F]
3	[3,4]	[T,T,T,T,F,F]
4	[4,5]	[T,T,T,T,T,F]
5	[5,6]	[T,T,T,T,T,T]
6	[6]	[T,T,T,T,T,T]
7	[]	[T,T,T,T,T,T]

Het is gemakkelijk om te controleren dat de wachtrij op elk moment knopen bevat van hoogstens twee verschillende “lagen”. ∎

Eigenschap 5.22 Wanneer een adjacentielijst-voorstelling gebruikt wordt voor een graaf G, dan is de uitvoeringstijd $T(n, m)$ van Algoritme 5.2 van de grootte-orde $\Theta(n + m)$. ∎

Bewijs Om te zien dat de uitvoeringstijd van breedte-eerst zoeken inderdaad $\Theta(n+m)$ is wanneer een adjacentielijst-voorstelling gebruikt wordt redeneren we als volgt:

1. Het initialiseren van de array D neemt tijd $\Theta(n)$.
2. Het is duidelijk dat elke knoop hoogstens één keer wordt toegevoegd aan de wachtrij, en dus wordt elke knoop er hoogstens één keer uit verwijderd. De totale tijd voor de lus op regel 8 is dus hoogstens

$$\sum_{v \in V} \#\big(\text{buren}(v)\big) = 2m \quad \text{in een ongerichte graaf}$$

of

$$\sum_{v \in V} \#\big(\text{buren}(v)\big) = m \quad \text{in een gerichte graaf.}$$

Dus we vinden inderdaad dat de uitvoeringstijd $\Theta(n+m)$ is aangezien alle bewerkingen in de lus een constante uitvoeringstijd hebben. ♢

5.3.3 Diepte-Eerst Zoeken

Diepte-eerst zoeken is eveneens een instantie van het generieke zoekalgoritme. Bij diepte-eerst zoeken gaan we zo snel mogelijk zo diep mogelijk in de graaf, i.e. we bekijken steeds de buren van de 'diepste' ontdekte knoop. Dit wordt bereikt met een algoritme gelijkaardig aan breedte-eerst zoeken, maar we vervangen de wachtrij van breedte-eerst zoeken door een *stapel* (stack). In Algoritme 5.3 geven we een recursieve implementatie en maken dus gebruik van de (impliciete) stapel van methode-oproepen (call-stack).

Voorbeeld 5.23 In Figuur 5.8 zien we hoe diepte-eerst zoeken wordt uitgevoerd op de ongerichte graaf in Figuur 5.1(a), startend vanaf top 1. Telkens wanneer regel 10 uitgevoerd wordt voegen we de boog (v, w) toe op de figuur. Hieronder zien we de verschillende recursieve oproepen die gebeuren bij het uitvoeren van DIEPTEEERST.

DIEPTEEERSTRECURSIEF(G, 1, [**false**, **false**, **false**, **false**, **false**, **false**])
  DIEPTEEERSTRECURSIEF(G, 2, [**true**, **false**, **false**, **false**, **false**, **false**])
    DIEPTEEERSTRECURSIEF(G, 4, [**true**, **true**, **false**, **false**, **false**, **false**])
      DIEPTEEERSTRECURSIEF(G, 3, [**true**, **true**, **false**, **true**, **false**, **false**])
        DIEPTEEERSTRECURSIEF(G, 5, [**true**, **true**, **true**, **true**, **false**, **false**])
          DIEPTEEERSTRECURSIEF(G, 6, [**true**, **true**, **true**, **true**, **true**, **false**])

Algoritme 5.3 Diepte-eerst zoeken in een graaf.

Invoer Een gerichte of ongerichte graaf $G = (V, E)$ met orde $n > 0$. Een knoop s waarvan het zoeken vertrekt. De knopen zijn genummerd van 1 tot n, i.e. $V = \{1, 2, \ldots, n\}$.

Uitvoer Een array D met $D[v] =$ **true** als en slechts als er een pad bestaat van s naar v.

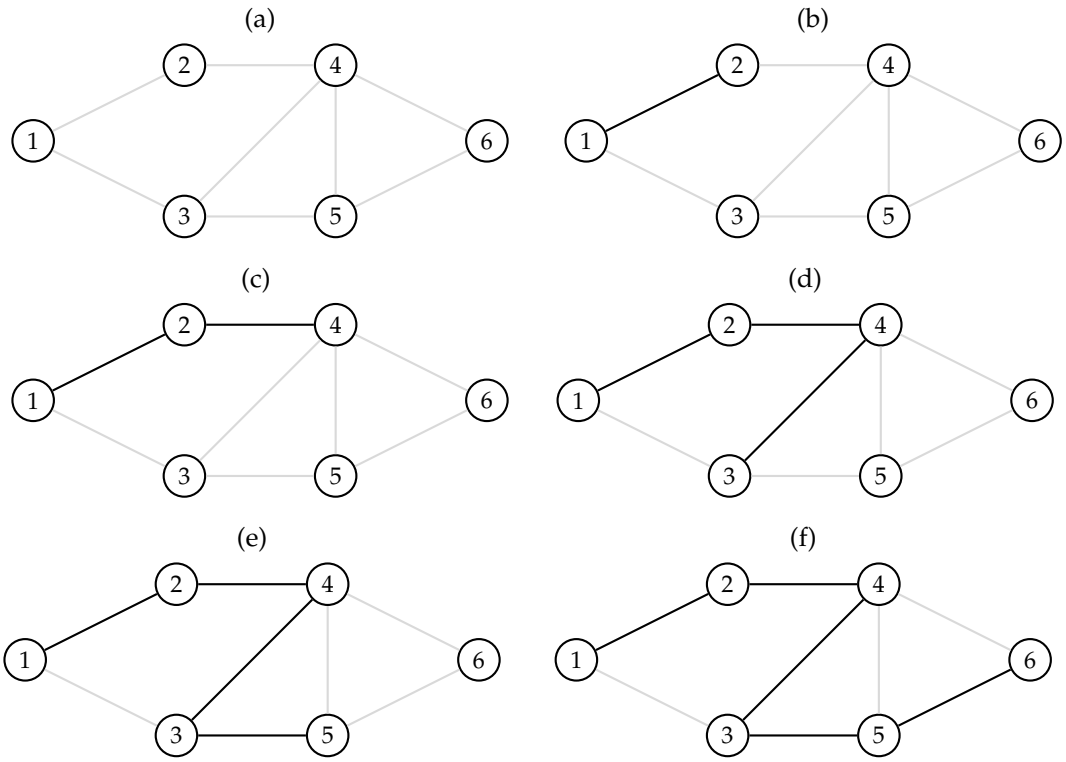
```
1: function DiepteEerst(G, s)
2:     D ← [false, false, ..., false]                ▷ n keer false
3:     DiepteEerstRecursief(G, s, D)
4:     return D
5: end function
6: function DiepteEerstRecursief(G, v, D)
7:     D[v] ← true                                   ▷ markeer v
8:     for all w ∈ buren(v) do
9:         if D[w] = false then                      ▷ w nog niet ontdekt
10:            DiepteEerstRecursief(G, w, D)
11:        end if
12:    end for
13: end function
```

Merk op dat het eerste statement van DiepteEerstRecursief ook aan $D[6]$ de waarde **true** zal toekennen. De uiteindelijke return-waarde zal dan ook een array zijn waarvan alle componenten de waarde **true** hebben. ∎

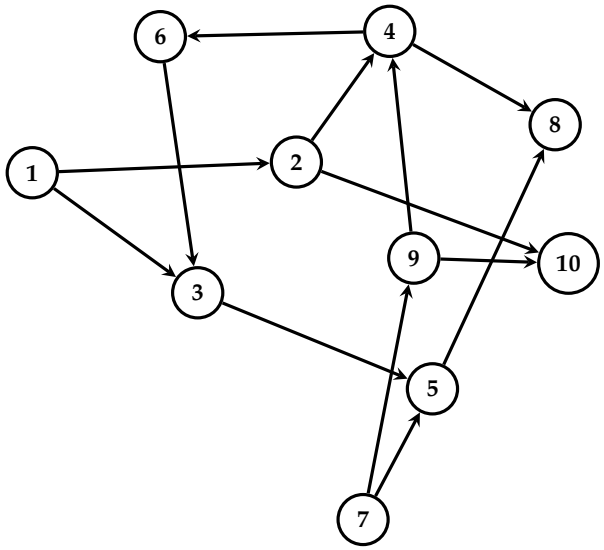
5.3.4 Toepassing: Topologisch Sorteren

We veronderstellen in deze sectie dat G een gerichte graaf is, zonder enkelvoudige cykels. Een voorbeeld van zo'n graaf is een PRECEDENTIEGRAAF, waarbij de verschillende knopen taken voorstellen. Er is een boog van knoop v naar w als taak v moet afgewerkt zijn alvorens taak w kan aangevat worden.

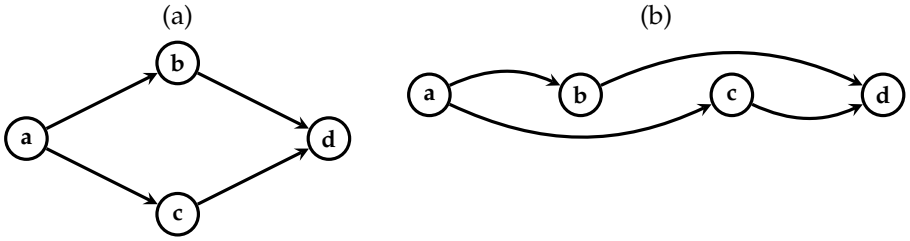
Voorbeeld 5.24 Veronderstel dat een softwareproject bestaat uit 10 modules. Sommige modules zijn afhankelijk van andere modules, en deze andere modules moeten dus eerst gecompileerd worden. We modelleren het project als een graaf, waarbij elke knoop een module voorstelt en er is een boog (v, w) als module w afhankelijk is van v, i.e. als v moet gecompileerd worden voor w. De graaf wordt gegeven in Figuur 5.9. We vragen ons af in welke volgorde we de modules kunnen compileren. ∎



Figuur 5.8: Stap voor stap uitvoering van diepte-eerst zoeken.



Figuur 5.9: Precedentiegraaf voor 10 software modules. Er is een boog van v naar w als v moet gecompileerd worden voor w.



Figuur 5.10: Links zie je een gerichte graaf zonder cykels, rechts een topologische sortering van deze graaf (impliciet gegeven door de volgorde van de knopen op de lijn).

Een topologische sortering is een volgorde van de taken zodanig dat voor elke taak w al zijn voorgaande taken zijn afgewerkt alvorens w wordt gestart. We maken dit meer formeel in de volgende definitie.

Definitie 5.25 Een TOPOLOGISCHE SORTERING van een gerichte graaf G kent aan elke knoop v een verschillend rangnummer $f(v)$ toe van 1 t.e.m. n zodanig dat de volgende eigenschap geldt:

$$\forall (u, v) \in E \colon f(u) < f(v), \tag{5.1}$$

m.a.w. als (u, v) een boog is in de graaf dan is het rangnummer van de kop v groter dan het rangnummer van de staart u. ∎

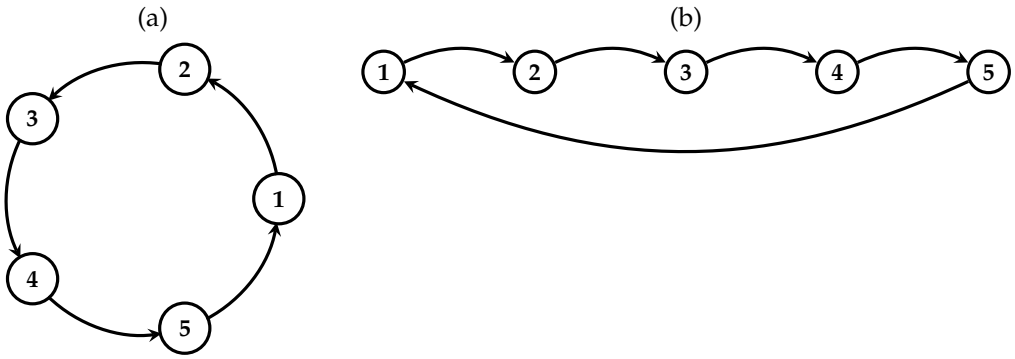
Wanneer we alle knopen op een rechte lijn tekenen, gesorteerd volgens het rangnummer van hun topologische sortering, dan zullen alle bogen vooruit wijzen.

Voorbeeld 5.26 In Figuur 5.10(a) zien we een gerichte graaf zonder cykels. Een mogelijke topologische sortering is de volgende:

$$a, b, c, d.$$

Inderdaad, kijken we naar Figuur 5.10(b) dan zien we een andere voorstelling van dezelfde graaf met alle knopen op een rechte lijn volgens de gegeven sortering. We zien onmiddellijk dat alle bogen vooruit wijzen, dus is dit een geldige topologische sortering. ∎

Opmerking 5.27 Het is duidelijk dat een gerichte graaf met een cykel *geen* topologische sortering heeft. Inderdaad, als we alle knopen van een cykel



Figuur 5.11: Een cykel met 5 knopen en een *poging* tot topologische sortering van diezelfde cykel. Merk op dat de boog $(5, 1)$ achteruit wijst.

op een rechte lijn tekenen, dan zal de boog die de cykel "sluit" steeds achteruit wijzen. ∎

Voorbeeld 5.28 In Figuur 5.11 ziet men een cykel samen met een poging tot topologische sortering van die cykel. De boog $(5, 1)$ die de cykel sluit wijst achteruit; dit is niet toegelaten voor een topologische sortering. ∎

Eigenschap 5.29 Wanneer een gerichte graaf G geen enkelvoudige cykels heeft, dan bestaat er een topologische sortering van G. ∎

Bewijs We bewijzen eerst dat elke gerichte graaf G zonder enkelvoudige cykels steeds een knoop zonder buren heeft.

We geven een bewijs uit het ongerijmde: veronderstel dat G een gerichte graaf is zonder cykels maar dat elke knoop steeds minstens één buur heeft. We kunnen dan een pad

$$(v_1, v_2, v_3, \ldots, v_{n+1}) \tag{5.2}$$

vormen. (We lopen nooit "vast" want elke knoop heeft minstens één buur.) Het pad (5.2) bestaat uit $n + 1$ knopen, maar de graaf G heeft slechts n knopen. Dit betekent dat minstens één knoop herhaald wordt in dit pad, en zo krijgen we een cykel in G. Dit is in strijd met de veronderstelling dat G geen cykel bevat. Er moet dus minstens één knoop zijn zonder buren.

Het is duidelijk dat wanneer G een topologische sortering heeft, dat dan de knoop met rangnummer n een knoop is zonder buren. Inderdaad, veronderstel dat v een knoop is die wel buren heeft en dat $f(v) = n$, dan betekent dit volgens eigenschap (5.1) dat $f(u) > n$ wanneer $(v, u) \in E$, maar dit is onmogelijk aangezien het grootste rangnummer dat we kunnen toekennen n is. M.a.w. in een topologische sortering zijn de enige knopen v waarvoor $f(v) = n$ kan zijn die knopen waarvoor $\text{buren}(v) = \emptyset$.

Een eenvoudig algoritme om een topologische sortering te vinden is dan het volgende.

1. Kies een knoop v zonder buren (zo is er minstens één), en stel $f(v) = n$.
2. Doe nu (recursief) hetzelfde voor de graaf $G - \{v\}$[2].

Dit algoritme werkt om de volgende reden: veronderstel dat de knoop v het rangnummer i krijgt, dan heeft deze knoop in de huidige graaf geen buren. Alle bogen die vertrekken vanuit v zijn dus reeds verwijderd uit de graaf G. Dat betekent dat voor een boog (v, u) er steeds geldt dat $f(u) > i$. ♢

Het bewijs van Eigenschap 5.29 geeft een algoritmische manier om een topologische sortering te vinden. We kunnen echter ook het algoritme voor diepte-eerst zoeken aanpassen zodanig dat het een topologische sortering produceert.

Het idee is het volgende: met diepte-eerst zoeken bekijken we vanuit *elke* knoop v steeds alle knopen w waarvoor er een pad bestaat van v naar w. Dit betekent dat in een topologische sortering v steeds vóór w zal moeten komen.

We houden een lijst van knopen S bij die initieel leeg is. Wanneer we *alle* knopen bereikbaar vanuit v hebben ontdekt dan voegen we v vooraan toe aan de lijst.

Het proces van diepte-eerst zoeken moet eventueel meerdere malen worden herhaald tot we alle knopen van de graaf hebben ontdekt. Het is immers mogelijk dat vanuit de gekozen startknoop niet alle knopen van de graaf bereikbaar zijn.

Om na te gaan of de graaf een cykel heeft of niet gaan we als volgt tewerk. Onontdekte knopen krijgen een statuscode 0; zolang een knoop nog niet is toegevoegd aan de lijst S (maar wel reeds ontdekt is) heeft die statuscode 1. Wanneer een knoop volledig is afgewerkt (en aan S wordt toegevoegd) dan krijgt deze statuscode 2. Stel nu dat we de buren van v bekijken en we zien een knoop w met statuscode 1. Dit betekent dat er een pad is van w naar v, maar de boog van v naar w sluit dit pad, en we hebben dus een cykel ontdekt in de graaf. De volledige pseudocode vind je in Algoritme 5.4.

Voorbeeld 5.30 We bepalen nu een mogelijke compilatievolgorde voor de modules. We voeren Algoritme 5.4 uit en we doorlopen de knopen steeds

[2] $G - \{v\}$ is de graaf die men vindt door v en al de bogen incident met v te verwijderen uit de graaf G. Uiteraard heeft deze graaf ook geen (enkelvoudige) cykels.

Algoritme 5.4 Topologisch sorteren van een graaf.

Invoer Een gerichte graaf $G = (V, E)$ met orde $n > 0$. De knopen zijn genummerd van 1 tot n, i.e. $V = \{1, 2, \ldots, n\}$.

Uitvoer Een topologische sortering van G indien mogelijk, **false** anders.

```
1:  function SorteerTopologisch(G)
2:      global cycleDetected ← false                    ▷ globale variabele
3:      D ← [0, 0, ..., 0]                              ▷ n keer 0
4:      S ← ∅                                           ▷ S is lege lijst
5:      for all s ∈ V do
6:          if D[s] = 0 then                            ▷ s nog niet gezien
7:              DfsTopo(G, s, D, S)                     ▷ S en D referentieparameters
8:              if cycleDetected = true then            ▷ controleer op cykel
9:                  return false
10:             end if
11:         end if
12:     end for
13:     return S
14: end function
15: function DfsTopo(G, v, D, S)
16:     D[v] ← 1                                        ▷ markeer v als 'bezig'
17:     for all w ∈ buren(v) do
18:         if D[w] = 0 ∧ cycleDetected = false then    ▷ w nog niet ontdekt
19:             DfsTopo(G, w, D, S)
20:         else if D[w] = 1 then                       ▷ cykel ontdekt w ⇝ v → w
21:             cycleDetected ← true
22:         end if
23:     end for
24:     D[v] ← 2                                        ▷ markeer v als 'voltooid'
25:     voeg v vooraan toe aan S                        ▷ ken rangnummer toe aan v
26: end function
```

volgens stijgend label. We houden de array D bij, de stapel van methode-oproepen, alsook de lijst S.

	1	2	3	4	5	6	7	8	9	10	stack	S
(a)	0	0	0	0	0	0	0	0	0	0		$\emptyset$
(b)	1	0	0	0	0	0	0	0	0	0	1	$\emptyset$
(c)	1	1	0	0	0	0	0	0	0	0	1, 2	$\emptyset$
(d)	1	1	0	1	0	0	0	0	0	0	1, 2, 4	$\emptyset$
(e)	1	1	0	1	0	1	0	0	0	0	1, 2, 4, 6	$\emptyset$
(f)	1	1	1	1	0	1	0	0	0	0	1, 2, 4, 6, 3	$\emptyset$
(g)	1	1	1	1	1	1	0	0	0	0	1, 2, 4, 6, 3, 5	$\emptyset$
(h)	1	1	1	1	1	1	0	1	0	0	1, 2, 4, 6, 3, 5, 8	$\emptyset$
(i)	1	1	1	1	1	1	0	2	0	0	1, 2, 4, 6, 3, 5	8
(j)	1	1	1	1	2	1	0	2	0	0	1, 2, 4, 6, 3	5, 8
(k)	1	1	2	1	2	1	0	2	0	0	1, 2, 4, 6	3, 5, 8
(l)	1	1	2	1	2	2	0	2	0	0	1, 2, 4	6, 3, 5, 8
(m)	1	1	2	2	2	2	0	2	0	0	1, 2	4, 6, 3, 5, 8
(n)	1	1	2	2	2	2	0	2	0	1	1, 2, 10	4, 6, 3, 5, 8
(o)	1	1	2	2	2	2	0	2	0	2	1, 2	10, 4, 6, 3, 5, 8
(p)	1	2	2	2	2	2	0	2	0	2	1	2, 10, 4, 6, 3, 5, 8
(q)	2	2	2	2	2	2	0	2	0	2		1, 2, 10, 4, 6, 3, 5, 8
(r)	2	2	2	2	2	2	1	2	0	2	7	1, 2, 10, 4, 6, 3, 5, 8
(s)	2	2	2	2	2	2	1	2	1	2	7, 9	1, 2, 10, 4, 6, 3, 5, 8
(t)	2	2	2	2	2	2	1	2	2	2	7	9, 1, 2, 10, 4, 6, 3, 5, 8
(u)	2	2	2	2	2	2	2	2	2	2		7, 9, 1, 2, 10, 4, 6, 3, 5, 8

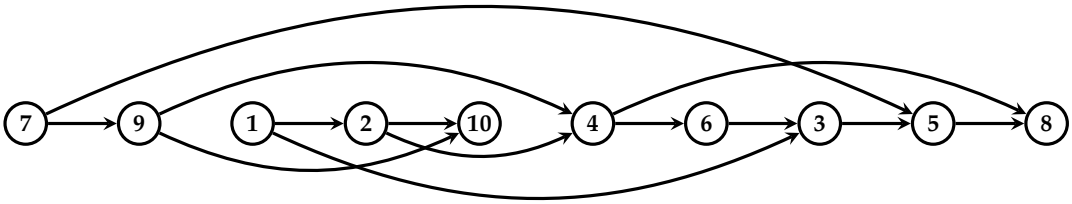
Uit de tabel leiden we af dat een mogelijke compilatievolgorde van de modules gegeven wordt door:

$$7, 9, 1, 2, 10, 4, 6, 3, 5, 8.$$

In Figuur 5.12 controleert men inderdaad dat alle bogen vooruit wijzen. ∎

5.3.5 Oefeningen

1. Een ongerichte graaf is GECONNECTEERD als en slechts als er een pad bestaat tussen elke twee knopen v en w.



Figuur 5.12: Een topologische sortering van de graaf in Figuur 5.9.

a) Ga na dat de bovenstaande definitie equivalent is met zeggen dat er een pad bestaat van een bepaalde knoop s naar alle andere knopen.

b) Schrijf een methode IsGeconnecteerd die nagaat of een ongerichte graaf geconnecteerd is (return-waarde **true**) of niet (return-waarde **false**). Doe dit door de methode BreedteEerst aan te passen.

2. Vind alle mogelijke topologische sorteringen van de graaf in Figuur 5.10.

3. Vind de compilatievolgorde van de modules in Figuur 5.9 wanneer de labels in dalende volgorde worden doorlopen.

4. Veronderstel nu dat er in de graaf van Figuur 5.9 een extra boog $(8, 6)$ wordt toegevoegd. Pas nu het algoritme voor topologisch sorteren toe.

5.4 Kortste Pad Algoritmen

5.4.1 Kortste Pad in een Ongewogen Graaf

Wanneer een graaf G ongewogen is, dan is een kortste pad van knoop v naar een knoop u een pad van v naar u dat het kleinste aantal bogen bevat. Omdat bij breedte-eerst zoeken de graaf "laag per laag" overlopen wordt kunnen we het algoritme gemakkelijk aanpassen om voor elke knoop v de lengte van het kortste pad van s naar v terug te geven. Dit gaat als volgt: het kortste pad van de knoop s naar zichzelf heeft uiteraard lengte 0. Telkens wanneer we een knoop w ontdekken m.b.v. de boog (v, w) dan noteren we

de afstand van s tot w als één meer dan de afstand van s tot v. De resulterende pseudocode vind je in Algoritme 5.5.

Algoritme 5.5 Kortste pad in een ongewogen graaf

Invoer Een gerichte of ongerichte ongewogen graaf $G = (V, E)$ met orde $n > 0$. Een knoop s waarvan het zoeken vertrekt. De knopen zijn genummerd van 1 tot n, i.e. $V = \{1, 2, \ldots, n\}$.

Uitvoer De array D met $D[v]$ de kortste afstand van s tot v; als $D[v] = \infty$ dan is er geen pad van s naar v.

```
1:  function KortstePadOngewogen(G, s)
2:      D ← [∞, ∞, ..., ∞]                                  ▷ n keer ∞
3:      D[s] ← 0              ▷ kortste pad van s naar zichzelf heeft lengte 0
4:      Q.init()                                             ▷ wachtrij van knopen
5:      Q.enqueue(s)
6:      while Q ≠ ∅ do
7:          v ← Q.dequeue()
8:          for all w ∈ buren(v) do
9:              if D[w] = ∞ then                             ▷ w nog niet ontdekt
10:                 D[w] ← D[v] + 1
11:                 Q.enqueue(w)
12:             end if
13:         end for
14:     end while
15:     return D
16: end function
```

Voorbeeld 5.31 We bekijken opnieuw de uitvoering van het breedte-eerst algoritme 5.2 in Voorbeeld 5.21. De uitvoering van Algoritme 5.5 ontdekt de knopen in exact dezelfde volgorde. In de tabel hieronder geven we de inhoud van de afstandenarray D weer telkens wanneer regel 6 wordt uitgevoerd.

	1	2	3	4	5	6
(a)	0	∞	∞	∞	∞	∞
(b)	0	1	1	∞	∞	∞
(c)	0	1	1	2	∞	∞
(d)	0	1	1	2	2	∞
(e)	0	1	1	2	2	3

Men controleert gemakkelijk dat de laatste lijn in deze tabel inderdaad de correcte afstanden aanduidt van knoop 1 naar alle (andere) knopen. ∎

5.4.2 Dijkstra's Algoritme

Wanneer de graaf gewogen is, i.e. wanneer elke boog e een gewicht, genoteerd gewicht(e), heeft, dan is men meestal niet geïnteresseerd in het pad met het minste aantal bogen, maar in het pad met het kleinste totale gewicht. In dit geval zal Algoritme 5.5 niet langer het correcte antwoord geven, zelfs niet als we regel 10 van het Algoritme 5.5 aanpassen om rekening te houden met de gewichten van de bogen.

Voorbeeld 5.32 Veronderstel dat we in Algoritme 5.5 regel 10 als volgt herschrijven:

$$D[w] \leftarrow D[v] + \text{gewicht}(v, w).$$

Wanneer we dit algoritme uitvoeren voor de gewogen graaf in Figuur 5.2 startend vanaf knoop 1, dan verandert de afstandenarray als volgt:

	1	2	3	4	5	6
(a)	0	∞	∞	∞	∞	∞
(b)	0	1	2	∞	∞	∞
(c)	0	1	2	6	∞	∞
(d)	0	1	2	6	3	∞
(e)	0	1	2	6	3	10

Men gaat echter eenvoudig na dat de lengte van het kortste pad van knoop 1 naar knoop 4, gewicht 4 heeft en niet 6 zoals dit algoritme aangeeft. Idem voor knoop 6: het kortste pad van knoop 1 naar knoop 6 heeft gewicht 8, i.p.v. 10 zoals aangegeven door dit foutieve algoritme. ∎

Een algoritme dat in het geval *alle gewichten positief zijn* het juiste antwoord geeft is Dijkstra's algoritme. De sleutelideeën van dit algoritme zijn de volgende:

1. Op elk moment in het algoritme zijn de knopen verdeeld in twee disjuncte verzamelingen: een verzameling S van knopen v waarvoor de kortste afstand van s tot v reeds gekend is, en een verzameling Q van knopen waarvoor de kortste afstand nog niet met zekerheid gekend is[3].

[3]Omdat S en Q elkaars complement zijn moet in een implementatie maar één van beide verzamelingen bijgehouden worden.

2. Voor elke knoop v die tot Q behoort houden we steeds de kortste afstand $D[v]$ bij van een pad van s naar v *dat enkel uit knopen van S bestaat, met uitzondering van de laatste knoop*.

3. We voegen telkens dié knoop v van Q toe aan S waarvoor $D[v]$ minimaal is onder alle knopen van Q. Voor de buren w van v die tot Q behoren moeten we eventueel $D[w]$ aanpassen. Het pad van s naar v (dat nu enkel uit knopen van S bestaat) uitgebreid met de boog (v, w) zou eventueel korter kunnen zijn dan het tot dan toe gevonden kortste pad.

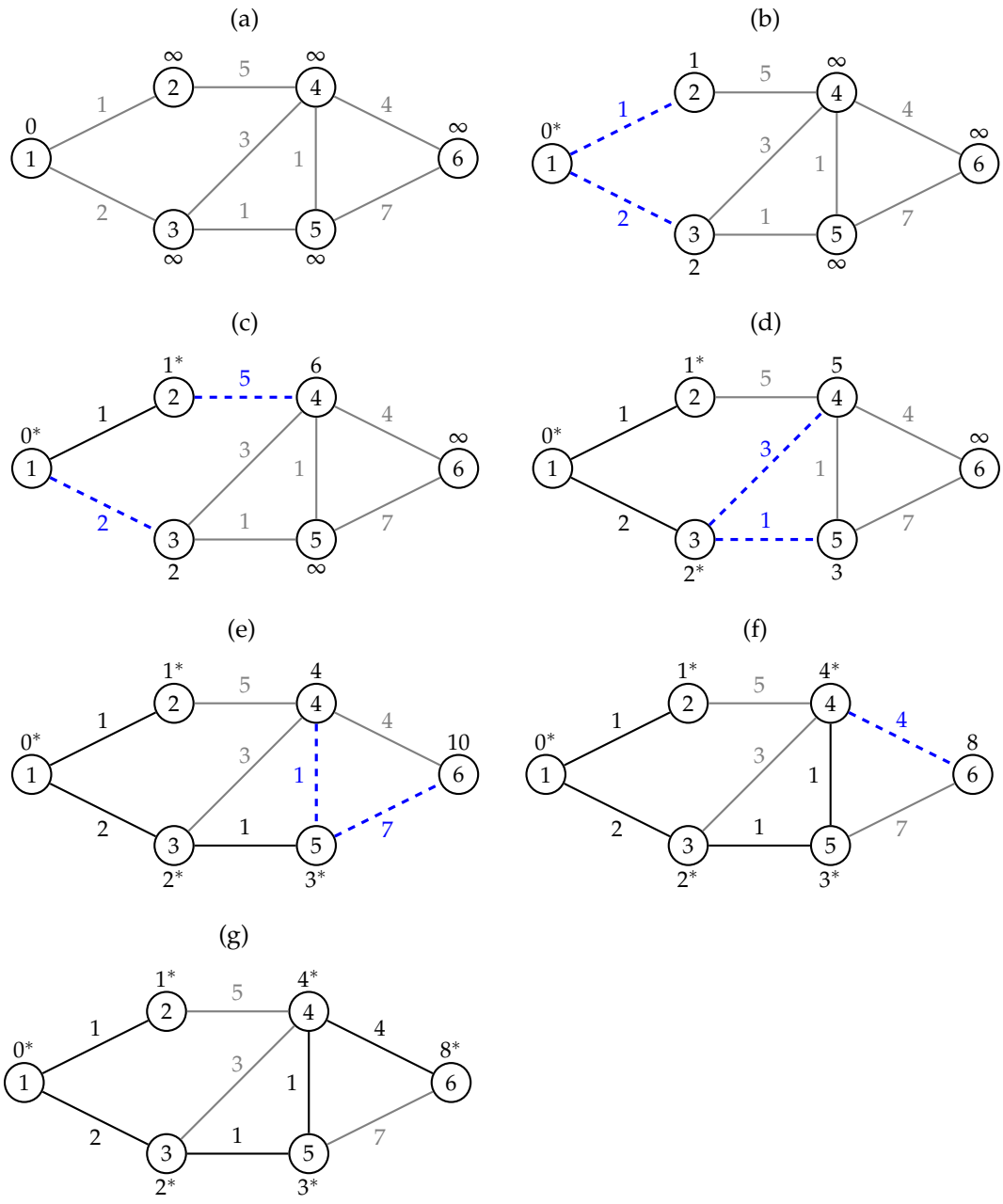
Algoritme 5.6 Het algoritme van Dijkstra

Invoer Een gerichte of ongerichte gewogen graaf $G = (V, E)$ met orde $n > 0$. Alle gewichten zijn positief. Een knoop s waarvan het zoeken vertrekt. De knopen zijn genummerd van 1 tot n, i.e. $V = \{1, 2, \ldots, n\}$.

Uitvoer De array D met $D[v]$ de kortste afstand van s tot v; als $D[v] = \infty$ dan is er geen pad van s naar v.

```
1:  function DIJKSTRA(G, s)
2:      D ← [∞, ∞, ..., ∞]                                  ▷ n keer ∞
3:      D[s] ← 0              ▷ kortste pad van s naar zichzelf heeft lengte 0
4:      Q ← V          ▷ knopen waarvan kortste afstand nog niet is bepaald
5:      while Q ≠ ∅ do
6:          zoek v ∈ Q waarvoor D[v] minimaal is (voor knopen in Q)
7:          verwijder v uit Q
8:          for all w ∈ buren(v) ∩ Q do
9:              if D[w] > D[v] + gewicht(v, w) then
10:                 D[w] ← D[v] + gewicht(v, w)          ▷ korter pad naar w
11:             end if
12:         end for
13:     end while
14:     return D
15: end function
```

Voorbeeld 5.33 We passen Dijkstra's algoritme toe op de graaf in Figuur 5.2. In onderstaande tabel geven we telkens de inhoud van de array D wanneer regel 5 uitgevoerd wordt. Wanneer er een sterretje bij een afstand staat betekent dit dat de overeenkomstige knoop verwijderd is uit Q.



Figuur 5.13: Stap voor stap uitvoering van het algoritme van Dijkstra. In elke stap wordt de tot dan toe gevonden kortste afstand tot knoop 1 aangeduid. Wanneer er een sterretje bij staat dan is dit met 100% zekerheid de kortste afstand. De blauwe bogen in stippellijn geven voor elke knoop u (met afstand verschillend van ∞) aan wat de boog is van een knoop van S naar u die het gevonden kortste pad beëindigt.

	1	2	3	4	5	6
(a)	0	∞	∞	∞	∞	∞
(b)	0^*	1	2	∞	∞	∞
(c)	0^*	1^*	2	6	∞	∞
(d)	0^*	1^*	2^*	5	3	∞
(e)	0^*	1^*	2^*	4	3^*	10
(f)	0^*	1^*	2^*	4^*	3^*	8
(g)	0^*	1^*	2^*	4^*	3^*	8^*

De verschillende stappen worden ook geïllustreerd in Figuur 5.13. Het is interessant om te zien dat bv. voor knoop 4 de afstand tot drie keer toe wordt verlaagd omdat er steeds kortere paden worden ontdekt. ∎

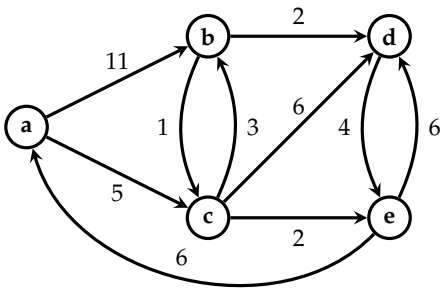
Implementatiedetails

We zien dat in Dijkstra's algoritme er herhaalde minimum berekeningen gebeuren: op regel 6 moet immers herhaaldelijk de knoop v gevonden worden met de minimale waarde voor $D[v]$. We hebben reeds gezien dat een *binaire hoop* hiervoor zeer geschikt is, omdat deze een prioriteitswachtrij implementeert. Een efficiënte implementatie van Dijkstra's algoritme zal dus gebruikmaken van een binaire hoop waarbij steeds aan de volgende twee invarianten voldaan is:

1. De elementen in de hoop zijn de elementen van Q.
2. De sleutel voor elk element v is de waarde van $D[v]$.

Op die manier wordt het ophalen en verwijderen van het minimum op regel 6 een efficiënte bewerking (logaritmisch in het aantal elementen van Q).

Op regel 10 moet echter de sleutelwaarde van w verkleind worden: deze operatie is standaard niet voorzien in een binaire hoop. Wanneer men echter *zou* weten wat de positie is van w in de binaire hoop, dan kan men gemakkelijk de sleutelwaarde aanpassen: dit komt neer op het laten omhoog bubbelen van w tot op zijn juiste plaats. Wat men dus moet doen is een array P bijhouden waarin voor elke knoop v zijn positie in de binaire hoop wordt bijgehouden. Operaties op de binaire hoop moeten dan natuurlijk ook deze array aanpassen (i.e. wanneer bv. twee elementen van plaats worden verwisseld in de binaire hoop dan moeten ook de overeenkomstige posities in de array P worden aangepast).



Figuur 5.14: Een gewogen, gerichte graaf.

5.4.3 Oefeningen

1. In Algoritme 5.5 wordt nu enkel de afstand van elke knoop v tot de startknoop s bijgehouden. In veel toepassingen heeft men echter ook een pad nodig dat deze minimale afstand realiseert.

   a) Pas de pseudo-code van Algoritme 5.5 aan zodanig dat er een tweede array P wordt teruggeven zodanig dat $P[v]$ de knoop geeft die de voorganger (predecessor) is van v op een kortste pad van s naar v.

   b) Pas je aangepaste algoritme toe op de gerichte graaf in Figuur 5.9 startend vanaf knoop 1. Ga ervan uit dat knopen steeds worden bezocht in stijgende volgorde. Hoe zit de array P er na afloop uit?

   c) Schrijf een algoritme dat als invoer de array P neemt en een knoop v. Het algoritme geeft een lijst terug die het kortste pad van s naar v bevat (in de juiste volgorde).

2. Beschrijf hoe je volgend probleem kan oplossen als een kortste pad probleem. Gegeven een lijst van Engelstalige 5-letterwoorden. Woorden worden *getransformeerd* door juist één letter van het woord te vervangen door een andere letter. Geef een algoritme dat nagaat of een woord w_1 omgezet kan worden in een woord w_2. Indien dit het geval is dan moet je algoritme ook de tussenliggende woorden tonen voor de kortste sequentie van transformaties die w_1 in w_2 omzet.

3. Vind voor de graaf in Figuur 5.4 de lengte van het kortste pad van Brugge naar alle andere steden. Voer hiertoe het algoritme van Dijkstra uit.

4. Vind voor de graaf in Figuur 5.14 (de lengte van) het kortste pad van de knoop *a* naar alle andere knopen. Voer hiertoe het algoritme van Dijkstra uit (en houd ook bij wat de kortste paden zijn).

5.5 Minimale Kost Opspannende Bomen

5.5.1 Minimale Kost Opspannende Bomen

Veronderstel dat een wegennetwerk gegeven is dat ervoor zorgt dat we van elke stad naar elke andere stad kunnen rijden, dan kunnen we ons afvragen welke wegen *essentieel* zijn om van elke stad naar elke andere stad te kunnen rijden. Wanneer het *aantal* wegen minimaal is, dan spreken we van een opspannende boom.

Definitie 5.34 Een OPSPANNENDE BOOM van een ongerichte graaf $G = (V, E)$ is een verzameling van bogen T, met $T \subseteq E$, zodanig dat $G' = (V, T)$ een pad heeft tussen elke twee knopen van V, en zodanig dat G' geen enkelvoudige cykels heeft. ∎

In het algemeen heeft een ongerichte graaf *veel* opspannende bomen. Wanneer de graaf gewogen is, dan zijn we vaak geïnteresseerd in de opspannende boom (of bomen) met het minimale gewicht, waarbij het gewicht (of de kost) van de boom gedefinieerd wordt als de som van de gewichten van zijn bogen, dus

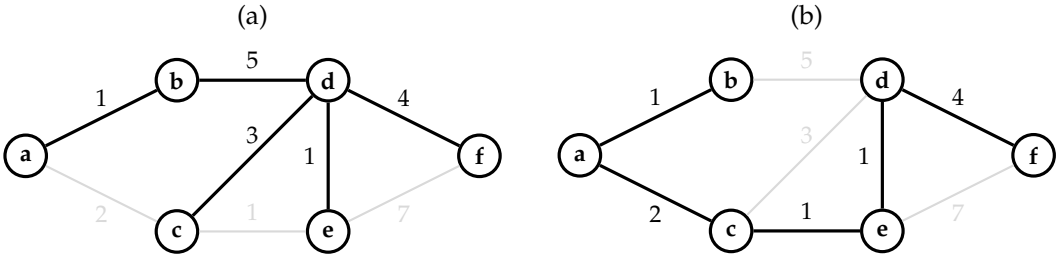
$$\text{gewicht}(T) = \sum_{t \in T} \text{gewicht}(t).$$

Definitie 5.35 Een MINIMALE KOST OPSPANNENDE BOOM T van de graaf G is een opspannende boom zodanig dat voor alle andere opspannende bomen T' van G geldt dat

$$\text{gewicht}(T) \leq \text{gewicht}(T').$$ ∎

Voorbeeld 5.36 In Figuur 5.15 ziet men twee opspannende bomen voor dezelfde graaf. Men controleert gemakkelijk dat dit inderdaad opspannende bomen zijn:

1. Het is duidelijk dat ze geen cykel bevatten.



Figuur 5.15: Twee opspannende bomen. De donkere bogen zijn deel van de opspannende boom.

2. Men ziet eenvoudig dat er een pad is tussen elke twee knopen van de graaf.

Voor de opspannende boom in Figuur 5.15(a) zien we dat

$$\begin{aligned}\text{gewicht}(T) &= \sum_{t \in T} \text{gewicht}(t) \\ &= 1 + 5 + 3 + 1 + 4 \\ &= 14.\end{aligned}$$

Het gewicht van de boom in Figuur 5.15(b) is

$$\begin{aligned}\text{gewicht}(T') &= \sum_{t \in T'} \text{gewicht}(t) \\ &= 1 + 2 + 1 + 1 + 4 \\ &= 9.\end{aligned}$$

De boom T is dus zeker *geen* minimale kost opspannende boom, want de boom T' heeft een lager gewicht. ∎

5.5.2 Prims Algoritme

Het algoritme voor generiek zoeken kan eenvoudig aangepast worden om een opspannende boom terug te geven. Inderdaad, in Figuur 5.6 ziet men in de laatste deelfiguur een opspannende boom. Het algoritme voor generiek zoeken genereert dus blijkbaar reeds een opspannende boom. We dienen enkel bij te houden welke bogen gekozen worden. Door de bogen op een *gulzige* manier te kiezen, i.e. door steeds de goedkoopste boog te kiezen die

het ontdekte gebied uitbreidt, bekomt men een minimale kost opspannende boom. Dit is de essentie van het algoritme van Prim.

Het is zo dat, in tegenstelling tot het algoritme voor generiek zoeken, er geen speciale startknoop is. Het blijkt dat dit niet uitmaakt: het algoritme werkt correct (zonder bewijs!) voor om het even welke startknoop.

Algoritme 5.7 Prims algoritme voor een minimale kost opspannende boom

Invoer Een ongerichte gewogen graaf $G = (V, E)$ met orde $n > 0$. De knopen zijn genummerd van 1 tot n, i.e. $V = \{1, 2, \ldots, n\}$.

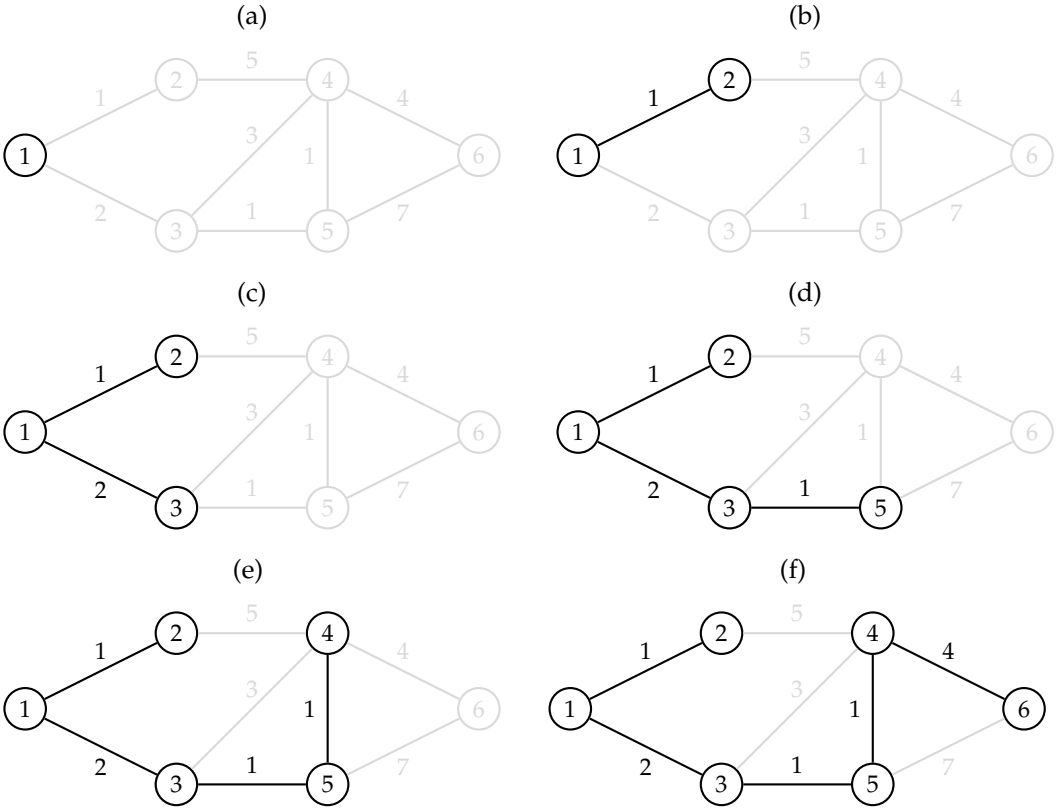
Uitvoer Een verzameling T van bogen die een minimale kost opspannende boom is.

```
1:  function PRIM(G)
2:      D ← [false, false, ..., false]                    ▷ n keer false
3:      D[1] ← true                                        ▷ kies knoop 1 als startknoop
4:      T ← ∅                                              ▷ gekozen bogen
5:      while ∃(u, v): D[u] = true ∧ D[v] = false do
6:          kies (u, v) met D[u] = true ∧ D[v] = false met minimaal gewicht
7:          D[v] ← true
8:          T ← T ∪ {(u, v)}                               ▷ Voeg boog (u, v) toe aan boom
9:      end while
10:     return T
11: end function
```

Voorbeeld 5.37 We voeren Prims algoritme uit op de gewogen graaf van Figuur 5.2. In onderstaande tabel kunnen we zien in welke volgorde de bogen worden gekozen:

gemarkeerde knopen	mogelijke bogen	gekozen boog
$\{1\}$	$(1,2), (1,3)$	$(1,2)$
$\{1,2\}$	$(1,3), (2,4)$	$(1,3)$
$\{1,2,3\}$	$(2,4), (3,4), (3,5)$	$(3,5)$
$\{1,2,3,5\}$	$(2,4), (3,4), (5,4), (5,6)$	$(5,4)$
$\{1,2,3,4,5\}$	$(4,6), (5,6)$	$(4,6)$
$\{1,2,3,4,5,6\}$	geen	geen

Figuur 5.16 illustreert dit proces. ∎



Figuur 5.16: Stap voor stap uitvoering van Prims algoritme. De knopen waarvoor $D[v] =$ **true** alsook de inhoud van T worden getoond telkens regel 5 wordt uitgevoerd.

5.5.3 Kruskals Algoritme

Kruskals algoritme is een alternatief algoritme om het probleem van het vinden van een minimale kost opspannende boom op te lossen. Net als Prims algoritme is het een gulzig algoritme, maar daar waar in Prims algoritme de gekozen bogen steeds met elkaar verbonden zijn is dat in Kruskals algoritme niet het geval.

Samengevat komt het algoritme van Kruskal erop neer dat men eerst de bogen sorteert volgens stijgend gewicht. Vervolgens loopt men deze lijst van bogen af, waarbij men telkens een boog selecteert op voorwaarde dat deze geen cykel veroorzaakt onder de bogen die reeds gekozen zijn.

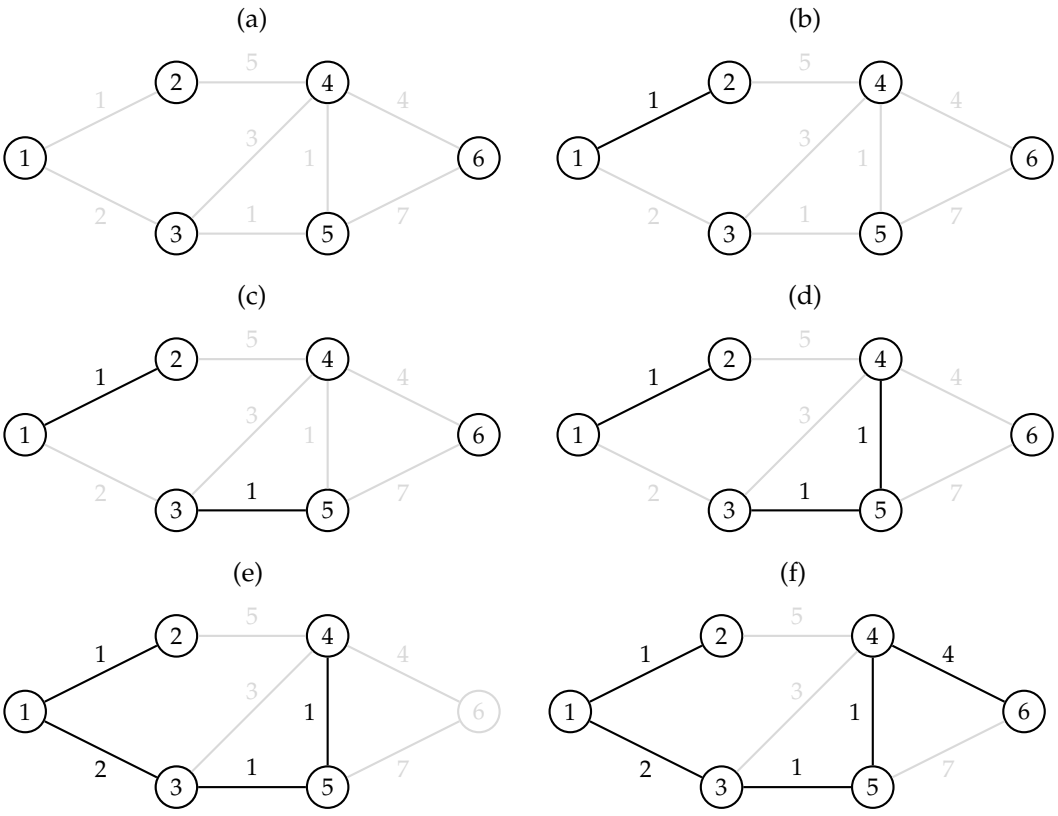
Voorbeeld 5.38 We voeren Kruskals algoritme uit op de graaf in Figuur 5.2.

Algoritme 5.8 Kruskals algoritme voor een minimale kost opspannende boom

Invoer Een ongerichte gewogen graaf $G = (V, E)$ met grootte $n > 0$. De knopen zijn genummerd van 1 tot n, i.e. $V = \{1, 2, \ldots, n\}$.

Uitvoer Een verzameling T van bogen die een minimale kost opspannende boom is.

```
1:  function KRUSKAL(G)
2:      T ← ∅                                   ▷ Start met lege boom
3:      E' ← sorteer E volgens stijgend gewicht
4:      for all e' ∈ E' do
5:          if T ∪ e' heeft geen cykel then
6:              T ← T ∪ e'
7:          end if
8:      end for
9:      return T
10: end function
```



Figuur 5.17: Stap voor stap uitvoering van Kruskals algoritme.

We sorteren de bogen volgens stijgend gewicht en we krijgen:

$$(1,2),(3,5),(4,5),(1,3),(3,4),(4,6),(2,4),(5,6).$$

Kruskals algoritme overloopt deze bogen één voor één en wanneer een boog geen cyckel veroorzaakt onder de reeds geselecteerde bogen wordt deze ook geselecteerd.

De eerste vier bogen worden zonder probleem geselecteerd. De volgende boog die in aanmerking komt is boog $(3,4)$, maar dan zouden we een cykel krijgen, namelijk $(3,5,4,3)$ (zie Figuur 5.17(e)), en dus wordt deze boog *niet* gekozen. De boog $(4,6)$ wordt wel gekozen. De laatste twee bogen tenslotte worden ook niet gekozen want deze veroorzaken allebei een cykel onder de reeds gekozen bogen. ∎

Opmerking 5.39 Merk op dat in de beschrijving van het algoritme *niet* wordt gezegd hoe we een cykel gaan detecteren. Dit kan m.b.v. (een variant van) diepte-eerst zoeken, maar er bestaan efficiëntere manieren om dit te doen. Dit wordt hier verder niet behandeld. ∎

5.5.4 Oefeningen

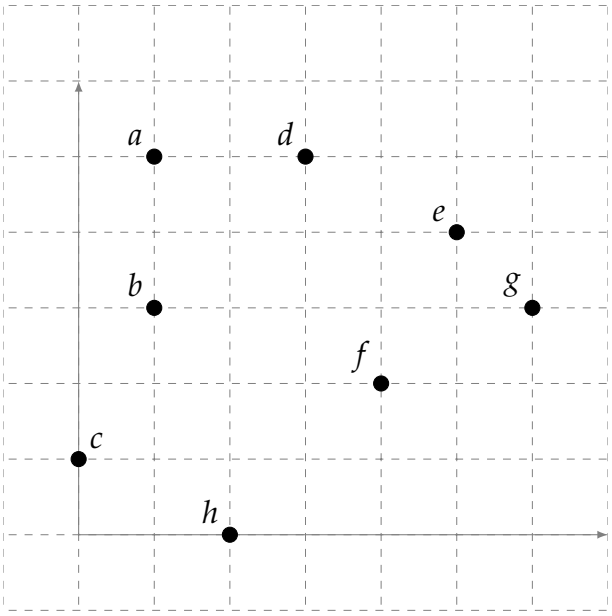
1. Vind een minimale kost opspannende boom m.b.v. het algoritme van Prim voor de graaf in Figuur 5.4. Neem als startknoop "Brugge".
2. Vind een minimale kost opspannende boom m.b.v. het algoritme van Kruskal voor de graaf in Figuur 5.4.

5.6 Het Handelsreizigersprobleem

Definitie 5.40 Het HANDELSREIZIGERSPROBLEEM is het volgende: gegeven een complete[4] gewogen ongerichte graaf G met niet-negatieve gewichten, vind dan een ordening van de knopen zodanig dat elke knoop juist éénmaal wordt bezocht (behalve de start- en eindknoop die samenvallen) en zodanig dat de som van de gewichten van de gekozen bogen minimaal is. ∎

[4]Een graaf is compleet als er een boog is tussen elke twee (verschillende) knopen.

Het handelsreizigersprobleem is van een veel grotere moeilijkheidsgraad dan de andere problemen die we reeds hebben besproken in de grafentheorie. Het is een zogenaamd NP-COMPLEET probleem[5], wat hoogstwaarschijnlijk betekent dat er *geen* polynomiaal algoritme bestaat dat alle gevallen correct kan oplossen. Men beperkt zich dan ook vaak tot snellere maar *benaderende* algoritmen, waarvan men hoopt (of kan bewijzen) dat de oplossing die ze produceren niet te ver afwijkt van de optimale oplossing.



Figuur 5.18: Acht steden in het vlak. Iedere stad is met iedere andere stad verbonden via een rechte lijn (lijnen niet getekend).

Voorbeeld 5.41 We beschouwen een graaf met 8 knopen die allen geheeltallige coördinaten hebben, zoals aangegeven in Figuur 5.18. De afstand tussen twee knopen (i.e. het gewicht van de overeenkomstige boog) wordt gegeven door de afstand in rechte lijn tussen de twee knopen. De adjacen-

[5]Juister gezegd: het overeenkomstig *beslissingsprobleem* ("bestaat er een rondreis met kost hoogstens k?") is NP-compleet. Het algemene probleem is echter minstens zo moeilijk als het beslissingsprobleem.

tiematrix (met gewichten) van de graaf is dus

$$A = \begin{pmatrix} 0 & 2 & \sqrt{17} & 2 & \sqrt{17} & \sqrt{18} & \sqrt{29} & \sqrt{26} \\ 2 & 0 & \sqrt{5} & \sqrt{8} & \sqrt{17} & \sqrt{10} & 5 & \sqrt{10} \\ \sqrt{17} & \sqrt{5} & 0 & 5 & \sqrt{34} & \sqrt{17} & \sqrt{40} & \sqrt{5} \\ 2 & \sqrt{8} & 5 & 0 & \sqrt{5} & \sqrt{10} & \sqrt{13} & \sqrt{26} \\ \sqrt{17} & \sqrt{17} & \sqrt{34} & \sqrt{5} & 0 & \sqrt{5} & \sqrt{2} & 5 \\ \sqrt{18} & \sqrt{10} & \sqrt{17} & \sqrt{10} & \sqrt{5} & 0 & \sqrt{5} & \sqrt{8} \\ \sqrt{29} & 5 & \sqrt{40} & \sqrt{13} & \sqrt{2} & \sqrt{5} & 0 & 5 \\ \sqrt{26} & \sqrt{10} & \sqrt{5} & \sqrt{26} & 5 & \sqrt{8} & 5 & 0 \end{pmatrix}$$

Aangezien de gewichten gebaseerd zijn op de Euclidische afstand zal het steeds korter (of juister gezegd: niet langer) zijn om rechtstreeks van een knoop v naar w te gaan dan om een omweg te maken via een knoop u. Wiskundig uitgedrukt voldoet deze graaf aan de driehoeksongelijkheid. ∎

Definitie 5.42 Een graaf G voldoet aan de DRIEHOEKSONGELIJKHEID wanneer voor alle knopen u, v en w geldt dat

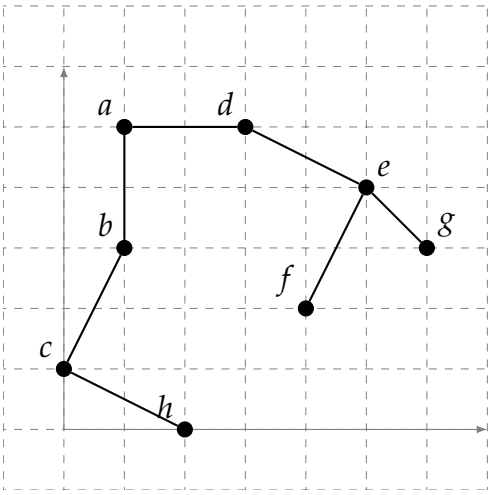
$$\text{gewicht}(v, w) \leq \text{gewicht}(v, u) + \text{gewicht}(u, w).$$ ∎

Een graaf voldoet m.a.w. aan de driehoeksongelijkheid wanneer rechtstreeks van v naar w gaan nooit langer is dan een omweg nemen via u. Wanneer de gewichten afgeleid zijn van de Euclidische afstand dan is steeds aan de driehoeksongelijkheid voldaan.

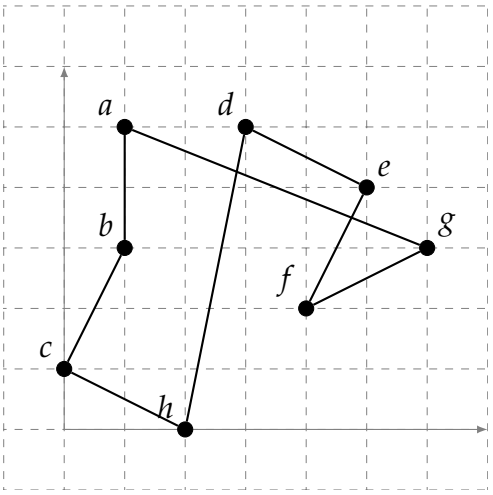
Wanneer de graaf aan de driehoeksongelijkheid voldoet dan geeft volgend *benaderingsalgoritme* een oplossing die hoogstens tweemaal zolang is als de optimale oplossing (zonder bewijs). Deze garantie is echter niet geldig wanneer de graaf niet aan de driehoeksongelijkheid voldoet.

1. Bereken een minimale kost opspannende boom T voor de graaf.
2. Kies willekeurig een wortel r van deze boom.
3. Geef de cykel terug die correspondeert met het in *preorde* doorlopen van deze boom.

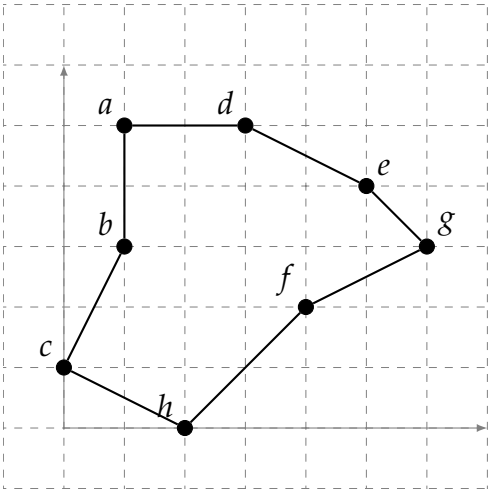
(a) Minimale Kost Opspannende boom



(b) Benaderende oplossing



(c) Optimale oplossing



Figuur 5.19: Illustratie van het benaderende algoritme voor het handelsreizigersprobleem.

Voorbeeld 5.43 In Figuur 5.19(a) is een minimale kost opspannende boom gegeven. Deze boom werd bekomen m.b.v. Prims algoritme waarbij bij gelijke kost steeds de lexicografische kleinste boog werd gekozen. De startknoop was knoop a.

Als we dan als wortel van deze boom a nemen, en de knopen van de boom in preorde doorlopen (met de kinderen lexicografisch geordend), dan vinden we:

$$a, b, c, h, d, e, f, g.$$

Dit geeft dan aanleiding tot de rondreis in Figuur 5.19(b). De kost van deze rondreis is:

$$2 + \sqrt{5} + \sqrt{5} + \sqrt{26} + \sqrt{5} + \sqrt{5} + \sqrt{5} + \sqrt{29} \approx 23.66.$$

De optimale oplossing wordt gegeven in Figuur 5.19(c), en deze rondreis heeft de volgende kost:

$$2 + \sqrt{5} + \sqrt{2} + \sqrt{5} + \sqrt{8} + \sqrt{5} + \sqrt{5} + 2 \approx 17.19.$$

De benaderende rondreis is dus ongeveer 38% langer dan de optimale rondreis. ∎

5.6.1 Oefeningen

1. Beschouw opnieuw de acht steden in Figuur 5.18, maar veronderstel nu dat het gewicht van een boog gegeven wordt door de zogenaamde Manhattan-distance tussen de twee knopen, dus

$$d\big((x_1, y_1), (x_2, y_2)\big) = |x_2 - x_1| + |y_2 - y_1|$$

   a) Ga na dat de Manhattan-distance aan de driehoeksongelijkheid voldoet. **Hint**: voor de absolute waarde geldt dat

$$|x + y| \leq |x| + |y|.$$

   b) Pas het benaderende algoritme voor het oplossen van het handelsreizigersprobleem toe op dit probleem. Gebruik Kruskals algoritme om de minimale opspannende boom te construeren. Wanneer meerdere bogen kunnen gekozen worden, kies dan steeds de lexicografisch kleinste boog. Neem de knoop a als wortel van de opspannende boom.

Deel III

Operationeel Onderzoek

HOOFDSTUK 6

Inleiding tot Operationeel Onderzoek

We starten dit hoofdstuk met een heel beknopte inleiding m.b.t. modelbouw en operationeel onderzoek. Daarna geven we een eerste voorbeeld van een LINEAIR PROGRAMMERINGSPROBLEEM (LP-probleem) in twee dimensies en bespreken de oplossing. In de oefeningen wordt er gevraagd om bijkomende LP-problemen op te stellen.

6.1 Wat is Operationeel Onderzoek?

Operationeel onderzoek (Eng. *Operations research* of *Operational Research*) is een technische discipline met als doel het ondersteunen van beslissingen. Hiervoor gebruikt men de beste beschikbare methode en in die zin is het een pragmatische discipline. In de praktijk gaat men dan ook vaak de exactheid van de oplossing gaan afwegen tegenover de kost/duur om een oplossing te bekomen. De bedoeling is om beslissingen te kunnen nemen op een manier die wetenschappelijk onderbouwd is, zodanig dat men niet overgeleverd is aan "gut feeling".

De oplossingsmethodes worden geïmplementeerd in software; er is dus ook een duidelijke link met de informatica.

Operationeel onderzoek is een zeer uitgebreid gebied waarover men gemakkelijk een "inleidend" boek van meerdere honderden pagina's schrijft. In operationeel onderzoek worden inzichten en begrippen uit de wiskunde, statistiek, speltheorie, grafentheorie, modellering en simulering enzovoort

gebruikt.

Eén van de eerste en een populaire definitie van operationeel onderzoek luidt als volgt:

> Operational research is a scientific method of providing executive departments with a quantitative basis for decisions regarding the operations under their control.

De "departments" waarvan hier sprake is zijn militaire afdelingen en de "operations" zijn de militaire operaties. Men ziet in deze definitie duidelijk de reeds opgesomde karakteristieken naar voren komen, zoals het feit dat het gaat over het ondersteunen van beslissingen op een kwantitatieve basis.

Tijdens de tweede wereldoorlog hielden teams van wetenschappers in Engeland en Amerika zich bezig met vragen als:

- Wat is de optimale grootte van een konvooi vrachtschepen en hoeveel begeleidende marineschepen moet men meesturen wannneer men het effect van de aanvallen van de vijandelijke onderzeeërs zo klein mogelijk wil maken?
- Op welke manier moeten zeemijnen worden gelegd als men de kans op het treffen van een door een mijnenveld varend vijandelijk schip zo groot mogelijk wil maken?
- Op welke plaatsen moet men bommenwerpers van extra bepantsering voorzien? Dit deed men door het analyseren van de kogelgaten in bommenwerpers wanneer die terug op de basis kwamen.

Tegenwoordig zijn er uiteraard en gelukkig ook veel vredelievende toepassingen zoals

- plannings- en scheduling problemen (supply chain analyse)
- de optimalisatie van layout van bv. magazijnen en productiehallen
- het ontwerpen van complexe (telecom) netwerken
- het ontwerpen van chips
- ...

6.2 Wiskundige vorm van een OR probleem

Heel wat vraagstukken binnen operationeel onderzoek kunnen geformuleerd worden als:

> Vind het optimum (maximum of minimum) van een bepaalde functie terwijl de onafhankelijke beslissingsvariabelen binnen bepaalde grenzen blijven.

De oplossingsmethode die men kan gebruiken om het probleem op te lossen hangt over het algemeen af van de volgende zaken:

1. Is de functie die moet geoptimaliseerd worden lineair of niet? Als ze niet lineair is, heeft ze dan een andere speciale vorm of eigenschap? Gaat het bv. over een kwadratische of een convexe functie?
2. Kunnen de beslissingsvariabelen reële waarden aannemen of kunnen ze enkel gehele of eventueel zelfs enkel binaire waarden aannemen?
3. Zijn de beperkingen waarin de beslissingsvariabelen moeten voldoen lineair of niet? Gaat het over "harde" of "zachte" beperkingen?
   - Een voorbeeld van een harde beperking is: bij het opstellen van een lessenrooster (een typisch OR-probleem) kan dezelfde lesgever geen twee lessen tegelijkertijd geven. Een lessenrooster waarin dit het geval zou zijn is een ongeldig lessenrooster.
   - Een voorbeeld van een zachte beperking is: de lesgever van het vak Probleemoplossend Denken II geeft liever niet om 8u15 les.
4. Bij het beslissen welke methode men zal gebruiken dient men ook rekening te houden met de kwaliteit van de informatie waarover men beschikt. Het heeft bv. weinig zin om veel tijd en energie te besteden aan het vinden van de exacte oplossing wanneer de waarden van de parameters hoogst onzeker zijn.

6.3 Modelbouwcyclus

Om een bepaald probleem in de praktijk te gaan aanpakken wordt vaak een (iteratief) proces gebruikt dat bestaat uit een aantal stappen. Deze stappen worden hieronder kort besproken.

1. **Specificeer het probleem.** Praktische OR-problemen zijn, in tegenstelling tot vraagstukken in boeken en cursussen, vaak nogal vaag en onduidelijk geformuleerd. In eerste instantie moet het relevante systeem bestudeerd worden en moet er gezorgd worden voor een duidelijke probleemomschrijving. Dit betekent onder andere dat men de juiste doelen moeten vooropstellen, dat men moet begrijpen wat de restricties zijn op wat er mogelijk is, dat eventuele relaties tussen het onderzoeksonderwerp en andere delen van de organisatie bepaald worden, enzovoort.

   Bij het beschrijven van het probleem worden kwantitatieve grootheden (zoals lengte, oppervlakte, hoeveelheden, ratio's, enzovoort) gebruikt.

   Dit is een cruciale stap: *het is immers zeer moeilijk om het "juiste" antwoord te verkrijgen uit het "verkeerde" probleem!*

2. **Stel het wiskundig model op.** In deze stap wordt het probleem dat in de vorige stap werd beschreven omgezet naar een vorm waarop wiskundige analyse kan toegepast worden. De kwantitatieve grootheden uit de vorige stap zullen de BESLISSINGSVARIABELEN worden; het doel is dan een functie van deze beslissingsvariabelen en wordt de DOELFUNCTIE genoemd. Typisch zullen de beslissingsvariabelen aan een aantal BEPERKINGEN moeten voldoen. Behalve de beslissingsvariabelen zullen in het model ook een heel aantal parameters (zoals coëfficiënten bij de beslissingsvariabelen en de rechterleden van de restricties) aanwezig zijn.

   Bij het opstellen van het wiskundig model moet ook steeds de afweging worden gemaakt hoe gedetailleerd en/of ingewikkeld het model mag/kan zijn. Meer eenvoudige modellen (bv. met lineaire restricties en lineaire doelfunctie) zijn eenvoudiger op te lossen dan meer ingewikkelde modellen. Aan de andere kant zullen de meer eenvoudige modellen ook minder goed overeenkomen met de realiteit.

3. **Los op én interpreteer.** In deze stap wordt het model opgelost. Bij het interpreteren van het model bekijkt men bv. het volgende. Ligt de oplossing op één van de restricties? Wat betekent dit dan? Wanneer de correcte waarden van de parameters in het model mogelijks onzeker zijn dan is het ook interessant om een SENSITIVITEITSANALYSE uit te voeren. Hier wordt bekeken wat het effect is op de oplossing van

een verandering in de parameters. Over het algemeen wil men dat een kleine verandering van de parameters in het model ook slechts een kleine invloed heeft op de oplossing. Het wordt problematisch wanneer een kleine verandering van de parameters een (zeer) grote invloed heeft op de oplossing.

4. **Vergelijk met de realiteit.** In deze stap wordt nagegaan of de bekomen oplossing realistisch is; men gaat ook na of men na implementatie de verwachte resultaten krijgt. Indien dit niet zo is kan men eventueel trachten te achterhalen welke beïnvloedingsfactoren over het hoofd werden gezien.

5. **Stuur indien nodig het model en/of probleemomschrijving bij.** Indien blijkt dat de oplossing niet voldoende overeenkomt met de realiteit dan kan gekeken worden of een ander (meer ingewikkeld) model soelaas kan brengen. Als dat niet het geval is moet men nagaan of eventueel de probleemomschrijving moet worden aangepast.

In deze cursus beperken wij ons hoofdzakelijk tot stappen (2) en (3); de problemen die wij bekijken in de cursus zijn in het algemeen ook veel kleiner dan de problemen die men in de praktijk ontmoet.

We geven nu een voorbeeld van een probleem dat vaak gebruikt zal worden doorheen deze tekst.

Voorbeeld 6.1 (Boer Frans) Boer Frans is een landbouwer en roept jouw hulp in om de optimale benutting van zijn land te bepalen. Boer Frans heeft in totaal 20 hectare grond ter beschikking en hij heeft de kennis en het materiaal om tarwe en/of gerst te verbouwen. De verkoopprijzen voor tarwe en gerst worden respectievelijk gegeven door 25 EUR en 20 EUR per 100 kg. De opbrengsten voor tarwe en gerst zijn respectievelijk 1000 kg en 1500 kg per hectare. De bovenstaande gegevens kunnen samengevat worden in de volgende tabel:

Gewas	Verkoopprijs	Opbrengst
Tarwe	25 EUR/100 kg	1000 kg/ha
Gerst	20 EUR/100 kg	1500 kg/ha

Zowel tarwe en gerst hebben nood aan meststoffen en pesticiden, elk met hun eigen aanbevolen hoeveelheid. Door de milieuwetgeving mag boer

Frans echter (gemiddeld) over zijn land hoogstens een bepaalde hoeveelheid mest en pesticiden gebruiken. De gegevens m.b.t. de meststoffen en pesticiden worden samengevat in de onderstaande tabel.

Wat?	Nodig tarwe	Nodig gerst	Prijs	Maximum
Bemesting	10 kg/ha	20 kg/ha	2 EUR/kg	15 kg/ha
Pesticide	2 kg/ha	1kg/ha	5 EUR/kg	1.8 kg/ha

Opstellen model Boer Frans moet beslissen hoeveel hectare tarwe en hoeveel hectare gerst hij zal kweken. Het is dus zinvol om twee beslissingsvariabelen x_t en x_g te definiëren met de volgende betekenis:

$$x_t: \text{aantal hectare tarwe}$$
$$x_g: \text{aantal hectare gerst.}$$

Aangezien het kweken van de tarwe en gerst voor boer Frans een economische activiteit is betekent een optimale benutting van zijn land dat zijn winst zo groot mogelijk is. Verschillende combinaties van x_t en x_g zullen boer Frans een andere winst opleveren, m.a.w. de gemaakte winst is een functie van de beslissingsvariabelen x_t en x_g. We stellen nu een formule op voor de winst. In het algemeen is

$$\text{winst} = \text{opbrengst} - \text{kosten}.$$

Wanneer er x_t (resp. x_g) hectare tarwe (resp. gerst) wordt gekweekt dan zal deze $1000x_t$ kg tarwe (resp. $1500x_g$ kg gerst) opleveren, die kan verkocht worden voor 25 EUR per 100 kg (resp. 20 EUR per 100 kg). De opbrengst is m.a.w.

$$\text{opbrengst} = 1000x_t \times \frac{25}{100} + 1500x_g \times \frac{20}{100} = 250x_t + 300x_g.$$

Boer Frans moet echter kosten maken om de meststoffen en de pesticide te kopen. Voor een bepaalde combinatie van x_t en x_g heeft boer Frans de volgende hoeveelheid meststoffen nodig (in kg):

$$\text{meststoffen} = 10x_t + 20x_g$$

en de volgende hoeveelheid pesticiden (in kg):

$$\text{pesticiden} = 2x_t + x_g.$$

De kosten (in EUR) gemaakt door boer Frans worden m.a.w. gegeven door:

$$\begin{aligned}\text{kosten} &= \text{meststoffen} \times 2 + \text{pesticiden} \times 5. \\ &= (10x_t + 20x_g) \times 2 + (2x_t + x_g) \times 5 \\ &= 30x_t + 45x_g.\end{aligned}$$

De winst gemaakt door boer Frans wordt m.a.w. gegeven door:

$$\text{winst} = 220x_t + 255x_g.$$

Op dit moment lijkt het alsof boer Frans oneindig rijk kan worden: immers hoe groter x_t en x_g hoe meer winst er wordt gemaakt. Natuurlijk kunnen x_t en x_g niet zomaar alle willekeurige waarden aannemen maar moeten ze aan een aantal beperkingen voldoen, die er dan voor zullen zorgen dat boer Frans niet oneindig rijk kan worden.

De eerste beperking is uiteraard het feit dat boer Frans slechts 20 hectare land ter beschikking heeft:

$$x_t + x_g \leq 20.$$

Verder mag boer Frans niet te veel meststoffen gebruiken. Aangezien hij 20 hectare land ter beschikking heeft mag hij maximaal $15 \times 20 = 300$ kg meststoffen gebruiken, of

$$10x_t + 20x_g \leq 300.$$

Voor de pesticiden vinden we op een gelijkaardige manier dat

$$2x_t + x_g \leq 36.$$

Ten slotte is het fysisch onmogelijk om een negatief aantal hectare van een bepaald gewas te kweken. Er moet m.a.w. ook gelden dat

$$x_t \geq 0 \quad \text{en} \quad x_g \geq 0.$$

Samenvattend moet het volgende optimalisatieprobleem opgelost worden:

$$\max D(x_t, x_g) = 220x_t + 255x_g$$

onder de beperkingen:

$$\begin{cases} x_t + x_g \leq 20 & \text{landbeperking} \\ 10x_t + 20x_g \leq 300 & \text{mestbeperking} \\ 2x_t + x_g \leq 36 & \text{pesticidebeperking} \end{cases}$$

en waarbij ook de niet-negativiteitsvoorwaarden van kracht zijn:

$$x_t \geq 0 \quad \text{en} \quad x_g \geq 0.$$

Vereenvoudigingen Het is zinvol om eens stil te staan bij een aantal vereenvoudigingen van dit model t.o.v. de werkelijkheid.

- De werkuren zijn niet in het model opgenomen, m.a.w. boer Frans heeft tijd zat.
- Er wordt geen rekening gehouden met "economies of scale" nl. dat het produceren goedkoper wordt voor grotere oppervlakten van hetzelfde product.
- Er wordt vanuit gegaan dat de opbrengstprijzen perfect gekend zijn[1].
- Er wordt geen rekening gehouden met het verhoogd risico dat monocultuur met zich meebrengt. Dit probleem kan eventueel opgevangen worden door een "strategische mix"-beperking toe te voegen.
- We bemesten steeds precies de aanbevolen hoeveelheid.
- Er werd aangenomen dat voor de beperking op pesticide en mest het gemiddelde over de totale oppervlakte mag genomen worden.
- ...

Oplossing Met de technieken die we later in de cursus zullen bespreken vindt men dat de optimale oplossing van het probleem gegeven wordt door:

$$x_t = 10, \quad \text{en} \quad x_g = 10 \quad \text{met bijhorende winst } D = 4750.$$

We controleren dat aan de beperkingen voldaan is:

$$\begin{cases} x_t + x_g = 20 \leq 20 & \text{landbeperking is actief} \\ 10x_t + 20x_g = 300 \leq 300 & \text{mestbeperking is actief} \\ 2x_t + x_g = 30 \leq 36 & \text{pesticidebeperking is niet actief} \end{cases}$$

We zien dat aan alle beperkingen voldaan is, maar dat voor bepaalde beperkingen linker- en rechterlid gelijk zijn. Deze beperkingen worden de ACTIEVE BEPERKINGEN genoemd.

[1] Dit zou je eventueel kunnen bereiken door op voorhand een contract af te sluiten.

Interpretatie We zien dat de optimale verhouding gegeven wordt door 10 hectare tarwe en 10 hectare gerst. Als boer Frans zijn winst wil verhogen moet hij op zoek naar zaaigoed waarvoor de bemestingsbehoefte lager is (omdat de mestbeperking actief is); zaaigoed met een lagere pesticidebehoefte zal zijn winst veel minder beïnvloeden. Inderdaad, wanneer beide soorten zaaigoed 10% minder bemesting nodig hebben dan verschuift de optimale mix naar $x_t = 6,667$ en $x_g = 13,333$ terwijl de winst verhoogt naar 4933 EUR. Wanneer beide soorten 10% minder pesticide nodig hebben dan blijft de optimale mix ongewijzigd en de winst stijgt een klein beetje naar 4765 EUR omdat de kosten voor de pesticiden ietsje gedaald zijn. ∎

6.4 Oefeningen

1. Een bedrijf vervaardigt twee soorten broeken, type A en type B. De stof van A-broeken kost 25 EUR per broek, die van de B-broeken 20 EUR per broek. Een arbeider werkt 60 minuten aan een A-broek, 20 minuten aan een B-broek. De verkoopprijzen bedragen respectievelijk 95 EUR en 60 EUR per broek. Het bedrijf heeft 8 arbeiders in dienst die maximaal 8 uur per dag werken aan 30 EUR per uur. Verder zijn er nog 2400 EUR vaste kosten per dag. Technische werkloosheid is niet mogelijk[2]. Uit marktonderzoek blijkt dat van de A-broeken ten hoogste 60 stuks per dag verkocht kunnen worden, en van de B-broeken hoogstens 100 stuks per dag. Per dag zijn er ook ten hoogste 120 ritssluitingen beschikbaar.

   Hoeveel broeken van elk type moeten er geproduceerd worden om een zo groot mogelijke winst te maken? Beantwoord hiertoe onderstaande vragen.

   a) Maak van dit probleem een wiskundig model. Voor de eenvoud mag je veronderstellen dat "fractionele" broeken mogelijk zijn.
   b) Geef een aantal vereenvoudigingen die aanwezig zijn in dit model.
   c) Geef de optimale oplossing (bv. door gebruik te maken van Excel) en interpreteer het resultaat.

[2]Dit betekent dat alle arbeiders steeds betaald worden, ook al hebben ze niets omhanden.

2. Een bank heeft 100 000 euro beschikbaar om te investeren gedurende het huidige jaar. De financiële analisten van de bank hebben de volgende investeringsmogelijkheden geselecteerd: bedrijfsleningen, persoonlijke leningen, preferente aandelen, gewone aandelen en staatsobligaties. Het jaarlijkse rendement van elke type investering wordt geschat op 12%, 17%, 10.5%, 11.5% en 9% respectievelijk. Om de risico's te reduceren hebben de analisten de volgende restricties opgelegd voor de portefeuille van de bank.

   a) In de leningen, noch in de aandelen mag meer dan 50% van beschikbare bedrag worden geïnvesteerd.

   b) De investering in staatsobligaties moet tenminste gelijk zijn aan 30% van de investering in leningen.

   c) De persoonlijke leningen mogen hooguit 40% voor hun rekening nemen van de totale investering in de leningen.

   Hoe moet de bank zijn geld investeren opdat het jaarlijke rendement op de portefeuille gemaximaliseerd wordt? Stel in eerste instantie het wiskundig model op voor dit probleem. Gebruik dan Excel (of een ander softwarepakket) om het probleem op te lossen.

3. Een vrachtvliegtuig beschikt over drie compartimenten om vracht te laden: vooraan, midden en achteraan. Elk compartiment heeft de volgende restricties qua gewicht en volume dat er kan in geladen worden:

Compartiment	Max gewicht (ton)	Max volume (m^3)
vooraan	10	6800
midden	16	8700
achteraan	8	5300

   Om het evenwicht van het vliegtuig te bewaren moet het gewicht van de cargo in elk compartiment steeds in dezelfde verhouding blijven als wanneer elk compartiment volledig was geladen. Het middelste compartiment moet bv. steeds precies dubbel zo veel gewicht bevatten als het achterste compartiment.

   Er zijn vier verschillende types lading beschikbaar. Deze types lading zijn zodanig dat er willekeurige fracties van kunnen meegenomen/aanvaard worden. De eigenschappen van deze types staan in de

tabel hieronder opgesomd. De kolom "Gewicht" geeft aan hoeveel ton er van elk type beschikbaar is.

Lading	Gewicht (ton)	Volume (m^3/ton)	Winst (EUR/ton)
L_1	18	480	310
L_2	15	650	380
L_3	23	580	350
L_4	12	390	285

Je taak is om te bepalen hoeveel van elke lading moet worden meegenomen én hoe deze lading moet verdeeld worden over de drie compartimenten om de winst te maximaliseren.

Stel hiertoe eerst het wiskundig model op. Gebruik vervolgens een softwarepakket om de optimale oplossing van dit probleem te bepalen.

HOOFDSTUK 7

Lineair Programmeren

In dit (uitgebreide) hoofdstuk starten we met een formele definitie van een LP-PROBLEEM. Wanneer een LP-probleem slechts twee beslissingsvariabelen heeft dan kan men dit eenvoudig oplossen a.d.h.v. de GRAFISCHE METHODE. Uit deze grafische methode kan men intuïtief inzien dat wanneer een LP-probleem een optimale oplossing heeft er steeds een optimale oplossing kan gevonden worden in een "hoekpunt" van het aanvaardbaar gebied.
Het basisidee van het SIMPLEXALGORITME bestaat er dan ook in om van hoekpunt naar hoekpunt te springen totdat men zich niet meer kan verbeteren. Om dit echter algebraïsch te kunnen uitvoeren zullen we het LP-probleem echter eerst herschrijven als een stelsel van lineaire vergelijkingen die de VERHOOGDE VORM wordt genoemd. Vervolgens wordt er getoond hoe men eenvoudig van de ene AANVAARDBARE BASISOPLOSSING naar een NABURIGE BASISOPLOSSING kan overgaan door het stelsel te gaan herschrijven m.b.v. een aantal eenvoudige REKENREGELS.
Opdat de simplexmethode zou werken is het nodig dat de oorsprong tot het aanvaardbaar gebied behoort. Wanneer dit niet zo is dan kan de II-FASEN METHODE worden gebruikt om in de eerste fase een aanvaardbare basisoplossing (en bijhorende vorm van het stelsel) te vinden waarna in de tweede fase de rekenregels van het "gewone" simplexalgoritme worden toegepast.
Het hoofstuk eindigt met enkele details van de simplexmethode zoals de vraag of deze steeds eindigt, wat de tijdscomplexiteit is en hoe de simplexmethode kan herkennen dat het aanvaardbaar gebied leeg is.

7.1 Inleiding

Heel veel problemen binnen het gebied van operationeel onderzoek zijn van de vorm: optimaliseer een *lineaire* doelfunctie waarbij de beslissingsvariabelen moeten voldoen aan een aantal *lineaire* restricties en waarbij de beslissingsvariabelen *niet-negatief* moeten zijn.

Andere problemen (bv. binnen de speltheorie) kunnen herleid worden tot het herhaaldelijk oplossen van zo'n problemen. Zoals we zullen zien in Hoofdstuk 8 is dit ook het geval voor het oplossen van geheeltallige lineaire problemen.

Definitie 7.1 We zeggen dat een reële functie f LINEAIR is in de veranderlijken x_1 t.e.m. x_n wanneer f de volgende vorm aanneemt:

$$f\colon \mathbb{R}^n \to \mathbb{R}\colon (x_1, x_2, \ldots, x_n) \mapsto c_0 + c_1x_1 + c_2x_2 + \cdots + c_nx_n,$$

waarbij de coëfficiënten c_i ($i \in \{0, 1, \ldots, n\}$) reële getallen zijn. ∎

Opmerking 7.2 Wanneer men een lineaire doelfunctie wil gaan optimaliseren, dan houdt men tijdens het optimalisatieproces meestal geen rekening met de coëfficiënt c_0, omdat die geen invloed heeft op de combinatie van de waarden van de variablen x_i waar het optimum bereikt wordt.

Wanneer men de oplossing wil gaan *interpreteren* dan is het uiteraard wel van belang dat de coëfficiënt c_0 in rekening wordt genomen, bv. om te bepalen of een bepaalde activiteit economisch zinvol is. In dit geval zou de coëfficiënt c_0 bv. de *vaste kosten* kunnen voorstellen. ∎

Definitie 7.3 We spreken over een LINEAIRE BEPERKING of LINEAIRE RESTRICTIE wanneer die van één van de volgende vormen is:

$$a_1x_1 + a_2x_2 + \cdots + a_nx_n \leq b,$$

of

$$a_1x_1 + a_2x_2 + \cdots + a_nx_n \geq b,$$

of

$$a_1x_1 + a_2x_2 + \cdots + a_nx_n = b,$$

waarbij de a_i (met $i \in \{1, \ldots, n\}$) en b reële getallen zijn. ∎

Definitie 7.4 Een LP-PROBLEEM bestaat erin om een *lineaire doelfunctie* in n beslissingsvariabelen te optimaliseren onder een verzameling van m *lineaire restricties* waarbij bovendien alle beslissingsvariabelen *niet-negatief* moeten zijn. ∎

Opmerking 7.5 "Optimaliseren" betekent dat er zowel gemaximaliseerd als geminimaliseerd kan worden. ∎

Voorbeeld 7.6 Het probleem opgesteld voor Boer Frans in Voorbeeld 6.1 is een LP-probleem in twee beslissingsvariabelen. De doelfunctie die moet gemaximaliseerd worden is

$$D(x_t, x_g) = 220x_t + 255x_g,$$

en er zijn drie lineaire beperkingen, nl. de land-, mest- en pesticidebeperking. De twee beslissingsvariabelen moeten eveneens niet-negatief zijn. ∎

Voorbeeld 7.7 Het volgende optimalisatieprobleem is eveneens een LP-probleem.

$$\max D(x_1, x_2, x_3) = x_1 + 2x_2 + 3x_3$$

onder de beperkingen

$$\begin{cases} x_1 + 2x_2 + 3x_3 \geq 12 \\ x_1 + 2x_2 + \ x_3 \geq 6 \\ x_1 + \ x_2 + \ x_3 \leq 6. \end{cases}$$

De niet-negativiteitsvoorwaarden zijn van kracht, of anders gezegd:

$$x_j \geq 0, \quad \text{voor } j \in \{1, 2, 3\}.$$

Dit LP-probleem heeft drie beslissingsvariabelen en drie lineaire restricties. ∎

7.2 De grafische oplossingsmethode

De grafische oplossingsmethode is een methode die kan gebruikt worden om een lineaire doelfunctie te optimaliseren onder een (willekeurig) aantal lineaire beperkingen wanneer er twee (of eventueel drie[1]) beslissingsvariabelen zijn.

[1] Dit is al een heel stuk moeilijker.

Voorbeeld 7.8 (Grafische oplossing van probleem Boer Frans) We illustreren de grafische oplossingsmethode voor het LP-probleem dat we hebben opgesteld in Voorbeeld 6.1.

Om te bepalen wat de optimale mix is van tarwe en gerst gaan we in eerste instantie trachten een duidelijk beeld te krijgen van de *aanvaardbare* verdelingen tussen tarwe en gerst. Zo is het bv. duidelijk dat het aanvaardbaar (of toelaatbaar) is om $x_t = 5$ en $x_g = 10$ te nemen. Men controleert dat aan alle drie de restricties voldaan is. Aan de andere kant is $x_t = 1$ en $x_g = 19$ niet toegestaan. In dit geval zou men te veel mest gebruiken.

Uit de niet-negativiteitsvoorwaarden weten we reeds dat x_t en x_g enkel waarden kunnen aannemen in het eerste kwadrant. We bekijken nu de eerste restrictie m.b.t. de beschikbare oppervlakte:

$$x_t + x_g \leq 20.$$

Wanneer men de ongelijkheid verandert in een gelijkheid dan zien we dat

$$x_t + x_g = 20,$$

de vergelijking van een rechte voorstelt in het (x_t, x_g)-vlak. Deze rechte verdeelt het eerste kwadrant in twee delen. Aan de ene "kant" van de rechte geldt dat

$$x_t + x_g < 20,$$

en dit is dus de "goede" kant, terwijl aan de andere kant geldt dat

$$x_t + x_g > 20,$$

en dat is dus de "slechte" kant. Om te weten welke kant de goede kant is kan men eenvoudigweg kijken of de originele restrictie voldaan is in de oorsprong. In dit geval is

$$0 + 0 = 0 \leq 20.$$

Dit betekent dat de kant van de oorsprong de goede kant is. In Figuur 7.1 ziet men links bovenaan dat de slechte kant van deze rechte aangeduid werd.

We weten nu reeds dat de mogelijke aanvaardbare verdelingen tussen tarwe en gerst zich steeds onder de rechte $x_t + x_g = 20$ bevinden. De landrestrictie is echter niet de enige restrictie, er is ook nog een restrictie op het gebruik van mest, nl.:

$$10x_t + 20x_g \leq 300.$$

We gaan nu opnieuw eventjes over op de gelijkheid

$$10x_t + 20x_g = 300,$$

en we realiseren ons dat dit opnieuw de vergelijking van een rechte is in het (x_t, x_g)-vlak. Aangezien in de oorsprong aan de restrictie voldaan is (want $0 + 0 = 0 \leq 300$) is de "onderkant" van de rechte de "goede" kant. Omdat geldige combinaties van x_t en x_g aan beide restricties (land en mest) moeten voldoen vinden we geldige combinaties enkel onder *beide* rechten. Dit ziet men rechts bovenaan in Figuur 7.1.

Er is nog een derde restrictie, de pesticiderestrictie,

$$2x_t + x_g \leq 36,$$

waarvoor we hetzelfde procedé toepassen. We gaan over op de gelijkheid, tekenen de rechte in het (x_t, x_g)-vlak en bepalen de "goede" kant van deze rechte. Dit is, in dit geval, opnieuw de kant waartoe de oorsprong behoort.

Conclusie: de aanvaardbare combinaties van x_t en x_g vindt men in het niet gearceerde deel van de figuur links onderaan in Figuur 7.1.

Dit is echter niet het hele verhaal. We willen nu uitvissen voor welke van de aanvaardbare combinaties van x_t en x_g de winst gemaakt door boer Frans maximaal is.

We kunnen ons bv. eerst afvragen of boer Frans in staat is om 2200 EUR winst te maken? De combinaties waarvoor 2200 EUR winst wordt gemaakt zijn diegene die voldoen aan:

$$220x_t + 255x_g = 2200. \tag{7.1}$$

Dit is opnieuw een rechte in het (x_t, x_g)-vlak. We zien rechts onderaan Figuur 7.1 dat deze rechte punten bevat die aanvaardbare combinaties zijn van x_t en x_g. M.a.w. boer Frans is in staat om 2200 EUR winst te maken.

Misschien kan boer Frans nog meer winst maken? Kan hij bv. 4000 EUR winst maken? Aangezien de rechte

$$220x_t + 255x_g = 4000, \tag{7.2}$$

aanvaardbare combinaties van x_t en x_g bevat is het antwoord affirmatief.

We merken op dat beide rechten gegeven door de vergelijkingen (7.1) en (7.2) evenwijdig zijn met elkaar, aangezien ze dezelfde richtingscoëfficiënt

hebben. Meer in het algemeen zijn alle DOELFUNCTIERECHTEN, i.e. rechten van de vorm

$$220x_t + 255x_g = D, \tag{7.3}$$

evenwijdig met elkaar. Op deze manier is het eenvoudig om te zien wat de maximale winst is die kan bereikt worden. Men verschuift de doelfunctierechte (in dit geval) zo veel mogelijk naar rechts, tot op het moment dat de doelfunctierechte nog net een aanvaardbare combinatie van x_t en x_g bevat. Nóg verder opschuiven zou betekenen dat de doelfunctierechte geen enkele aanvaardbare combinatie meer bevat.

In dit geval wordt dit bereikt in het punt dat het snijpunt is van de rechten

$$x_t + x_g = 20,$$

en

$$10x_t + 20x_g = 300.$$

Als we de coördinaten van dit snijpunt bepalen, dan vinden we,

$$x_t = 10 \quad \text{en} \quad x_g = 10,$$

en de bijhorende winst is

$$220 \times 10 + 255 \times 10 = 4750.$$ ∎

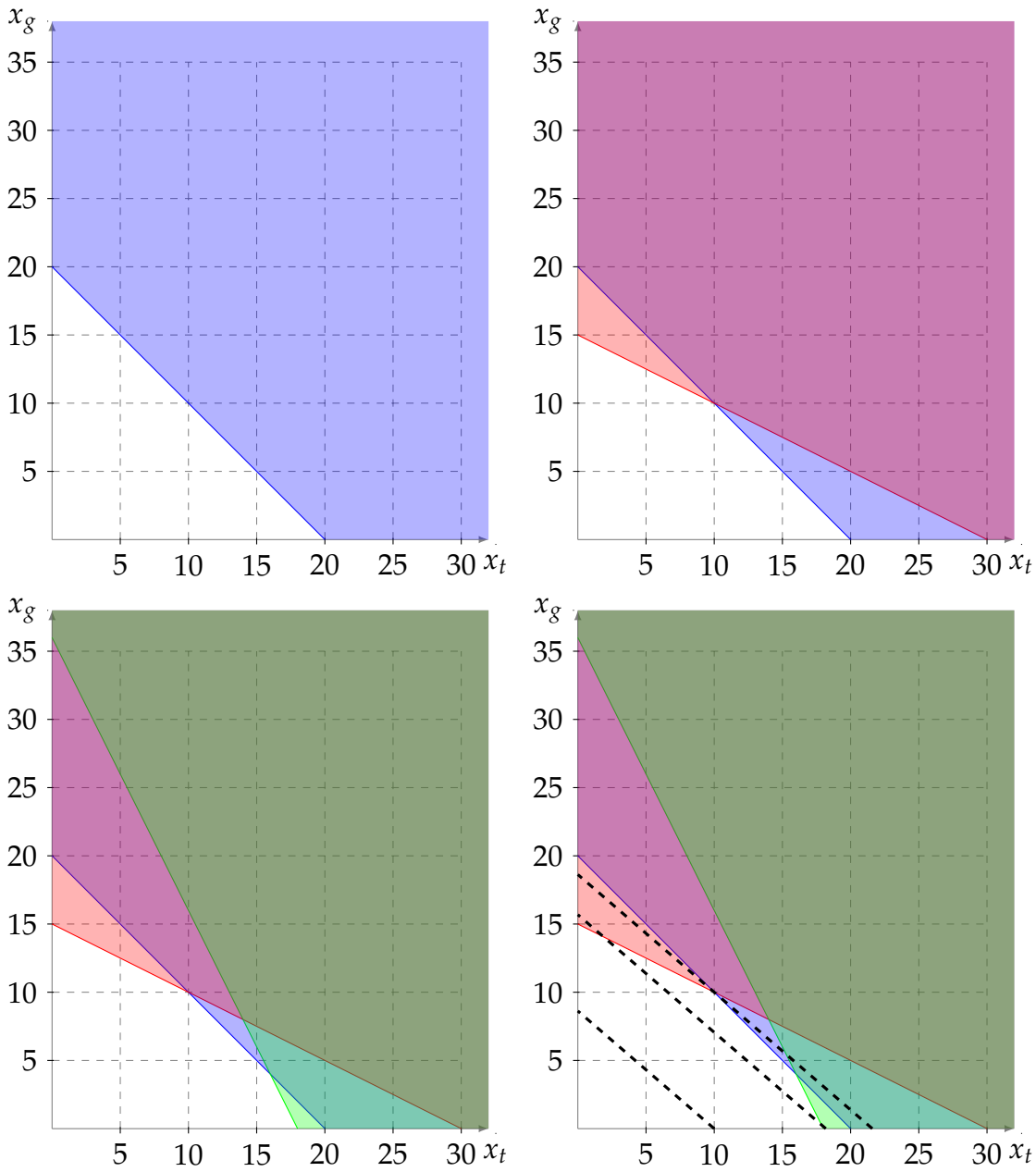
De GRAFISCHE METHODE bestaat m.a.w. uit de volgende stappen:

1. Bepaal het aanvaardbaar gebied[2] als volgt: teken de rechte die overeenkomt met elk van de restricties (na omzetten naar een gelijkheid[3]). Bepaal de punten die tegelijkertijd aan alle "goede" kanten van al deze rechten liggen.

2. Bepaal een doelfunctierechte. Verschuif deze evenwijdig naar de juiste kant tot op het moment net voor deze rechte geen punt meer gemeenschappelijk heeft met het aanvaardbaar gebied.

[2] Algemene definitie komt later.

[3] Wanneer de restrictie reeds een gelijkheid was dan liggen de aanvaardbare punten precies op de rechte.



Figuur 7.1: Opeenvolgende figuren bij het grafisch oplossen van het "Boer Frans" LP-probleem. Men tekent eerst het aanvaardbaar gebied door alle restricties één voor één te tekenen waarbij men telkens de "verkeerde" kant arceert. Hier zijn in volgorde de land-, mest- en pesticiderestrictie toegevoegd. Tenslotte tekent men de (richting van) de doelfunctierechten. Hiermee kan nu gemakkelijk het hoekpunt worden bepaald waarin de doelfunctiewaarde wordt gemaximaliseerd. In de figuur ziet men drie doelfunctierechten waarvoor de winst respectievelijk 2000, 4000 en 4750 EUR is.

3. Meestal zal de doelfunctierechte nu net nog het aanvaardbaar gebied "raken" in het snijpunt van twee rechten die de rand van het aanvaardbaar gebied uitmaken. Vind de coördinaten van dit snijpunt door het oplossen van een eenvoudig lineair stelsel met twee vergelijkingen en twee onbekenden.

4. Bepaal de optimale waarde van de doelfunctie en geef de oplossing.

7.2.1 Bijzondere gevallen

Meestal zal een LP-probleem in twee beslissingsvariabelen een unieke oplossing hebben. Dit is bv. het geval voor het LP-probleem dat opgelost werd in Voorbeeld 7.8. Er zijn echter ook andere gevallen mogelijk.

Oneindig veel oplossingen. Het is mogelijk dat een LP-probleem oneindig veel oplossingen heeft. In twee dimensies is dit mogelijks het geval wanneer de doelfunctierechte evenwijdig is met één van de grensrechten van het aanvaardbaar gebied.

Voorbeeld 7.9 (LP-probleem met oneindig veel oplossingen)
Veronderstel dat de de doelfunctie die moet gemaximaliseerd worden in het LP-probleem van Voorbeeld 6.1 gegeven wordt door

$$200x_t + 400x_g.$$

In dit geval merkt men dat de doelfunctierechten

$$200x_t + 400x_g = D$$

evenwijdig zijn met de mestrestrictie:

$$10x_t + 20x_g = 300.$$

Men ziet dan grafisch dat de optimale doelfunctiewaarde (nl. $D = 6000$) niet alleen bereikt wordt in de hoekpunten

$$x_t = 0, \quad x_g = 15$$

en

$$x_t = 10, \quad x_g = 10$$

maar óók in alle punten die deze hoekpunten verbinden. ∎

Doelfunctie wordt oneindig groot/klein. Het is mogelijk dat het aanvaardbaar gebied niet begrensd is. In dit geval kan het gebeuren dat de doelfunctie oneindig groot of klein kan gemaakt worden. Dit duidt *meestal* op het feit dat er een restrictie vergeten werd. In realistische situaties is het niet vaak het geval dat de winst oneindig groot of de kosten oneindig klein kunnen worden gemaakt.

Voorbeeld 7.10 (Doelfunctie is onbegrensd) Beschouw het volgende LP-probleem:

$$\max D(x_1, x_2) = x_1 + 3x_2$$

onder de beperking

$$2x_1 + x_2 \geq 1$$

terwijl de niet-negativiteitsvoorwaarden van kracht zijn:

$$x_i \geq 0, \quad \text{voor } i \in \{1, 2\}.$$

Zoals je ziet kan je in dit voorbeeld x_1 én x_2 onbeperkt laten toenemen; de enige restrictie blijft steeds voldaan. Dit heeft als gevolg dat men de doelfunctie in dit geval elke willekeurig grote waarde kan laten aannemen. In dit geval is er geen optimale oplossing. ∎

Aanvaardbaar gebied is leeg. Het kan gebeuren dat de restricties elkaar tegenspreken. In dit geval is het aanvaardbaar gebied leeg. Dit gebeurt bv. wanneer er een fout geslopen is bij het opstellen van het model (of wanneer het probleem effectief geen oplossing kan hebben).

Voorbeeld 7.11 (Leeg aanvaardbaar gebied) Beschouw het volgende LP-probleem:

$$\max D(x_1, x_2) = x_1 + 3x_2$$

onder de beperkingen

$$\begin{cases} 2x_1 + x_2 \leq 1 \\ x_1 + x_2 \geq 2 \end{cases}$$

terwijl de niet-negativiteitsvoorwaarden van kracht zijn:

$$x_i \geq 0, \quad \text{voor } i \in \{1, 2\}.$$

Wanneer men voor dit probleem het aanvaardbaar gebied tekent dan merkt men dat er *geen enkele combinatie* van waarden van x_1 en x_2 is die aan de restricties voldoet. ∎

7.3 Het simplexalgoritme

De meest voor de hand liggende beperking[4] van de grafische oplossingsmethode is uiteraard dat deze enkel kan toegepast worden voor LP-problemen waarbij het aantal beslissingsvariabelen 2 (of eventueel 3) is. We bespreken nu een algemene oplossingsmethode die kan gebruikt worden om LP-problemen met een willekeurig aantal beslissingsvariabelen en een willekeurig aantal restricties (meestal) efficiënt op te lossen. Dit algoritme, het simplexalgoritme, werd in de tweede helft van de jaren 1940 opgesteld door George Dantzig. Meer dan 70 jaar na zijn uitvinding blijft dit één van de belangrijkste algoritmes binnen de informatica.

7.3.1 De algemene en de standaardvorm van een LP-probleem

We starten met de definitie van de algemene vorm van een LP-probleem. Dit is een meer expliciete versie van Definitie 7.4.

Definitie 7.12 Een LP-probleem in ALGEMENE VORM bestaat uit n beslissingsvariabelen x_1 t.e.m. x_n waarbij een lineaire doelfunctie

$$D(x_1, x_2, \ldots, x_n) = c_1x_1 + c_2x_2 + \cdots + c_nx_n,$$

hetzij gemaximaliseerd hetzij geminimaliseerd moet worden onder een verzameling van m lineaire restricties,

$$\begin{cases} a_{11}x_1 + a_{12}x_2 + \cdots + a_{1n}x_n & \{\leq | = | \geq\}\ b_1 \\ a_{21}x_1 + a_{22}x_2 + \cdots + a_{2n}x_n & \{\leq | = | \geq\}\ b_2 \\ \quad\vdots & \quad\vdots \\ a_{m1}x_1 + a_{m2}x_2 + \cdots + a_{mn}x_n & \{\leq | = | \geq\}\ b_m \end{cases}$$

en waarbij de beslissingsvariabelen voldoen aan de niet-negativiteitsvoorwaarden:

$$x_j \geq 0, \quad \text{voor } j \in \{1, 2, \ldots, n\}.$$

Hierbij zijn de coëfficiënten c_j, a_{ij} en b_i vaste reële getallen. ∎

Opmerking 7.13 De notatie $\{\leq | = | \geq\}$ betekent dat er precies één van de drie relationele operatoren moet worden gekozen. Deze keuze kan echter wel per restrictie verschillend zijn. ∎

[4] Vergeef me de woordspeling.

Zoals men ziet laat deze algemene vorm zowel maximalisatie als minimalisatie van de doelfunctie toe. Verder kunnen de beperkingen geschreven worden met één van de drie beschikbare relationele operatoren. In de algemene vorm worden er geen eisen opgelegd aan de waarden van de coëfficiënten b_i, maar er wordt wel steeds geëist dat alle beslissingsvariabelen niet-negatief zijn.

Dit betekent dat er nog veel variatie mogelijk is wanneer men een LP-probleem opschrijft in zijn algemene vorm. Deze variatie zorgt er ook meteen voor dat het moeilijker is om een algoritme te ontwerpen dat zo'n LP-probleem in algemene vorm kan aanpakken.

Elk LP-probleem in algemene vorm kan omgezet worden naar een LP-probleem in standaardvorm. We geven eerst de definitie van deze standaardvorm en argumenteren dan dat deze precies dezelfde modelleringskracht heeft als de algemene vorm.

Definitie 7.14 Bij een LP-probleem in STANDAARDVORM moet men de waarde van n beslissingsvariabelen x_1 t.e.m. x_n bepalen om een lineaire doelfunctie te *maximaliseren*, i.e. bepaal

$$\max D(x_1, \ldots, x_n) = c_1x_1 + c_2x_2 + \cdots + c_nx_n,$$

onder de beperkingen

$$\begin{cases} a_{11}x_1 + a_{12}x_2 + \cdots + a_{1n}x_n \leq b_1 \\ a_{21}x_1 + a_{22}x_2 + \cdots + a_{2n}x_n \leq b_2 \\ \quad\vdots \qquad\qquad\qquad\qquad \vdots \\ a_{m1}x_1 + a_{m2}x_2 + \cdots + a_{mn}x_n \leq b_m \end{cases}$$

en waarbij de beslissingsvariabelen voldoen aan de niet-negativiteitsvoorwaarden:

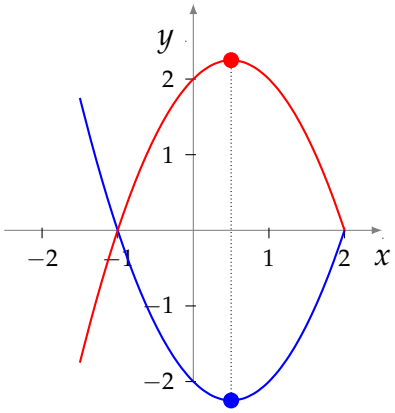
$$x_j \geq 0, \quad \text{voor } j \in \{1, 2, \ldots, n\}.$$

Hierbij zijn de coëfficiënten c_j, a_{ij} en b_i vaste reële getallen. ∎

Opmerking 7.15 Het verschil tussen de algemene en de standaardvorm is m.a.w. dat in de standaardvorm steeds wordt *gemaximaliseerd* en dat alle beperkingen van de $\leq$-vorm zijn. ∎

Veronderstel dat men een doelfunctie

$$D_{\min}(x_1, \ldots, x_n) = c_1x_1 + c_2x_2 + \cdots + c_nx_n$$



Figuur 7.2: Omzetten van minimalisatie van $(x+1)(x-2)$ over $[-1,2]$ naar maximalisatie van $-(x+1)(x-2)$ over hetzelfde interval. Het minimum van $(x+1)(x-2)$ over $[-1,2]$ wordt bereikt voor $x = 1/2$ (het midden van de nulpunten) met bijhorende functiewaarde $y = -9/4$. Het maximum van $-(x+1)(x-2)$ wordt bereikt voor dezelfde x-waarde $1/2$ maar de bijhorende functiewaarde is nu $y = 9/4$.

wenst te minimaliseren. Dit is equivalent met het maximaliseren van de doelfunctie

$$D_{\max}(x_1, \ldots, x_n) = -c_1x_1 - c_2x_2 - \cdots - c_nx_n.$$

Eens men weet voor welke waarde $D_{\max}$ maximimaal is, weet men ook wanneer $D_{\min}$ minimaal is. Dit is namelijk voor precies dezelfde combinatie van de beslissingsvariabelen, en de bijhorende minimale waarde is precies de tegengestelde van de gevonden maximale waarde.

Voorbeeld 7.16 In Figuur 7.2 ziet men hoe men een minimalisatieprobleem van een kwadratische functie kan omzetten naar een equivalent maximalisatieprobleem. Men ziet dat de x-waarde waarvoor het maximum wordt bereikt dezelfde x-waarde is als waarvoor het minimum wordt bereikt. Om te weten wat de effectieve minimale functiewaarde is hoeft men enkel maar de negatie te nemen van de maximale functiewaarde. ∎

Een restrictie van de vorm

$$a_1x_1 + a_2x_2 + \cdots + a_nx_n \geq b,$$

is, na vermenigvuldiging met -1, equivalent met

$$-a_1x_1 - a_2x_2 - \cdots - a_nx_n \leq -b,$$

zodat elke $\geq$-restrictie onmiddellijk kan omgezet worden in een restrictie in $\leq$-vorm.

Tenslotte is een restrictie van de vorm

$$a_1x_1 + a_2x_2 + \cdots + a_nx_n = b,$$

equivalent met de twee restricties

$$a_1x_1 + a_2x_2 + \cdots + a_nx_n \leq b$$

en

$$a_1x_1 + a_2x_2 + \cdots + a_nx_n \geq b.$$

Van deze laatste restrictie weten we reeds hoe we deze kunnen omzetten naar $\leq$-vorm.

7.3.2 Basisidee van het simplexalgoritme

We starten met enkele definities en een aantal eigenschappen van LP-problemen waaronder één zeer belangrijke eigenschap. Deze eigenschap hebben we reeds kunnen vaststellen bij het grafisch oplossen van LP-problemen, maar ze is ook algemeen geldig.

Definitie 7.17 Het AANVAARDBAAR GEBIED van een LP-probleem bestaat uit de deelverzameling van punten uit $\mathbb{R}^n$ die zowel aan de m restricties, als aan de n niet-negativiteitsvoorwaarden voldoen. ∎

Hieronder wordt de belangrijke eigenschap van LP-problemen geformuleerd:

Stelling 7.18 Wanneer een LP-probleem minstens één optimale oplossing heeft, dan kan er steeds een optimale oplossing gevonden worden in een "hoekpunt" van het aanvaardbaar gebied. ∎

Opmerking 7.19 In twee dimensies (i.e. wanneer $n = 2$) is een hoekpunt in het algemeen het snijpunt van twee restrictierechten, in drie dimensies is een hoekpunt in het algemeen het snijpunt van drie restrictievlakken, en in n dimensies is een hoekpunt in het algemeen het snijpunt van n restrictie-hypervlakken[5]. ∎

[5]Een hypervlak is de naam die wordt gegeven aan de verzameling van de punten in $\mathbb{R}^n$ die voldoen aan een lineaire vergelijking in n variabelen.

Het basisidee van het simplexalgoritme bestaat er in om, startend vanuit een bepaald hoekpunt, *iteratief van hoekpunt naar hoekpunt te "springen"* tot men vaststelt dat men zich niet meer kan verbeteren. Op dit moment wordt de optimale oplossing gevonden.

In de volgende sectie tonen we aan hoe men dit idee op een *algebraïsche* wijze kan bewerkstelligen.

7.3.3 Het simplexalgoritme

In deze sectie beperken we ons tot problemen in standaardvorm waarbij bovendien geldt dat alle coëfficiënten b_i niet-negatief zijn. Dit betekent dat de oorsprong tot het aanvaardbaar gebied behoort. Een uitbreiding wordt later besproken.

In een eerste stap gaan we de m restricties in $\leq$-vorm omzetten naar gelijkheden door het toevoegen van SPELINGSVARIABELEN (Eng. *slack variables*), één per restrictie. De restrictie

$$a_{i1}x_1 + a_{i2}x_2 + \cdots + a_{in}x_n \leq b_i$$

wordt omgezet in

$$a_{i1}x_1 + a_{i2}x_2 + \cdots + a_{in}x_n + y_i = b_i,$$

hierbij is y_i een niet-negatieve spelingsvariabele die de speling aangeeft tussen het linker- en rechterlid. Vervolgens worden de spelingsvariabelen ook toegevoegd aan de doelfunctie met een coëfficiënt gelijk aan nul.

Definitie 7.20 De VERHOOGDE VORM (Eng. *augmented form*) van het LP-probleem in standaardvorm wordt gegeven door:

$$\max D(x_1, \ldots, x_n, y_1, \ldots, y_m) = c_1x_1 + c_2x_2 + \cdots + c_nx_n + 0y_1 + \cdots + 0y_m$$

onder de beperkingen

$$\begin{cases} a_{11}x_1 + a_{12}x_2 + \cdots + a_{1n}x_n + y_1 & = b_1 \\ a_{21}x_1 + a_{22}x_2 + \cdots + a_{2n}x_n + y_2 & = b_2 \\ \vdots & \vdots \\ a_{m1}x_1 + a_{m2}x_2 + \cdots + a_{mn}x_n + y_m & = b_m \end{cases}$$

en waarbij de beslissingsvariabelen voldoen aan de niet-negativiteitsvoorwaarden:

$$x_j \geq 0, \quad \text{voor } j \in \{1, 2, \ldots, n\}$$

en

$$y_i \geq 0, \quad \text{voor } i \in \{1, 2, \ldots, m\}.$$ ∎

Voorbeeld 7.21 De verhoogde vorm van het LP-probleem in Voorbeeld 6.1 wordt gegeven door[6]:

$$\max D(x_1, x_2, y_1, y_2, y_3) = 220x_1 + 255x_2 + 0y_1 + 0y_2 + 0y_3$$

onder de beperkingen:

$$\begin{cases} x_1 + x_2 + y_1 = 20 \\ 10x_1 + 20x_2 + y_2 = 300 \\ 2x_1 + x_2 + y_3 = 36 \end{cases}$$

en waarbij ook de niet-negativiteitsvoorwaarden van kracht zijn:

$$x_j \geq 0 \quad \text{voor } j \in \{1, 2\}$$

en

$$y_i \geq 0 \quad \text{voor } i \in \{1, 2, 3\}.$$ ∎

De verhoogde vorm bevat steeds m (functionele) restricties die samen een stelsel van m lineaire vergelijkingen in $n + m$ onbekenden vormen.

Eigenschap 7.22 Het stelsel lineaire vergelijkingen van de verhoogde vorm van een LP-probleem is steeds oplosbaar. Meer in het bijzonder heeft het stelsel n vrijheidsgraden. ∎

Bewijs Uit de cursus Math4IT weten we reeds dat een stelsel lineaire vergelijkingen oplossingen heeft als en slechts als de rang van de coëfficiëntenmatrix gelijk is aan die van de verhoogde matrix. In dit geval is de rang van coëfficiëntenmatrix maximaal; ze is immers steeds gelijk aan m aangezien de laatste m kolommen (overeenkomend met de variabelen y_i) de eenheidsmatrix vormen. De rang van de verhoogde matrix kan niet groter zijn dan m aangezien er maar m rijen zijn.

[6]We schrijven nu eventjes x_1 en x_2 i.p.v. x_t en x_g.

Verder weten we uit de cursus Math4IT dat een stelsel lineaire vergelijkingen met N onbekenden en waarvoor de rang van de coëfficiëntenmatrix en de verhoogde matrix gelijk zijn aan r over $N - r$ vrijheidsgraden bezit.

In dit geval is $N = n + m$ en is $r = m$, zodat het stelsel van de verhoogde vorm inderdaad n vrijheidsgraden heeft. ♢

Opmerking 7.23 Wanneer een stelsel lineaire vergelijkingen $N - r$ vrijheidsgraden heeft dan betekent dit in de praktijk dat men aan $N - r$ willekeurig gekozen variabelen een willekeurige waarde mag geven. De waarden van de overige r variabelen liggen dan eenduidig vast. ∎

Voorbeeld 7.24 Het stelsel van de verhoogde vorm in Voorbeeld 7.21 bevat 5 variabelen; de rang van de coëfficiëntenmatrix is 3 en bijgevolg bezit dit stelsel twee vrijheidsgraden.

Zo kan men bv. als vrij te kiezen variabelen x_1 en x_2 kiezen en deze respectievelijk de waarde 5 en 10 geven. De waarde van y_1 t.e.m. y_3 wordt dan gegeven door

$$y_1 = 20 - 5 - 10 = 5, \quad y_2 = 300 - 50 - 200 = 50 \text{ en } y_3 = 36 - 10 - 10 = 16.$$

We zien dat in deze oplossing alle variabelen een niet-negatieve waarde aannemen. Men verifieert (op de figuur van het aanvaardbaar gebied in Figuur 7.1) dat we hier te maken hebben met een combinatie van x_1 en x_2 die aanvaardbaar is.

Men kan echter evengoed x_2 en y_1 een willekeurige waarde geven, bv. $x_2 = 15$ en $y_1 = 0$. In dit geval worden de waarden van de overige variabelen gegeven door[7]:

$$x_1 = 5, \quad y_2 = -50, \quad \text{en} \quad y_3 = 11,$$

Hier ziet men dat er een variabele (nl. y_2) is die een negatieve waarde aanneemt. Men controleert dat $x_1 = 5$ en $x_2 = 15$ *geen* geldige combinatie van landgebruik is. Er wordt immers te veel mest gebruikt:

$$10 \times 5 + 20 \times 15 = 350 > 300.$$

Men ziet dat de spelingsvariabele y_2 die met deze restrictie werd geassocieerd de waarde -50 aanneemt om aan te geven dat de totale hoeveelheid toegestane mest met 50 eenheden werd overschreden.

[7]Gebruik hiervoor eventueel een programma als www.wolframalpha.com om het stelsel op te lossen.

Merk op dat voor $x_1 = 5$ en $x_2 = 15$ al het beschikbare land (nl. 20 hectare) wordt gebruikt. Dit wordt gereflecteerd door het feit dat de spelingsvariabele y_1 in dit geval de waarde nul heeft. ∎

We zien dat de spelingsvariabelen de volgende betekenis hebben.

1. Als y_i een waarde heeft strikt groter dan nul, dan ligt de oplossing (van de x-en) strikt aan de "goede" kant van deze restrictie. De i-de restrictie is m.a.w. voldaan.
2. Als y_i de waarde nul heeft, dan ligt de oplossing (van de x-en) precies "op" de i-de restrictie, en deze is dan ook voldaan.
3. Als y_i een negatieve waarde aanneemt, dan ligt de oplossing (van de x-en) aan de "verkeerde" kant van deze restrictie. De i-de restrictie is m.a.w. *niet* voldaan.

Een gelijkaardige observatie geldt voor de beslissingsvariabelen x_j.

Oplossingen van het stelsel lineaire vergelijkingen waarvoor alle variabelen een niet-negatieve waarde hebben worden AANVAARDBARE OPLOSSINGEN genoemd; wanneer één of meerdere variabelen een negatieve waarde hebben dan spreken we over een NIET-AANVAARDBARE OPLOSSING.

Wanneer in een bepaalde oplossing van het stelsel lineaire vergelijkingen nu meerdere, stel k in totaal, variabelen x_j en/of y_i de waarde nul aannemen dan ligt het bijhorende punt x dus in de doorsnede van de k hypervlakken horende bij de niet-negativiteitsvoorwaarden en/of restricties.

Zoals reeds vermeld in Opmerking 7.19 bestaat de doorsnede van n willekeurige hypervlakken in $\mathbb{R}^n$ in het algemeen uit precies één punt.

We spreken over een BASISOPLOSSING wanneer de n vrij te kiezen variabelen de waarde nul krijgen. De n gekozen variabelen (met de waarde nul) worden de NIET-BASISVARIABELEN genoemd, de andere m variabelen worden de BASISVARIABELEN genoemd. Normaalgezien hebben deze m variabelen een waarde verschillend van nul[8]. Wanneer in een basisoplossingen alle variabelen een niet-negatieve waarde aannemen, dan wordt dit een AANVAARDBARE BASISOPLOSSING genoemd.

[8] In bepaalde ontaarde gevallen kan een basisvariabele ook de waarde nul aannemen.

	niet-basisvariabelen	basisvariabelen	aanvaardbaar?
(a)	$x_1 = 0, x_2 = 0$	$y_1 = 20, y_2 = 300, y_3 = 36$	Ja
(b)	$x_1 = 0, y_1 = 0$	$x_2 = 20, y_2 = -100, y_3 = 16$	Nee
(c)	$x_1 = 0, y_2 = 0$	$x_2 = 15, y_1 = 5, y_3 = 21$	Ja
(d)	$x_1 = 0, y_3 = 0$	$x_2 = 36, y_1 = -16, y_2 = -420$	Nee
(e)	$x_2 = 0, y_1 = 0$	$x_1 = 20, y_2 = 100, y_3 = -4$	Nee
(f)	$x_2 = 0, y_2 = 0$	$x_1 = 30, y_1 = -10, y_3 = -24$	Nee
(g)	$x_2 = 0, y_3 = 0$	$x_1 = 18, y_1 = 2, y_2 = 120$	Ja
(h)	$y_1 = 0, y_2 = 0$	$x_1 = 10, x_2 = 10, y_3 = 6$	Ja
(i)	$y_1 = 0, y_3 = 0$	$x_1 = 16, x_2 = 4, y_2 = 60$	Ja
(j)	$y_2 = 0, y_3 = 0$	$x_1 = 14, x_2 = 8, y_1 = -2$	Nee

Tabel 7.1: De 10 basisoplossingen van het LP-probleem in verhoogde vorm in Voorbeeld 7.21. Deze zijn gevonden door tienmaal een stelsel lineaire vergelijkingen op te lossen (m.b.v. www.wolframalpha.com).

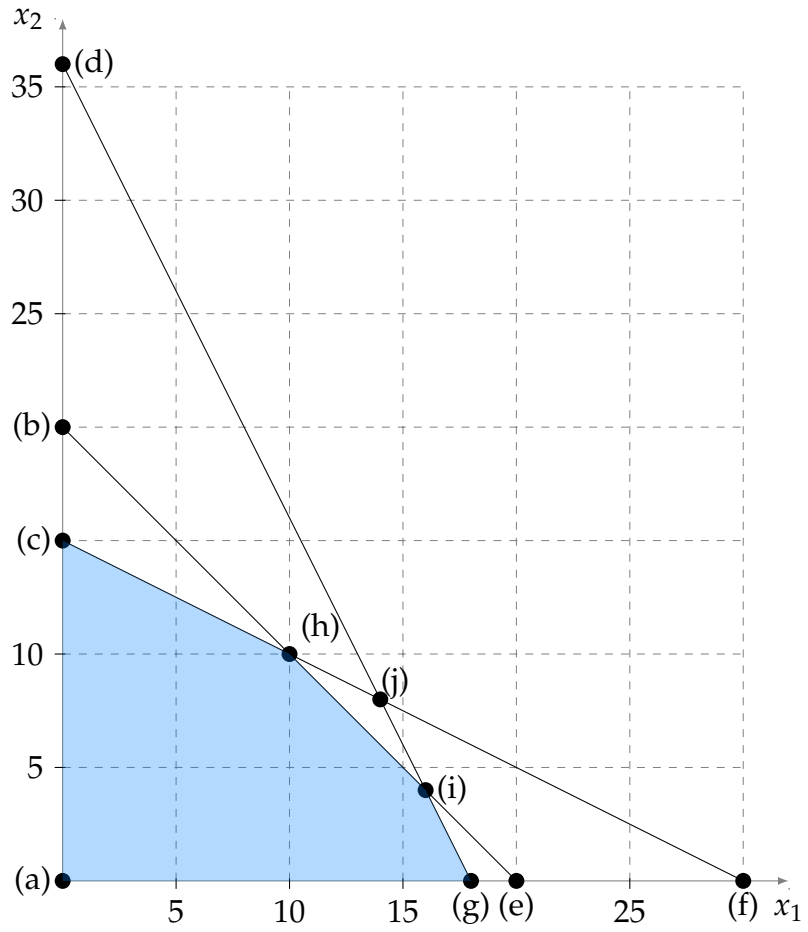
Voorbeeld 7.25 We beschouwen opnieuw het LP-probleem in verhoogde vorm van Voorbeeld 7.21. Voor het vinden van een basisoplossing kiest men willekeurig twee variabelen en geeft deze de waarde nul. De waarden van de andere drie variabelen worden bepaald door het stelsel lineaire vergelijkingen op te lossen.

In dit geval zijn er $\binom{5}{2} = 10$ basisoplossingen. We sommen deze op in Tabel 7.1. Zoals je kan zien op Figuur 7.3 komen de aanvaardbare basisoplossingen precies overeen met de hoekpunten van het aanvaardbaar gebied. Deze eigenschap is steeds geldig. ∎

Dit voorbeeld illustreert de volgende zeer belangrijke eigenschap.

Eigenschap 7.26 De aanvaardbare basisoplossingen van het stelsel lineaire vergelijkingen van een LP-probleem in verhoogde vorm komen precies overeen met de hoekpunten van het aanvaardbaar gebied van dit LP-probleem. ∎

Dit betekent dat we een *algebraïsche* manier hebben gevonden om de hoekpunten van het aanvaardbaar gebied te identificeren.



Figuur 7.3: Het aanvaardbaar gebied voor het LP-probleem uit Voorbeeld 7.21. Alle 10 de basisoplossingen staan aangeduid op deze figuur. Je ziet duidelijk dat de *aanvaardbare* basisoplossingen overeenkomen met de hoekpunten van het aanvaardbaar gebied (dat hier in het lichtblauw gekleurd is).

n	m	aantal basisoplossingen
2	3	10
10	6	8008
30	20	$47129212243960 \approx 4.7 \times 10^{13}$
100	30	$29372339821610944823963760 \approx 2.9 \times 10^{25}$

Tabel 7.2: Aantal basisoplossingen voor enkele combinaties van n en m. Men ziet dat zelfs voor relatief kleine waarden van n en m het aantal basisoplossingen veel te groot wordt om deze allemaal te gaan beschouwen.

Opmerking 7.27 In principe hebben we nu een algoritme om een LP-probleem op te lossen: enumereer over alle deelverzamelingen van grootte n uit een verzameling van $n + m$ elementen. Bepaal voor elk van die deelverzamelingen de bijhorende basisoplossing door het oplossen van het stelsel lineaire vergelijkingen. Controleer, wanneer de basisoplossing aanvaardbaar is, of de doelfunctiewaarde beter is dan de huidige beste. Indien ja, onthou dat deze aanvaardbare basisoplossing de huidige beste is.

Het probleem met de hierboven geschetste werkwijze is dat ze in de praktijk veel te lang gaat duren. Er zijn immers

$$\binom{n+m}{n} = \frac{(n+m)!}{n!m!}$$

basisoplossingen, waarvan de meeste niet aanvaardbaar zijn. Zoals blijkt uit Tabel 7.2 is het voor praktische problemen inderdaad niet haalbaar om alle basisoplossingen te overlopen. ∎

Het simplexalgoritme lost dit probleem op door

1. énkel aanvaardbare basisoplossingen te beschouwen én
2. ervoor te zorgen dat de basisoplossing snel (i.e. min of meer "op het zicht") kan bepaald worden.

Hiervoor introduceert men het concept van een naburige basisoplossing.

Definitie 7.28 Een NABURIGE BASISOPLOSSING van een basisoplossing is een basisoplossing waarvan alle basisvariabelen *op één na* gelijk zijn aan de

basisoplossing	naburige basisoplossingen
(a)	(b), (c), (d), (e), (f), (g)
(b)	(a), (c), (d), (e), (h), (i)
(c)	(a), (b), (d), (f), (h), (j)
(d)	(a), (b), (c), (g), (i), (j)
(e)	(a), (b), (f), (g), (h), (i)
(f)	(a), (c), (e), (g), (h), (j)
(g)	(a), (d), (e), (f), (i), (j)
(h)	(b), (c), (e), (f), (i), (j)
(i)	(b), (d), (e), (g), (h), (j)
(j)	(c), (d), (f), (g), (h), (i)

Tabel 7.3: Naburige basisoplossingen voor het LP-probleem in verhoogde vorm van Voorbeeld 7.21. De namen van de basisoplossingen verwijzen naar Tabel 7.1. Zoals verwacht heeft elke basis $m \times n = 3 \times 2 = 6$ naburige basisoplossingen.

basisvariabelen van de eerste basisoplossing. Of anders gezegd, twee basisoplossingen zijn naburig wanneer ze precies $m - 1$ basisvariabelen gemeenschappelijk hebben. ∎

Opmerking 7.29 Elke basisoplossing heeft $m \times n$ naburige basisoplossingen. Inderdaad, er zijn m mogelijkheden voor de variabele die men uit de basis gaat verwijderen, en voor elk van die m mogelijkheden zijn er n keuzes voor de niet-basisvariabele die men in de basis gaat brengen. ∎

Voorbeeld 7.30 In Tabel 7.1 staan alle basisoplossingen voor het LP-probleem van Voorbeeld 7.21 opgesomd. In Tabel 7.3 duiden we aan welke basisoplossingen *naburig* zijn van elkaar. ∎

Grafisch betekent een naburige basisoplossing dat men zich verplaatst langs een rechte aan de rand van het aanvaardbaar gebied.

We herinneren aan het reeds vermelde basisidee van het simplexalgoritme, nl. *spring van hoekpunt naar hoekpunt tot men zich niet meer kan verbeteren.*

We weten nu reeds uit Eigenschap 7.26 dat een hoekpunt van het aanvaardbaar gebied overeenkomt men een aanvaardbare basisoplossing. Wanneer alle coëfficiënten b_i groter of gelijk zijn aan nul, dan is

$$x_1 = 0, x_2 = 0, \ldots, x_n = 0, y_1 = b_1, y_2 = b_2, \ldots, y_m = b_m$$

een aanvaardbare basisoplossing. *Deze aanvaardbare basisoplossing kan onmiddellijk worden afgelezen uit de vorm van het stelsel, want elke basisvariabele komt precies éénmaal voor in het stelsel.*

Het simplexalgoritme voert elementaire rijoperaties uit om het stelsel lineaire vergelijkingen te transformeren naar een andere maar equivalente vorm zodanig dat deze eigenschap, nl. *elke basisvariabele heeft zijn eigen vergelijking*, geldig blijft gedurende de uitvoering van het algoritme.

Om gemakkelijk te kunnen bepalen wanneer het algoritme eindigt én welke niet-basisvariabele in de basis wordt opgenomen wordt er een extra gelijkheid toegevoegd aan het stelsel. Deze komt overeen met de doelfunctie, en wordt de DOELFUNCTIEVERGELIJKING genoemd. In deze doelfunctievergelijking worden alle variabelen in het linkerlid geschreven. Dit betekent dat de tekens van de coëfficiënten van de variabelen omkeren.

Om te beslissen welke niet-basis variabele de basis binnenkomt wordt er gekeken naar de doelfunctievergelijking. De niet-basisvariabele die de doelfunctie het snelst laat stijgen per eenheid stijging van de niet-basisvariabele is diegene die in de basis wordt opgenomen.

Voorbeeld 7.31 De doelfunctievergelijking voor het LP-probleem in Voorbeeld 7.21 is

$$-220x_1 - 255x_2 + D = 0.$$

De variabele x_1 laat de waarde van D stijgen met 220 eenheden per eenheid stijging van x_1; voor de variabele x_2 is dit 255 eenheden per eenheid stijging van x_2.

In dit geval zou x_2 in de basis worden opgenomen. ∎

Eens gekozen is welke variable in de basis zal komen zijn er nog m naburige basissen mogelijk. De meeste hiervan zijn echter niet-aanvaardbaar. Er is echter een eenvoudige rekenregel om te bepalen welke variabele uit de basis moet verdwijnen. We bepalen deze rekenregel a.d.h.v. een voorbeeld.

Voorbeeld 7.32 We beschouwen volgend LP-probleem in standaardvorm:

$$\max D(x_1, x_2) = 3x_1 + 2x_2$$

onder de beperkingen

$$\begin{cases} -4x_1 + 6x_2 \leq 21 \\ 2x_1 + 6x_2 \leq 39 \\ 4x_1 + 2x_2 \leq 33. \end{cases}$$

De niet-negativiteitsvoorwaarden zijn van kracht.

Als eerste stap herschrijven we het probleem in zijn verhoogde vorm, daartoe voegen we spelingsvariabelen toe. In dit voorbeeld stappen we af van de benaming y voor de spelingsvariabelen en noemen we deze ook eenvoudigweg x. De verhoogde vorm voor dit probleem is:

$$\max D(x_1, x_2, x_3, x_4, x_5) = 3x_1 + 2x_2 + 0x_3 + 0x_4 + 0x_5$$

onder de beperkingen

$$\begin{cases} -4x_1 + 6x_2 + x_3 = 21 \\ 2x_1 + 6x_2 + x_4 = 39 \\ 4x_1 + 2x_2 + x_5 = 33. \end{cases}$$

en $x_i \geq 0$ voor $i \in \{1, 2, 3, 4, 5\}$.

Vervolgens nemen we de doelfunctievergelijking op in het stelsel en we krijgen.

$$\begin{cases} -3x_1 - 2x_2 + D = 0 \\ -4x_1 + 6x_2 + x_3 = 21 \\ 2x_1 + 6x_2 + x_4 = 39 \\ 4x_1 + 2x_2 + x_5 = 33 \end{cases}$$

met $x_i \geq 0$ voor $i \in \{1, 2, 3, 4, 5\}$. Merk nogmaals op dat de coëfficiënten van de doelfunctievergelijking het tegengestelde teken hebben t.o.v. de doelfunctie zelf.

De triviale basis is (x_3, x_4, x_5, D). We kiezen $x_1 = x_2 = 0$ en dan lezen we de oplossing onmiddellijk af uit het stelsel: $(x_3, x_4, x_5, D) = (21, 39, 33, 0)$ (en $x_1 = x_2 = 0$).

De doelfunctievergelijking heeft twee negatieve coëfficiënten, nl. bij x_1 en x_2. Omdat de coëfficiënt bij x_1 het meest negatief is (en dus de doelfunctie het meest stijgt per eenheid stijging van de niet-basisvariabele) wordt besloten om x_1 in de basis te brengen.

Welke variabele moet de basis verlaten, x_3, x_4 of x_5? We proberen ze één voor één.

- Als x_3 de basis verlaat, dan zal x_1 een waarde krijgen gelijk aan $-21/4$. Dit volgt immers onmiddellijk uit de tweede vergelijking als we er rekening mee houden dat in dit geval x_2 en x_3 allebei de waarde nul aannemen. Dit is een niet-aanvaardbare basisoplossing!
- Als x_4 de basis verlaat lezen we uit de derde vergelijking onmiddellijk af dat x_1 de waarde $39/2$ krijgt. Deze waarde is positief, maar voor x_5 volgt dan uit de laatste vergelijking dat
$$x_5 = 33 - 4 \times \frac{39}{2} = -45 < 0.$$
Opnieuw hebben we te maken met een niet-aanvaardbare basisoplossing!
- Als x_5 de basis verlaat dan zien we uit de laatste vergelijking onmiddellijk dat x_1 de waarde $33/4$ krijgt. Uit de voorlaatste vergelijking volgt dan dat
$$x_4 = 39 - 2x_1 = 39 - \frac{33}{2} = \frac{45}{2}$$
en uit de tweede vergelijking volgt dat
$$x_3 = 21 + 4x_1 = 21 + 33 = 54.$$

Eens de keuze gemaakt is dat x_1 een basisvariabele wordt is de enige mogelijkheid om over te gaan op een aanvaardbare naburige basis om x_5 uit de basis te laten verdwijnen. Merk op dat de vergelijking voor x_5 diegene is waarvoor de verhouding van zijn rechterlid en de coëfficiënt van x_1 (nl. $33/4$) de kleinste is onder alle niet-negatieve verhoudingen.

Alhoewel we nu wel weten (door het uitvoeren van een kleine berekening) wat de waarden zijn van de naburige basisoplossing (x_3, x_4, x_1) is het niet mogelijk om deze oplossing op het zicht te bepalen uit de huidige vorm van het stelsel.

We gaan nu het stelsel herschrijven zodanig dat x_1 slechts éénmaal zal voorkomen in het stelsel, nl. in de vergelijking waar x_5 nu in staat. Dit kan men realiseren door x_1 te bepalen uit de laatste vergelijking, nl.
$$x_1 = \frac{1}{4}(33 - 2x_2 - x_5)$$

en deze uitdrukking voor x_1 te substitueren in de andere drie vergelijkingen. Voor de eerste vergelijking krijgen we

$$\begin{aligned} & -3x_1 - 2x_2 + D = 0 \\ \iff & -\frac{3}{4}(33 - 2x_2 - x_5) - 2x_2 + D = 0 \\ \iff & -\frac{99}{4} + \frac{3}{2}x_2 + \frac{3}{4}x_5 - 2x_2 + D = 0 \\ \iff & -\frac{1}{2}x_2 + \frac{3}{4}x_5 + D = \frac{99}{4}. \end{aligned}$$

Dit wordt de nieuwe doelfunctievergelijking.

Voor de vergelijking voor x_3, i.e. voor de tweede vergelijking krijgen we:

$$\begin{aligned} & -4x_1 + 6x_2 + x_3 = 21 \\ \iff & -(33 - 2x_2 - x_5) + 6x_2 + x_3 = 21 \\ \iff & 8x_2 + x_3 + x_5 = 54. \end{aligned}$$

Voor de vergelijking van x_4 krijgen we

$$\begin{aligned} & 2x_1 + 6x_2 + x_4 = 39 \\ \iff & \frac{1}{2}(33 - 2x_2 - x_5) + 6x_2 + x_4 = 39 \\ \iff & 5x_2 + x_4 - \frac{1}{2}x_5 = \frac{45}{2}. \end{aligned}$$

Samenvattend kunnen we zeggen dat we het stelsel hebben omgevormd naar

$$\begin{cases} -\frac{1}{2}x_2 + \frac{3}{4}x_5 + D = \frac{99}{4} \\ 8x_2 + x_3 + x_5 = 54 \\ 5x_2 + x_4 - \frac{1}{2}x_5 = \frac{45}{2} \\ 4x_1 + 2x_2 + x_5 = 33. \end{cases}$$

Merk op dat we nu opnieuw in de situatie zijn waarin we de waarde van de basisvariabelen, nl. x_3, x_4 en x_1 "op het zicht" kunnen aflezen. Ook de waarde van de doelfunctie (nl. $99/4$) kan onmiddellijk afgelezen worden.

We zijn m.a.w. in een situatie die zeer gelijkend is aan de beginsituatie. Dezelfde stappen kunnen bijgevolg herhaald worden, nl.:

- Kies de variabele die in de basis wordt gebracht.
- Kies de variabele die uit de basis verdwijnt.
- Herschrijf het stelsel zodat elke basisvariabele zijn "eigen" vergelijking heeft en precies éénmaal voorkomt in het stelsel. ∎

7.3.4 Rekenregels van het simplexalgoritme

Beslissen welke variabele in de basis wordt gebracht. We kiezen steeds de niet-basisvariabele wiens coëfficiënt in de huidige doelfunctievergelijking het meest negatief is. Wanneer zo'n niet-basisvariabele niet bestaat dan stopt het algoritme. Wanneer er gelijkspel is tussen meerdere niet-basisvariabelen, dan wordt steeds de "eerste" gekozen[9].

Beslissen welke variabele uit de basis verdwijnt. Voor elke basisvervariabele berekenen we de verhouding van het rechterlid van zijn vergelijking en de coëfficiënt van de variabele waarvan beslist is dat die in de basis komt. De vergelijking (i.e. basisvariabele) met de kleinste niet-negatieve verhouding is de variabele die uit de basis verdwijnt.

Herschrijven van het stelsel lineaire vergelijkingen. Nadat beslist is welke variabele in de basis komt en welke eruit verdwijnt moet het stelsel herschreven worden zodanig dat de nieuwe basisvariabele precies één keer voorkomt in het stelsel. Deze variabele moet m.a.w. geëlimineerd worden uit alle vergelijkingen behalve uit de vergelijking waar de "oude" basisvariabele in voorkomt. In principe kan men dit doen zoals voorgedaan in Voorbeeld 7.32, maar het is eenvoudiger om dit te realiseren a.d.h.v. elementaire rijoperaties op de verhoogde matrix van dit stelsel. Dit is volledig equivalent met de elementaire rijoperaties die werden uitgevoerd bij de methode van Gauss-Jordan en de methode van Gauss bij het oplossen van stelsels lineaire vergelijkingen[10]. Deze methodes werden eveneens (als oefening) geïmplementeerd in de cursus Probleemoplossend Denken I, wat erop duidt dat het bepalen en uitvoeren van dit soort van elementaire rijoperaties gemakkelijk te automatiseren is.

[9] Volgens een bepaalde volgorde van de variabelen.
[10] Zie cursus Math4IT.

Voorbeeld 7.33 (Vervolg simplexmethode) Het stelsel heeft momenteel de volgende vorm:

$$\begin{cases} -\frac{1}{2}x_2 + \frac{3}{4}x_5 + D = \frac{99}{4} \\ 8x_2 + x_3 + x_5 = 54 \\ 5x_2 + x_4 - \frac{1}{2}x_5 = \frac{45}{2} \\ 4x_1 + 2x_2 + x_5 = 33. \end{cases}$$

of in matrixnotatie (waarbij voor de duidelijkheid de rijen geannoteerd worden met de basisvariabelen en de kolommen met de variabele die ze representeren):

$$\begin{array}{c} \\ D \\ x_3 \\ x_4 \\ x_1 \end{array} \begin{array}{c} \begin{array}{ccccccc} x_1 & x_2 & x_3 & x_4 & x_5 & D & \text{RL} \end{array} \\ \begin{pmatrix} 0 & -1/2 & 0 & 0 & 3/4 & 1 & 99/4 \\ 0 & 8 & 1 & 0 & 1 & 0 & 54 \\ 0 & 5 & 0 & 1 & -1/2 & 0 & 45/2 \\ 4 & 2 & 0 & 0 & 1 & 0 & 33 \end{pmatrix} \end{array}$$

Aangezien x_2 de enige variabele is met een negatieve coëfficiënt in de doelfunctievergelijking is dit de variabele die in de basis komt. Om te beslissen wie uit de basis verdwijnt berekenen we de verhoudingen van de rechterleden en de coëfficiënt van x_2. We vinden voor x_3

$$54/8 = 6.75,$$

voor x_4

$$45/2/5 = 4.5$$

en voor x_1

$$33/2 = 16.5$$

In dit geval verdwijnt x_4 uit de basis. De variabele x_2 moet nu geëlimineerd worden uit alle vergelijkingen behalve uit de derde vergelijking. Hiertoe voeren we de volgende elementaire rijoperaties uit:

$$R_1 \leftarrow R_1 - \frac{-1/2}{5} R_3,$$

of equivalent en eenvoudiger te berekenen[11]

$$R_1 \leftarrow 10R_1 + R_3.$$

[11] althans met de hand

Voor de tweede rij vinden we de volgende elementaire rijoperatie:

$$R_2 \leftarrow R_2 - \frac{8}{5}R_3 \quad \text{of eenvoudiger} \quad R_2 \leftarrow 5R_2 - 8R_3,$$

en tenslotte voor de vierde rij.

$$R_4 \leftarrow R_4 - \frac{2}{5}R_3 \quad \text{of eenvoudiger} \quad R_4 \leftarrow 5R_4 - 2R_3.$$

Merk op dat steeds R_3 wordt gebruikt omdat dát de rij is die ongewijzigd moet worden overgenomen. Na het uitvoeren van de elementaire rijoperaties wordt de verhoogde matrix

$$\begin{array}{c} \\ D \\ x_3 \\ x_2 \\ x_1 \end{array} \begin{array}{c} \begin{array}{ccccccc} x_1 & x_2 & x_3 & x_4 & x_5 & D & \text{RL} \end{array} \\ \left(\begin{array}{ccccccc} 0 & 0 & 0 & 1 & 7 & 10 & 270 \\ 0 & 0 & 5 & -8 & 9 & 0 & 90 \\ 0 & 5 & 0 & 1 & -1/2 & 0 & 45/2 \\ 20 & 0 & 0 & -2 & 6 & 0 & 120 \end{array} \right) \end{array}$$

Hieruit lezen we onmiddellijk de huidige oplossing af:

$$D = 270/10 = 27, x_3 = 90/5 = 18, x_2 = 45/10 = 9/2, x_1 = 6$$

en de niet-basisvariabelen:

$$x_4 = x_5 = 0.$$

Omdat in de doelfunctievergelijking alle coëfficiënten positief zijn betekent dit dat we de optimale oplossing van dit LP-probleem hebben gevonden. ∎

Voorbeeld 7.34 (Grafische interpretatie van de simplexmethode) Het is interessant om voor dit kleine LP-probleem na te gaan welke basisoplossingen werden beschouwd en in welke volgorde dit gebeurde.

We startten in de oorsprong:

$$x_1 = 0, x_2 = 0, x_3 = 21, x_4 = 39, x_5 = 33, D = 0.$$

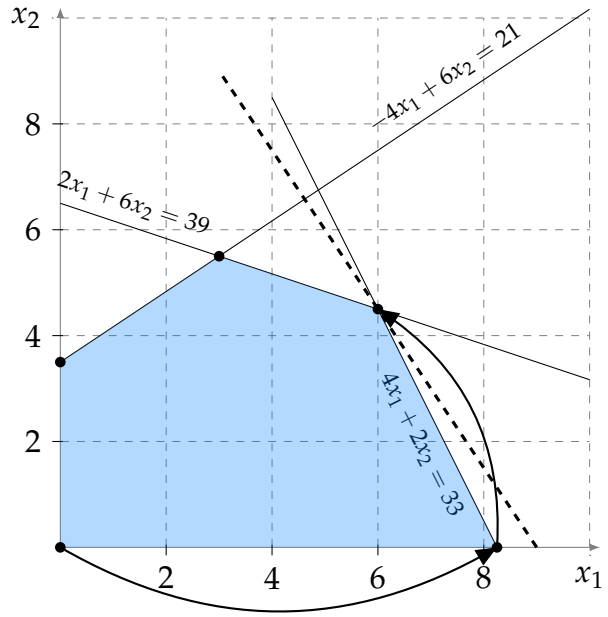
De variabele x_1 kwam in de basis ten koste van x_5 en we kregen de oplossing:

$$x_1 = 33/4, x_2 = 0, x_3 = 54, x_4 = 45/2, x_5 = 0, D = 99/4.$$

Vervolgens werd x_2 in de basis gebracht ten koste van x_4 en we vonden

$$x_1 = 6, x_2 = 9/2, x_3 = 18, x_4 = 0, x_5 = 0, D = 27.$$

Merk op dat we inderdaad van hoekpunt naar hoekpunt gesprongen zijn; dit kan men zien in Figuur 7.4. ∎



Figuur 7.4: De basisoplossingen bezocht door het simplexalgoritme. Het aanvaardbaar gebied is aangeduid. De oorsprong is de startoplossing. In dit geval wordt de optimale oplossing bereikt na twee iteraties. De doelfunctierechte waarvoor de optimale doelfunctiewaarde wordt bereikt is aangeduid.

7.4 De II-fasen methode

Wanneer de oorsprong tot het aanvaardbaar gebied behoort dan is het eenvoudig om een startoplossing (en bijhorend stelsel) te vinden voor het simplexalgoritme. *Hoe vinden we echter op een eenvoudige manier een aanvaardbare basisoplossing en de vorm van het bijhorend stelsel wanneer de oorsprong niet tot het aanvaardbaar gebied behoort?*

Voorbeeld 7.35 Beschouw het volgende LP-probleem in twee variabelen:

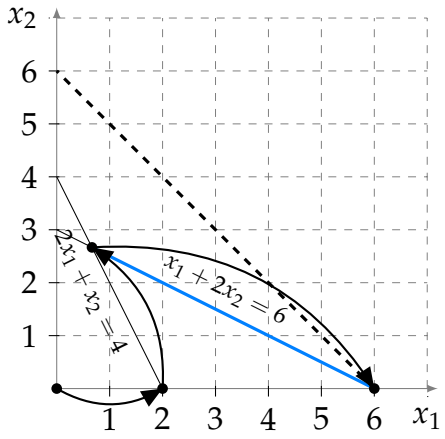
$$\max D(x_1, x_2) = x_1 + x_2$$

onder de beperkingen

$$\begin{cases} 2x_1 + \ \ x_2 \geq 4 \\ \ \ x_1 + 2x_2 = 6. \end{cases}$$

en waarbij de niet-negativiteitsvoorwaarden van kracht zijn:

$$x_j \geq 0 \quad \text{voor } j \in \{1, 2\}.$$



Figuur 7.5: Een LP-probleem waarvoor de oorsprong niet tot het aanvaardbaar gebied behoort. Het aanvaardbaar gebied bestaat enkel uit het in het blauw aangeduide lijnstuk. Men ziet ook de punten, startend in de oorsprong, zoals die worden bezocht door de II-fasen methode. De eerste fase eindigt wanneer men de rand van het aanvaardbaar gebied bereikt.

Men gaat onmiddellijk na dat de oorsprong, nl. $x_1 = 0$ en $x_2 = 0$ niet tot het aanvaardbaar gebied behoort. ∎

Opmerking 7.36 Je ziet dat in het voorgaand voorbeeld er een restrictie voorkomt van de $=$-vorm. We hebben vroeger vermeld dat zo'n restrictie equivalent is met twee restricties, nl. één in de $\leq$-vorm en één in de $\geq$-vorm. Dit is correct maar het is efficiënter om de oplossingswijze te volgen die we hierna voorop stellen. ∎

Wanneer men een restrictie heeft van de $\geq$-vorm, waarvan de coëfficiënt b positief is, dan betekent dit dat in dit geval een eventuele spelingsvariabele negatief zou moeten zijn. We willen echter het idee om enkel met niet-negatieve variabelen te werken behouden. Daarom voegen we de spelingsvariabele aan zo'n restrictie toe met een minteken.

M.a.w. de restrictie

$$a_1x_1 + a_2x_2 + \cdots + a_nx_n \geq b,$$

met $b > 0$, wordt in eerste instante omgevormd naar de gelijkheid:

$$a_1x_1 + a_2x_2 + \cdots + a_nx_n - y = b,$$

maar wanneer alle $x_j = 0$ worden gesteld dan is $y = -b < 0$ wat niet toegelaten is! Om te kunnen starten in de oorsprong voegen we *nog een extra variabele toe*, dit wordt een KUNSTMATIGE VARIABELE genoemd, of

$$a_1x_1 + a_2x_2 + \cdots + a_nx_n - y + z = b,$$

zodanig dat $x_j = 0$ en $y = 0$ en $z = b$ een aanvaardbare oplossing is voor deze vergelijking.

Voor een restrictie in de $=$-vorm is er zelfs helemaal geen speling. In dit geval wordt enkel een kunstmatige variabele ingevoerd. Met andere woorden de restrictie

$$a_1x_1 + a_2x_2 + \cdots + a_nx_n = b$$

wordt omgezet in

$$a_1x_1 + a_2x_2 + \cdots + a_nx_n + z = b,$$

waarbij z een kunstmatige variabele voorstelt.

Voorbeeld 7.37 Het stelsel lineaire vergelijkingen (met toevoeging van de doelfunctievergelijking) voor het voorgaand LP-probleem is:

$$\begin{cases} -x_1 - x_2 + D = 0 \\ 2x_1 + x_2 - x_3 + \underline{x_4} = 4 \\ x_1 + 2x_2 + \underline{x_5} = 6. \end{cases}$$

Hierbij is x_3 een spelingsvariabele en werden de kunstmatige variabelen voor de duidelijkheid eventjes onderlijnd. ∎

Die extra kunstmatige variabelen willen we echter helemaal niet! *We zoeken een aanvaardbare oplossing waarvoor alle kunstmatige variabelen de waarde nul aannemen.* De minimale waarde van de som van alle kunstmatige variabelen is gelijk aan nul. Bovendien, wanneer de som van alle kunstmatige variabelen gelijk is aan nul dan zijn alle individuele kunstmatige variabelen gelijk aan nul. We kunnen m.a.w. de oorspronkelijke doelfunctie eventjes "vergeten" en vervangen door

$$\min D_{\min} = \text{som van alle kunstmatige variabelen.}$$

Omdat de procedure zoals besproken echter enkel werkt voor maximalisatie gaan we nu over op

$$\max D_{\max} = -(\text{som van alle kunstmatige variabelen}),$$

zodanig dat de doelfunctievergelijking wordt

$$\text{som van alle kunstmatige variabelen} + D_{\max} = 0.$$

Voorbeeld 7.38 Voor het voorgaande LP-probleem vinden we:

$$\begin{cases} \underline{x}_4 + \underline{x}_5 + D_{\max} = 0 \\ 2x_1 + x_2 - x_3 + \underline{x}_4 = 4 \\ x_1 + 2x_2 + \underline{x}_5 = 6. \end{cases}$$

In dit geval is $\underline{x}_4 = 4$ en $\underline{x}_5 = 6$ en $x_1 = x_2 = x_3 = 0$ een aanvaardbare oplossing van dit stelsel vergelijkingen. Er is echter een probleem, de correcte waarde van $D_{\max}$ nl. -10 kan niet onmiddellijk worden afgelezen uit dit stelsel. De reden hiervoor is dat elke kunstmatige variabele twee keer voorkomt in het stelsel, één keer in zijn "eigen" vergelijking en één keer in de doelfunctievergelijking. Dit probleem kan men eenvoudig oplossen door de kunstmatige variabelen te elimineren uit de doelfunctievergelijking door het toepassen van een gepaste elementaire rijoperatie.

Wanneer we de elementaire rijoperatie

$$R_1 \leftarrow R_1 - R_2 - R_3$$

toepassen dan wordt het stelsel

$$\begin{cases} -3x_1 - 3x_2 + x_3 + D_{\max} = -10 \\ 2x_1 + x_2 - x_3 + \underline{x}_4 = 4 \\ x_1 + 2x_2 + \underline{x}_5 = 6, \end{cases}$$

en staat het wel in de correcte vorm voor het toepassen van de rekenregels van het simplexalgoritme. ∎

We zien m.a.w. het volgende: *alvorens te starten met de rekenregels van het simplexalgoritme moet men de kunstmatige variabelen elimineren uit de doelfunctievergelijking zodanig dat elke basisvariabele (kunstmatig of niet) precies éénmaal voorkomt in het stelsel!*

Voorbeeld 7.39 Wanneer we nu de rekenregels van het simplexalgoritme toepassen op het stelsel

$$\begin{array}{c} \\ D_{\max} \\ \underline{x}_4 \\ \underline{x}_5 \end{array} \begin{array}{c} \begin{array}{ccccccc} x_1 & x_2 & x_3 & \underline{x}_4 & \underline{x}_5 & D_{\max} & \text{RL} \end{array} \\ \begin{pmatrix} -3 & -3 & 1 & 0 & 0 & 1 & -10 \\ 2 & 1 & -1 & 1 & 0 & 0 & 4 \\ 1 & 2 & 0 & 0 & 1 & 0 & 6 \end{pmatrix} \end{array}$$

met basisoplossing $(\underline{x}_4, \underline{x}_5, D_{\max}) = (4, 6, -10)$ dan vinden we dat x_1 in de basis komt ten koste van $\underline{x}_4$. We passen de volgende elementaire rijoperaties toe:

$$R_1 \leftarrow 2R_1 + 3R_2, \quad \text{en} \quad R_3 \leftarrow 2R_3 - R_2$$

en we vinden

$$\begin{array}{c} \\ D_{\max} \\ x_1 \\ \underline{x}_5 \end{array} \begin{array}{c} \begin{array}{ccccccc} x_1 & x_2 & x_3 & \underline{x}_4 & \underline{x}_5 & D_{\max} & \text{RL} \end{array} \\ \begin{pmatrix} 0 & -3 & -1 & 3 & 0 & 2 & -8 \\ 2 & 1 & -1 & 1 & 0 & 0 & 4 \\ 0 & 3 & 1 & -1 & 2 & 0 & 8 \end{pmatrix} \end{array}$$

We zien dat $\underline{x}_4$ de waarde nul heeft gekregen en dat $D_{\max}$ gestegen is naar -4. Op dit moment brengen we x_2 in de basis ten koste van $\underline{x}_5$. Hierdoor zal $\underline{x}_5$ eveneens de waarde nul krijgen en zullen we een hoekpunt van het aanvaardbaar gebied bereikt hebben. We moeten enkel nog $\underline{x}_5$ elimineren uit de eerste en de tweede vergelijking. Dit doen we door het uitvoeren van de volgende elementaire rijoperaties:

$$R_1 \leftarrow R_1 + R_3, \quad \text{en} \quad R_2 \leftarrow 3R_2 - R_3.$$

We vinden:

$$\begin{array}{c} \\ D_{\max} \\ x_1 \\ x_2 \end{array} \begin{array}{c} \begin{array}{ccccccc} x_1 & x_2 & x_3 & \underline{x}_4 & \underline{x}_5 & D_{\max} & \text{RL} \end{array} \\ \begin{pmatrix} 0 & 0 & 0 & 2 & 2 & 2 & 0 \\ 6 & 0 & -4 & 4 & -2 & 0 & 4 \\ 0 & 3 & 1 & -1 & 2 & 0 & 8 \end{pmatrix} \end{array}$$

met bijhorende basisoplossing $(x_1, x_2, D_{\max}) = (2/3, 8/3, 0)$. ∎

Op het moment dat we een basisoplossing hebben gevonden waarvoor alle kunstmatige variabelen gelijk zijn aan nul, is de rol van de kunstmatige variabelen uitgespeeld. We verwijderen deze uit het stelsel. Uiteraard voeren we ook de originele doelfunctievergelijking terug in, want we zijn uiteindelijk geïnteresseerd in het optimaliseren van de originele doelfunctie.

Voorbeeld 7.40 Wanneer we de kunstmatige variabelen verwijderen uit het stelsel en de originele doelfunctievergelijking terug invoeren dan vinden we:

$$\begin{cases} -x_1 - x_2 + D = 0 \\ 6x_1 - 4x_3 = 4 \\ 3x_2 + x_3 = 8. \end{cases}$$

De gevonden basisoplossing is: $(x_1, x_2, D) = (2/3, 8/3, 10/3)$, maar opnieuw kunnen we op dit stelsel niet rechtstreeks de rekenregels van de simplexprocedure toepassen omdat de basisvariabelen x_1 en x_2 ook voorkomen in de doelfunctievergelijking. ∎

We merken het volgende: *na het verwijderen van de kunstmatige variabelen en het terug invoeren van de doelfunctievergelijking zal het over het algemeen zo zijn dat één of meerdere basisvariabelen voorkomen in de doelfunctievergelijking. Deze moeten hieruit geëlimineerd worden alvorens verder te gaan.*

Voorbeeld 7.41 In matrixvorm ziet het stelsel er op dit moment als volgt uit:

$$\begin{array}{c} \\ D \\ x_1 \\ x_2 \end{array} \begin{array}{c} \begin{array}{ccccc} x_1 & x_2 & x_3 & D & \text{RL} \end{array} \\ \begin{pmatrix} -1 & -1 & 0 & 1 & 0 \\ 6 & 0 & 4 & 0 & 4 \\ 0 & 3 & 1 & 0 & 8 \end{pmatrix} \end{array}$$

We voeren de rijoperatie

$$R_1 \leftarrow 6R_1 + R_2 + 2R_3$$

uit om x_1 en x_2 te elimineren uit de eerste vergelijking en we vinden

$$\begin{array}{c} \\ D \\ x_1 \\ x_2 \end{array} \begin{array}{c} \begin{array}{ccccc} x_1 & x_2 & x_3 & D & \text{RL} \end{array} \\ \begin{pmatrix} 0 & 0 & -2 & 6 & 20 \\ 6 & 0 & -4 & 0 & 4 \\ 0 & 3 & 1 & 0 & 8 \end{pmatrix} \end{array}$$

Nu kunnen de rekenregels van de simplexprocedure worden toepast. De variable x_3 komt in de basis en x_2 verlaat de basis. Na het uitvoeren van de elementaire rijoperaties

$$R_1 \leftarrow R_1 + 2R_3 \quad \text{en} \quad R_2 \leftarrow R_2 + 4R_3$$

wordt het stelsel

$$\begin{array}{c} \\ D \\ x_1 \\ x_3 \end{array} \begin{array}{c} \begin{array}{ccccc} x_1 & x_2 & x_3 & D & \text{RL} \end{array} \\ \begin{pmatrix} 0 & 6 & 0 & 6 & 36 \\ 6 & 12 & 0 & 0 & 36 \\ 0 & 3 & 1 & 0 & 8 \end{pmatrix} \end{array}$$

Alle coëfficiënten in de doelfunctievergelijking zijn positief. Bijgevolg heeft de simplexprocedure de optimale oplossing gevonden. Die wordt gegeven door

$$x_1 = 6, x_2 = 0, x_3 = 8, D = 6.$$

Figuur 7.5 toont de verschillende punten in het (x_1, x_2)-vlak die werden bezocht tijdens de uitvoering van de II-fasen methode. ∎

7.4.1 II-fasen methode: samenvatting

Samenvattend kunnen we zeggen dat men om een LP-probleem op te lossen de volgende stappen doorloopt.

1. Zorg ervoor dat elke restrictie een rechterlid b_i heeft dat groter of gelijk is aan nul. Dit kan steeds bereikt worden door te vermenigvuldigen met -1 indien nodig.
2. Voeg spelings- en kunstmatige variabelen toe:
   - Een restrictie in de $\leq$-vorm krijgt een spelingsvariabele met een plusteken.
   - Een restrictie in de $\geq$-vorm krijgt een spelingsvariabele met een minteken en een kunstmatige variabele met een plusteken.
   - Een restrictie in de $=$-vorm krijgt een kunstmatige variabele met een plusteken.
3. Indien er kunstmatige variabelen werden toegevoegd moeten deze verwijderd worden door de tijdelijke doelfunctievergelijking

   $$\text{som van de kunstmatige variabelen} + D_{\max} = 0$$

   toe te voegen. Dit gaat als volgt:
   a) Elimineer de kunstmatige variabelen uit de doelfunctievergelijking.
   b) Pas de rekenregels van het simplexalgoritme toe totdat $D_{\max}$ de waarde nul aanneemt. Als dit niet mogelijk is, dan is het aanvaardbaar gebied voor dit LP-probleem leeg.
4. Indien er kunstmatige variabelen werden toegevoegd, dan worden deze verwijderd en de originele doelfunctievergelijking wordt terug ingevoerd. In het algemeen zullen er basisvariabelen uit de doelfunctievergelijking moeten worden geëlimineerd.
5. Nu staat het stelsel in de correcte vorm voor het toepassen van de rekenregels van de simplexprocedure. Herhaal deze totdat de simplexprocedure stopt.

7.5 Details van de simplexprocedure

Eindigt de simplexprocedure steeds? Wanneer in elke stap de waarde van elke basisvariabele strikt positief is (m.a.w. wanneer er nooit degeneratie optreedt) dan wordt de waarde van de doelfunctie in elke stap strikt groter. Dit betekent dat het algoritme nooit twee keer dezelfde basis zal ontmoeten; aangezien het aantal basissen eindig is moet het algoritme eindigen na een eindig aantal stappen.

Deze redenering is niet langer geldig wanneer er degeneratie optreedt, m.a.w. wanneer er basisvariabelen de waarde nul aannemen, omdat in dit geval de waarde van de doelfunctie gelijk kan blijven van de ene iteratie naar de volgende. Er zijn voorbeelden geconstrueerd waarvoor de simplexprocedure in de versie zoals we ze in deze tekst hebben besproken inderdaad in een oneindige lus raakt. In Oefening 4 vind je een voorbeeld hiervan.

In de praktijk komt het gelukkig niet vaak voor dat de simplexprocedure vastraakt in een oneindige lus, maar theoretisch kan het wel gebeuren.

Uitvoeringstijd van de simplexprocedure Wanneer de simplexprocedure eindigt dan gebeurt dit *meestal* na een "beperkt" aantal iteraties. Experimentele studies hebben aangetoond dat het gemiddeld aantal iteraties (basisveranderingen) dat men nodig heeft om de optimale oplossing te bereiken evenredig is met m.

Er zijn echter problemen (die hier speciaal voor opgesteld werden) waarvoor de simplexprocedure een exponentieel aantal stappen nodig heeft. Eén zo'n voorbeeld is de "kubus" van Klee-Minty die in het algemeen opgebouwd kan worden in een willekeurig aantal dimensies n. De "kubus" bestaat uit n restricties en heeft 2^n hoekpunten. Wanneer een bepaalde doelfunctie moet worden gemaximaliseerd over deze kubus dan bezoekt het simplexprocedure alle 2^n hoekpunten! De tijdscomplexiteit van de simplexprocedure op dit voorbeeld is m.a.w. exponentieel in het aantal veranderlijken (en het aantal restricties). In Oefening 5 wordt dit geïllustreerd.

Hoe herkent de simplexprocedure dat het aanvaardbaar gebied leeg is? Wanneer de eerste fase eindigt en de oplossing bevat artificiële variabelen die verschillend zijn van nul, dan betekent dit dat het aanvaardbaar gebied leeg is.

Voorbeeld 7.42 Beschouw de beperkingen van het LP-probleem uit Voorbeeld 7.11:

$$\begin{cases} 2x_1 + x_2 \leq 1 \\ x_1 + x_2 \geq 2 \end{cases}$$

Als we deze beperkingen omzetten naar hun verhoogde vorm dan krijgen we

$$\begin{cases} 2x_1 + x_2 + x_3 = 1 \\ x_1 + x_2 - x_4 + \underline{x}_5 = 2 \end{cases}$$

waarbij x_3 en x_4 spelingsvariabelen zijn en $\underline{x}_5$ een kunstmatige variabele is. Wanneer we nu de negatie van de som van de kunstmatige variabelen gaan maximaliseren, dan voegen we een doelfunctievergelijking toe aan het stelsel en we krijgen:

$$\begin{cases} \underline{x}_5 + D_{\max} = 0 \\ 2x_1 + x_2 + x_3 = 1 \\ x_1 + x_2 - x_4 + \underline{x}_5 = 2 \end{cases}$$

Waarbij we vertrekken van de oplossing $(x_3, x_5, D_{\max}) = (1, 2, -2)$ (en alle andere variabelen gelijk aan nul). Uiteraard moeten we eerst het stelsel opnieuw klaarzetten voor Gaussische eliminatie en we doen dit door de derde vergelijking af te trekken van de eerste. We krijgen dan:

$$\begin{cases} -x_1 - x_2 + x_4 + D_{\max} = -2 \\ 2x_1 + x_2 + x_3 = 1 \\ x_1 + x_2 - x_4 + \underline{x}_5 = 2 \end{cases}$$

We brengen eerst x_1 in de basis, ten koste van x_3. Het stelsel wordt

$$\begin{cases} -x_2 + x_3 + 2x_4 + 2D_{\max} = -3 \\ 2x_1 + x_2 + x_3 = 1 \\ x_2 - x_3 - 2x_4 + 2\underline{x}_5 = 3 \end{cases}$$

De waarde van de doelfunctie is nu $-3/2$ (en is m.a.w. gestegen t.o.v. -2) Vervolgens wordt x_2 in de basis gebracht (ten koste van x_1) en het stelsel wordt nu

$$\begin{cases} 2x_1 + 2x_3 + 2x_4 + 2D_{\max} = -2 \\ 2x_1 + x_2 + x_3 = 1 \\ -2x_1 - 2x_3 - 2x_4 + 2x_5 = 2 \end{cases}$$

De waarde van de doelfunctie is nu -1, maar aangezien alle coëfficiënten in de doelfunctievergelijking niet-negatief zijn kan deze waarde niet langer verbeterd worden. De eerste fase eindigt met een oplossing waarin een kunstmatige variabele ($\underline{x}_5$ in dit geval) een niet-nul waarde heeft. Zo herkent het algoritme dat het aanvaardbaar gebied leeg is. ∎

Hoe herkent de simplexprocedure een onbegrensde oplossing? Veronderstel dat op een bepaald moment beslist wordt dat de variabele x_s in de basis komt, en dat alle coëfficiënten $a_{i,s}$ kleiner of gelijk aan nul zijn. Dit betekent dat alle testratio's hoogstens nul zijn aangezien de coëfficiënten b_i steeds groter of gelijk zijn aan nul. Het feit dat beslist werd dat x_s in de basis komt betekent dat de coëfficiënt bij x_s in de doelfunctievergelijking strikt negatief is, en dat een grotere waarde voor x_s bijgevolg aanleiding geeft tot een grotere waarde van de doelfunctie.

Voor de andere vergelijkingen krijgen we dan

$$\text{basisvariabele} = b_i - a_{i,s}x_s.$$

Merk op dat b_i de huidige waarde is van deze basisvariabele, en dus $b_i \geq 0$. Bovendien hebben we aangenomen dat $a_{i,s}$ kleiner of gelijk aan nul is. Dit betekent dat er geen "natuurlijke" begrenzing voor x_s. Anders gezegd, men kan aan x_s een willekeurig grote waarde geven, en men zal een aanvaardbare oplossing bekomen (die evenwel geen basisoplossing is) van het stelsel lineaire vergelijkingen waardoor de doelfunctie willekeurig groot kan worden gemaakt.

Voorbeeld 7.43 Beschouw het LP-probleem uit Voorbeeld 7.10. Na het einde van de eerste fase (die succesvol verloopt) verkrijgt men na het uitvoeren van de Gaussische eliminatie het volgend stelsel:

$$\begin{cases} \quad -5x_2 - x_3 + 2D = 1 \\ 2x_1 + \quad x_2 - x_3 \qquad = 1. \end{cases}$$

Na het inbrengen van x_2 in de basis krijgt men de volgende vorm:

$$\begin{cases} 10x_1 \qquad -6x_3 + 2D = 6 \\ \quad 2x_1 + x_2 - \quad x_3 \qquad = 1. \end{cases}$$

Hier zou x_3 in de basis moeten komen. De coëfficiënt van x_3 in de tweede (en enige) vergelijking is -1. Dit betekent dat

$$x_2 = 1 + x_3,$$

en dit betekent dat men x_3 een willeurig grote waarde kan geven zonder dat de oplossing niet-aanvaardbaar wordt. Aangezien

$$D = 3 + 3x_3$$

wordt ook de waarde van D willekeurig groot. ∎

7.6 Oefeningen

1. Los het broekenprobleem op m.b.v. de grafische methode. Ter herinnering, dit probleem wordt gegeven door:

   $$\max D(x_1, x_2) = 70x_1 + 40x_2$$

   onder de beperkingen

   $$\begin{cases} 3x_1 + x_2 \leq 192 \\ x_1 \leq 60 \\ x_2 \leq 100 \\ x_1 + x_2 \leq 120 \end{cases}$$

   en waarbij verder de niet-negativiteitsvoorwaarden van kracht zijn. Merk op dat we hier de vaste kost achterwege hebben gelaten. Om te bepalen of de activiteit al dan niet economisch zinvol is moet deze natuurlijk wel in rekening worden gebracht.

2. Los het volgende LP-probleem op m.b.v. de grafische methode:

   $$\min D(x_1, x_2) = -x_1 + 2x_2$$

   onder de beperkingen

   $$\begin{cases} -x_1 + x_2 \leq 6 \\ 3x_1 + x_2 \geq 8 \\ -2x_1 + x_2 \geq 3 \end{cases}$$

   waarbij de niet-negativiteitsvoorwaarden van kracht zijn:

   $$x_1 \geq 0 \quad \text{en} \quad x_2 \geq 0.$$

3. Los het LP-probleem uit opgave 2 op m.b.v. de II-fasen methode. Maak gebruik van een programma zoals Excel of R om de elementaire rij-operaties snel en correct te kunnen uitvoeren. Duid op de figuur aan welke punten werden bezocht in het (x_1, x_2)-vlak. Ga in het bijzonder na waar de simplexprocedure is aanbeland op het einde van de eerste fase.

4. **Simplex procedure vast in oneindige lus** Het volgende voorbeeld is speciaal geconstrueerd opdat de simplexprocedure in een oneindige lus zou raken:

$$\max D(x_1, x_2, x_3, x_4) = 2.3x_1 + 2.15x_2 - 13.55x_3 - 0.4x_4$$

onder de beperkingen

$$\begin{cases} 0.4x_1 + 0.2x_2 - 1.4x_3 - 0.2x_4 \leq 0 \\ -7.8x_1 - 1.4x_2 + 7.8x_3 + 0.4x_4 \leq 0 \\ x_1 \leq 1 \\ x_2 \leq 1. \end{cases}$$

Gebruik Excel om twee iteraties van de simplexprocedure toe te passen. Wanneer meerdere variabelen uit de basis kunnen worden verwijderd kies dan diegene met de grootste pivotwaarde. (Deze regel is ook interessant vanuit het standpunt van numerieke stabiliteit.)

Wat merk je na twee iteraties? Wat zal er gebeuren na 6 iteraties?

Opmerking: Het feit dat de simplexprocedure vastloopt in een oneindige lus betekent *niet* dat dit probleem geen optimale oplossing heeft. Gebruik een software-pakket om deze optimale oplossing te bepalen.

5. **Simplexprocedure op Klee-Minty kubus** Beschouw het volgende LP-probleem met drie beslissingsvariabelen en met drie restricties:

$$\max D(x_1, x_2, x_3) = 2^2 x_1 + 2x_2 + x_3$$

onder de volgende beperkingen:

$$\begin{cases} x_1 \leq 5 \\ 2^2 x_1 + x_2 \leq 5^2 \\ 2^3 x_1 + 2^2 x_2 + x_3 \leq 5^3, \end{cases}$$

en waarbij verder de niet-negativiteitsvoorwaarden van kracht zijn:

$$x_j \geq 0 \quad \text{voor } j \in \{1, 2, 3\}.$$

Het aanvaardbaar gebied is in dit geval een "vervormde" kubus waarbij één van de 2^3 hoekpunten in de oorsprong ligt.

Gebruik een software-pakket om na te gaan dat de simplexprocedure in dit geval alle 8 de hoekpunten bezoekt om uiteindelijk te eindigen in de optimale oplossing $(0, 0, 5^3)$.

Men kan dit probleem veralgemenen naar een willekeurig aantal dimensies n waarbij het dan 2^n hoekpunten zal bezoeken. Dit probleem werd speciaal ontworpen om aan te tonen dat de simplexprocedure in het slechtste geval een exponentiële tijdscomplexiteit heeft.

HOOFDSTUK 8

Geheeltallig Lineair Programmeren

In dit hoofdstuk bekijken we GEHEELTALLIGE LP-PROBLEMEN. We zien eerst en vooral dat de voor de hand liggende oplossingsmethoden zoals afronden van de oplossingen van de LP-RELAXATIE en enumereren van alle oplossingen voor praktische problemen niet bruikbaar zijn.
De BRANCH-AND-BOUND methode is een zoekproces dat iteratief (gewone) LP-problemen oplost om zo een optimale oplossing te vinden voor een geheeltallig LP-probleem. Dit proces kan visueel worden voorgesteld als een gewortelde boom. Wanneer duidelijk is dat een bepaalde tak de optimale oplossing van het geheeltallig LP-probleem niet kan bevatten dan wordt deze niet verder uitgewerkt. Op die manier hoopt men om relatief snel tot een oplossing te komen, al is dit natuurlijk niet steeds mogelijk.
We eindigen het hoofdstuk met de definitie van het KNAPZAKPROBLEEM, een geheeltallig LP-probleem met een bijzondere vorm. Ook hier kan de branch-and-bound methode worden toegepast, vooral omdat het zeer eenvoudig is om de LP-relaxaties op te lossen m.b.v. een gulzig algoritme.

8.1 Inleiding

Het simplexalgoritme en de II-fasen methode kunnen gebruikt worden om LP-problemen op te lossen waarbij alle beslissingsvariabelen niet-negatieve *reële* waarden aannemen. In bepaalde gevallen is het echter niet zinvol dat

één of meerdere beslissingsvariabelen een reële waarde zouden aannemen. Als men bijvoorbeeld moet gaan beslissen hoeveel printers men gaat aankopen dat is het niet zinvol om anderhalve printer te kopen; of wanneer men gaat beslissen welke voorwerpen men gaat meenemen dan is het (als we er van uitgaan dat de voorwerpen ondeelbaar zijn) ook niet zinvol om slechts een deel van het voorwerp mee te nemen.

Definitie 8.1 Een LP-probleem waarin *minstens één* beslissingsvariabele een gehele waarde moet aannemen wordt een GEHEELTALLIG LP-PROBLEEM genoemd. Binnen de geheeltallige LP-problemen wordt nog een onderscheid gemaakt tussen problemen waarbij *alle* beslissingsvariabelen geheeltallig zijn, de zogenaamde ZUIVERE GEHEELTALLIGE LP-PROBLEMEN en diegene waarbij dit niet het geval is, de GEMENGDE GEHEELTALLIGE LP-PROBLEMEN. Een bijzondere klasse van zuivere geheeltallige LP-problemen zijn diegene waarbij alle beslissingsvariabelen *binair* zijn, en deze worden BINAIRE GEHEELTALLIGE LP-PROBLEMEN genoemd. ∎

Voorbeeld 8.2 (Geheeltallig LP-probleem) Het volgende probleem is een voorbeeld van een geheeltallig LP-probleem:

$$\max D(x_1, x_2, x_3, x_4) = 4x_1 - 2x_2 + 7x_3 - x_4$$

onder de volgende beperkingen:

$$\begin{cases} x_1 + \quad 5x_3 \quad \leq 10 \\ x_1 + x_2 - x_3 \quad \leq 1 \\ 6x_1 - 5x_2 \quad \leq 0 \\ -x_1 \quad + 2x_3 - 2x_4 \leq 3 \end{cases}$$

waarbij de niet-negativiteitsvoorwaarden

$$x_j \geq 0 \quad \text{voor } j \in \{1, 2, 3, 4\}$$

van kracht zijn en bovendien moet gelden dat

$$x_j \text{ is geheel} \quad \text{voor } j \in \{1, 2, 3\}. \tag{8.1}$$

Merk op dat x_4 om het even welke positieve reële waarde kan aannemen maar dat de eerste drie beslissingsvariabelen natuurlijke getallen moeten zijn. Aangezien niet alle beslissingsvariabelen geheeltallig moeten zijn hebben we te maken met een gemengd geheeltallig LP-probleem. ∎

Wanneer we methodes zullen bespreken om geheeltallige LP-problemen op te lossen dan zal het vaak interessant zijn om in eerste instantie de geheeltalligheidseisen van het probleem eenvoudigweg te negeren. We bekomen dan de LP-relaxatie van het probleem.

Definitie 8.3 De LP-RELAXATIE van een geheeltallig LP-probleem is het LP-probleem dat men bekomt door alle geheeltalligheidseisen weg te laten. ∎

Voorbeeld 8.4 De LP-relaxatie van het geheeltallig LP-probleem in Voorbeeld 8.2 is:

$$\max D(x_1, x_2, x_3, x_4) = 4x_1 - 2x_2 + 7x_3 - x_4$$

onder de volgende beperkingen:

$$\begin{cases} x_1 + 5x_3 \leq 10 \\ x_1 + x_2 - x_3 \leq 1 \\ 6x_1 - 5x_2 \leq 0 \\ -x_1 + 2x_3 - 2x_4 \leq 3 \end{cases}$$

waarbij de niet-negativiteitsvoorwaarden

$$x_j \geq 0 \quad \text{voor } j \in \{1, 2, 3, 4\}$$

van kracht zijn.

Merk op dat de eisen in vergelijking (8.1) eenvoudigweg weggelaten werden en dat alle beslissingsvariabelen nu reële waarden kunnen aannemen. ∎

Voorbeeld 8.5 (LP-relaxatie van een binaire variabele) Veronderstel dat x_j een binaire variabele is, i.e. een variabele die slechts de waarden 0 of 1 kan aannemen, of anders gezegd

$$x_j \in \{0, 1\}$$

is een restrictie in het originele probleem. Deze restrictie wordt in de LP-relaxatie vervangen door de restrictie

$$x_j \leq 1$$

en de niet-negativiteitsvoorwaarde

$$x_j \geq 0,$$

zodanig dat x_j een reële waarde tussen 0 en 1 moet aannemen. ∎

8.2 Tweedimensionale geheeltallige LP-problemen

Wanneer men te maken heeft met een tweedimensionaal geheeltallig LP-probleem dan kan men opnieuw de grafische methode gebruiken om dit probleem aan te pakken. Men moet er enkel rekening mee houden dat het aanvaardbaar gebied nu in minstens één en mogelijks twee dimensies beperkt is tot geheeltallige waarden.

Voorbeeld 8.6 (Grafische methode voor een ILP) We beschouwen het volgende zuiver geheeltallig LP-probleem:

$$\max D(x_1, x_2) = 5x_1 + 2x_2$$

onder de beperkingen

$$\begin{cases} -x_1 + 2x_2 \geq 4 \\ \quad x_1 + 3x_2 \leq 14. \end{cases}$$

waarbij

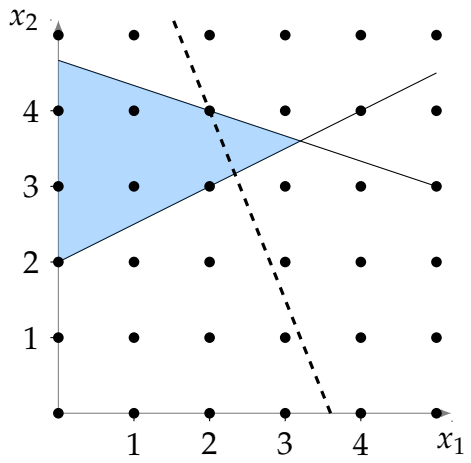
$$x_j \in \mathbb{N} \quad \text{voor } j \in \{1, 2\}.$$

Het aanvaarbaard gebied van de LP-relaxatie bepaalt men als voorheen en wordt getoond in Figuur 8.1. Het aanvaardbaar gebied van het ILP bestaat enkel uit de punten op het geheeltallig grid die binnen het aanvaardbaar gebied van de LP-relaxtie liggen.

Wanneer men nu de doelfunctierechte toevoegt, dan ziet men dat de optimale oplossing van het ILP gegeven wordt door

$$x_1 = 2, \quad x_2 = 4 \quad \text{en} \quad D = 18. \qquad \blacksquare$$

Uit dit voorbeeld kunnen we een zeer belangrijke conclusie trekken, nl. dat *de optimale oplossing van een geheeltallig LP-probleem niet langer gegarandeerd in een hoekpunt van het aanvaardbaar gebied ligt!* Dit betekent onmiddellijk ook dat de oplossingsmethodes die we hebben gezien niet langer rechtstreeks toepasbaar zijn.



Figuur 8.1: De grafische methode toegepast op een zuiver geheeltallig LP-probleem. Het aanvaardbaar gebied bestaat enkel uit die geheeltallige punten die binnen het aangeduide gebied liggen. De optimale doelfunctierechte is getekend. De optimale oplossing ($(2, 4)$ in dit geval) ligt *niet* in een hoekpunt! In dit geval ligt de optimale oplossing wel op een restrictierechte, maar dit is toeval en zeker niet algemeen geldig.

8.3 Oplossen (?!) van zuivere geheeltallige LP-problemen

8.3.1 Afronden oplossing LP-relaxatie

Op het eerste zicht zou men kunnen denken dat voor het oplossen van een geheeltallig LP-probleem het volstaat om zijn LP-relaxatie op te lossen en vervolgens de gevonden oplossing (voor de geheeltallige variabelen) op de één of andere manier af te ronden zodanig dat aan de geheeltalligheidseisen voldaan wordt.

We tonen nu m.b.v. een tweetal voorbeelden aan dat dit niet steeds werkt.

Voorbeeld 8.7 Beschouw het geheeltallig LP-probleem uit Voorbeeld 8.6. Met behulp van de grafische methode, zie ook Figuur 8.1, vindt men gemakkelijk dat de oplossing van de LP-relaxatie gegeven wordt door

$$x_1 = 3.2 \quad x_2 = 3.6 \quad \text{en} \quad D = 23.2$$

Op welke manier we x_1 en x_2 ook afronden we bekomen steeds een punt dat niet tot het aanvaard gebied van het geheeltallig LP-probleem behoort.

Wanneer we bv. x_1 en x_2 allebei naar beneden afronden (afkappen) dan vinden we

$$x_1 = 3 \quad \text{en} \quad x_2 = 3,$$

wat niet aan alle restricties voldoet. Ook afronden naar het dichtsbijzijnde gehele getal werkt niet, dan bekomen we immers

$$x_1 = 3 \quad \text{en} \quad x_2 = 4,$$

wat opnieuw niet aan alle restricties voldoet. Ook de twee resterende manieren van afronden leveren dezelfde problemen. ∎

Voorbeeld 8.8 Beschouw het volgend zuiver geheeltallig LP-probleem:

$$\max D(x_1, x_2) = x_1 + 5x_2,$$

onder de beperkingen

$$\begin{cases} x_1 + 10x_2 \leq 20 \\ x_1 \qquad\quad \leq 2. \end{cases}$$

waarbij

$$x_j \in \mathbb{N} \quad \text{voor } j \in \{1, 2\}.$$

Uit Figuur 8.2 leidt men af dat de oplossing van de LP-relaxatie gegeven wordt door

$$x_1 = 2, \quad x_2 = 1.8 \quad \text{en} \quad D = 11.$$

Als we deze oplossing naar beneden afronden dan krijgen we

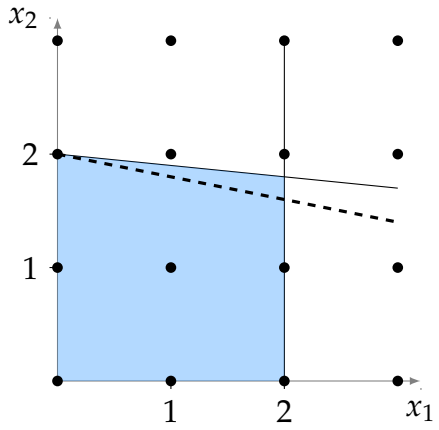
$$x_1 = 2, \quad x_2 = 1 \quad \text{en} \quad D = 7.$$

Dit punt voldoet aan alle restricties maar is geen optimale oplossing van het geheeltallig LP-probleem. Deze wordt immers gegeven door

$$x_1 = 0, \quad x_2 = 2 \quad \text{en} \quad D = 10.$$ ∎

Deze twee voorbeelden tonen aan dat afronden van de optimale oplossing van de LP-relaxatie kan leiden tot

- een “oplossing” die niet aanvaardbaar is;
- of een suboptimale “oplossing”.



Figuur 8.2: Het aanvaardbaar gebied voor de LP-relaxatie van het probleem uit Voorbeeld 8.8 is aangeduid. De optimale doelfunctierechte voor het geheeltallig LP-probleem is getekend. Men ziet dat de optimale oplossing van de LP-relaxatie gegeven wordt door het hoekpunt rechtsboven met coördinaten $(2, 9/5)$.

.

8.3.2 Enumereren van alle aanvaardbare punten

Wanneer we ons beperken tot zuivere geheeltallige LP-problemen dan is het aantal punten dat tot het aanvaardbaar gebied van dit probleem behoort steeds eindig[1]. In principe kunnen we dus (m.b.v. een of andere methode) alle punten van het aanvaardbaar gebied overlopen. Op die manier is het gemakkelijk om na te gaan voor welke punten van het aanvaardbaar gebied de optimale doelfunctiewaarde wordt bereikt.

Het fundamentele probleem hier is dat hoewel het aantal te controleren punten eindig is, het in de praktijk voor realistische problemen *veel te groot* zal zijn. Het aantal punten van het aanvaardbaar gebied stijgt immers exponentieel met het aantal beslissingsvariabelen. Wanneer men zich bv. beperkt tot binaire geheeltallige LP-problemen met n beslissingsvariabelen dan zijn er a priori 2^n combinaties die mogelijks aanvaardbaar zijn. Wanneer bv. $n = 100$ dan is het duidelijk dat deze methode niet praktisch bruikbaar is.

Hoewel deze manier van werken in theorie gegarandeerd een optimale oplossing zal vinden is deze in de praktijk enkel bruikbaar voor “kleine” pro-

[1] Als we ervan uitgaan dat het aanvaardbaar gebied begrensd is.

blemen met een beperkt aantal beslissingsvariabelen.

8.4 De branch-and-bound methode

De twee technieken besproken in de voorgaande sectie zijn in de praktijk onbruikbaar omdat ze enerzijds niet noodzakelijk een optimale oplossing opleveren of anderzijds een veel te grote uitvoeringstijd hebben.

De branch-and-bound methode tracht op een slimme manier "alle" aanvaardbare punten te overlopen, maar het is de bedoeling om (hopelijk) veel aanvaardbare punten *niet expliciet te gaan beschouwen* omdat men reeds weet dat ze toch niet de optimale oplossing kunnen zijn.

De branch-and-bound methode zal herhaaldelijk LP-problemen oplossen en maakt dus gebruik van een (niet nader bepaald) oplossingsalgoritme voor LP-problemen. De reeds besproken oplossingsmethodes kunnen hiervoor gebruikt worden.

Alvorens de branch-and-bound methode te bespreken geven we twee belangrijke eigenschappen van LP-relaxaties.

Eigenschap 8.9 De doelfunctiewaarde van een optimale oplossing van de LP-relaxatie van een geheeltallig LP-probleem is steeds minstens even goed als de doelfunctiewaarde van een optimale oplossing van het geheeltallig LP-probleem. ∎

Bewijs Als een punt $(x_1, x_2, \ldots, x_n)$ een aanvaardbaar punt is voor het geheeltallig LP-probleem dan behoort het onmiddellijk ook tot het aanvaarbaar gebied van de LP-relaxatie. Dit betekent dat de LP-relaxatie steeds minstens evenveel aanvaardbare punten heeft als het ILP-probleem maar dat het aanvaardbaar gebied van de LP-relaxatie eventueel extra punten kan bevatten; deze extra punten kunnen de doelfunctiewaarde alleen maar beter maken. ♢

De volgende belangrijke eigenschap is een onmiddellijk gevolg van deze eerste eigenschap.

Eigenschap 8.10 Wanneer een optimale oplossing van de LP-relaxatie van een geheeltallig LP-probleem aan alle geheeltalligheidseisen voldoet dan is dit meteen ook een optimale oplossing van het geheeltallig LP-probleem. ∎

Opmerking 8.11 Deze tweede eigenschap van LP-relaxaties zegt eigenlijk dat men soms "geluk" kan hebben. Wanneer men een LP-relaxatie oplost en de optimale oplossing voldoet aan de geheeltalligheidseisen dan hebben we onmiddellijk ook het geheeltallig LP-probleem opgelost. ∎

Voor de eenvoud beperken we ons in de rest van deze sectie tot *maximalisatieproblemen*, maar het algoritme kan eenvoudig aangepast worden om ook rechtstreeks te werken met minimalisatieproblemen.

De branch-and-bound methode bouwt conceptueel een zoekboom[2] op. De wortel van de zoekboom stelt de LP-relaxatie van het originele geheeltallige LP-probleem voor. Elke andere top stelt een LP-probleem voor waarbij wordt gezocht naar een oplossing *in een deel* van het aanvaardbaar gebied van de LP-relaxatie. Dit wordt bereikt door iedere keer er een bijkomende top van de boom wordt gegenereerd zijn LP-probleem een extra restrictie te geven t.o.v. het LP-probleem van zijn ouder.

Veronderstel nu eventjes dat men reeds een oplossing van het geheeltallig LP-probleem heeft gevonden met als optimale doelfunctiewaarde D^*. Veronderstel dat in een bepaalde top t de optimale doelfunctiewaarde gegeven wordt door D_t. Dit betekent, wegens Eigenschap 8.9, dat voor elk punt dat aanvaardbaar is voor het geheeltallig LP-probleem en dat ligt in dit bepaald deel van het aanvaardbaar gebied *de doelfunctiewaarde hoogstens gelijk is aan* D_t. Wanneer men tijdens de uitvoering van het algoritme merkt dat $D_t < D^*$, dan weet men dat men dit deel van het aanvaardbaar gebied niet verder hoeft te onderzoeken.

Opmerking 8.12 Het is dus alsof de waarde D_t aankondigt: "hierlangs kan je oplossingen vinden met een doelfunctiewaarde hoogstens gelijk aan D_t". Wanneer je reeds een betere oplossing hebt gevonden dan ga je uiteraard dat deel van het aanvaardbaar gebied niet verder gaan verkennen. ∎

Opmerking 8.13 Wat je moet doen wanneer $D_t = D^*$ hangt af van het feit of je genoegen neemt met één optimale oplossing of niet. Wanneer je alle optimale oplossingen wil vinden dan moet je ook wanneer $D_t = D^*$ deze top nog verder onderzoeken. Wanneer het vinden van één optimale oplossing voldoende is dan is dit niet nodig. ∎

[2]Niet te verwarren met binaire zoekbomen.

Wanneer het LP-probleem geassocieerd met een bepaalde top geen oplossing heeft (i.e. het aanvaardbaar gebied is leeg), dan hoeft men deze top uiteraard ook niet verder te gaan onderzoeken.

Wanneer men merkt dat de oplossing van het LP-probleem geassocieerd met een bepaalde top voldoet aan alle geheeltalligheidseisen, dan hoeft men deze top ook niet meer verder te onderzoeken. Wegens Eigenschap 8.10 heeft men *in dit deel van het aanvaardbaar gebied* de optimale oplossing van het geheeltallig LP-probleem gevonden.

Samenvatting branch-and-bound methode

De branch-and-bound methode houdt een OPEN LIJST bij van LP-problemen die nog verder moeten bekeken worden. **Initieel** bevat deze open lijst enkel de LP-relaxatie van het originele geheeltallig LP-probleem (wanneer deze een oplossing heeft)[3]. Het algoritme houdt eveneens bij wat de beste waarde D^* (en bijhorende oplossing x^*) is van de doelfunctie voor het geheeltallig LP-probleem dat het reeds heeft ontmoet. Initieel is $D^* = -\infty$ om aan te duiden dat er nog geen oplossing van het geheeltallig LP-probleem gevonden is. Het algoritme start dan een **hoofdlus** waarbij in elke iteratie het volgende gebeurt:

Opsplitsen Een LP-probleem van de open lijst wordt gekozen en ervan verwijderd. Wij kiezen steeds het probleem waarvoor de doelfunctiewaarde maximaal is[4]. Bij constructie heeft dit probleem een oplossing met een doelfunctiewaarde die minstens D^* is én bestaat er een eerste variabele x_j met een waarde x_j^* die geheel zou moeten zijn maar het niet is. Twee nieuwe deelproblemen worden aangemaakt, één met de extra restrictie $x_j \leq \lfloor x_j^* \rfloor$ en de andere met de extra restrictie $x_j \geq \lfloor x_j^* \rfloor + 1$.

Begrenzing Bepaal de doelfunctiewaarde voor de twee nieuw aangemaakte LP-problemen, bv. met behulp van het simplexalgoritme.

Controle Elk van de deelproblemen wordt op de open lijst geplaatst tenzij aan minstens één van de volgende drie voorwaarden voldaan is.

a) De doelfunctiewaarde van de gevonden oplossing is kleiner of gelijk aan D^*.

[3] Wanneer de LP-relaxatie van het geheeltallig LP-probleem geen oplossing heeft dan heeft uiteraard ook het geheeltallig LP-probleem geen oplossing.

[4] Andere selectiemechanismes zijn mogelijk.

b) Het LP-probleem heeft geen oplossingen.

c) De oplossing van het LP-probleem voldoet aan alle geheeltalligheidseisen (en is m.a.w. een oplossing van het geheeltallig LP-probleem). Nu zijn er twee mogelijkheden.

   i. De doelfunctiewaarde van het LP-probleem is niet beter dan de huidig beste waarde D^*. In dit geval moet er verder niets worden gedaan.

   ii. De doelfunctiewaarde van het LP-probleem is beter dan de huidige beste. We vervangen D^* en x^* door de nieuwe beste oplossing. Vervolgens verwijderen we alle problemen van de open lijst waarvoor de doelfunctiewaarde kleiner of gelijk is aan de nieuwe waarde van D^*.

De hoofdlus eindigt wanneer de open lijst leeg is. Op dit moment zijn er twee mogelijkheden.

1. Het algoritme heeft een oplossing ontdekt en D^* en x^* hebben een zinvolle waarde. In dit geval zijn dit de retourwaarden van het algoritme.

2. Het algoritme heeft geen oplossing ontdekt. Dit betekent dat het geheeltallig LP-probleem geen oplossing heeft en dit wordt dan ook zo gerapporteerd.

Voorbeeld 8.14 (Uitgewerkt voorbeeld branch-and-bound) We geven een uitgewerkt voorbeeld voor de branch-and-bound methode. We beschouwen het geheeltallig LP-probleem dat ook al aan bod kwam in Voorbeeld 8.2:

$$\max D(x_1, x_2, x_3, x_4) = 4x_1 - 2x_2 + 7x_3 - x_4 \tag{8.2}$$

met de volgende beperkingen:

$$\begin{cases} x_1 + 5x_3 \leq 10 \\ x_1 + x_2 - x_3 \leq 1 \\ 6x_1 - 5x_2 \leq 0 \\ -x_1 + 2x_3 - 2x_4 \leq 3 \end{cases}$$

met

$$\begin{aligned} &x_j \geq 0 \quad \text{voor } j \in \{1,2,3,4\} \\ &x_j \text{ is geheel} \quad \text{voor } j \in \{1,2,3\}. \end{aligned} \tag{8.3}$$

We starten de branch-and-bound methode.

Initialisatie
Probleem ()[5]: de LP-relaxatie van het probleem laat alle geheeltalligheidseisen vallen. M.b.v. de simplexmethode[6] lossen we dit probleem op en we vinden de volgende optimale oplossing:

$$(x_1, x_2, x_3, x_4) = (\frac{5}{4}, \frac{3}{2}, \frac{7}{4}, 0) \quad \text{en } D = \frac{57}{4} = 14.25\,. \tag{8.4}$$

De oplossing van dit probleem voldoet niet aan alle geheeltalligheidseisen en het moet m.a.w. verder uitgewerkt worden. We plaatsen het op de open lijst.

Eerste iteratie
Er staat slechts één probleem op de open lijst en dit wordt dan uiteraard gekozen. De variabele x_1 is de eerste variabele die geheel zou moeten zijn maar het niet is en dus wordt dit onze vertakkingsvariabele en we verkrijgen Probleem ($x_1 \leq 1$) en Probleem ($x_1 \geq 2$). De waarde van x_1 ligt immers tussen 1 en 2 en dus worden de twee deelproblemen als volgt opgesteld.

Probleem ($x_1 \leq 1$): Dit probleem heeft de volgende oplossing:

$$(x_1, x_2, x_3, x_4) = (1, \frac{6}{5}, \frac{9}{5}, 0) \quad \text{en } D = \frac{71}{5} = 14.2\,. \tag{8.5}$$

De oplossing van dit probleem voldoet nog niet aan alle geheeltalligheidseisen en $D > D^* = -\infty$. Dit probleem moet m.a.w. verder uitgewerkt worden en wordt op de open lijst geplaatst.

Probleem ($x_1 \geq 2$): Dit probleem heeft geen aanvaardbare oplossingen, en moet bijgevolg niet verder uitgewerkt worden; het wordt m.a.w. niet op de open lijst geplaatst.

Tweede iteratie
We halen nu Probleem ($x_1 \leq 1$) van de open lijst: de variabele x_2 is niet geheel, en dit wordt onze vertakkingsvariabele. We creëren twee nieuwe deelproblemen, één waaraan $x_2 \leq 1$ als restrictie wordt toegevoegd, en één waaraan $x_2 \geq 2$ als restrictie wordt toegevoegd.

[5]Problemen worden genoteerd m.b.v. de extra restricties die worden toegevoegd aan de originele LP-relaxatie.

[6]Zoals geïmplementeerd in een heel aantal softwarepakketten.

Probleem ($x_1 \leq 1, x_2 \leq 1$): De optimale oplossing van dit probleem is:

$$(x_1, x_2, x_3, x_4) = (\frac{5}{6}, 1, \frac{11}{6}, 0) \quad \text{en } D = \frac{85}{6} \approx 14.167\,. \tag{8.6}$$

De bovengrens voor problemen die vertakken onder dit probleem is 14.167. Aangezien de oplossing van dit probleem niet-geheel is in x_1 en x_3 en $D \geq D^* = -\infty$ zullen we dit probleem verder moeten uitwerken; het wordt toegevoegd aan de open lijst.

Probleem ($x_1 \leq 1, x_2 \geq 2$): De optimale oplossing van dit probleem is:

$$(x_1, x_2, x_3, x_4) = (\frac{5}{6}, 2, \frac{11}{6}, 0) \quad \text{en } D = \frac{73}{6} \approx 12.167\,. \tag{8.7}$$

Voor dit deelprobleem is de bovengrens 12.167. Dit probleem dient ook nog later uitgewerkt te worden en het wordt aan de open lijst toegevoegd.

Derde iteratie
We kiezen ervoor om eerst Probleem ($x_1 \leq 1, x_2 \leq 1$) verder uit te werken aangezien dit de beste bovengrens biedt (14.167 versus 12.167) en we hopen om hier sneller een goede oplossing te bekomen.

In Probleem ($x_1 \leq 1, x_2 \leq 1$) dienen we x_1 *opnieuw* als vertakkingsvariabele te nemen. De twee deelproblemen voegen de beperkingen $x_1 \leq 0$ en $x_1 \geq 1$ toe aan Probleem ($x_1 \leq 1, x_2 \leq 1$).

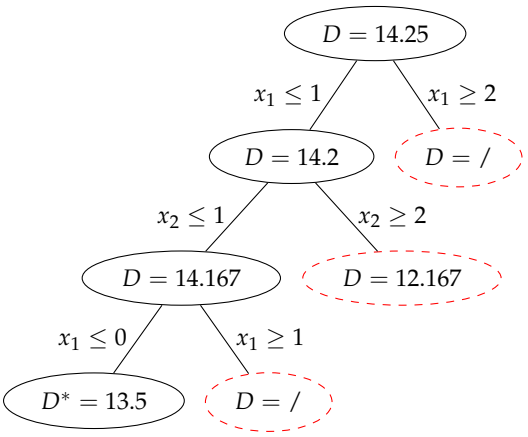
Probleem ($x_1 \leq 1, x_2 \leq 1, x_1 \leq 0$): Merk op dat dit betekent dat $x_1 = 0$. De optimale oplossing van dit deelprobleem is

$$(x_1, x_2, x_3, x_4) = (0, 0, 2, \frac{1}{2}) \quad \text{en } D = \frac{27}{2} = 13.5\,. \tag{8.8}$$

Merk op dat deze oplossing voldoet aan *alle* eisen van het oorspronkelijke probleem. Op dit moment hebben we een eerste aanvaardbare oplossing van het oorspronkelijke probleem gevonden en we stellen $D^* = 13.5$. Het Probleem ($x_1 \leq 1, x_2 \geq 2$) wordt van de open lijst verwijderd omdat de optimale doelfunctiewaarde van dit probleem, nl. $12,167$ kleiner is dan D^*.

Probleem ($x_1 \leq 1, x_2 \leq 1, x_1 \geq 1$): Merk op dat dit betekent dat $x_1 = 1$ (want x_1 was reeds hoogstens 1). In dit geval zijn er geen aanvaardbare oplossingen en het probleem wordt niet aan de open lijst toegevoegd.

Besluit
Er zijn geen verdere deelproblemen uit te werken of anders gezegd de open



Figuur 8.3: Verloop van het branch-and-bound algoritme voor het geheeltallig LP-probleem uit Voorbeeld 8.14. Deze boom wordt niet expliciet bijgehouden door het algoritme. Deze is slechts een visuele voorstelling van het zoekproces.

lijst is leeg. De optimale oplossing van het oorspronkelijke probleem is m.a.w:

$$(x_1, x_2, x_3, x_4) = (0, 0, 2, \frac{1}{2}) \quad \text{en } D = \frac{27}{2} = 13.5\,. \tag{8.9}$$

Het verloop van het branch-and-bound algoritme voor dit voorbeeld kan je eenvoudig volgen in Figuur 8.3 ∎

8.5 Het knapzakprobleem

Het knapzakprobleem is een binair geheeltallig LP-probleem met een bijzondere vorm. We starten met een eenvoudig voorbeeld om deze vorm duidelijk te maken.

Voorbeeld 8.15 Een dief heeft een rugzak met een capaciteit van 50 kg. De volgende drie voorwerpen zijn beschikbaar:

Voorwerp	Gewicht	Waarde
Beeldje	30 kg	4000 EUR
Schilderij 1	25 kg	2500 EUR
Schilderij 2	25 kg	2500 EUR

De dief wenst uiteraard de waarde van de meegenomen voorwerpen te maximaliseren, zonder daarbij de capaciteit van zijn rugzak te overschrijden.

Om een model voor dit probleem op te stellen zien we dat we drie zaken moeten beslissen: voor elk van de drie voorwerpen moeten we namelijk beslissen of we het voorwerp meenemen of niet. Dit zijn drie ja/nee beslissingen, en we stellen bijgevolg

$$x_j = \begin{cases} 1 & \text{als voorwerp } j \text{ gekozen wordt} \\ 0 & \text{als voorwerp } j \text{ niet gekozen wordt.} \end{cases}$$

Met behulp van deze beslissingsvariabelen kan de waarde van de rugzak als volgt worden uitgedrukt:

$$4000x_1 + 2500x_2 + 2500x_3.$$

Inderdaad, wanneer het beeldje gekozen wordt dan levert dit een bijdrage van 4000 EUR aan de waarde van de rugzak, en wanneer het niet gekozen wordt dan is er natuurlijk geen bijdrage. Dit is precies wat de term $4000x_1$, samen met de betekenis van de beslissingsvariabele x_1, uitdrukt! De andere termen komen op een gelijkaardige manier tot stand.

Het gewicht van de rugzak wordt voor een bepaalde keuze van voorwerpen uitgedrukt als:

$$30x_1 + 25x_2 + 25x_3.$$

Het optimalisatieprobleem waar de dief voor staat wordt bijgevolg gegeven door

$$\max D(x_1, x_2, x_3) = 4000x_1 + 2500x_2 + 2500x_3$$

onder de beperkingen

$$30x_1 + 25x_2 + 25x_3 \leq 50$$

en

$$x_j \in \{0, 1\} \quad \text{voor } j \in \{1, 2, 3\}.$$ ∎

Uit dit voorbeeld leiden we nu eenvoudig de algemene vorm van een knapzakprobleem af. Veronderstel stel dat er n ondeelbare voorwerpen zijn. Elk voorwerp heeft een bepaald gewicht g_j en een bepaalde waarde w_j. Het

maximale gezamenlijke gewicht van de gekozen voorwerpen is G. De vraag die moet beantwoord worden is: welke voorwerpen moeten gekozen worden opdat de waarde van de gekozen voorwerpen maximaal is wanneer het maximale gewicht G niet mag overschreden worden.

Definitie 8.16 Gegeven n positieve reële gewichten g_j ($j \in \{1, 2 \ldots, n\}$) en n positieve reële waarden w_j ($j \in \{1, 2 \ldots, n\}$) en een maximaal positief reëel gewicht G. Een KNAPZAKPROBLEEM heeft de volgende vorm:

$$\max D(x_1, x_2, \ldots, x_n) = w_1 x_1 + w_2 x_2 + \cdots + w_n x_n$$

onder de beperkingen

$$g_1 x_1 + g_2 x_2 + \cdots + g_n x_n \leq G$$

en

$$x_j \in \{0, 1\} \quad \text{voor } j \in \{1, 2, \ldots, n\}.$$ ∎

De wiskundige vorm van dit probleem is zeer eenvoudig en men zou misschien denken dat het eenvoudig is om dit probleem op te lossen. Immers, welke voorwerpen zijn het interessants om mee te nemen? Dat zijn de voorwerpen die niet veel wegen en toch veel waard zijn, i.e. de voorwerpen die *per eenheid gewicht* het meeste waard zijn. Een gulzig algoritme neemt dan eerst het voorwerp met de grootste waarde per gewicht ratio (wanneer dit in de rugzak past), neemt vervolgens het voorwerp met de tweede grootste waarde per gewicht ratio, enzoverder tot alle voorwerpen bekeken zijn.

Voorbeeld 8.17 (Gulzig algoritme voor het knapzakprobleem) We passen het gulzig algoritme toe op het knapzakprobleem in Voorbeeld 8.15.

We bekijken de waarde per gewicht ratio van de drie voorwerpen.

Voorwerp	Gewicht	Waarde	Waarde/Gewicht
Beeldje	30 kg	4000 EUR	$4000/30 \approx 133.33$
Schilderij 1	25 kg	2500 EUR	$2500/25 = 100$
Schilderij 2	25 kg	2500 EUR	$2500/25 = 100$

Volgens het gulzig algoritme moeten we eerst het beeldje nemen (want het past in de rugzak). De rugzak heeft nu nog een capaciteit van $50 - 30 = 20$ kg. Geen van de twee schilderijen past nog in de rugzak.

De rugzak samengesteld door het gulzig algoritme heeft een totale waarde van 4000 EUR. Echter, de twee schilderijen passen samen ook in de rugzak en hebben een gecombineerde waarde van 5000 EUR. ∎

Uit dit voorbeeld blijkt duidelijk dat het gulzig algoritme *niet noodzakelijk de optimale oplossing* van het knapzakprobleem vindt. Daarom noemen we het algoritme dan ook het BENADEREND GULZIG ALGORITME.

Wannneer we de branch-and-bound methode willen toepassen om het knapzakprobleem op te lossen, dan zullen we herhaaldelijk LP-relaxaties van knapzakproblemen moeten oplossen. Het blijkt nu dat er een eenvoudig gulzig algoritme bestaat om zo'n LP-relaxatie op te lossen. Men sorteert opnieuw de voorwerpen volgens dalende waarde per gewicht ratio. Men overloopt de voorwerpen in deze volgorde. Wanneer de rugzak nog voldoende capaciteit heeft dan wordt een voorwerp volledig meegenomen (en wordt de capaciteit van de rugzak overeenkomstig verminderd), in het andere geval wordt de maximaal mogelijke fractie van het voorwerp meegenomen. Stel dat de resterende capaciteit van de rugzak gegeven wordt door g en dat we aanbeland zijn bij voorwerp j met gewicht g_j, dan wordt de fractie g/g_j van het voorwerp meegenomen.

De oplossingen gegeneerd door het gulzig algoritme nemen dus steeds een aantal voorwerpen volledig mee, terwijl er hoogstens één voorwerp voor een fractie verschillend van 1 wordt meegenomen.

Voorbeeld 8.18 De oplossing gevonden door het gulzig algoritme voor het knapzakprobleem in Voorbeeld 8.15 is

$$x_1 = 1, x_2 = 20/25, x_3 = 0, D = 6000.$$

Men verifieert (bv. m.b.v. Excel of de simplexmethode) dat dit inderdaad de optimale oplossing is van dit fractioneel knapzakprobleem. Dit is altijd het geval zoals blijkt uit de volgende eigenschap. ∎

Eigenschap 8.19 Het zonet besproken gulzig algoritme levert steeds de oplossing voor de *LP-relaxatie* van een knapzakprobleem, i.e. voor een fractioneel knapzakprobleem waarbij willekeurige fracties van de items kunnen meegenomen worden. ∎

Om het knapzakprobleem op te lossen kunnen we gebruikmaken van het branch-and-bound algoritme met de volgende aanpassingen.

1. In plaats van D^* te initialiseren op $-\infty$ kunnen we m.b.v. het *benaderend* gulzig algoritme reeds een eerste oplossing vinden. Dit zal toelaten om hopelijk meer deelproblemen vroegtijdig af te sluiten (op basis van hun bovengrens).
2. Een extra restrictie van de vorm $x_j \leq 0$ vereenvoudigt tot $x_j = 0$ en correspondeert met een knapzakprobleem waarbij het j-de voorwerp niet mag gekozen worden.
3. Een extra restrictie van de vorm $x_j \geq 1$ vereenvoudigt tot $x_j = 1$ en correspondeert met een knapzakprobleem waarbij het j-de voorwerp steeds moet gekozen worden (en men de capaciteit van de knapzak dus vermindert met g_j).

Voorbeeld 8.20 We lossen het volgende knapzakprobleem op m.b.v. de branch-and-bound methode:

$$\max D(x_1, x_2, x_3, x_4) = 12x_1 + 11x_2 + 8x_3 + 6x_4 \tag{8.10}$$

met

$$8x_1 + 8x_2 + 6x_3 + 5x_4 \leq 20 \tag{8.11}$$

en

$$x_j \in \{0, 1\} \quad \text{voor } j \in \{1, 2, 3, 4\}. \tag{8.12}$$

Ordenen we de vier voorwerpen v_i volgens dalende orde van waarde per gewicht verhouding dan vinden we: v_1, v_2, v_3 en v_4. Dit is nu *toevallig* gelijk aan de oorspronkelijke volgorde maar dit hoeft niet zo te zijn.

Initialisatie
Het benaderend gulzig algoritme vindt de volgende oplossing:

$$(x_1, x_2, x_3, x_4) = (1, 1, 0, 0) \quad \text{en} \quad D = 12 + 11 = 23. \tag{8.13}$$

Dit is de "voorlopig beste" oplossing en we noteren dit als $D^* = 23$.

Probleem (): Het oorspronkelijke probleem waarbij de restricties (8.12) vervangen werden door de restricties:

$$0 \leq x_j \leq 1 \quad \text{voor } j \in \{1, 2, 3, 4\}. \tag{8.14}$$

Het gulzige algoritme[7] voor zo'n fractioneel knapzakprobleem (of de simplex-methode) vindt voor dit probleem de oplossing:

$$(x_1, x_2, x_3, x_4) = (1, 1, \frac{2}{3}, 0) \quad \text{en} \quad D = 12 + 11 + \frac{2}{3}8 = \frac{85}{3} \approx 28.33\,. \tag{8.15}$$

[7] Dit algoritme is eenvoudig "met de hand" uit te voeren.

Aangezien de optimale oplossing van het probleem niet-geheel is en bovendien de D-waarde groter is dan D^*, moeten we dit probleem verder gaan uitwerken en we plaatsen het op de open lijst.

Iteratie 1
We verwijderen het enige probleem van de open lijst en zien dat x_3 niet geheeltallig is. We krijgen dus twee deelproblemen die we als Probleem ($x_3 = 0$) en Probleem ($x_3 = 1$) benoemen.

Probleem ($x_3 = 0$): Probleem () met de restricties (8.14) vervangen door

$$0 \leq x_j \leq 1 \quad \text{voor } j \in \{1, 2, 4\} \quad \text{en} \quad x_3 = 0. \tag{8.16}$$

Dit probleem heeft als optimale oplossing (gemakkelijk gevonden met het gulzig algoritme)

$$(x_1, x_2, x_3, x_4) = (1, 1, 0, \frac{4}{5}) \quad \text{en} \quad D = 12 + 11 + \frac{4}{5}6 = \frac{139}{5} = 27.8\,. \tag{8.17}$$

Aangezien dit probleem een niet-gehele optimale oplossing heeft en de bovengrens, 27, strikt groter is dan de waarde van de huidig beste oplossing, 23, zullen we dit probleem later nog moeten uitwerken en we plaatsen het op de open lijst. Merk op dat de bovengrens voor een oplossing die aan alle geheeltalligsheidsvoorwaarden voldoet en die kan verkregen worden door de restrictie $x_3 = 0$ toe te voegen inderdaad 27 is (en niét 27.8) aangezien de waarde van de knapzak steeds geheel zal zijn.

Probleem ($x_3 = 1$): Probleem () met de restricties (8.14) vervangen door

$$0 \leq x_j \leq 1 \quad \text{voor } j \in \{1, 2, 4\} \quad \text{en} \quad x_3 = 1. \tag{8.18}$$

Dit probleem heeft als optimale oplossing

$$(x_1, x_2, x_3, x_4) = (1, \frac{3}{4}, 1, 0) \quad \text{en} \quad D = 12 + \frac{3}{4}11 + 8 = \frac{113}{4} = 28.25\,. \tag{8.19}$$

Aangezien dit probleem een niet-gehele optimale oplossing heeft en de bovengrens, 28, strikt groter is dan de huidig beste oplossing, 23, plaatsen we het op de open lijst.

Iteratie 2
Aangezien Probleem ($x_3 = 1$) een betere bovengrens heeft dan Probleem ($x_3 = 0$) kiezen we ervoor om dit probleem eerst uit te werken. We krijgen

dan opnieuw twee deelproblemen: Probleem ($x_3 = 1, x_2 = 0$) en Probleem ($x_3 = 1, x_2 = 1$).

Probleem ($\mathbf{x_3 = 1, x_2 = 0}$): Probleem ($x_3 = 1$) met de restricties (8.18) vervangen door

$$0 \leq x_j \leq 1 \quad \text{voor } j \in \{1, 4\} \quad \text{en} \quad x_2 = 0,\ x_3 = 1. \tag{8.20}$$

Dit probleem heeft als optimale oplossing

$$(x_1, x_2, x_3, x_4) = (1, 0, 1, 1) \quad \text{en} \quad Z = 12 + 8 + 6 = 26. \tag{8.21}$$

Aangezien deze oplossing voldoet aan de restricties van het oorspronkelijke probleem moeten we dit deelprobleem niet verder gaan uitwerken. Bovendien is de D-waarde voor dit probleem beter dan de huidige beste D-waarde: $26 > 23$. Op dit moment is er dus een nieuwe waarde voor D^*: $D^* = 26$. Alle problemen op de open lijst hebben een bovengrens die groter is dan 26; we kunnen m.a.w. geen enkel probleem verwijderen van de open lijst.

Probleem ($\mathbf{x_3 = 1, x_2 = 1}$): Probleem ($x_3 = 1$) met de restricties (8.18) vervangen door

$$0 \leq x_i \leq 1 \quad \text{voor } i \in \{1, 4\} \quad \text{en} \quad x_2 = 1,\ x_3 = 1. \tag{8.22}$$

Dit probleem heeft als optimale oplossing

$$(x_1, x_2, x_3, x_4) = (\frac{3}{4}, 1, 1, 0) \quad \text{en} \quad D = \frac{3}{4}12 + 11 + 8 = 28. \tag{8.23}$$

Aangezien dit probleem een niet-gehele optimale oplossing heeft en de bovengrens, 28, strikt groter is dan de huidig beste oplossing, 26, plaatsen we het op de open lijst.

Iteratie 3

Op dit moment zijn er nog twee openstaande deelproblemen: Probleem ($x_3 = 0$) en Probleem ($x_3 = 1, x_2 = 1$). In dit voorbeeld kiezen we ervoor om het probleem met de beste bovengrens eerst uit te werken. In dit geval is dit dus Probleem ($x_3 = 1, x_2 = 1$). De vertakkingsvariabele is x_1 en we krijgen twee deelproblemen: Probleem ($x_3 = 1, x_2 = 1, x_1 = 0$) en Probleem ($x_3 = 1, x_2 = 1, x_1 = 1$).

Probleem ($\mathbf{x_3 = 1, x_2 = 1, x_1 = 0}$): : Probleem ($x_3 = 1, x_2 = 1$) met de restricties (8.22) vervangen door

$$0 \leq x_i \leq 1 \quad \text{voor } i \in \{4\} \quad \text{en} \quad x_1 = 0,\ x_2 = 1,\ x_3 = 1. \tag{8.24}$$

Dit probleem heeft als optimale oplossing

$$(x_1, x_2, x_3, x_4) = (0, 1, 1, 1) \quad \text{en} \quad D = 11 + 8 + 6 = 25. \tag{8.25}$$

Dit is een oplossing die voldoet aan de restricties van het originele probleem, maar ze is *niet beter* dan de huidig beste oplossing. Dit deelprobleem moeten we dus niet verder uitwerken en het wordt bijgevolg niet toegevoegd aan de open lijst.

Probleem ($\mathbf{x_3 = 1, x_2 = 1, x_1 = 1}$): : Probleem ($x_3 = 1, x_2 = 1$) met de restricties (8.22) vervangen door

$$0 \leq x_i \leq 1 \quad \text{voor } i \in \{4\} \quad \text{en} \quad x_1 = 1,\ x_2 = 1,\ x_3 = 1. \tag{8.26}$$

Dit probleem heeft geen aanvaardbare oplossingen, en moet dus ook niet verder uitgewerkt worden.

Iteratie 4
Probleem ($x_3 = 0$) staat nog op de open lijst aangezien de bovengrens voor dit probleem, 27, strikt groter is dan de waarde van de huidig beste oplossing, namelijk 26. We werken dit probleem verder uit door x_4 als vertakkingsvariabele te gebruiken en krijgen twee deelproblemen: Probleem ($x_3 = 0, x_4 = 0$) en Probleem ($x_3 = 0, x_4 = 1$).

Probleem ($\mathbf{x_3 = 0, x_4 = 0}$): Probleem ($x_3 = 0$) maar met de restricties (8.16) vervangen door:

$$0 \leq x_i \leq 1 \quad \text{voor } i \in \{1, 2\} \quad \text{en} \quad x_3 = 0,\ x_4 = 0. \tag{8.27}$$

De optimale oplossing voor dit probleem is:

$$(x_1, x_2, x_3, x_4) = (1, 1, 0, 0) \quad \text{en} \quad D = 12 + 11 = 23. \tag{8.28}$$

Aangezien deze oplossing voldoet aan de restricties van het oorspronkelijke probleem dienen we dit probleem niet verder uit te werken. De D-waarde voor deze oplossing is echter niet beter dan de beste D-waarde tot nu toe, dus we kijgen geen nieuwe beste oplossing en er moet verder geen actie ondernomen worden.

Probleem ($\mathbf{x_3 = 0, x_4 = 1}$): Probleem ($x_3 = 0$) maar met de restricties (8.16) vervangen door:

$$0 \leq x_i \leq 1 \quad \text{voor } i \in \{1, 2\} \quad \text{en} \quad x_3 = 0,\ x_4 = 1. \tag{8.29}$$

De optimale oplossing voor dit probleem is:

$$(x_1, x_2, x_3, x_4) = (1, \frac{7}{8}, 0, 1) \quad \text{en} \quad D = 12 + \frac{7}{8}11 + 6 = \frac{221}{8} = 27.625\,. \tag{8.30}$$

We hebben hier te maken met een niet-gehele oplossing wiens bovengrens, 27, beter is dan de huidig beste oplossing $D^* = 26$. Dit probleem moet dus nog verder uitgewerkt worden.

Iteratie 5
De vertakkingsvariabele voor het enige probleem op de open lijst, nl. Probleem $(x_3 = 0, x_4 = 1)$, is x_2 en we krijgen twee deelproblemen: Probleem $(x_3 = 0, x_4 = 1, x_2 = 0)$ en Probleem $(x_3 = 0, x_4 = 1, x_2 = 1)$.

Probleem $(\mathbf{x_3 = 0, x_4 = 1, x_2 = 0})$: Probleem $(x_3 = 0, x_4 = 1)$ maar met de restricties (8.29) vervangen door:

$$0 \leq x_i \leq 1 \quad \text{voor } i \in \{1\} \quad \text{en} \quad x_2 = 0,\ x_3 = 0,\ x_4 = 1. \tag{8.31}$$

De optimale oplossing voor dit probleem is:

$$(x_1, x_2, x_3, x_4) = (1, 0, 0, 1) \quad \text{en} \quad D = 12 + 6 = 18. \tag{8.32}$$

Dit probleem dient niet verder uitgewerkt te worden want het is een geheeltallige oplossing die bovendien niet beter is dan de huidig beste oplossing.

Probleem $(\mathbf{x_3 = 0, x_4 = 1, x_2 = 1})$: Probleem $(x_3 = 0, x_4 = 1)$ maar met de restricties (8.29) vervangen door:

$$0 \leq x_i \leq 1 \quad \text{voor } i \in \{1\} \quad \text{en} \quad x_2 = 1,\ x_3 = 0,\ x_4 = 1. \tag{8.33}$$

De optimale oplossing voor dit probleem is:

$$(x_1, x_2, x_3, x_4) = (\frac{7}{8}, 1, 0, 1) \quad \text{en} \quad D = \frac{7}{8}12 + 11 + 6 = \frac{55}{2} = 27.5\,. \tag{8.34}$$

Dit probleem heeft een niet-gehele oplossing met een bovengrens, 27, die nog steeds strikt beter is dan de huidig beste waarde van D^*. We plaatsen het op de open lijst.

Iteratie 6
We werken het enige probleem op de open lijst verder uit door x_1 te gebruiken als vertakkingsvariabele. We krijgen twee deelproblemen.

Probleem ($\mathbf{x_3 = 0, x_4 = 1, x_2 = 0, x_1 = 0}$): Probleem ($x_3 = 0, x_4 = 1, x_2 = 0$) maar met de restricties (8.33) vervangen door:

$$x_1 = 0,\ x_2 = 1,\ x_3 = 0,\ x_4 = 1. \tag{8.35}$$

De optimale oplossing voor dit probleem is:

$$(x_1, x_2, x_3, x_4) = (0, 1, 0, 1) \quad \text{en} \quad D = 11 + 6 = 17. \tag{8.36}$$

Dit probleem dient niet verder uitgewerkt te worden.

Probleem ($\mathbf{x_3 = 0, x_4 = 1, x_2 = 0, x_1 = 1}$): Probleem ($x_3 = 0, x_4 = 1, x_2 = 0$) maar met de restricties (8.33) vervangen door:

$$x_1 = 1,\ x_2 = 1,\ x_3 = 0,\ x_4 = 1. \tag{8.37}$$

Dit probleem heeft geen oplossing en moet bijgevolg ook niet verder uitgewerkt te worden.

Besluit:
Op dit moment zijn er geen deelproblemen meer die nog verder uitgewerkt dienen te worden; de open lijst is leeg. Het algoritme stopt en de optimale oplossing

$$(x_1, x_2, x_3, x_4) = (1, 0, 1, 1) \quad \text{met } D = 12 + 8 + 6 = 26 \tag{8.38}$$

werd gevonden.

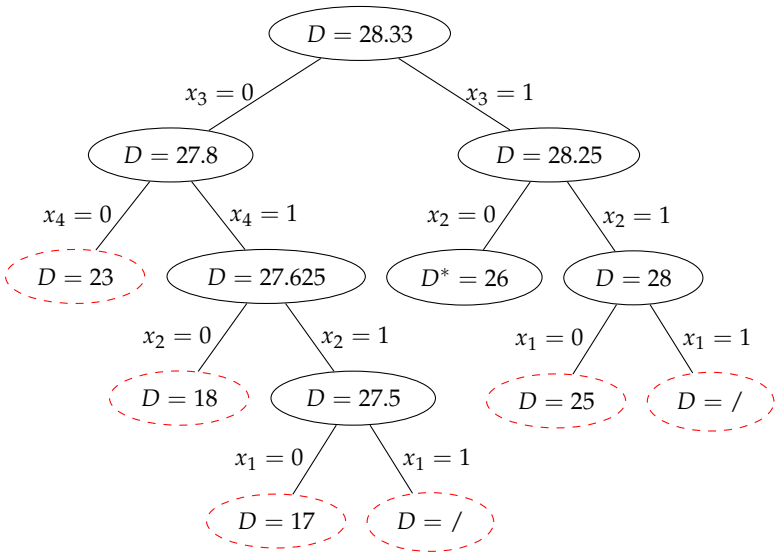
De eenvoudigste manier om dit proces (visueel) bij te houden is aan de hand van een oplossingsboom zoals getoond in Figuur 8.4. ∎

Opmerking 8.21 We merken op de beslissingsversie van het knapzakprobleem NP-compleet is. Op dit moment is er geen gekend algoritme om alle instanties van een NP-compleet efficiënt, i.e. in polynomiale tijd op te lossen. Ook het branch-and-bound algoritme biedt geen garantie dat de oplossing "snel" zal gevonden worden. Vaak is dit wel het geval maar zeker niet altijd. ∎

8.6 Oefeningen

1. Los het volgende gemengde ILP-probleem op m.b.v. de grafische methode:

$$\max D(x_1, x_2) = -3x_1 + 2x_2$$



Figuur 8.4: De oplossingsboom zoals die wordt opgebouwd door het branch-and-bound algoritme voor het oplossen van het knapzakprobleem in Voorbeeld 8.20.

onder de beperkingen

$$\begin{cases} 4x_1 - 2x_2 \geq 3 \\ 2x_1 + 2x_2 \leq 19 \\ 4x_1 + 2x_2 \leq 33 \end{cases}$$

waarbij $x_1 \in \mathbb{N}$ en waar bovendien de niet-negativiteitsvoorwaarden van kracht zijn:

$$x_1 \geq 0 \quad \text{en} \quad x_2 \geq 0.$$

2. Los volgend binair geheeltallig LP-probleem m.b.v. de branch-and-bound methode. Om de LP-relaxaties op te lossen kan je gebruikmaken van een softwarepakket. Houd het verloop van het proces bij a.d.h.v. een oplossingsboom.

$$\max D(x_1, x_2, x_3, x_4) = 9x_1 + 5x_2 + 6x_3 + 4x_4$$

onder de beperkingen

$$\begin{cases} 6x_1 + 3x_2 + 5x_3 + 2x_4 \leq 10 \\ x_3 + x_4 \leq 1 \\ -x_1 + x_3 \leq 0 \\ -x_2 + x_4 \leq 0. \end{cases}$$

en

$$x_j \in \{0,1\} \quad \text{voor } j \in \{1,2,3,4\}.$$

3. Een coach van een zwemteam wil een team samenstellen voor de 4×50 meter wisselslag. Hij beschikt over 5 zwemmers, die elk verschillende tijden neerzetten voor de verschillende zwemstijlen. De coach wil uiteraard het snelste team samenstellen.

   De tijden (in seconden) van de zwemmers worden in onderstaande tabel gegeven:

	Lucas	Liam	Vince	Finn	Louis
Rugslag	37,7	32,9	33,8	37,0	35,4
Schoolslag	43,4	33,1	42,2	34,7	41,8
Vlinderslag	33,3	28,5	38,9	30,4	33,6
Vrije slag	29,2	26,4	29,6	28,5	31,1

   a) Gebruik de binaire beslissingsvariabelen $x_{i,j}$, waarbij $x_{i,j} = 1$ betekent dat "slag" i gezwommen wordt door "zwemmer" j. Bijvoorbeeld $x_{1,2} = 1$ betekent dat Liam rugslag voor zijn rekening neemt.

   b) Hoe druk je uit dat de tijd van het beste team minimaal is? Gebruik $t_{i,j}$ om aan te geven wat de tijd is zwemmer j voor de slag i.

   c) Druk uit dat elke slag door juist één zwemmer wordt gezwommen.

   d) Druk uit dat elke zwemmer hoogstens één slag mag zwemmen. Merk op dat één zwemmer buiten de ploeg zal vallen.

   Los het probleem op m.b.v. Excel-solver en formuleer je advies aan de coach.

4. **Klassiek arbeidsplanningprobleem** Een restaurant is zeven dagen per week open. Men wil op elk van de zeven dagen minimaal het volgende aantal personeelsleden aan het werk hebben in het restaurant:

Dag	Ma	Di	Woe	Do	Vr	Zat	Zo
Aantal	14	13	15	16	19	18	11

Het werkregime is zodanig dat een arbeider steeds 5 dagen werkt en daarna (verplicht) twee dagen thuis is. Dit patroon zet zich (oneindig) door.

Vind het minimaal aantal personeelsleden dat nodig is om aan de bestaffingsvereisten van het restaurant te voldoen. Stel hiertoe eerst het wiskundig model op en los dit vervolgens op m.b.v. een softwarepakket.

Tip: denk goed na over de *betekenis* van je beslissingsvariabelen. Wanneer je de (voor de hand liggende) keuze zou gebruiken die zegt dat x_i het aantal personeelsleden is dat werkt op dag i, dan wordt het zeer moeilijk om het model op te stellen.

5. Stel dat in een bepaalde regio met 5 steden brandweercentrales gevestigd moeten worden. Elke stad is een potentiële vestigingsplaats voor een brandweercentrale. Het is vereist dat elke stad op hoogstens 10 minuten reistijd ligt van een brandweercentrale. In de tabel hieronder worden de reistijden in minuten tussen elk paar steden gegeven.

Van \Naar	Stad 1	Stad 2	Stad 3	Stad 4	Stad 5
Stad 1	0	11	12	9	15
Stad 2	11	0	14	12	10
Stad 3	9	8	0	15	12
Stad 4	10	12	15	0	13
Stad 5	17	12	10	13	0

Merk op dat de reistijden niet symmetrisch zijn. Het duurt bv. 12 minuten om van stad 1 naar stad 3 te reizen, maar omgekeerd duurt het slechts 9 minuten.

Men wil zo weinig mogelijk brandweercentrales plaatsen zodanig dat aan de reistijdvoorwaarden wordt voldaan. Stel het wiskundig model op voor dit probleem (en los het op), i.e. hoeveel brandweerkazernes zijn nodig en waar moeten ze worden geplaatst?

BIJLAGE **A**

Complexiteitstheorie

In deze appendix geven we een korte en informele inleiding tot de complexiteitstheorie. Dat is de theorie die problemen classificeert volgens hun moeilijkheidsgraad. Deze eerste klasse van problemen die besproken wordt is de klasse **P**, de klasse van de "eenvoudige" problemen. We zien hierbij ook dat niet alle problemen tot de klasse **P** behoren, méér zelfs, we zien dat er problemen bestaan, zoals Turing's STOPPROBLEEM, die ONBESLISBAAR zijn. REDUCTIES geven een manier om aan te tonen dat een bepaald probleem minstens zo moeilijk is als een ander probleem. Dit leidt dan onmiddellijk tot het begrip COMPLEETHEID voor een bepaalde klasse. De klasse **NP** wordt geïntroduceerd als de klasse waarvan oplossingen eenvoudig te verifiëren zijn. De klasse **NP** is zeer groot en bevat ook een heel aantal **NP**-complete problemen. Het is een open vraag of de klasse **NP** strikt groter is dan de klasse **P**, en dit is één van de belangrijkste open vragen binnen de informatica!

A.1 De complexiteitsklasse P

In de complexiteitstheorie is men voornamelijk geïnteresseerd om "gemakkelijke" van "moeilijke" problemen te onderscheiden en dit op basis van hun tijdscomplexiteit.

We starten met de definitie van de klasse **P** wat de klasse van de "gemakkelijke" problemen is.

Definitie A.1 De KLASSE **P** is de verzameling van alle problemen die in polynomiale tijd kunnen opgelost worden door een (deterministisch) algo-

ritme. ∎

Opmerking A.2 Met polynomiale tijd bedoelen we dat de uitvoeringstijd van het algoritme $O(n^k)$ is waarbij n de grootte van de invoer voorstelt en k een constante is (onafhankelijk van n). ∎

De meeste van de problemen die we in deze cursus hebben gezien behoren tot de klasse **P**, zoals bv. het vinden van een kortste pad tussen twee knopen in een gewogen graaf met positieve gewichten (Dijkstra), het vinden van een minimale kost opspannende boom (algoritmes van Prim en Kruskal).

Men kan zich nu afvragen of *alle* problemen tot de klasse **P** behoren. Het negatief antwoord op deze vraag werd reeds in 1936 gegeven door Turing (Turing, 1936). Turing's STOPPROBLEEM (halting problem) is het volgende: "schrijf een algoritme A (programma) dat bepaalt of een willekeurig programma P met als input I stopt of niet". Het blijkt dat het stopprobleem een ONBESLISBAAR probleem is: het algoritme A bestaat niet!

Eigenschap A.3 Turing's stopprobleem is onbeslisbaar. ∎

Bewijs We schetsen het bewijs. We moeten aantonen dat er geen algoritme A bestaat dat voor alle programma's P en elke mogelijke invoer I kan bepalen of het programma P stopt voor de invoer I of niet.

We geven een bewijs uit het ongerijmde en veronderstellen m.a.w. dat het programma A wel bestaat:

$$A(P, I) = \begin{cases} 1 & \text{als programma } P \text{ stopt voor invoer } I \\ 0 & \text{als programma } P \text{ niet stopt voor invoer } I. \end{cases}$$

Met dit programma A kunnen we een programma Q construeren met als invoer een willekeurig programma P. Dit programma stopt als P niet stopt wanneer het wordt uitgevoerd met als invoer zichzelf en omgekeerd:

$$Q(P) = \begin{cases} \text{stopt} & \text{als } A(P, P) = 0 \\ \text{stopt niet} & \text{als } A(P, P) = 1. \end{cases}$$

Het programma Q kan gemakkelijk geschreven worden door programma A aan te roepen.

We voeren nu het programma Q uit met zichzelf als invoer en we krijgen:

$$Q(Q) = \begin{cases} \text{stopt} & \text{als } A(Q,Q) = 0, \\ & \text{i.e. programma } Q \text{ stopt niet met zichzelf als invoer} \\ \text{stopt niet} & \text{als } A(Q,Q) = 1, \\ & \text{i.e. programma } Q \text{ stopt wel met zichzelf als invoer.} \end{cases}$$

Beide gevallen leiden tot een contradictie die een logisch gevolg is van het bestaan van het programma A. Dit toont aan dat het programma A niet kan bestaan en dat het stopprobleem inderdaad onbeslisbaar is. ♢

Turing's stopprobleem toont aan dat er wel degelijk grenzen zijn aan wat er kan *berekend* worden m.b.v. een computer. Dit is een interessante observatie voor een informaticus.

A.2 Reducties

Reducties zijn een fundamenteel begrip in de informatica en worden in de praktijk ook vaak gebruikt.

Voorbeeld A.4 (Voorbeeld reductie) Om de mediaan van een rij getallen te vinden kunnen we eenvoudigweg deze rij sorteren en dan het middelste getal of het gemiddelde van de twee middelste getallen teruggeven. Het vinden van de mediaan *reduceert* tot sorteren, of anders gezegd sorteren kan gebruikt worden om het probleem van het vinden van de mediaan op te lossen[1]. ∎

Voorbeeld A.5 (Voorbeeld reductie) Als we bv. de kortste afstanden willen vinden tussen *alle* paren van knopen in een graaf met positieve gewichten dan kunnen we het algoritme van Dijkstra n keer aanroepen, één keer voor elk van de n startknopen. Het probleem van het vinden van de afstanden tussen *alle* paren van knopen is m.a.w. *gereduceerd* tot het n keer aanroepen van het algoritme van Dijkstra. Het algoritme van Dijkstra kan gebruikt worden om het oorspronkelijke probleem op te lossen. ∎

We maken dit nu iets formeler met de volgende definitie.

[1] Er zijn snellere manieren om de mediaan te vinden.

Definitie A.6 Een probleem π_1 reduceert tot een probleem π_2 wanneer een polynomiaal algoritme voor π_2 kan gebruikt worden om probleem π_1 op te lossen in polynomiale tijd. ∎

Als probleem π_1 reduceert tot π_2 dan betekent dit dat π_1 "gemakkelijk" is wanneer π_2 "gemakkelijk" is:

$$\pi_2 \text{ gemakkelijk} \implies \pi_1 \text{ gemakkelijk}.$$

Als we hiervan de contrapositie nemen dan betekent dit

$$\pi_1 \text{ moeilijk} \implies \pi_2 \text{ moeilijk}$$

of

$$\pi_1 \notin \mathbf{P} \implies \pi_2 \notin \mathbf{P}.$$

M.a.w. **als** π_1 **reduceert tot** π_2 **dan is** π_2 **minstens zo moeilijk als** π_1, want je kan π_2 niet alleen gebruiken om π_1 op te lossen maar eventueel ook om nog andere zaken te doen.

A.3 Compleetheid en de klasse NP

Veronderstel dat C één of andere klasse van problemen is. We wensen nu een definitie die zegt dat het probleem π het moeilijkste probleem is van deze klasse of juister gezegd dat het minstens zo moeilijk is als alle problemen uit deze klasse. Dit is de definitie van C-compleetheid.

Definitie A.7 Als C een klasse van problemen is dan is een probleem π C-COMPLEET als en slechts als π tot de klasse C behoort en alle problemen uit de klasse C reduceren naar π. ∎

Als we van een bepaald probleem willen aantonen dat het een moeilijk probleem is, dan kunnen we proberen aan te tonen dat het een C-compleet probleem is voor een zeer grote klasse van problemen C.

Als we wensen aan te tonen dat het handelsreizigersprobleem "moeilijk" is dan zouden we kunnen proberen om als klasse C *alle* computationele problemen te nemen. Helaas werkt dit niet want Turing's stopprobleem is strikt moeilijker dan het handelsreizigersprobleem want dit laatste probleem kan

opgelost worden m.b.v. een algoritme met exponentiële uitvoeringstijd terwijl voor het stopprobleem helemaal geen algoritme bestaat.

Zoals net gezegd kan het handelsreizigersprobleem opgelost worden door een brutekracht algoritme. We wensen nu aan te tonen dat het handelsreizigersprobleem minstens zo moeilijk is als alle problemen die kunnen opgelost met zo'n brutekracht algoritme.

Definitie A.8 De KLASSE **NP** bestaat uit de problemen waarvoor oplossingen een lengte hebben die hoogstens polynomiaal is in de lengte van de invoer en waarvoor de correctheid van een oplossing kan *geverifieerd* worden in polynomiale tijd. ∎

Opmerking A.9 De klasse **NP** bestaat dus uit die problemen waarvoor het eenvoudig is om te *herkennen* dat je een oplossing hebt gevonden. Dit is anders dan de klasse P waar het eenvoudig is om een oplossing te *vinden*. ∎

Voorbeeld A.10 Veronderstel dat men zich in het handelsreizigersprobleem afvraagt of er een rondreis bestaat met kost hoogstens k.

Wanneer iemand je een *voorstel* van een rondreis geeft dan is het eenvoudig om te verifiëren of de totale kost van deze rondreis hoogstens k is. Inderdaad, je moet enkel maar controleren of de som van kosten van de bogen hoogstens k is. Dit kan gebeuren in lineaire tijd. ∎

Alles wat nodig is om tot de klasse **NP** te behoren is dat men op een efficiënte manier oplossingen kan herkennen en dit betekent dat de klasse **NP** enorm veel problemen omvat.

Wanneer een probleem **NP**-COMPLEET is dan betekent dit (door definitie van compleetheid) dat dit probleem minstens zo moeilijk is als alle problemen in **NP**. Dit suggereert dat het waarschijnlijk zeer moeilijk, zo niet onmogelijk is om voor een **NP**-compleet probleem een algoritme te bedenken dat kan uitgevoerd worden in polynomiale tijd. Bovendien: het vinden van zo'n algoritme voor een **NP**-compleet probleem zou betekenen dat *alle* problemen in **NP** kunnen opgelost worden in polynomiale tijd!

De hamvraag blijft natuurlijk of er wel zo'n **NP**-compleet probleem bestaat. Deze vraag werd, onafhankelijk van elkaar, door Cook in 1971 (Cook, 1971) en Levin in 1973 (Levin, 1973) positief beantwoord. In 1972 werd door Karp (Karp, 1972) een lijst van 21 **NP**-complete problemen beschreven. Sindsdien

is van honderden problemen bewezen dat ze **NP**-compleet zijn, waaronder ook het handelsreizigersprobleem en het knapzakprobleem.

Om aan te tonen dat een probleem **NP**-compleet is gaat men als volgt tewerk. Eerst toont men aan dat het probleem tot de klasse **NP** behoort. Dit is meestal gemakkelijk. In een tweede stap reduceert men een gekend **NP**-compleet probleem tot het nieuwe probleem.

Voorbeeld A.11 (TSP is NP-compleet) We beschouwen de *beslissingsversie* van het handelsreizigersprobleem: "gegeven een gewogen complete ongerichte graaf $G = (V, E)$ en een reëel getal d, bestaat er een rondreis die alle knopen juist éénmaal bezoekt zodanig dat de kost van de rondreis hoogstens d is." We noemen dit probleem TSP.

Het is duidelijk dat dit beslissingsprobleem tot de klasse **NP** behoort: als iemand je een voorstel van rondreis doet dan is het eenvoudig om te controleren of het een geldige rondreis is en of de kost inderdaad hoogstens d is. Dit kan gebeuren in een tijd $\Theta(n)$.

We *veronderstellen* nu dat we reeds weten dat het Hamiltoniaanse cykel probleem een **NP**-compleet probleem is. Het Hamiltoniaanse cykel probleem is het volgende: "gegeven een ongerichte (ongewogen) graaf $G = (V, E)$, bestaat er een Hamiltoniaanse cykel, i.e. bestaat er een cykel die alle knopen van de graaf bevat?" Dit probleem noemen we HC.

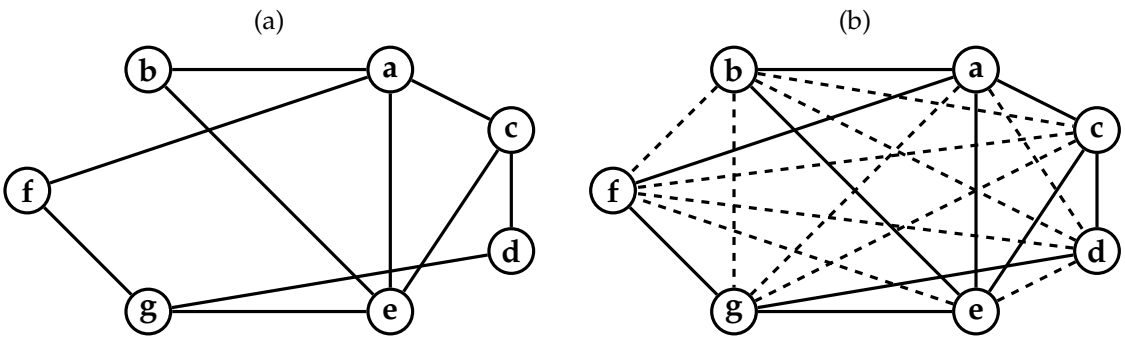
We tonen nu aan dat HC reduceert tot TSP. Gegeven een graaf $G = (V, E)$ waarop we HC willen toepassen. Construeer nu een nieuwe (gewogen) en complete graaf $G' = (V, E')$ (met dezelfde knopenverzameling) als volgt:

$$\text{gewicht}(e'_{i,j}) = \begin{cases} 1 & \text{als knopen } i \text{ en } j \text{ adjacent zijn in } G \\ 2 & \text{als knopen } i \text{ en } j \text{ niet adjacent zijn in } G. \end{cases}$$

We tonen nu aan dat het antwoord op HC voor $G = (V, E)$ "ja" is als en slechts als het antwoord op TSP voor $G' = (V, E')$ met $d = n$ "ja" is.

Veronderstel dat $G = (V, E)$ een Hamiltoniaanse cykel bezit, dan kunnen in $G' = (V, E')$ al de bogen met gewicht 1 gevolgd worden om een rondreis te bekomen met gewicht gelijk aan n. En dus als HC "ja" heeft als antwoord heeft ook TSP "ja" als antwoord.

Omgekeerd: veronderstel dat er een rondreis is met kost hoogstens n in $G' = (V, E')$. Aangezien deze rondreis n bogen moet bevatten kan deze



Figuur A.1: Links ziet men de originele ongewogen graaf $G = (V, E)$ voor probleem HC. Rechts ziet men de gewogen graaf $G' = (V, E')$. Om de figuur niet te zwaar te maken zijn de bogen met gewicht 1 met een volle lijn getekend, terwijl de bogen met gewicht 2 met een streepjeslijn zijn aangeduid. Het is duidelijk dat de rechtse graaf een rondreis heeft met kost hoogstens 7 als en slechts de linkse graaf een Hamiltoniaanse cykel heeft. We kunnen TSP dus gebruiken om HC op te lossen, of anders gezegd HC reduceert naar TSP.

enkel bogen met gewicht gelijk aan 1 bevatten. Deze bogen vormen een Hamiltoniaanse cykel in de oorspronkelijke graaf. M.a.w. als TSP "ja" heeft als antwoord heeft ook HC "ja" als antwoord.

Figuur A.1 geeft een illustratie van dit proces. ∎

A.4 De klasse P versus NP

Het is duidelijk dat de klasse **P** tot de klasse **NP** behoort. Inderdaad, wanneer men efficiënt een oplossing kan berekenen (de klasse **P**) kan men ook efficiënt controleren of een gegeven oplossing correct is (de klasse **NP**). Er geldt dus zeker

$$\mathbf{P} \subseteq \mathbf{NP}.$$

Dé cruciale vraag is echter of de twee klassen samenvallen of niet. Tot op heden is het antwoord hierop nog onbekend al gaan de meeste computerwetenschappers er wel van uit dat de klasse **NP** strikt groter is dan de klasse **P**.

Het **P** versus **NP** probleem is één van de zeven problemen op de lijst van "Millennium Prizes" van het Clay Institute of Mathematics. Een definitief

antwoord op deze vraag is dan ook 1 miljoen dollar waard!

Wanneer men in de praktijk te maken heeft met een **NP**-compleet probleem dan kan men dit probleem meestal slechts voor "kleine" instanties exact oplossen binnen een redelijke tijd. Voor grotere instanties neemt men vaak zijn toevlucht tot benaderingsalgoritmen.

Bibliografie

Cook, S. A. (1971). The Complexity of Theorem-proving Procedures. In *Proceedings of the Third Annual ACM Symposium on Theory of Computing*, STOC '71, pages 151–158, New York, NY, USA. ACM.

Karp, R. M. (1972). *Reducibility Among Combinatorial Problems*, pages 85–103. Springer US, Boston, MA.

Levin, L. A. (1973). Universal sequential search problems. *Problemy Peredachi Informatsii*, 9(3):115–116.

Turing, A. (1936). On computable numbers. *Proceedings of the London Mathematical Society*, 42:230–265.