

# 5

## Bomen

### 5.1 Inleiding

Een van de belangrijkste onderwerpen in de theoretische informatica is de grafen-theorie, met als veelgebruikt onderdeel daarvan de bomen.

Een graaf (Engels: graph) is letterlijk niets meer dan een 'plaatje' (een 'grafiek'). Met zo'n 'plaatje' willen we dan sommige dingen (eigenschappen en dergelijke) duidelijker weergeven. Daarom moet een graaf niet zomaar een plaatje zijn, maar aan een aantal eisen voldoen. Aan de hand van een voorbeeldje zullen we zien wat nu wel en wat nu géén graaf genoemd kan worden.

Later zullen we zien dat een boom niets anders is dan een bijzonder soort graaf.

*Voorbeeld 1* Stel je een aantal verschillende computers voor, bijvoorbeeld een Vax, drie PC's en een Gould, die tezamen in één 'netwerk' aan elkaar gekoppeld zijn. In een plaatje zou dat er als volgt uit kunnen zien:

$$T_1 = \{(a, b), \{b, c\}, \{b, b\}\} \text{ en } P_2 = \{(a, b), (b, c), (b, b)\}.$$

Merk op dat een *tak* dus weergegeven wordt door een *verzameling van twee knopen* (bijvoorbeeld  $\{a, b\}$ ), terwijl een *pijl* weer gegeven wordt door een *geordend paar van twee knopen* (bijvoorbeeld  $(a, b)$ ). Dat is ook logisch want bij een tak is de richting niet van belang, maar bij een pijl wel.

We noemen  $G_1$  een *ongerichte graaf* en  $G_2$  een *gerichte graaf*.

We spreken af dat een graaf ofwel slechts pijlen, ofwel slechts takken heeft en dus nooit zowel pijlen als takken.

Bovendien geldt, dat de knopen-verzameling niet leeg mag zijn, maar de verzameling takken (pijlen) wel.  
Verder staat altijd de benaming bij elke knoop vermeld.

### Gerichte grafen

Een *ongerichte graaf* is een plaatje (tekening) waarin één of meer knopen (•) en nul of meer takken (tussen twee knopen) getekend zijn. We spreken van een *ongerichte graaf*, omdat in een tak nu eenmaal geen richting wordt aangegeven. Formeel kunnen we dit als volgt formuleren.

**Definitie 5.1** Een *ongerichte graaf* bestaat uit een *verzameling knopen* (de verzameling mag *niet* leeg zijn) en een *verzameling takken*, waarvan elk element twee knopen verbindt (de verzameling takken mag *wel* leeg zijn).

We beschrijven een *ongerichte graaf*  $G$  met het geordend paar  $(K, T)$ , waarbij  $K$  de verzameling knopen voorstelt en  $T$  de verzameling takken.

We kunnen een *ongerichte graaf* dus *tekenen* maar ook *beschrijven* door (op de juiste wijze) de twee verzamelingen te definiëren.

Met een paar eenvoudige grafen als voorbeeld zullen we enkele belangrijke begrippen over *ongerichte grafen* behandelen.

- $G$  bestaat uit één component dan als ze verbonden is.
- In  $G$  zijn  $n$  afzonderlijke, verbonden grafen zichtbaar, dan en slechts dan als  $G$  uit  $n$  componenten bestaat.  
Van de bovenstaande grafen bestaat  $G_1$  uit twee componenten en  $G_2$  uit één component.

**Voorbeeld 3** Beschouw de volgende twee (ongerichte) grafen  $G_1$  en  $G_2$ .

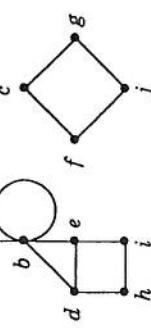
$$G_1 = (K_1, T_1)$$

$$G_2 = (K_2, T_2)$$



$$G_1 = (K_1, T_1)$$

$$G_2 = (K_2, T_2)$$



We zien dat graaf  $G_1$  uit twee delen bestaat (we noemen ze *componenten*), er is dus geen pad mogelijk van bijvoorbeeld knoop  $b$  naar knoop  $c$ . Daarentegen zien we dat er in graaf  $G_2$  een pad is vanuit *elke* knoop naar *elke* knoop: graaf  $G_2$  bestaat uit één component.

□ We zeggen dat  $G_2$  *verbonden* is en  $G_1$  niet.

### Enkele begrippen bij ongerichte grafen

**Definitie 5.2** Voor elke ongerichte graaf  $G$  gelden de volgende begrippen.

- We bedoelen met het pad  $pad_1 = a \rightarrow b \rightarrow c$  het pad dat achtereenvolgens langs de knopen  $a$ ,  $b$  en  $c$  voert.
- De padlengte van  $pad_1$  is het aantal takken dat  $pad_1$  gebruikt. (De padlengte kan 0 zijn, maar niet negatief.)  
Bovengenoemd  $pad_1$  heeft dus lengte 2.
- $G$  is *verbonden* dan en slechts dan als er een pad is van *elke* knoop naar *elke* knoop. (Zo'n pad mag lengte 0 hebben.)  
Van de bovenstaande grafen is  $G_2$  wel verbonden, maar  $G_1$  niet.

- De graad van knoop  $a$  is het aantal takken dat in  $a$  uitmondt. We schrijven graad ( $a$ ). Zo geldt in graaf  $G_1$  van voorbeeld 3: graad ( $d$ ) = 3 en graad ( $b$ ) = 5.
- voorbeeld 3: graad ( $d$ ) = 3 en graad ( $b$ ) = 5.

### Gerichte grafen

Een *gerichte graaf* is een plaatje (tekening) waarin één of meer knopen (•) en nul of meer pijlen (tussen twee knopen) getekend zijn. We spreken van een gerichte graaf omdat met een pijl een richting wordt aangegeven.

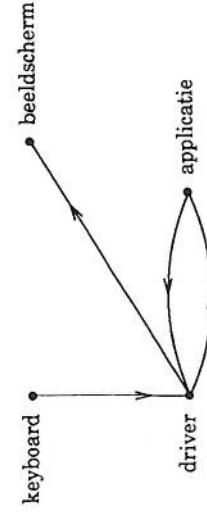
Formeel kunnen we dit als volgt formuleren.

- Definitie 5.3** Een gerichte graaf bestaat uit een *verzameling* knopen (de verzameling mag *niet* leeg zijn) en een *verzameling* pijlen, waarvan elk element twee knopen verbindt (de verzameling pijlen mag *wel* leeg zijn). Een gerichte graaf beschrijven we met het geordend paar  $(K, P)$ , waarbij  $K$  de verzameling knopen voorstelt en  $P$  de verzameling pijlen.

We kunnen een gerichte graaf dus, net als een ongerichte, *tekenen* maar ook *beschrijven* door (op de juiste wijze) de verzamelingen  $K$  en  $P$  te definiëren.

Een voorbeeld van een *gerichte* graaf is het volgende.

- Voorbeeld 4** We kunnen ons een *terminal-sessie* als volgt voorstellen: er is een toetsenbord (keyboard), een beeldscherm, een driver (stuk software, dat zorgt voor de besturing van de data) en een verbinding met de applicatie (een gebruikers-programma). Het 'plaatje' zou er als volgt uit kunnen zien:



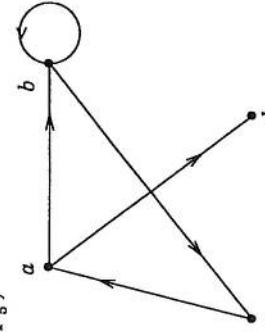
- Duidelijk zal zijn dat er vanuit de driver met de applicatie 'gecommuniceerd' kan worden, en vice versa. Ook zal duidelijk zijn dat er alleen *naar* het beeldscherm informatie (tekst) kan worden verzonden en alleen *vanuit* het keyboard.

Alles wat we intikken op het keyboard wordt (door de driver) onmiddellijk naar het scherm gestuurd. Bovendien zal deze data (tekst) ook naar het gebruikers-programma gezonden worden. Ook dit gebeurt door de driver.  
Naar een keyboard iets versturen is onzin, vanaf beeldscherm iets verzenden ook. Het is duidelijk dat de *richting* erg belangrijk is: we kunnen dit daarom alleen goed weergeven in  een *gerichte* graaf.

### Enkele begrippen bij gerichte grafen.

**Voorbeeld 5** Beschouw de volgende (gerichte) graaf:

$$G_5 = (K_5, P_5)$$



In de graaf is duidelijk te zien dat er vanuit knoop  $a$  in totaal *vier* pijlen *vertrekken* en één pijl *aankomt*. We spreken van de *ingraad* en *uitgraad* van  $a$ , die respectievelijk 4 en 1 bedragen. Zo geldt dat  $\text{ingraad}(a) = 4$  en  $\text{uitgraad}(a) = 1$ .

We zien verder dat er bij knoop  $b$  één pijl is, die  $b$  zowel in- als uitslaat. We spreken bij zo'n pijl van een *lus*. (ook wel het Engelse woord 'loop' genoemd.) Daarmee wordt de  $\text{ingraad}(b) = 2$ ,  $\text{uitgraad}(b) = 2$ .

- We zien ook dat er vanuit  $a$  een 'weg' mogelijk is om via  $b$  in  $c$  te komen: we spreken van een *pad* van  $a$  naar  $c$ . Formeel beschrijven we zo'n pad in een graaf door de opeenvolgende knopen te benoemen waar het pad langs voert. Hier dus:  $a \rightarrow b \rightarrow c$ . Een pad geven we vaak een naam als  $pad_1$  (dus bijvoorbeeld:  $pad_1 = a \rightarrow b \rightarrow c$ ). We zien dat  $pad_1 = a \rightarrow b \rightarrow c$  twee pijlen 'gebruikt': we spreken van *padlengte 2*. Als we in bijvoorbeeld knoop  $a$  zijn, kunnen we ook spreken van een pad  $a$  met lengte 0 (dit kan uiteraard vanuit *elke* knoop).

$\square$

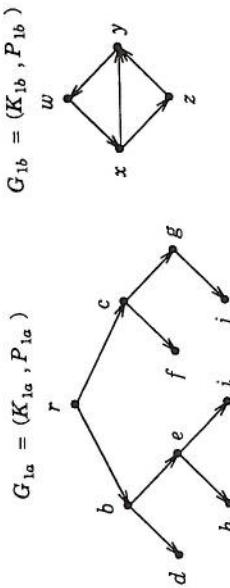
#### Definitie 5.4 Gegeven een gerichte graaf $G = (K, P)$ , dan geldt:

- Ingraad( $a$ ) = 'de ingraad van knoop  $a$ ' = het aantal pijlen dat in  $a$  aankomt.  
In bovengetekende graaf  $G_5$  geldt dat ingraad( $a$ ) = 1 en  
ingraad( $b$ ) = 2.
- Uitgraad( $a$ ) = 'de uitgraad van knoop  $a$ ' = het aantal pijlen dat uit  $a$  vertrekt.  
In bovengetekende graaf  $G_5$  geldt dat uitgraad( $a$ ) = 2 en  
uitgraad( $b$ ) = 2.
- Een pad is *gericht* als er steeds rekening gehouden is met de richting van de pijlen.  
Een voorbeeld van een gericht pad in graaf  $G_5$  is  
 $pad_1 = a \rightarrow b \rightarrow c$  maar  $pad_2 = a \rightarrow b \rightarrow b \rightarrow a$  is  
geen gericht pad.
- $pad_3$  is een *ongericht* pad als er geen rekening gehouden is met de richting van de pijlen. Dit betekent dat er wel overall een verbinding (pijl) moet zijn, maar dat een pijl ook genomen mag worden tegen zijn richting in.  
Een voorbeeld van een ongericht pad in graaf  $G_5$  is  
 $pad_3 = a \rightarrow b \rightarrow b \rightarrow a$  maar  $pad_4 = a \rightarrow b \rightarrow b \rightarrow d$  is  
geen ongericht pad.

### 5.3 De Boom: een bijzondere graaf

Een in het vakgebied van de informatica veelgebruikte toepassing van de graaf is de *boom*. Aan de hand van een eenvoudig voorbeeldje zullen we bekijken wat precies een boom is en wat niet.

#### Voorbeeld 1



In bovenstaande (verbonden) grafen  $G_{1a}$  en  $G_{1b}$  zien we dat in  $G_{1a}$  er precies één knoop is waar geen enkele pijl aankomt (knoop  $r$ ); deze knoop noemen we de *wortel*. In *elke* andere knoop komt precies één pijl aan en gaan er 0 of meer uit. De *ingraad* van elke knoop is 1 (op die van  $r$  na), terwijl de *uitgradiënt* van elke knoop 0 of meer mag zijn (willekeurig dus). In  $G_{1b}$  zien we dat er geen enkele knoop is met ingraad 0, terwijl bovendien knoop  $Y$  ingraad 2 heeft.

We noemen een graaf zoals  $G_{1a}$  een *boom* en graaf  $G_{1b}$  niet.

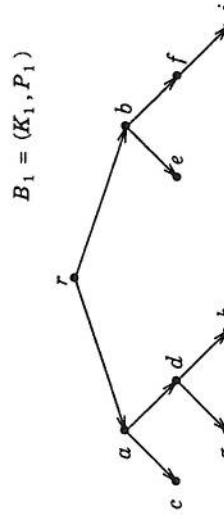
**Definitie 5.5** Een boom is een verbonden, gerichte graaf met als extra eisen:

- Er is precies één knoop met ingraad 0. Deze knoop noemen we de *wortel*.
- Alle andere knopen hebben ingraad 1.

graaf op dat ook deze gerichte graaf een boom is (dat wil zeggen dat aan alle eisen van de boom-definitie voldaan is).

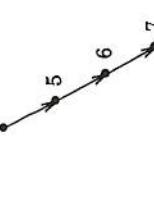
**Definitie 5.6** We noemen een boom een *binaire boom* als *elke* knoop als uitgradiënt maximaal twee heeft.

Merk op dat volgens deze definitie de volgende twee grafen binaire bomen zijn (hoe onnuttig wellicht dan ook):



**Voorbeeld 3**

$$B_{3a} = (K_{3a}, P_{3a}) \quad B_{3b} = (K_{3b}, P_{3b})$$



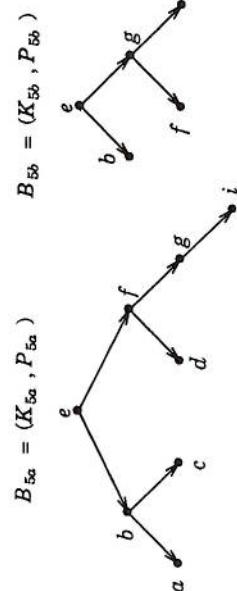
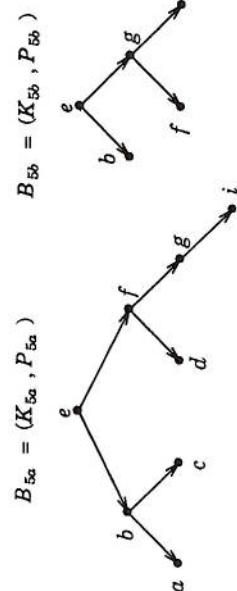
□

**Definitie 5.7** We noemen een binaire boom *compleet* als *elke* knoop of precies 0 of precies 2 zonen (opvolgers) heeft.

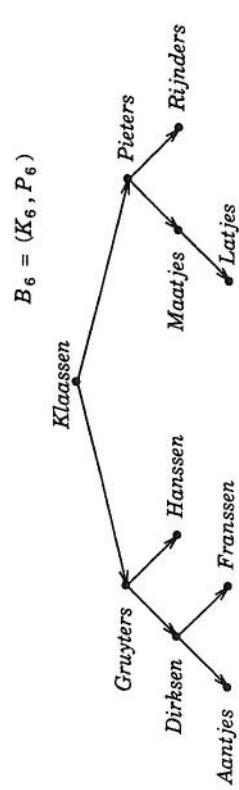
**Voorbeeld 4**

$$B_4 = (K_4, P_4)$$

□

**Voorbeeld 5**

□

**Voorbeeld 6**

□

Merk op dat bovenstaande boom een (binaire) *complete* boom is: *elke* knoop heeft ofwel uitgraad 0 ofwel uitgraad 2, maar nooit uitgraad 1.

□

**Geordende boom**

We noemen een binaire boom *geordend* wanneer er een vaste ordening in alle knoop-waarden te herkennen is en wel zodanig dat voor elke knoop in de *linker-subboom* alle knoop-waarden kleiner zijn dan de onderhavige knoop-waarde en in de *rechter-subboom* alle knoop-waarden groter zijn dan de onderhavige knoop-waarde.

N.B.: deze definitie is 'de meest gebruikelijke'. In theorie echter kan een binaire boom ook geordend zijn als alle linker-afstammelingen een *grote* knoop-waarde hebben en alle rechter-afstammelingen een *kleinere* knoop-waarde.

Voorop staat dat de ordening (groter of kleiner) in de gehele boom consequent doorgevoerd is.

Van de onderstaande bomen is de linker niet geordend, maar de rechter wel. (Ga dit na.)

Bekijk bovenstaande boom, waarin je moet voorstellen dat er (echte) records in de knopen opgeslagen zijn: knoop *Klaassen* stelt dus een record voor waarin alle gegevens van de heer *Klaassen* opgeslagen zijn (naam, adres, woonplaats, enzovoort).

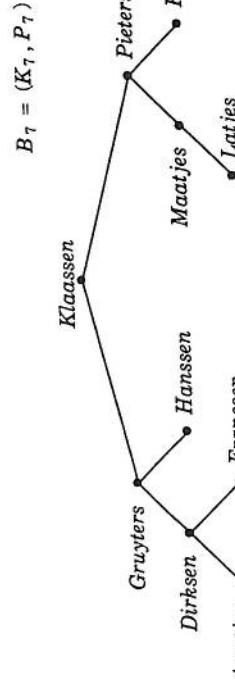
Merk op dat er een (alfabetische) ordening in de boom is.

Stel dat we de gegevens van de heer (of vrouw) *Maatjes* willen hebben, dan kunnen we het juiste record zoeken door in de wortel van de boom te beginnen (dit is de enige knoop met ingraad 0; hier zullen we beginnen). We vergelijken de naam *Maatjes* met *Klaassen* en daar *Maatjes* > *Klaassen* moeten we hier 'rechtsaf'. Zo komen we via *Pieters* tenslotte bij het record van *Maatjes* aan (ga na dat dit indertijd de weg is die afgelegd moet worden om van de wortel naar het juiste record te gaan). Dit lukt natuurlijk alleen maar omdat we *alle* records *geordend* (hier: alfabetisch) in de boom hebben opgeslagen.

Stel dat we nu het record van de heer of mevrouw *Mijndert* zouden willen hebben, dan zouden we er bij het bereiken van knoop *Maatjes* achter komen dat dit record er niet is (hier zouden we dan namelijk weer 'rechtsaf' moeten gaan. Als we een en ander in een gewone file (sequentieel) opgeslagen hadden, zouden we daar pas achter komen na het lezen van achtereenvolgenselementen: *Aantjes, Dirksen, Franssen, Gruyters, Hanssen, Klaassen, Latjes en Maatjes*. Het zal duidelijk zijn dat deze laatste methode veel meer lezen en vergelijken vergt.

Wanneer een en ander geen verwarring kan wekken, wil men de pijlen in de tekening van de boom nog wel eens vervangen door takken: de graaf *lijkt* dan een ongerichte graaf geworden, maar we bedoelen er wel degelijk een *gerichte* mee. Dat is ook wat we willen, zoals blijkt uit het voorgaande voorbeeld over het opslaan van records in een (binaire) boom.

Onze boom zou er dan als volgt uit kunnen zien:



Uit de getekende structuur van de boom moet dan blijken wat de wortel is en wat eigenlijk de richting van de takken (die pijlen moeten zijn) is. We gaan er van uit dat *elke* boom een *gerichte* graaf is, ook als de richting niet expliciet met behulp van pijlen is aangegeven, maar de knopen door middel van takken met elkaar verbonden zijn: de richting kunnen we er dan gemakkelijk zelf bij denken!

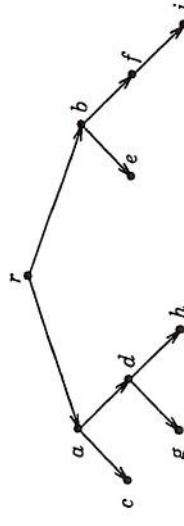
## 5.4 Begrippen

We hebben in de vorige paragrafen al kennis gemaakt met enkele termen en begrippen over bomen: hier zullen we nog een aantal nieuwe behandelen en tevens de oude herhalen zodat we ze allemaal bij elkaar hebben staan.

Gegeven de volgende (binaire) boom aan de hand waarvan we de begrippen zullen uitleggen:

*Voorbeeld 1*

$$B_1 = (K_1, P_1)$$



□

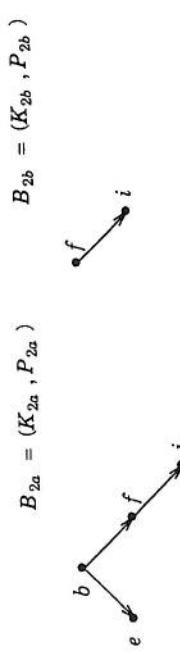
- De gegeven boom telt precies tien knopen.
- Er is precies één knoop met ingraad 0 (namelijk knoop *r*). Deze knoop noemen we de *wortel*.
- Knoop *b* heeft twee *zonen* (ook wel *opvolgers* genoemd) namelijk *e* en *f*.
- Knoop *b* heeft drie *afstammelingen*, namelijk *e*, *f* en *i*.
- Knoop *b* is de *vader* (ook wel *voorganger* genoemd) van de knopen *e* en *f*.
- Knoop *f* heeft twee *voorouders*, namelijk *b* en *r*. Knoop *h* heeft drie voorouders: *d*, *a* en *r*.
- Knoop *b* wordt wel de *grootvader* van knoop *i* genoemd.
- Knoop *i* is dan een *kleinzoon* van knoop *b*.
- Knoop *r* ligt op niveau nul, knopen *a* en *b* op niveau één, enzovoort.
- Het hoogst mogelijke niveau in deze boom is drie (knopen *g*, *h* en *i*), men zegt dan dat de *diepte* (ook wel *hoogte* genaamd) van de boom drie is.

- Onder andere knoop  $c$  heeft géén afstammelingen: men noemt  $c$  dan wel een *blad*. Andere bladeren zijn  $g, h, e$  en  $i$ .
- Onder andere knoop  $a$  heeft wel afstammelingen: men noemt  $a$  dan wel een *interne knoop*. Andere interne knopen zijn  $b, d, f$  en  $r$ .
- Zoals we in de vorige paragraaf geleerd hebben is de boom binair en niet compleet.

### ubbomen

Met bovenstaande boom kunnen we nog veel meer doen. Zo kunnen we bijvoorbeeld uit die boom onder andere de volgende bomen halen.

### Voorbeeld 2



Merk op dat  $B_{2a}$  en  $B_{2b}$  op zichzelf ook (binaire) bomen zijn, terwijl ze elk bovendien een 'onderdeel' vormen van de oorspronkelijke boom  $B_1$  uit voorbeeld 1. We noemen ze *subbömen* van  $B_1$ .

- $\square$  *men van  $B_1$*
- Definitie 5.8** Gegeven een boom  $B_1$  kunnen we een *subboom* maken uit  $B_1$  door een knoop uit  $K_1$  te nemen met al zijn afstammelingen uit  $B_1$ .

- Elke boom heeft tenminste één subboom: de boom zelf is zijn eigen subboom (vergelijk dit met het begrip *deelverzameling*: elke verzameling heeft zichzelf als deelverzameling).
- $\square$

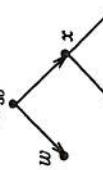
Ga zelf na dat  $B_1$  van deze paragraaf *precies* tien subbömen heeft.

### Gebalanceerde boom

Vergelijk de twee onderstaande (binaire) bomen.

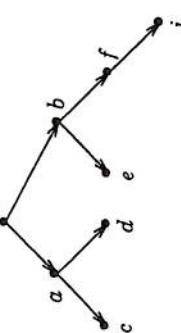
$$B_{3a} = (K_{3a}, P_{3a})$$

$$R_{3a}$$



$$B_{3b} = (K_{3b}, P_{3b})$$

$$R_{3b}$$



Merk op dat beide bomen inderdaad binair zijn; dat is namelijk van belang voor het begrip *gebalanceerd*.

Je kunt zien dat in  $B_{3b}$  knoop  $R_{3b}$  links één afstammeling heeft en *rechts* drie (ga dit na).

In boom  $B_{3a}$  heeft  $R_{3a}$  links drie afstammelingen en rechts vier. Knoop  $b$  heeft links één afstammeling en rechts twee.

Knoop  $f$  heeft er links nul en rechts één enzovoort. Merk op (en ga dit zelf na) dat in boom  $B_{3a}$  geldt dat voor *geen enkele* knoop het verschil tussen het aantal afstammelingen links en rechts méér dan één is! In  $B_{3b}$  geldt dit niet; het aantal afstammelingen links en rechts van de wortel  $R_{3b}$  verschilt twee.

We noemen dan  $B_{3a}$  *gebalanceerd* en  $B_{3b}$  niet.

**Definitie 5.9** Een binaire boom is gebalanceerd als voor *elke* knoop geldt dat het aantal afstammelingen links en rechts *maximaal één verschilt*.

$\square$

### Perfect gebalanceerde (binaire) boom

Analoog aan het begrip gebalanceerd kunnen we nu het begrip *perfect gebalanceerd* behandelen: hierbij mag het aantal afstammelingen links en rechts van *elke knoop helemaal niet* verschillen!

**Definitie 5.10** Een binaire boom heet *perfect gebalanceerd* als voor *elke* knoop geldt dat het aantal afstammelingen links en rechts *niet* verschillt.

In de informatica zijn de begrippen gebalanceerd en (in mindere mate) perfect gebalanceerd buitengewoon belangrijk. Als de boom in de vorige paragraaf niet gebalanceerd zou zijn, zouden we onnodig veel niveau's diep moeten zoeken.

### 5.5 Infix-, prefix- en postfix-notatie

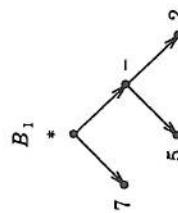
We herinneren ons wat we onder een geordende binaire boom verstaan (paragraaf 5.3). Een bijzonder soort geordende boom komen we in deze paragraaf tegen: een geordende gelabelde (binaire) boom. In zo'n geordende gelabelde boom zit niet alleen een ordening in de knopen (zoals in elke geordende boom), maar er wordt bovendien onderscheid gemaakt tussen de soort gegevens die in de bladeren mag worden opgeslagen en de soort gegevens in de interne knopen.

#### Geordende gelabelde boom

Behalve getallen, records en dergelijke kunnen we ook een 'expressie' in een boom opslaan: dit gebeurt in feite in elke rekenmachine om een rekenkundige uitdrukking 'uit te rekenen'.

Aan de hand van een voorbeeldje zullen we dit duidelijk maken.

**Voorbeeld 1** Stel we willen het rekensommetje  $7*(5-2)$  in een boom opslaan, zodat het 'stapsgewijs' uitgerekend kan worden. Dit gaat dan op de volgende manier:



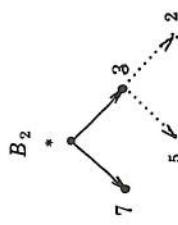
Bekijken we deze boom dan zien we het volgende:

- De *getallen* 7, 5 en 2 staan alle in de bladeren.
- De *operatoren* \* en - staan alle in de interne knopen.

Dit is een eis die we voortaan aan elke boom stellen waarmee we een formule willen weergeven. Zo'n boom noemen we een *geordende gelabelde boom* en we kunnen ermee aangeven hoe in een computer (of een zakrekenmachine) een rekenopgave uitgerekend wordt. Dit laatste (het uitrekenen) gaat als volgt:

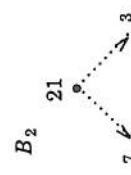
**Definitie 5.11** Een *geordende gelabelde boom* is een (binaire) geordende boom waarin we een (bijvoorbeeld) rekenkundige, wiskundige, logische expressie kunnen weergeven en die aan de volgende eisen voldoet:

- In de bladeren van de boom komen *de operanden* te staan.
  - In de interne knopen van de boom komen *de operatoren* te staan.
- 



In het rechterblad staat nu een 3 omdat  $5 - 2 = 3$  is uitgerekend. (Het zojuist geëvalueerde subboomje is voor de duidelijkheid nog gestippeld getekend: in werkelijkheid wordt zo'n subboom 'opgeruimd' op het moment dat hij geëvalueerd is en bestaat hij dus niet meer.)  
Bij het uitrekenen wordt altijd begonnen met de kleinste subboomjes waar een op zichzelf staand rekensommetje staat (hier dus 5-2).

Rekenen we deze boom verder uit, dan is er nog slechts het sommetje  $7 * 3$  uit te rekenen en de uitkomst ervan (21) in de wortel van de boom te zetten:



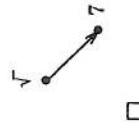
In het algemeen wordt een geordende gelabelde boom als volgt geëvalueerd:

- Eerst worden de kleinste subboomjes (die nog één rekenkundige expressie bevatten) uitgerekend en de uitkomst ervan in de wortel van het subboomje gezet. (Het subboomje 'verdwijnt' dan dus, op de wortel ervan na.)
- Daarna worden weer de kleinste subboomjes uitgerekend. Dit proces gaat door tot slechts de wortel een waarde bevat (de uitkomst).

Aan de hand van dit voorbeeld gaan we eerst vastleggen wat we precies onder een *geordende gelabelde boom* verstaan; we bedoelen er een boom mee waarin een (bijvoorbeeld) rekenkundige, logische, wiskundige uitdrukking kan worden opgeslagen, zodat die per subboom kan worden geëvalueerd.

Hierboven hebben we alleen operatoren gebruikt die steeds *twee* operanden verbinden: de '\*' bijvoorbeeld 'hoort bij' 7 en bij 3 in  $B_2$ . Zulke operatoren noemen we daarom wel *binaire* operatoren. Er zijn ook *unaire* operatoren: bijvoorbeeld  $\sqrt{\phantom{x}}$ . Deze laatste 'hoort bij' slechts één operand (getal). Iets als  $\sqrt{7^2}$  heeft geen duidelijke betekenis, maar een uitdrukking als  $\sqrt{7}$  wel (althans als element van  $\mathbb{R}$ ). De  $\sqrt{\phantom{x}}$ -operator is een voorbeeld van een *unaire* operator. In de logica is bijvoorbeeld de ontkening ( $\bar{P}$ ) er ook een. Bij afspraak tekenen we de operand *altijd rechts onder* de unaire operator, zoals in onderstaand voorbeeld:

### Voorbeeld 2



### Infix-notatie

Nu willen we wel eens vanuit een gegeven geordende gelabelde boom de erbijbehorende uitdrukking terugkrijgen. Hoe krijgen we bijvoorbeeld uit  $B_2$  het oorspronkelijke rekenkundige sommetje terug? Hiervoor moeten we duidelijke afspraken maken, zodat het nooit verwarring kan wekken welk sommetje er precies mee bedoeld werd. Een van de (in totaal drie) methoden is met behulp van de zogenaamde *infix-notatie*. Deze werkt als volgt: bij elke (binaire) operator wordt eerst de linkerzoon opgeschreven, dan de operator en tenslotte de rechterzoon. Indien bijvoorbeeld de rechterzoon zelf een subboom is wordt deze subboom ook infix genoteerd. We krijgen dan uit  $B_1: 7 * 5 - 2$  (ga dit na). Andersom kunnen we uit een infix-notatie de bijbehorende geordende gelabelde boom tekenen.

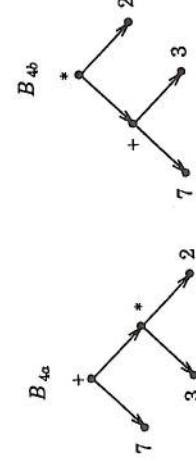
N.B.: Een infix-notatie is een notatie zonder haakjes en zonder prioriteiten van de operatoren (dus zonder Meneer Van Dalen Wacht Op Antwoord -regel). Dit leidt soms tot problemen en daarom zullen we nog andere notatiwijzen behandelen.

**Voorbeeld 3** Gegeven de infix-notatie  $12 + 13$ . De bijbehorende geordende gelabelde boom wordt dan:



Hier is één vergissing mogelijk: *elke* andere boom hoort niet bij de gegeven infix-notatie.

**Voorbeeld 4** Nu is gegeven  $7 + 3 * 2$  als infix-notatie. De bijbehorende geordende gelabelde boom kan nu ook op twee verschillende manieren getekend worden:



Wanneer we beide bomen zouden evalueren, krijgen we uit  $B_{4a}$  als antwoord 13 en uit  $B_{4b}$  20. Uit dit voorbeeld blijkt duidelijk dat de infix-notatie (hoeveel veel gebruikt om historische redenen) niet echt handig is (ga zelf na dat beide bomen wel degelijk goede bomen zijn, behorende bij de gegeven infix-notatie).

De prefix-notatie is uitgevonden door de Poolse wiskundige Jan Lukasiewics (in 1951) en wordt daarom ook wel de Poolse notatie genoemd.

**Voorbeeld 5** Gegeven de volgende boom:

$B_5$



In infix-notatie zou het er als volgt uitzien:  $3 * 2$ . Bij de prefix-notatie schrijven we eerst de operator en daarna de linker operand en tenslotte de rechter operand. Hier wordt dat dus: \* 3 2 (merk op dat wel duidelijk moet zijn dat 3 en 2 twee verschillende operanden zijn en niet samen het getal '32' voorstellen).

**Voorbeeld 6** Gegeven de volgende twee geordende gelabelde bomen (dezelfde als in voorbeeld 4):

$B_{6a}$



Bij de prefix-notatie schrijven we eerst de operator, daarna de linker en dan de rechter operand. Als er geen rechter operand is maar in de plaats daarvan een subboom, dan wordt daarvan de (prefix-) notatie gegeven. Bij  $B_{6a}$  wordt dat dus: + 7 \* 3 2 en bij  $B_{6b}$  wordt dit \* + 7 3 2 (ga dit zelf na).

We zien dat beide notaties nu wel verschillend zijn.

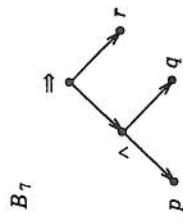
De prefix-notatie is uitgevonden door de Poolse wiskundige Jan Lukasiewics (in 1951) en wordt daarom ook wel de Poolse notatie genoemd.

### Prefix-notatie

We willen dus een notatie-wijze waarbij altijd maar één geordende gelabelde boom hoort bij een gegeven uitdrukking. Dit kan met behulp van de zogenaamde prefix-notatie. Eerst zullen we zien wat dat precies voorstelt.



**Voorbeeld 7** Gegeven de volgende prefix-notatie  $\wedge p q r$ . Teken de bijbehorende (binaire) geordende gelabelde boom. Oplossing: we zetten haakjes bij elke subboom (bij elke binaire operator samen met de twee operanden die erna staan). Dit levert ( $\Rightarrow (\wedge p q) r$ ). De boom wordt dan als volgt:

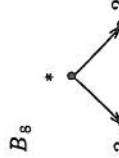


- De bijbehorende logische uitdrukking is:  
  $(p \wedge q) \Rightarrow r$
- N.B.: Bij elke gegeven prefix-notatie hoort altijd precies één geordende gelabelde boom.

#### Postfix-notatie

Aan de hand van al eerder genoemd voorbeeld 5 laten we deze laatste notatie-wijze zien:

**Voorbeeld 8** Gegeven de volgende boom:



In prefix-notatie zou de bijbehorende expressie er als volgt uitzien: \*32. Bij de postfix-notatie schrijven we eerst de linker operand, daarna de rechter operand en tenslotte de operator. Hier wordt dat dus: 32\* (merk op dat wel duidelijk moet zijn dat 3 en 2 twee verschillende operanden zijn en niet samen het getal '32' voorstellen).

We nemen weer een al eerder behandeld voorbeeld voor een ingewikkelder uitdrukking te beschrijven:

**Voorbeeld 9** Gegeven de volgende twee geordende gelabelde bomen (dezelfde als uit voorbeeld 6):



Bij de **postfix**-notatie schrijven we eerst de linker operand daarna de rechter operand en tenslotte de operator. Als er geen rechter operand is maar in de plaats daarvan een subboom, dan wordt daarvan de (postfix-) notatie gegeven: 732\*+. En bij  $B_{9b}$  wordt dit 73+2\* (ga dit zelf na).  
 We zien dat beide notaties nu wel verschillend zijn. De postfix-notatie wordt ook wel *reversed Polish* genoemd.  
 N.B.: Bij elke gegeven postfix-notatie hoort altijd precies één geordende gelabelde boom.

## 5.6 Preorder-, inorder- en postorder-notatie

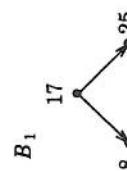
Wanneer we records in een (binaire) boom opgeslagen hebben en we zouden bijvoorbeeld hiervan een lijstje willen hebben (op papier) dan moeten we op de een of andere manier 'door de boom wandelen' en record na record in het werkgeheugen inlezen, zodat het kan worden afdrukkt. Analoog aan de prefix- en infixnotatie van de vorige paragraaf kunnen we dat ook hier doen, alleen betreft het dan natuurlijk geen geordende *gelabelde* boom meer (*wel geordend, niet gelabeld*).

### Preorder

*W.L.Q.*

Aan de hand van een eenvoudig voorbeeld zullen we zien hoe een en ander in zijn werk gaat bij een boom met records.

**Voorbeeld 1** Stel dat we drie records in een (binaire) boom opgeslagen hebben, uiteraard op geordende wijze. We kunnen deze records voorstellen door hun sleutelwaarden te benoemen. Onze boom zou er als volgt uit kunnen zien:



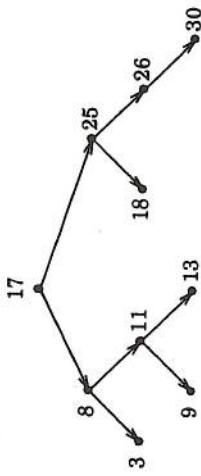
Als we vanuit een computer-programma de boom benaderen, is de wortel (hier: '17') het eerste record dat we 'tegenkomen'. Het ligt voor de hand om hier dan ook mee te beginnen met het afdrukken. Daarna kunnen we linksaf (hier: '8' afdrukken) en tenslotte rechtsaf ('25').

- We krijgen dan bij elkaar: 17, 8, 25.

Deze volgorde noemen we *preorder*, analoog aan onze eerder gevonden *prefix*-notatie in de vorige paragraaf. De preordervolgorde van een (binaire) boom verkrijgen we dus uit: *eerst de wortel, daarna linksaf, tenslotte rechtsaf*. Ingewikkelder wordt het wanneer we meer dan drie records in een boom opgeslagen hebben. Toch kunnen we, met behoud van ons algoritme, de hele boom 'doorwandelen'.

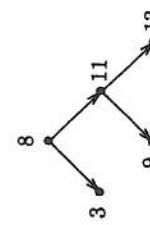
### Voorbeeld 2

$B_2$



De records die in de boom opgeslagen zijn worden hier weer voorgesteld door de sleutelwaarde. Alle records zijn *geordend* opgeslagen in onze (binaire) boom en het is dus inderdaad een □ *geordende binaire boom* (ga dit zelf na).

Vanuit een computerprogramma kunnen we altijd alleen de wortel (hier: '17') direct benaderen en via de wortel alle andere knopen. Ook hier gaan we dus eerst de wortel afdrukken. Daarna kunnen we *links* afslaan en komen dan in een subboom terecht, waar '8' de wortel van is. Om het algoritme éénduidig te houden drukken we ook hier weer eerst de wortel (van de subboom dus) af en gaan dan weer linksaf. Deze subboom ziet er als volgt uit:



Ook de inhoud van deze boom gaan we preorder uitlijsten: eerst de wortel ('8') dan linksaf ('3') tenslotte rechtsaf ('subboomje' met '11' als wortel). Merk op dat we na '3' niet verder linksaf kunnen en daarom wel rechtsaf moeten slaan. We hebben nu dus al: 17, 8, 3 afdrukt. Vanuit '8' rechtsaf staand komen we in de volgende subboom terecht (ga dit zelf na):



Ook deze subboom wordt in de volgorde wortel, links, rechts afgedrukt. Deze subboom levert dus 11, 9, 13 op.

Zo verdergaand krijgen we in totaal:  
 $17, 8, 3, 11, 9, 13, 25, 18, 26, 30$ .  
 Ga zelf na dat dit klopt! We noemen deze volgorde *preorder*, analog aan de eerder gevonden *prefix-notatie*.

**Definitie 5.12** Gegeven een *binaire boom* kunnen we de inhoud ervan *preorder* weergeven als volgt:

- *Eerst* wordt de wortel van de boom afgedrukt.
- *Daarna* gaan we de linker-subboom (*preorder*) afdrukken.
- • *Tenslotte* gaan we de rechter-subboom (*preorder*) afdrukken.

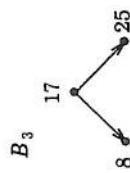
### Inorder

*Inorder*

We zien dat dit weliswaar 'werkt' (met een eenvoudig algoritme), maar de *ordening* die we in de boom aangebracht hadden is compleet zoek door onze manier van uitlijsten. Reden om een ander algoritme te zoeken dat wel de 'juiste' volgorde geeft (dat wil zeggen die de ordening in de boom ook op papier weer geeft). Om dit door een computerprogramma te kunnen laten doen, moeten we dit ook in een vast algoritme beschrijven, net zoals bij de *preorder-volgorde*.

Eerst weer het eenvoudige voorbeeld:

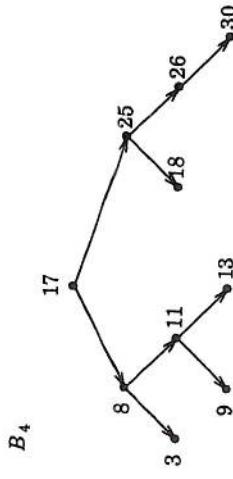
### Voorbeeld 3



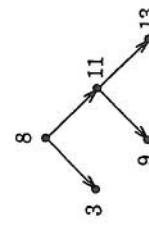
De 'juiste' volgorde is hier: 8, 17, 25.  
 Dit betekent dat we in de boom *niet eerst* de wortel moeten afdrukken, maar *eerst* de linker-knoop, *daarna* de wortel, *tenslotte* de rechter-knoop (ga na dat dit inderdaad de volgorde 8, 17, 25 oplevert).

In gewikkelder wordt het weer als we een grotere boom nemen, bijvoorbeeld die uit voorbeeld 2.

### Voorbeeld 4

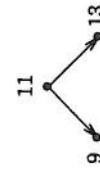


- Ook hier moeten we dus niet beginnen met de wortel, maar eerst links afslaan: we komen dan in de subboom met '8' als wortel terecht:



Ook hier moeten we niet eerst de wortel ('8') hebben, maar eerst linksaf gaan; bij '3' aangekomen kunnen we niet verder en we kunnen nu '3' afdrukken (ga *zelf* na dat dit ook *altijd* het kleinste getal zal zijn in een juist-geordende boom). Na de linker subboom (die hier slechts '3' bevat) komt de wortel, hier dus '8', tenslotte gaan we rechtsaf.

Hier komen we weer een subboompje tegen dat er zo uitziet:



Nu deze nog in dezelfde volgorde (inorder) afdrukken levert hier: 9, 11, 13.

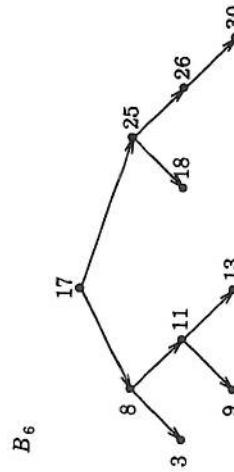
Daarna kan de wortel van de gehele boom aan de beurt komen (we hebben immers de gehele linker-subboom gehad), tenslotte gaan we de rechter-subboom van  $B_4$  in.

De gehele lijst ziet er dan als volgt uit (ga dit zelf na): 3, 8, 9, □ 11, 13, 17, 18, 25, 26, 30.

**Definitie 5.13** Gegeven een *binaire boom* kunnen we de inhoud ervan in order weergeven als volgt:

- *Eerst* gaan we vanuit de wortel de linker-subboom in en drukken die in order af.
  - *Daarna* drukken we de wortel af.
  - *Tenslotte* gaan we de rechter-subboom in en drukken die in order af.
- 

#### Voorbeeld 6

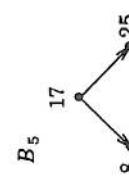


In plaats van een linkerknoop staat er weer een linker-subboom, die we dan eerst (postorder) moeten afdrukken.

**Postorder** De laatste volgorde waarin we een binaire boom kunnen uitlijsten is de zogenaamde 'postorder'-volgorde.

Eerst weer een eenvoudig voorbeeld.

#### Voorbeeld 5

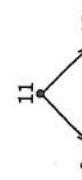


De postorder volgorde is hier: 8, 25, 17.  
We drukken daarvoor dus *eerst* de linkerknoop, *daarna* de rechterknoop en *tenslotte* de wortel af.

□

Ietwat ingewikkelder weer is het in het volgende voorbeeld (hetzelfde als voorbeeld 2 en 4).

Als we nu beginnen met de linkerknoop dan drukken we dus knoop '3' af en gaan daarna de rechter-subboom te lijf:



Deze kunnen we gewoon postorder afdrukken: 9, 13, 11.

Ga zelf na dat de postorder-volgorde van boom  $B_6$  wordt: 3, 9;  
□ 13, 11, 8, 18, 30, 26, 25, 17.

Merk op dat bij deze volgorde de wortel van de (sub) boom altijd achteraan staat. We kunnen dan ook de postorder 'van achter naar voren' schrijven, door 'eerst' (dus achteraan) de wortel van de boom af te drukken, daarna de wortel van de rechter-subboom enzovoort.

**Definitie 5.14** Gegeven een *binaire boom* kunnen we de inhoud ervan postorder weergeven als volgt:

- *Eerst* gaan we de linker-subboom in en drukken die postorder af.
- *Daarna* gaan we de rechter-subboom in en drukken die postorder af.

- • *Tenslotte* drukken we de wortel af.

#### Preorder en prefix, inorder en infix, postorder en postfix

Het zal je wellicht opgevallen zijn dat er verrassend veel overeenkomst is tussen de preorder-notatie en de prefix-notatie. Als we een geordende gelabelde (binaire) boom hebben, dan komt de preorder uitlijsting indertijd precies overeen met de erbij behorende prefix-notatie. Bij inorder en infix is dit niet zo. Het verschil zit hem hierin, dat preorder een uitlijsting definieert van *wat voor binaire boom dan ook*, terwijl de prefix-notatie alléén past bij een geordende en *gelabelde* (binaire) boom. Ofwel: preorder heeft meer te maken met het maken van een lijstje van alle records die in een boom opgeslagen zijn, maar prefix is een bepaalde schrijfwijze om een rekenkundige, wiskundige of logische expressie eenduidig vast te leggen.

#### 5.7 Recursie toegepast op bomen

Nu we in de vorige paragraaf kennis gemaakt hebben met drie in de informatica veelgebruikte 'uitlijst-methoden', gaan we eens kijken hoe we hiervoor handig een Pascal-procedure kunnen maken om het uitlijsten door de computer te laten doen.

- • Eerst definiëren we een datatype voor een wijzer, die we in de te behandelen procedures zullen gebruiken:

```
type wijzer = ^rec;
rec = record
  sleutel : integer;
  data : char;
  linkerzoon : wijzer;
  rechterzoon : wijzer;
end;

procedure preorder (wortel: wijzer);
begin
  if wortel <> NIL then
    begin
      writeln (wortel.data);
      preorder (wortel^.linkerzoon);
      preorder (wortel^.rechterzoon);
    end;
end;
```

We zien dat dit een recursieve procedure is. Met een recursieve (Pascal) functie hebben we al eerder kennis gemaakt. Ga na dat deze procedure ook werkt als er (nog) geen boom is (zoals dat hoort bij netjes programmeren!) en dat hij inderdaad de juiste (preorder) volgorde van een boom geeft.

#### Toepassing van preorder, inorder en postorder

Preorder-uitlijsting vindt een belangrijke toepassing bij zogenaamde hiërarchische databases: in zulke databases ligt het alsof de records in een boom staan, maar in werkelijkheid staan ze (op schijf) in preorder volgorde.

## 5.10 Opgaven

5.2.1 Teken de volgende grafen:

- a.  $G_1 = (K_1, T_1)$  met  $K_1 = \{x, y, z\}$  en  
 $T_1 = \{\{x, y\}, \{y, y\}, \{z, y\}\}$
- b.  $G_2 = (\{a, b, c, d\}, \{\{a, b\}, \{a, c\}, \{d, a\}, \{d, c\}\})$
- c.  $G_3 = (\{a, b, c, d\}, \{(a, b), (a, c), (d, a), (d, c)\})$
- d.  $G_4 = (\{a, b, c, d\}, \{(a, b), (a, c), (d, a), (c, d)\})$

5.2.2 De volgende vragen gaan over de grafen uit de vorige opgave.

- a. Geef bij elke graaf een pad (geef aan of het pad gericht is of ongericht).
- b. Hoe groot is de grootst mogelijke padlengte bij  $G_2$ ? Leg uit.
- c. Geef bij elke graaf een (het?) langst mogelijke pad dat aan de volgende eis voldoet: een knoop mag *maximaal één keer bezocht* worden, met als uitzondering de begin- en eindknoop: deze *mogen dezelfde zijn*. (zo'n pad noemen we *enkelvoudig*, Engels: 'simple path').

5.2.3 Geef een voorbeeld van een verbonden (ongerichte) graaf van vier knopen, waarbij een niet-verbonden graaf ontstaat door één tak (willekeurig welke) weg te halen.

5.2.4 Geef een voorbeeld van een volledig verbonden (gerichte) graaf van vier knopen, waarbij een niet-volledig verbonden graaf ontstaat door één tak (willekeurig welke) weg te halen.

5.2.5 Gegeven is de (binaire) relatie  $R$  op verzameling  $A = \{1, 2, 3, 4, 5\}$  met  $x R y \Leftrightarrow x \leq y$ .

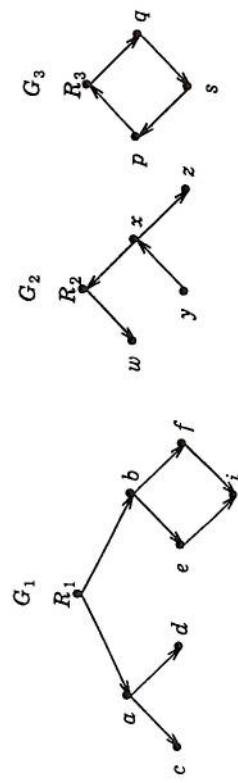
- a. Geef  $R$  weer als verzameling geordende paren.
- b. Teken de bijbehorende graaf.

5.3.1 Van een boom is gegeven dat hij dertig pijlen heeft. Hoeveel knopen heeft hij?

### 5.3.2 Beantwoord de volgende vragen:

- a. Kan een boom een volledig verbonden (gerichte) graaf zijn?  
Geef een korte uitleg bij je antwoord.
  - b. Iemand beweert dat aan de definitie van een boom (zie definitie 5.5) moet worden toegevoegd:
    - Er is een gericht pad van de wortel naar *elke* andere knoop.  
Ga na dat deze eis niet iets wezenlijks *toenegt*. (dus: ga na dat deze eis eigenlijk wordt geimpliceerd door de eisen die in definitie 5.5 staan). Geef een korte uitleg bij je antwoord.
    - c. Iemand anders beweert dat de definitie uit het boek onjuist is en dat ze moet zijn als volgt:  
Een boom is een gerichte graaf met als extra eisen:
      - Er is precies één knoop met ingraad 0. Deze knoop noemen we de *wortel*.
      - Alle andere knopen hebben ingraad 1.
      - Er is een gericht pad van de wortel naar elke andere knoop.
- Ga na dat deze definitie niet *fout* is, alleen maar *anders*, dus dat ook bij deze definitie elke boom voldoet. Geef een korte uitleg bij je antwoord.

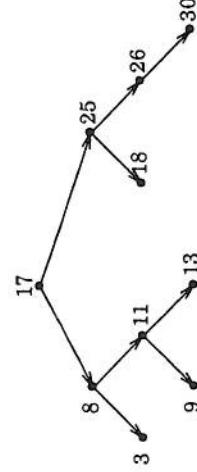
- d. Weer iemand anders beweert dat er bomen zijn waarbij sommige knopen vanuit de wortel te bereiken zijn via twee of meer *verschillende* paden.  
Ga na dat deze uitspraak *onjuist* is, dus dat elke graaf waarbij dit wel mogelijk is, *geen* boom kan zijn. Geef een korte uitleg aan de hand van een zelfverzonnen voorbeeldje.
- 5.3.3 Geef bij elk van de onderstaande grafen aan of het een boom is of niet.**  
Zo ja: geef de wortel.  
Zo neen: leg uit waarom niet (noem alle, indien er meerdere redenen zijn).



- 5.3.4 a. Teken (zo mogelijk) de volgende boom:**  
 $B = ([1, 2, 3, 4, 5, 6], \{(2, 1), (2, 3), (4, 2), (4, 6), (6, 5)\})$ .
- b. Wat is volgens jou de wortel van de bij a gevonden boom (zo eer een is)?
- c. Geef het langste gerichte pad. Geef ook de padlengte.
- d. Hoe lang is het langste on-gerichte pad? Leg uit.
- e. Is deze boom binair? Geef de nodige uitleg.
- f. Is deze boom geordend? Geef de nodige uitleg.
- g. Is deze boom compleet? Geef de nodige uitleg.

5.4.1 Gegeven is de volgende boom:

$$B_1 = (K_1, P_1)$$



- a. Geef alle knopen van nivo 2.
- b. Geef de diepte (hoogte) van de boom.
- c. Teken alle mogelijke subbomen van de gegeven boom.
- d. Wie is de grootvader van 13? En wie is de vader van 25?
- e. Geef de opvolgers van knoop 8.
- f. Welke knopen zijn de bladeren? En welke de interne knopen?
- g. Is deze boom gebalanceerd? Geef de nodige uitleg hierbij.
- h. Is deze boom geordend? Geef de nodige uitleg hierbij.
- i. Is deze boom compleet? Geef de nodige uitleg hierbij.
- 5.4.2 a. Kun je een perfect gebalanceerde (binaire) boom construeren met precies acht knopen erin?  
Zo ja: doe dat.  
Zo neen: construeer een zo klein mogelijke perfect gebalanceerde (binaire) boom die tenminste acht knopen heeft.
- b. Kun je een gebalanceerde (binaire) boom construeren met precies acht knopen erin?  
Zo ja: doe dat.  
Zo neen: construeer een zo klein mogelijke gebalanceerde (binaire) boom die tenminste acht knopen heeft.
- c. Geef de rij getallen (zo die er is) die voorstelt: alle aantallen knopen die perfect gebalanceerde (binaire) bomen kunnen hebben.
- d. Geef de rij getallen (zo die er is) die voorstelt: alle aantallen knopen die gebalanceerde (binaire) bomen kunnen hebben.