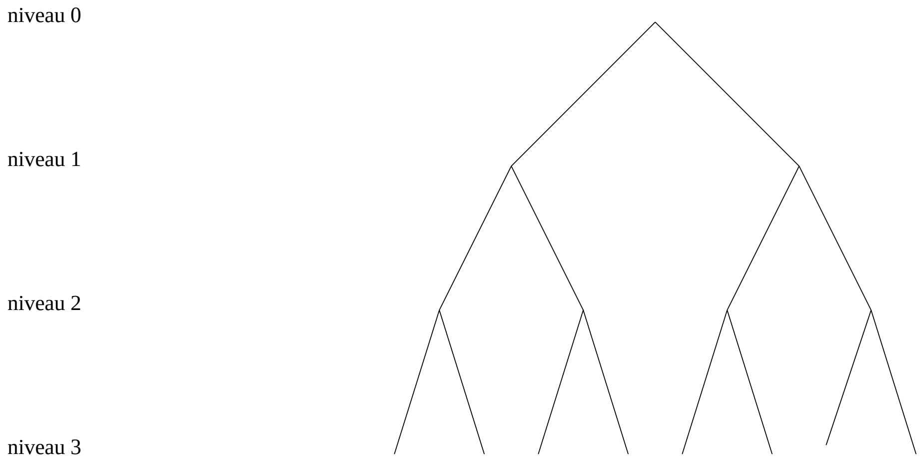
Hoofdstuk 2

Inleiding tot de grafentheorie

2.6 Bomen

2.6.1 Definitie

boom = eindige geconnecteerde graaf zonder cykels.



Figuur 2.22: Een boom

2.6.2 Elementaire eigenschappen

(1) Boom = altijd geconnecteerd

tussen twee knopen steeds (uniek) pad

(2) (Oefening)

1. G is een boom.
2. G is geconnecteerd en $|E(G)| = |V(G)| - 1$.
3. G bevat geen cykels en $|E(G)| = |V(G)| - 1$.
4. Tussen elk tweetal knopen x en y van G bestaat precies één pad.
5. G bevat geen cykels en toevoegen van tak creëert precies één cykel in G.

2.6.3 Ordening in een boom

Bomen T = grafen met partiële ordening:

een grootste element = *wortel*

minimale elementen = *bladeren*

voor elke $x \in T$, $\{y \in T||x \leq y\}$ eindig, totaal geordend

2.6.4 Voorstelling van boom

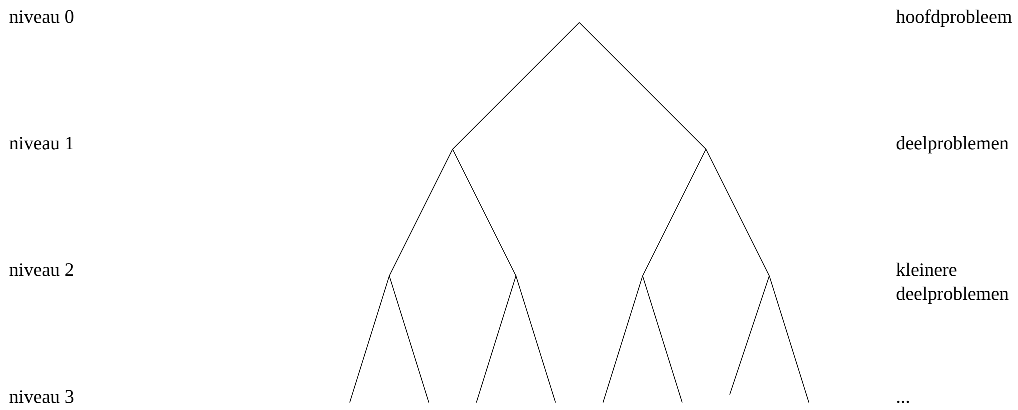
(1) niveau nul: wortel

niveau 1: elementen adjacent met wortel

niveau 2: elementen adjacent met elementen op niveau 1,...

Onderaan: bladeren

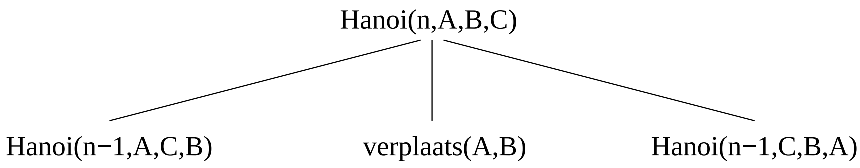
(2) (2.1) oplossingen voor problemen



Figuur 2.23: Splitsing in deelproblemen

2.6.5 Voorbeeld

Beschrijving van het algoritme voor de torens van Hanoi



Figuur 2.24: Hanoi(n, A, B, C)

(2.2) nuttig voor zoekalgoritmes: *backtracking*.

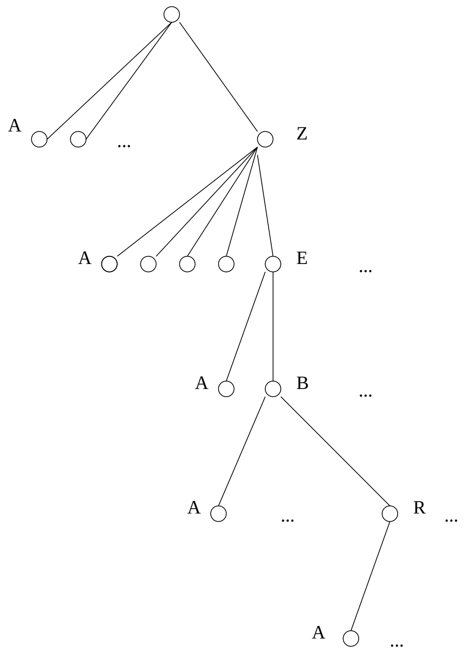
(2.3) vinden efficiënte zoekalgoritmes

2.6.6 Voorbeeld

Vinden van woord in woordenboek.

Woorden vijf letters

Gebruikelijke methode: eerste correcte letter vinden, dan tweede correcte letter, ...



Figuur 2.26: Voorstelling van zoeken naar woord zebra

2.6.7 Binaire bomen

binaire boom = elke knoop adjacent met hoogstens twee knopen op volgend niveau.

2.6.8 Zoekalgoritmes en bomen

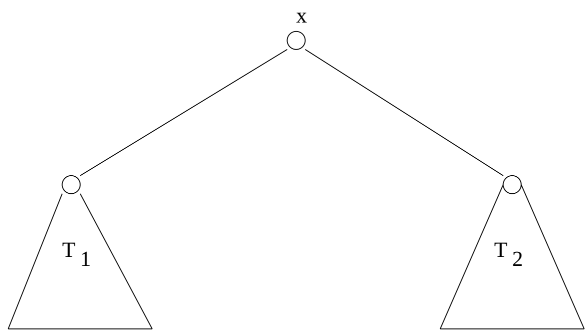
Complexiteit van zoekalgoritmes

Eindige binaire bomen: recursief gedefinieerd/geconstrueerd

één enkele knoop x, $x \leq x$

en

twee disjuncte binaire bomen T_1 en T_2



Figuur 2.27: Recursieve structuur van binaire boom

Partiële ordening op $T_1 \cup T_2 \cup \{x\}$

$$
\begin{aligned}
y &\leq x, \text{ voor alle } y \in T_1 \cup T_2, \\
y &\leq z, \text{ als } y \text{ en } z \text{ tot dezelfde boom } T_i \text{ behoren en de} \\
&\quad \text{ongelijkheid } y \leq z \text{ geldig is binnen } T_i.
\end{aligned}
$$

x is grootste element van $T_1 \cup T_2 \cup \{x\}$;

(1) **in-order**

1. doorloop linkse subboom T_1;
2. passeer de wortel x;
3. doorloop de rechtse subboom T_2.

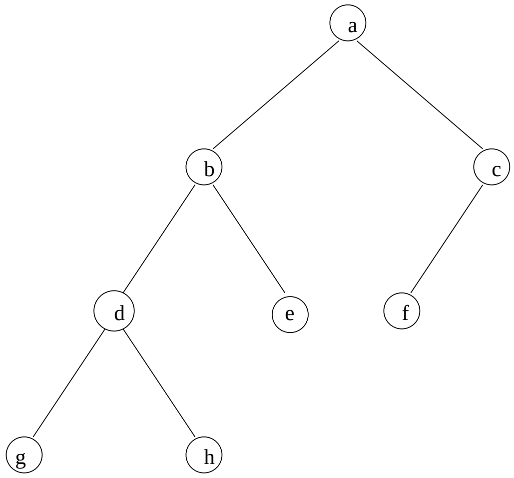
(2) **pre-order**

1. passeer de wortel x;
2. doorloop linkse subboom T_1;
3. doorloop de rechtse subboom T_2.

(3) **post-order**

1. doorloop linkse subboom T_1;
2. doorloop de rechtse subboom T_2;
3. passeer de wortel x.

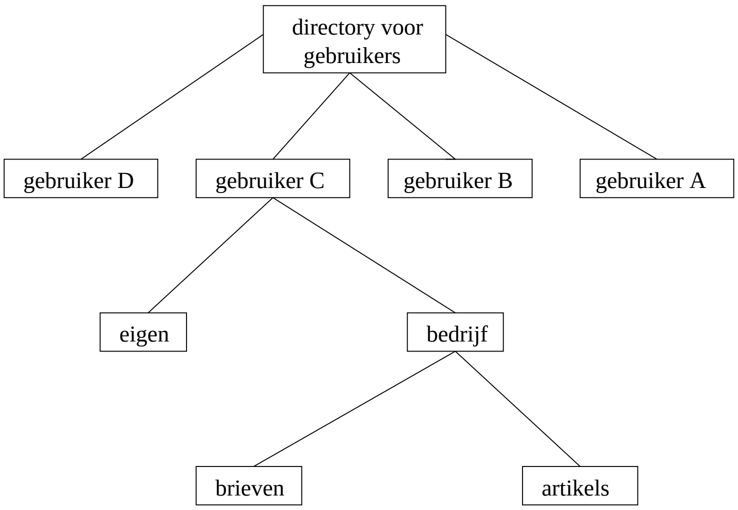
2.6.9 Voorbeeld



Figuur 2.28: Doorlopen van een boom

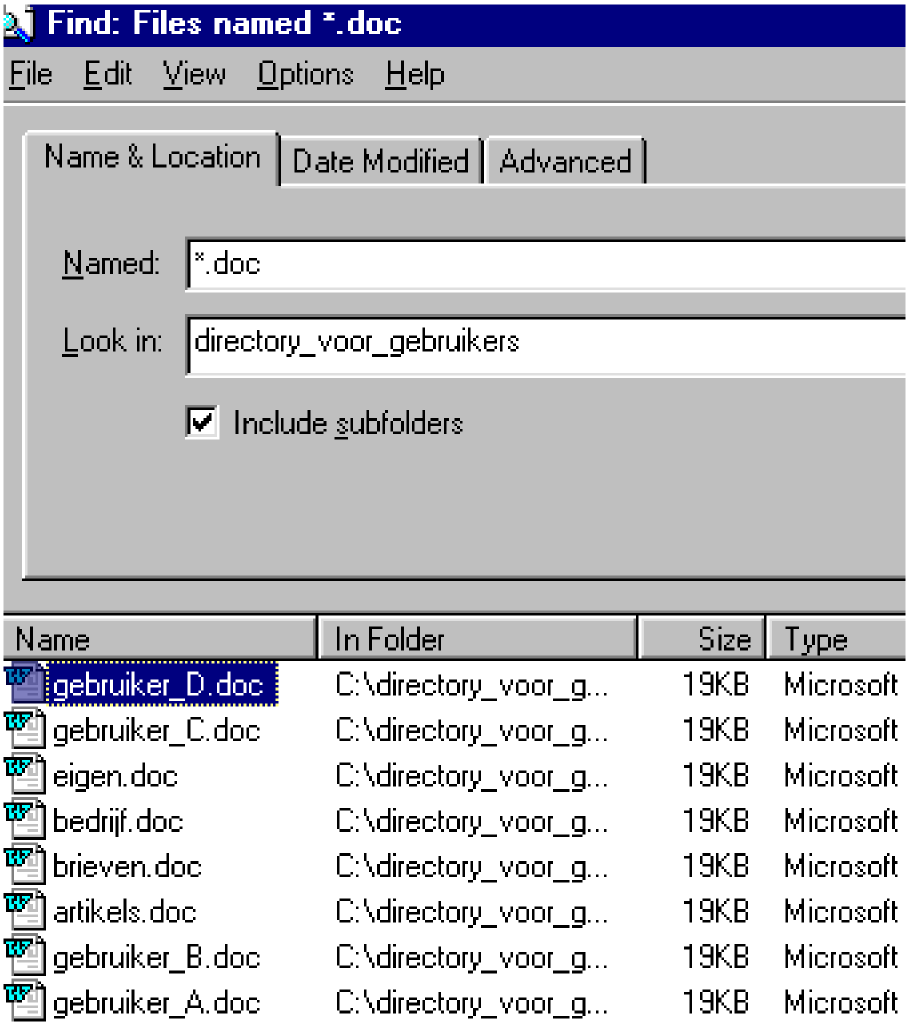
in-order:	$g\ d\ h\ b\ e\ a\ f\ c$
pre-order:	$a\ b\ d\ g\ h\ e\ c\ f$
post-order:	$g\ h\ d\ e\ b\ f\ c\ a$

2.6.10 Doorlopen van directories



Figuur 2.29: Directory structuur

Directory structuur: in pre-order doorlopen



Figuur 2.30: Doorzoeken van directories

2.7 Systematisch zoeken in graaf

2.7.1 Depth-first search

zo ver mogelijk geraken in graaf en eventueel terugkrabbelen

2.7.2 Breadth-first search

- Kies eerste knoop r

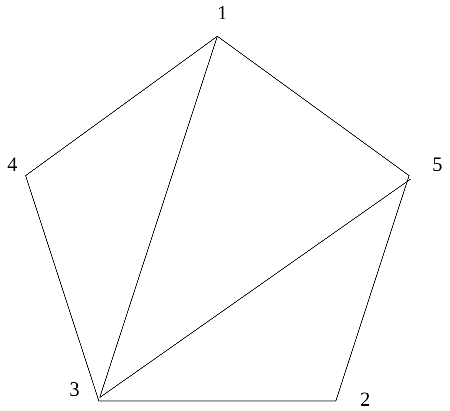
  - Controleer alle knopen $s_1, \dots$ adjacent met r

  - Controleer alle knopen adjacent met $s_1, \dots$

  - ...

  Nadeel: Veel tussenresultaten opslaan

2.7.3 Voorbeeld: Hamiltoniaans pad



Figuur 2.17: Hamiltonpad

- eerste knoop: 1

- twee knopen: 14, 13, 15

- drie knopen: 143, 134, 135, 132, 153, 152

- vier knopen: 1435, 1432, 1352, 1325, 1532, 1534, 1523

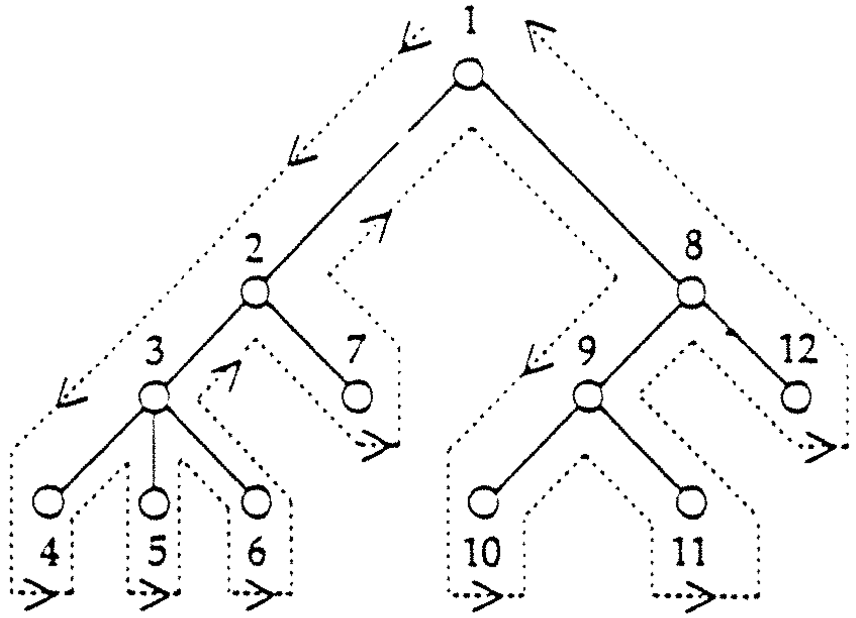
- vijf knopen: 14352, 14325, 15234

2.7.4 Systematisch zoeken in bomen

op unieke wijze van elke knoop naar de wortel (backtracking).

(1) *breadth-first search*: knopen niveau per niveau doorlopen. Eerst de wortel, dan alle knopen op niveau 1, dan alle knopen op niveau 2,...

(2) *depth-first search*: boom doorlopen in pre-order.



Figuur 2.31: Depth-first search in boom

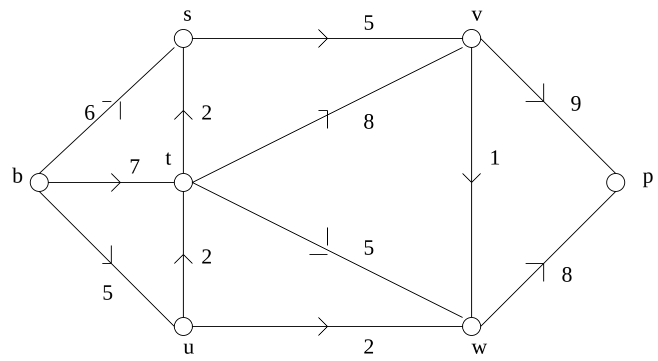
2.8 Netwerken

2.8.1 Definitie

netwerk $N = (T, B, c)$ = gerichte graaf $G = (V(G), E(G))$

elke pijl (x, y): positief getal $c(x, y)$ = *capaciteit* pijl

2.8.2 Voorbeeld



Figuur 2.32: Een netwerk