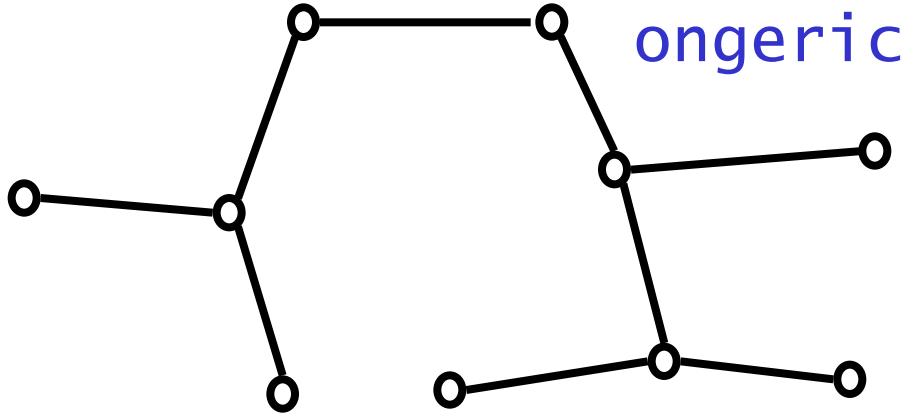
10 Bomen

deel 2

Tiende college

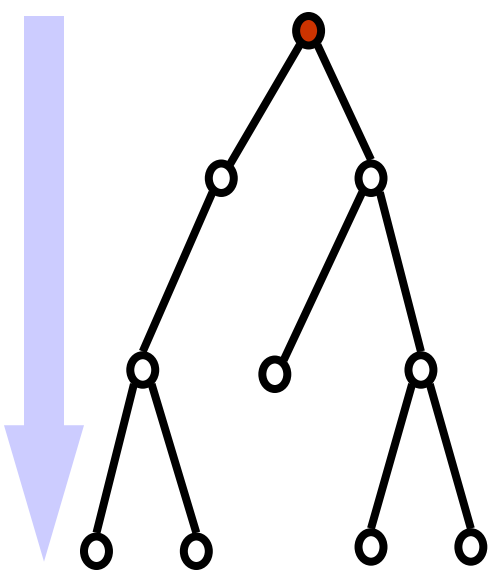
§8.8 ongerichte bomen
§9.4 gerichte bomen
ch 10. binaire bomen

arboretum

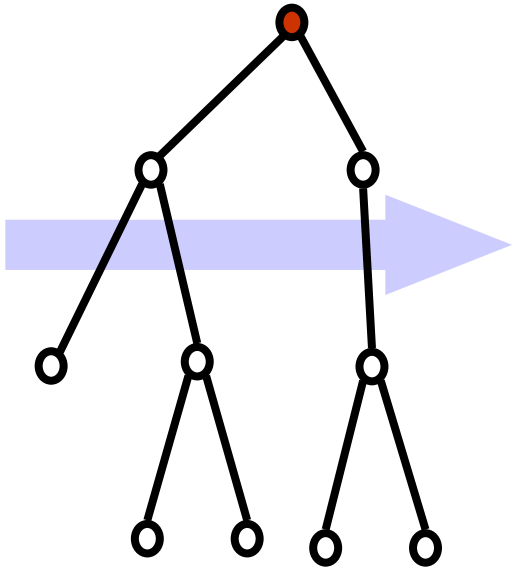


ongericht

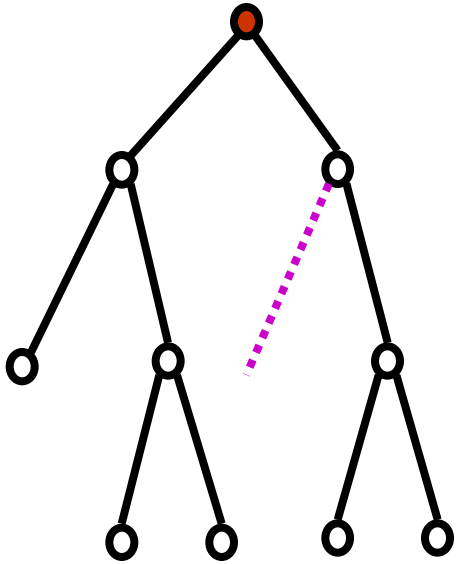
§8.8 tree graphs
§9.4 rooted trees
ch.10 binary trees



gericht



geordend



links/rechts

bomen

Een *boom* is een *samenhangende* ongerichte graaf zonder cykels

Een *bos* is een ongerichte graaf zonder cykels

Een *gewortelde boom* heeft een speciale knoop (de *wortel*), en daarmee een *richting* (vanuit de wortel)

Bij een *geordende gewortelde boom* zijn de kinderen van elke knoop geordend: eerste (oudste) kind, tweede kind, etc.
Recursieve definitie: zie §10.9

Een *geordend bos* (forest) is een geordende collectie geordende gewortelde bomen

eigenschappen

Voor een boom $T = (V,E)$ geldt:

- tussen elk tweetal knopen is er *één* simpel pad

- het aantal takken is één minder dan het aantal knopen: $|E| = |V|-1$

(bewijs met inductie: zie college 9)

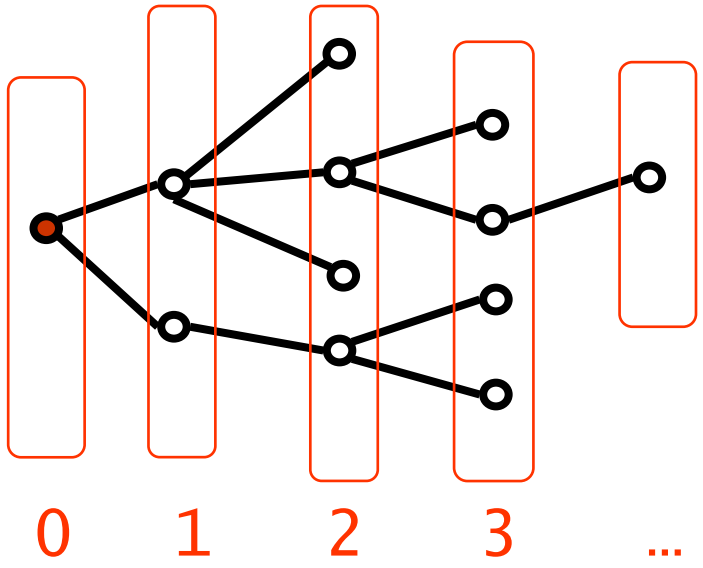
- T heeft ten minste twee knopen van graad 1

(mits T ten minste één tak heeft)

stelling

Het aantal takken van een [ongerichte] boom T = (V,E) is één minder dan het aantal knopen van T: |E| = |V|-1.

Bewijs (alternatief)



Kies een willekeurig beginknoop, dan alle buren, etc.
Acyclisch, dus we vinden nooit eerdere bezochte knopen
Elke knoop heeft één inkomende tak, behalve …

exercise 8.38

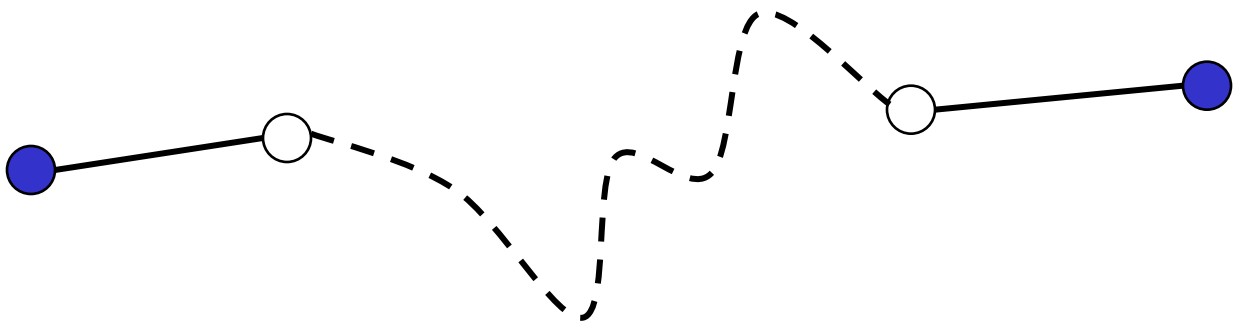
In het boek van Schaum wordt voor de stelling van de vorige sheets een inductief bewijs gegeven dat gebruikmaakt van de stelling die gegeven wordt in exercise 8.38.
Dat levert een inductiestap op waarbij de situatie voor n knopen wordt teruggebracht tot n-1 knopen. Zie Schaum, p183, exercise 8.14.

eigenschap

(Exercise 8.38)

Een (samenhangende) graaf zonder cykels met ten minste één tak heeft ten minste twee knopen van graad 1

Bewijs

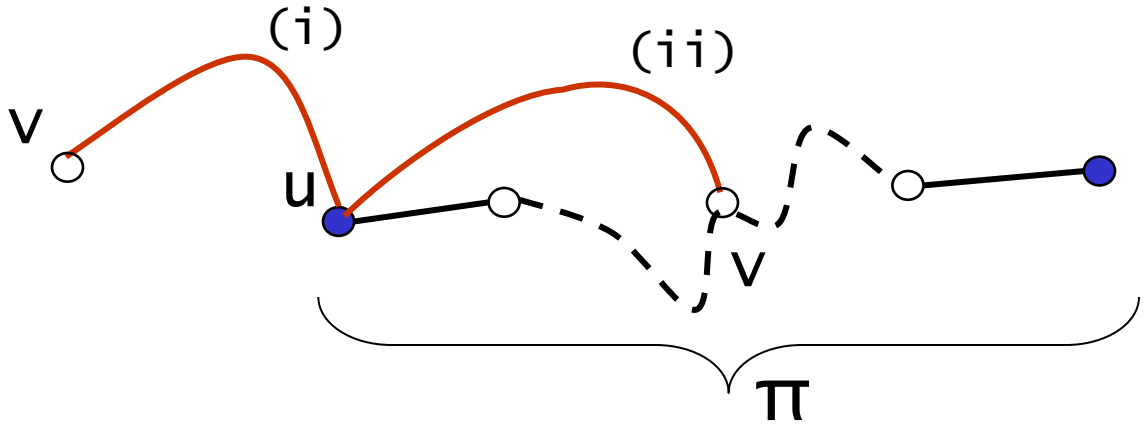


maximaal simpel pad

alternatief: graden tellen (som van de graden is …)

bewijs Ex.8.38

Een samenhangende graaf zonder cykels met ten minste één tak heeft ten minste twee knopen van graad 1



Bewijs.
Neem een maximaal simpel pad π in graaf G en laat u een eindpunt van π zijn. Heeft knoop u nog andere buren? Nee: want dan zou òf (i) het pad langer worden, òf (ii) er zou een cykel ontstaan. Dat kan allebei niet, dus: u heeft graad 1 □

alternatief bewijs Ex.8.38

Een samenhangende graaf zonder cykels met ten minste één tak heeft ten minste twee knopen van graad 1

Graden tellen: *bewijs uit het ongerijmde* (hiervoor moet al wel bewezen zijn dat voor bomen |E| = |V|-1; bijv. via slide 5)

Neem aan: G=(V,E) heeft maximaal één knoop van graad 1 (en geen knopen van graad 0)

Laat σ som van de graden van G, dan geldt

(1) ten minste $\sigma \geq 1 + 2(|V|-1)$

(2) stelling $\sigma = 2|E|$

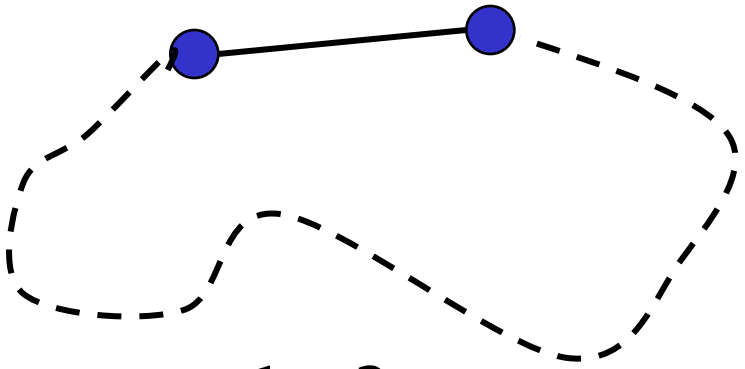
(3) voor bomen $|E| = |V|-1$

$2|E| = \sigma \geq 1 + 2(|V|-1) = 1+2|E|$ tegenspraak

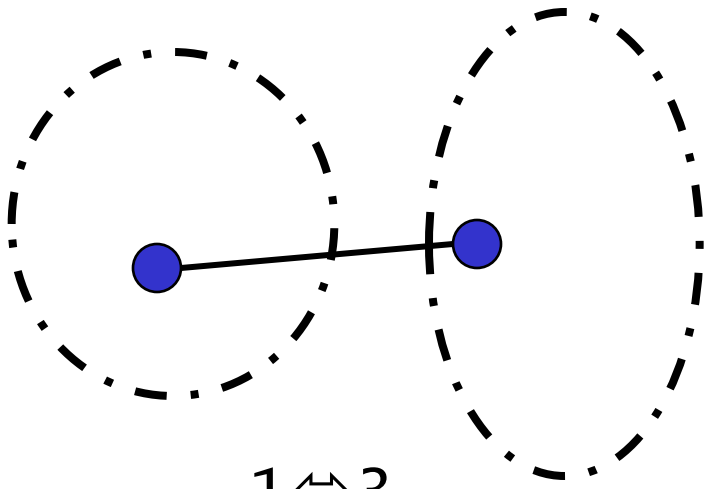
Dus moeten er (ten minste) twee knopen van graad 1 zijn.

karakterisatie I

1. (ongerichte) boom
   samenhangend, acyclisch
2. maximaal acyclisch
   lijn erbij ⇒ cykel
3. minimaal samenhangend
   lijn weg ⇒ onsamenhangend



1⇔2

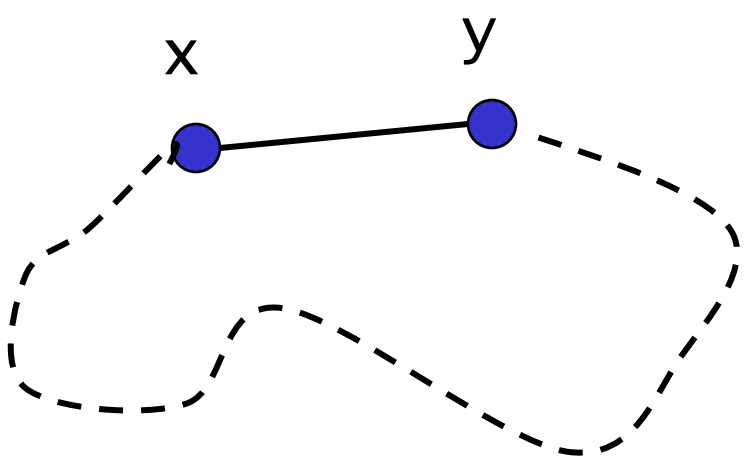


1⇔3

(Exercise 8.13)

bewijs 1⇔2

1. boom
   samenhangend, acyclisch
2. maximaal acyclisch
   lijn erbij ⇒ cykel
3. minimaal samenhangend
   lijn weg ⇒ onsamenhangend



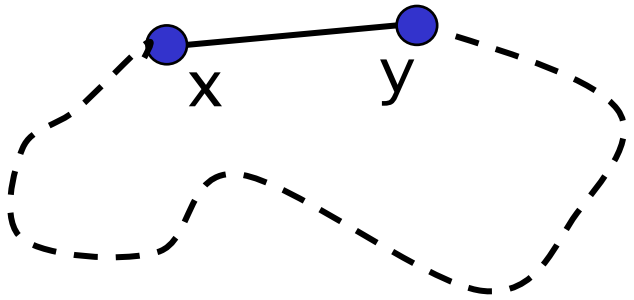
1⇒2
maximaal
2⇒1
samenhangend

bewijs 1⇔2

1. boom — samenhangend, acyclisch
2. maximaal acyclisch — tak erbij ⇒ cykel

1⇒2 Voeg lijn {x,y} toe aan G. Omdat G samenhangend is bestond er al een (simpel) pad tussen x en y. Samen met de lijn ontstaat een cykel.

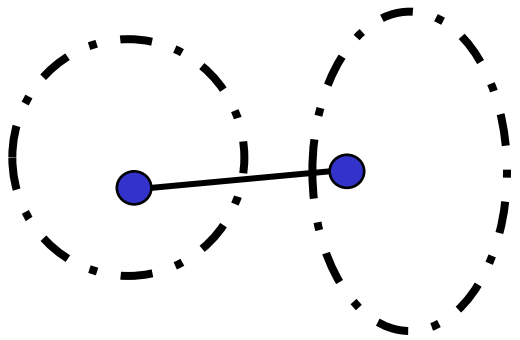
2⇒1 We moeten nog laten zien dat G samenhangend is. Kies willekeurige x en y. We beredeneren dat x en y verbonden zijn.
Als er een lijn tussen x en y zit, dan zijn ze verbonden. Als er geen lijn is voeg die dan toe. Volgens het gegeven ontstaat een cykel. Dat betekent dat er al een pad tussen x en y moest bestaan.



bewijs 1⇔3

zie college 9

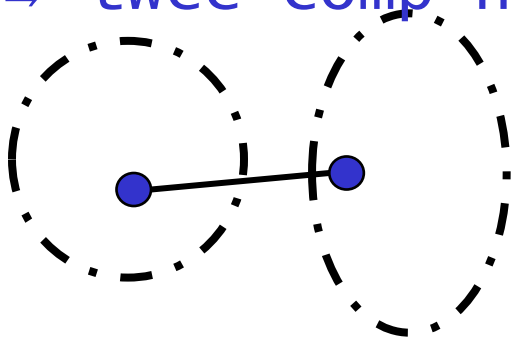
1. boom
   samenhangend, acyclisch
2. maximaal acyclisch
   lijn erbij ⇒ cykel
3. minimaal samenhangend
   lijn weg ⇒ onsamenhangend



1⇒3
   minimaal
3⇒1
   acyclisch

bewijs 1⇔3

1. boom — samenhangend, acyclisch
3. minimaal samenh. — lijn eraf ⇒ twee comp'n



1⇒3 Verwijder lijn {x,y} uit G. Nu is G niet langer samenhangend, want x en y zijn niet meer verbonden via een pad: immers, anders zou er oorspronkelijk een cykel geweest zijn.
3⇒1 We moeten nog laten zien dat G acyclisch is. Stel G bevat wél een cykel. Verwijderen van een lijn in de cykel levert nog steeds een samenhangende graaf, dus de graaf is dan niet minimaal samenhangend.
Tegenspraak. G heeft dus geen cykels.

stelling

Gevolg van het voorgaande:
Een samenhangende graaf met n knopen heeft ten minste n-1 takken.

Immers, men kan uit G alle takken verwijderen die de samenhangendheid niet verstoren. Zo krijgt men een minimaal samenhangende graaf H met n knopen. Volgens de voorgaande karakterisatie is H dus een boom, en heeft derhalve n-1 takken. Dus: #takken(G) ≥ #takken(H) = n-1.

karakterisatie II

*definitie: een *boom* is een samenhangende ongerichte graaf zonder cykels

*eigenschap: T = (V,E) dan |E| = |V|-1

Theorem 8.6 (Exercise 8.14)

G is een graaf met n knopen.
De volgende beweringen zijn equivalent:
(i) G is een boom (samenhangend, zonder cykels)
(ii) G is zonder cykels, heeft n-1 takken
(iii) G is samenhangend, heeft n-1 takken

boom, kies twee uit:
- G is zonder cykels
- G is samenhangend
- G heeft n-1 takken

Theorem 8.6

Schaum gebruikt inductie om Theorem 8.6 te bewijzen. Dit is niet nodig als we al los van Thm 8.6 bewezen hebben dat een boom met n knopen n-1 takken heeft (en als we weten dat een samenhangende graaf altijd *ten minste* n-1 takken heeft).

(i) $\Rightarrow$ (ii) duidelijk

(ii) $\Rightarrow$ (iii) gebruik dat #takken $T_i = n_i - 1$, want T_i is een boom

(iii) $\Rightarrow$ (i) H samenhangend, dus #takken H $\geq$ n-1 enerzijds, en anderzijds #takken H = #takken G - 1 = n-2. Tegenspraak

Bekijk het bewijs uit het boek, pagina 183, zelf!!

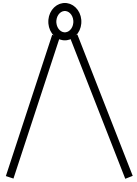
Zie ook **opgave 53**!

10

binaire bomen

binaire bomen

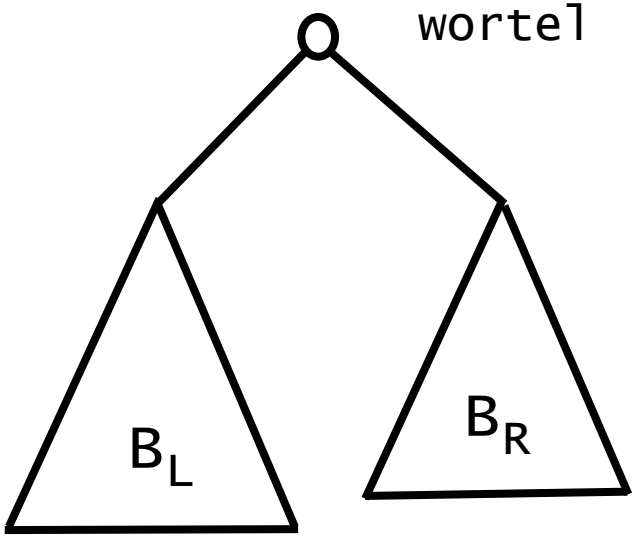
Een *binaire boom* B is
(1) leeg
of
(2) bestaat uit een speciale knoop (de *wortel*) en een linkersubboom B_L en een rechtersubboom B_R (disjunct) die beide ook binaire bomen zijn



links/rechts

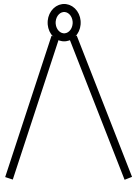
recursie

B =



binair

Links en rechts is belangrijk in binaire bomen: zelfs als er maar één kind is maakt het uit of dat links dan wel rechts zit.



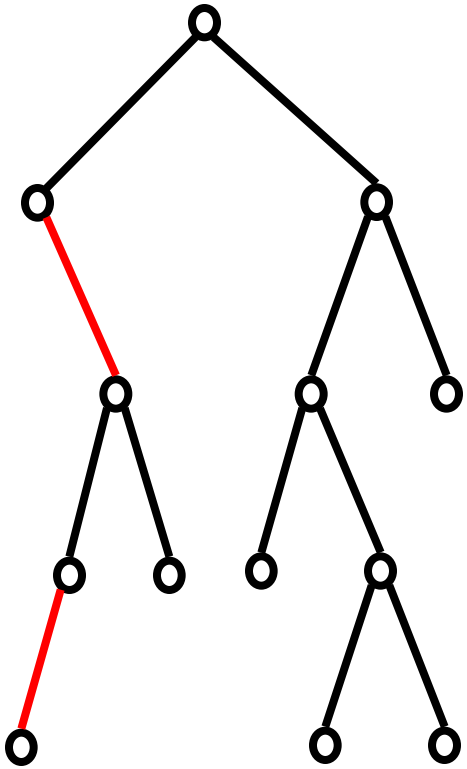
links/rechts

Het begrip is natuurlijk binnen de informatica want past bij een standaard implementatie van bomen met pointers.

Veelgebruikt. Bijvoorbeeld binaire zoekbomen.

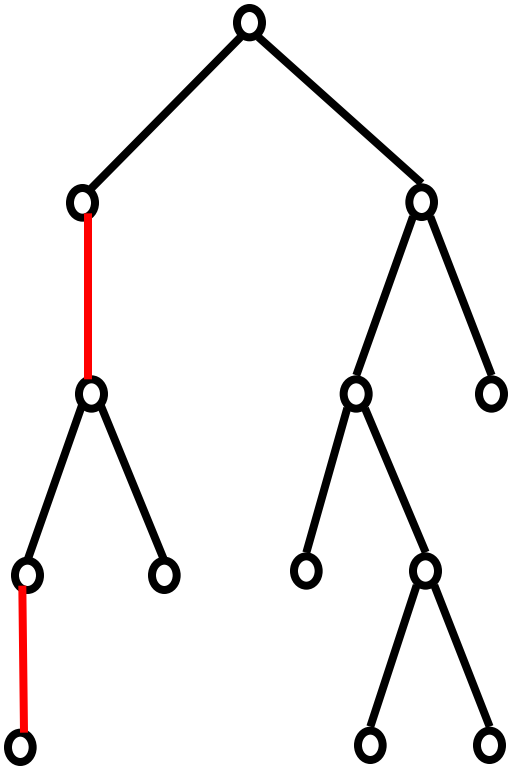
één kind

linkerkind of
rechterkind



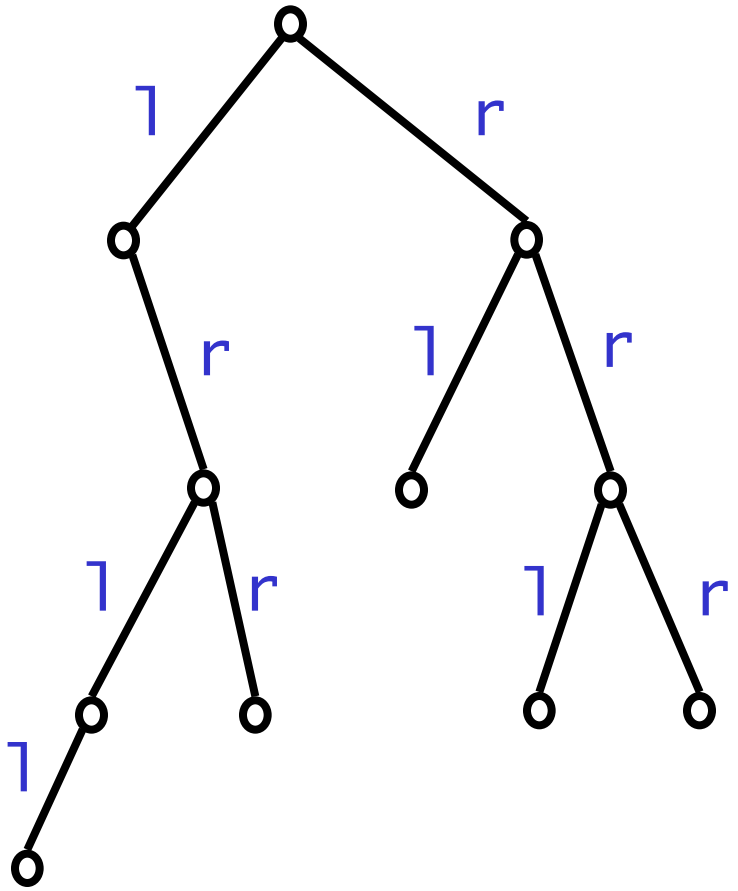
binaire
boom

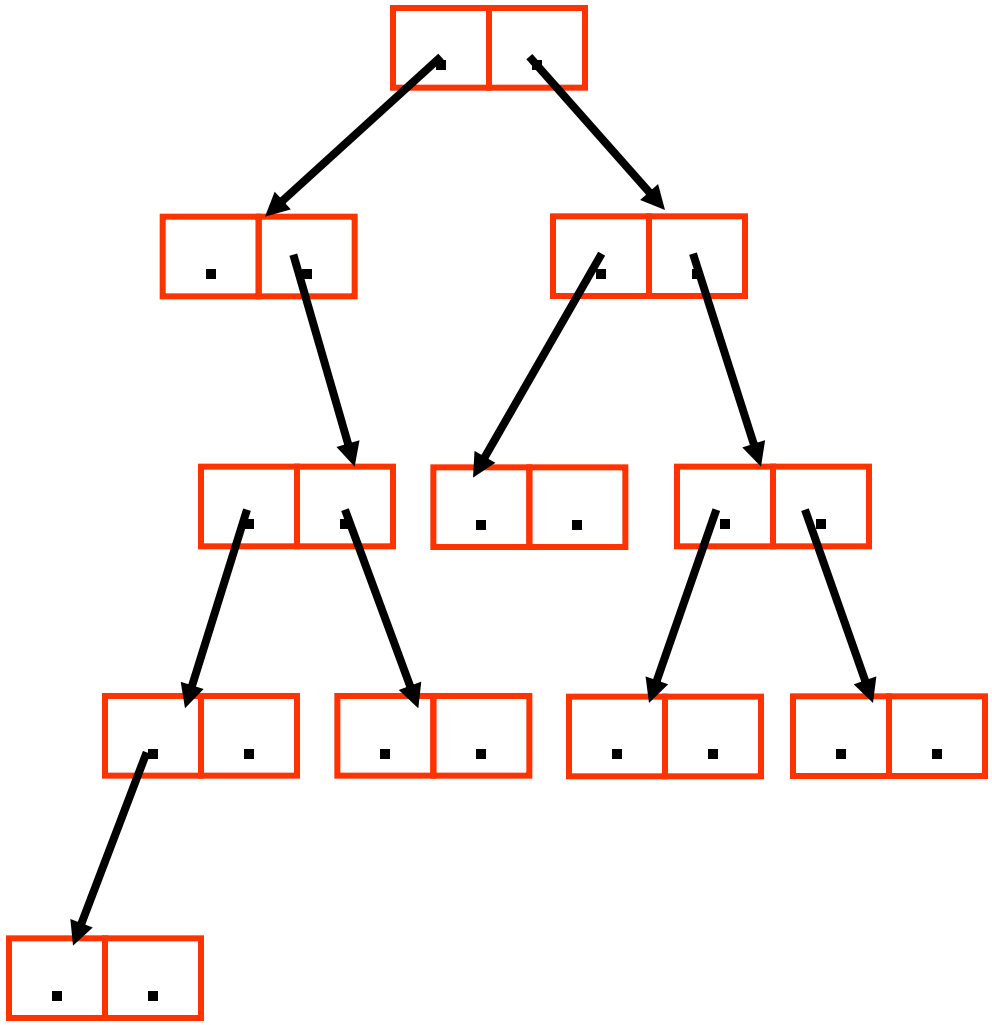
één kind



geordende
gewortelde boom

§10.4 binaire boom: pointerstructuur

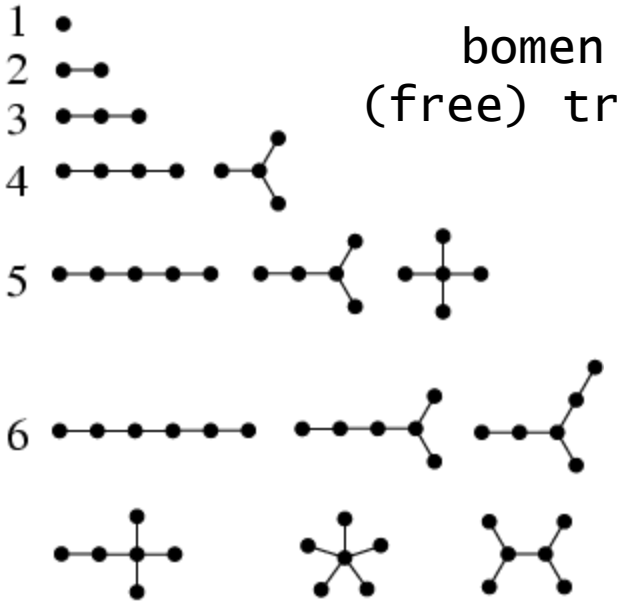




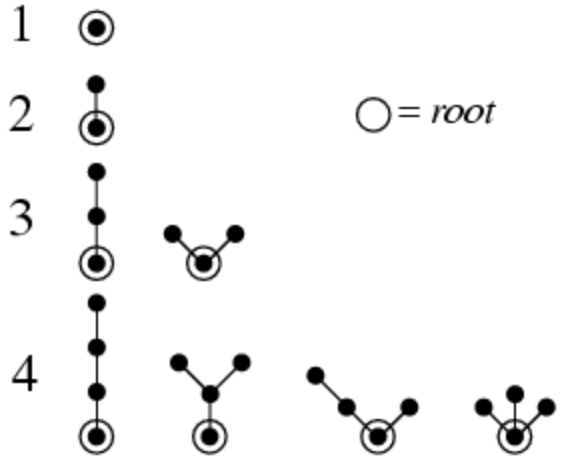
classificatie bomen

tree (graph)	boom (als graaf) ongerichte boom
rooted tree	gewortelde boom georiënteerde boom gerichte boom (wortel naar bladeren)
ordered rooted tree	geordend en gericht (standaard (stam)boom ?)
binary tree	binaire boom links en rechts (informatica!)

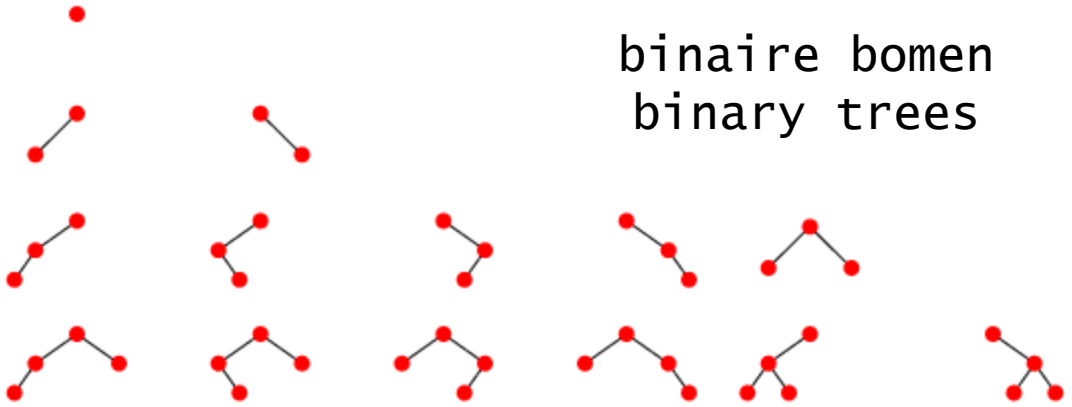
bomen (trees)



bomen
(free) trees



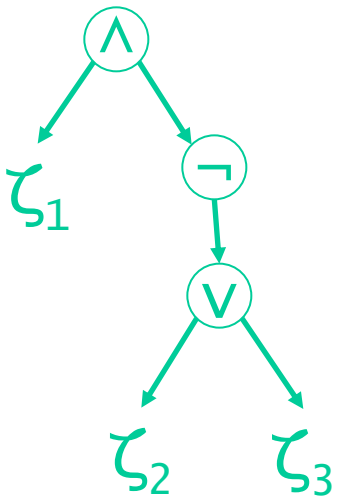
gewortelde bomen
rooted trees



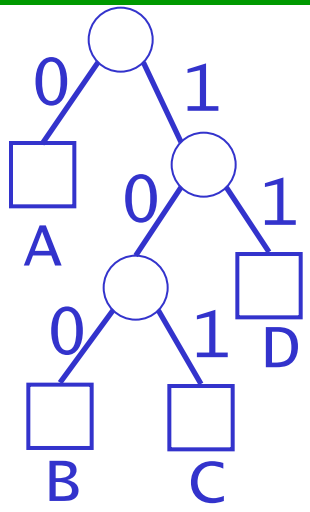
binaire bomen
binary trees

http://mathworld.wolfram.com/

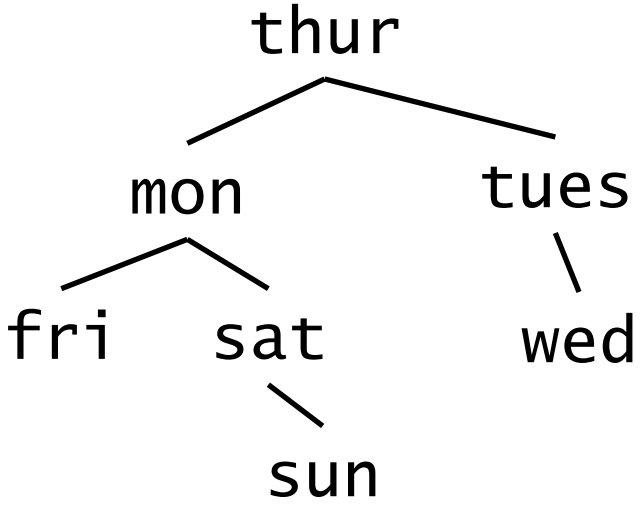
bomen, bomen, bomen …



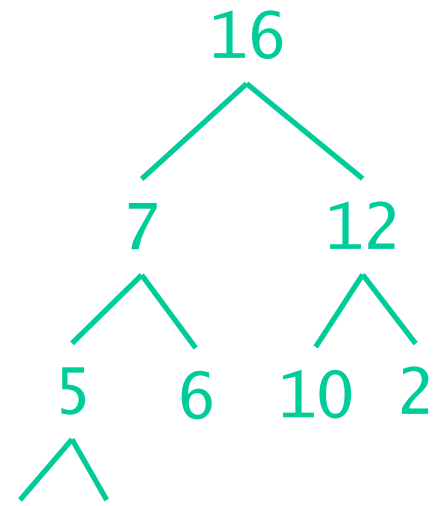
expressie



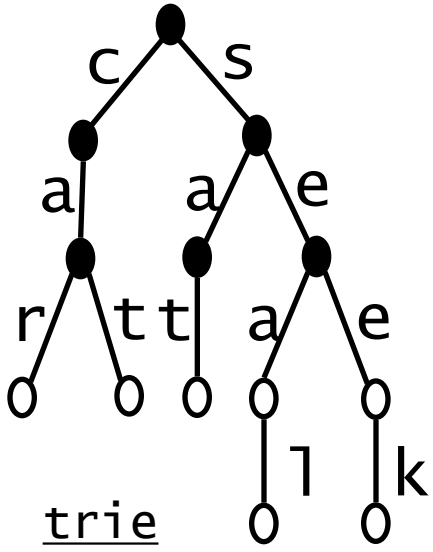
code



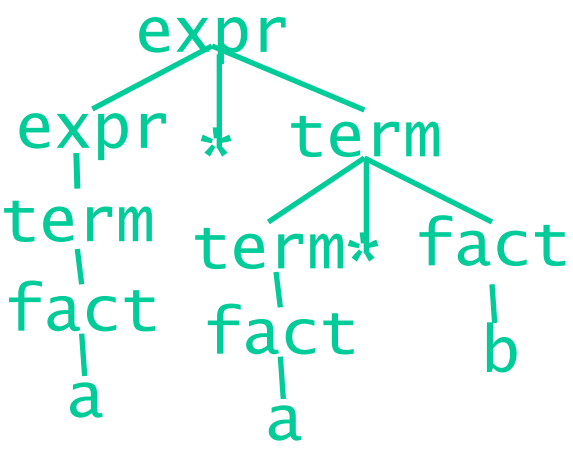
binaire zoekboom



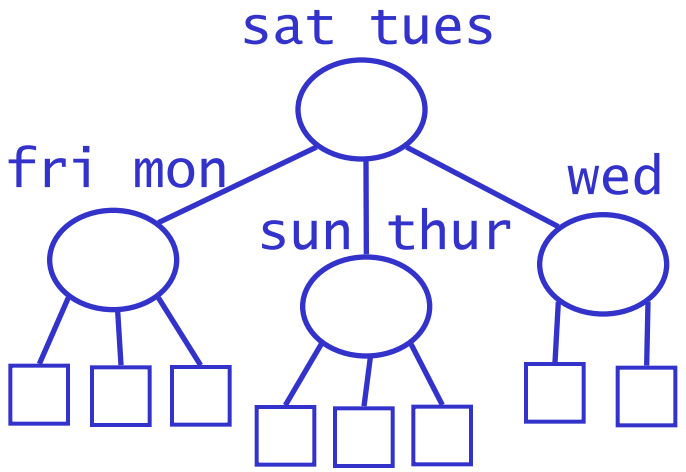
heap



trie



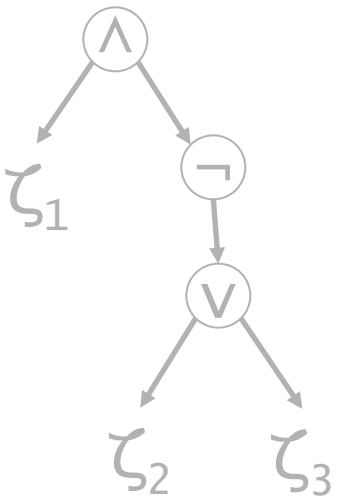
syntax



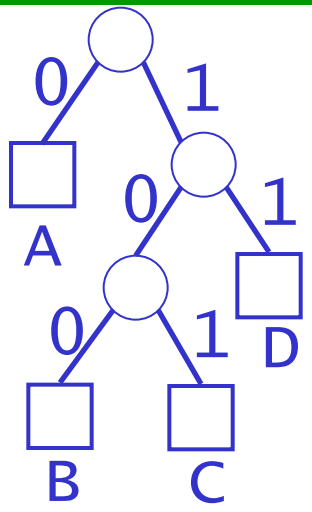
2,3 boom

en nog veel meer

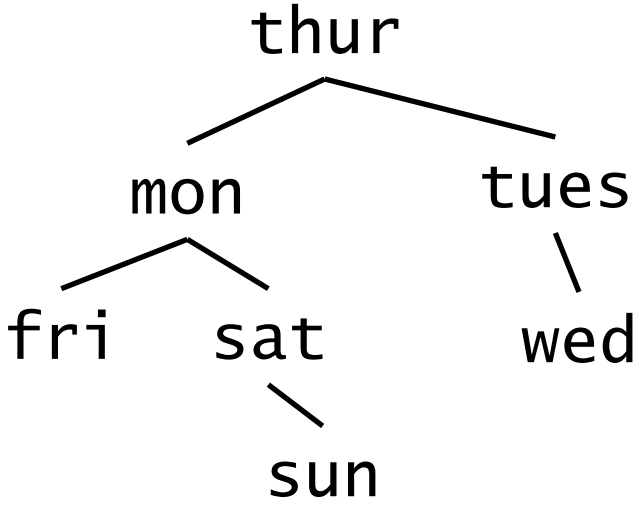
bomen, bomen, bomen …



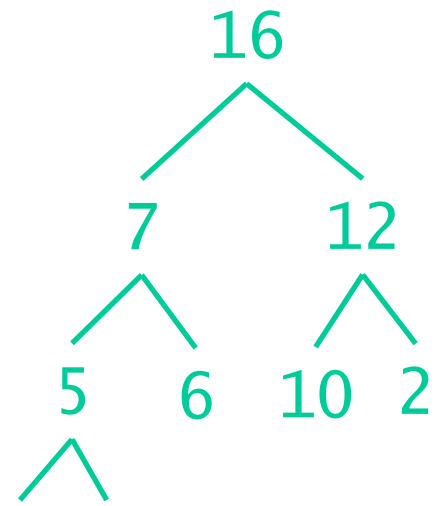
expressie



code

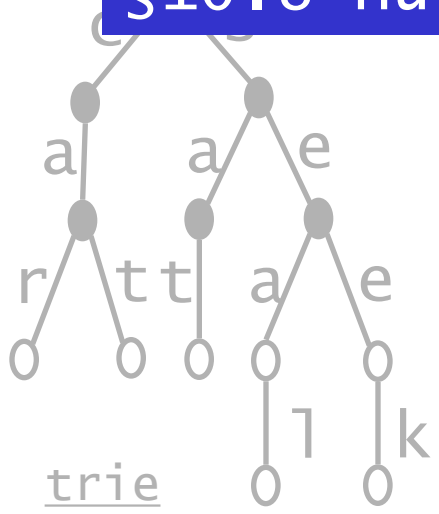


binaire zoekboom

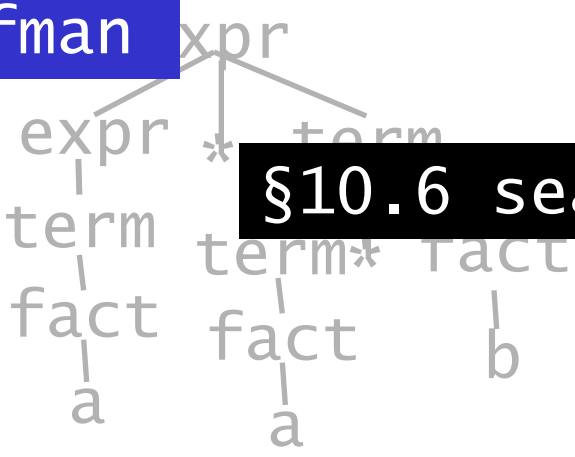


heap

§10.8 Huffman



trie



syntax

§10.7 heap

§10.6 search trees

2,3 boom

bomen

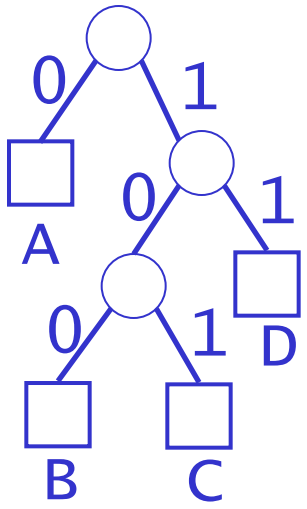
Bomen komen aan de orde bij de colleges Algoritmiek en Datastructuren.

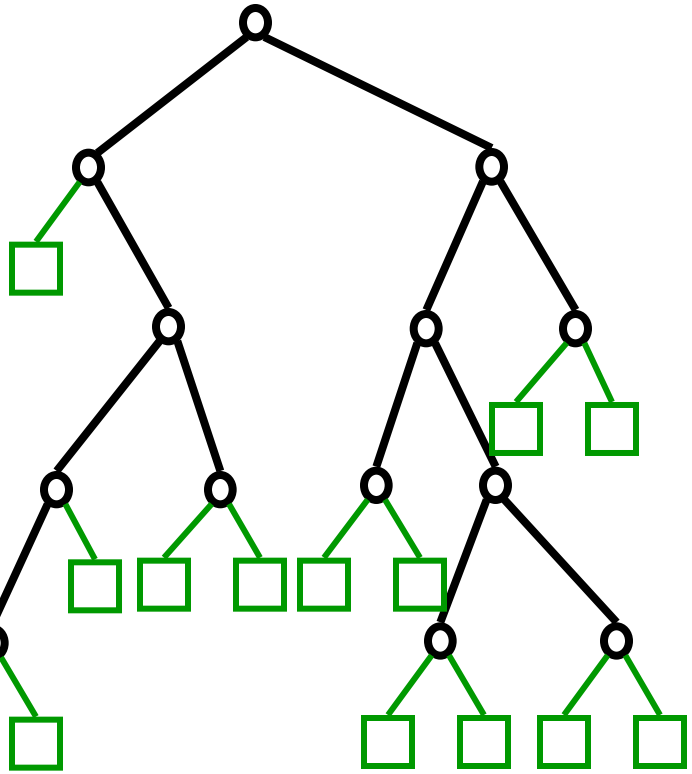
Huffman code: binaire representatie van letters zodat de boodschap minimale lengte heeft bij gegeven frequentie van de letters

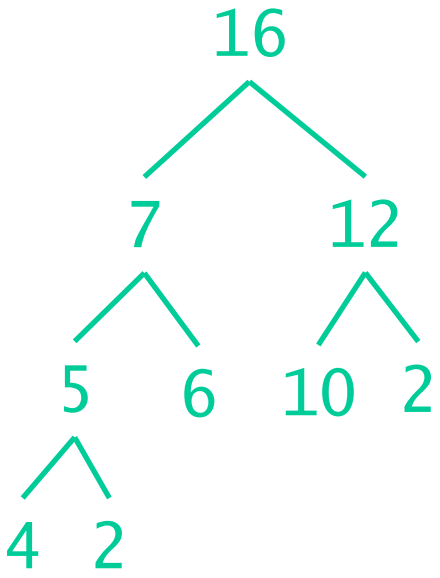
Zoekbomen (voor snel zoeken): vind elk element, als kleiner dan naar links, als groter dan naar rechts

Heap: ordening op 'prioriteit' van boven naar beneden, om snel grootste element te vinden (en verwijderen). Er zijn slimme algoritmen om elementen toe te voegen en te verwijderen

§10.3 mooie vormen







complete binaire boom

2-boom (volle binaire boom)

uitgebreide boom (extended tree)

vol: elke knoop 0 of 2 kinderen

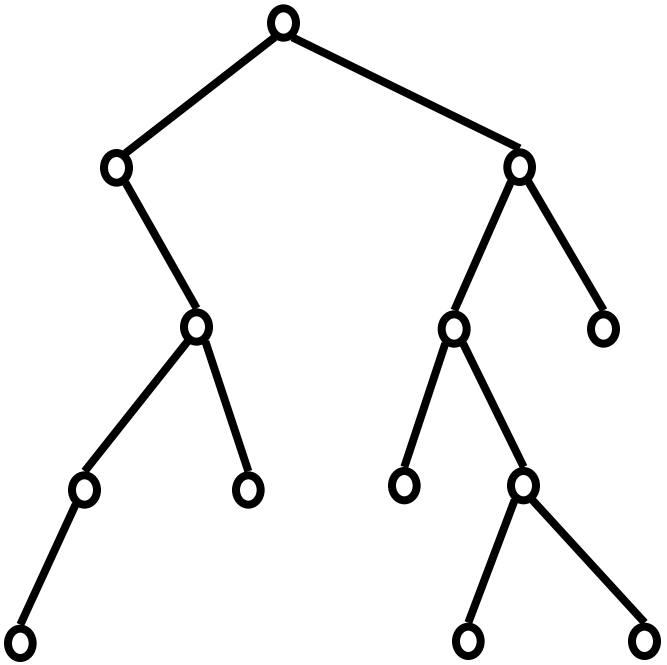
vormen

- vol: élke knoop is óf een blad of heeft twee kinderen. Er zijn dus geen knopen met één kind.

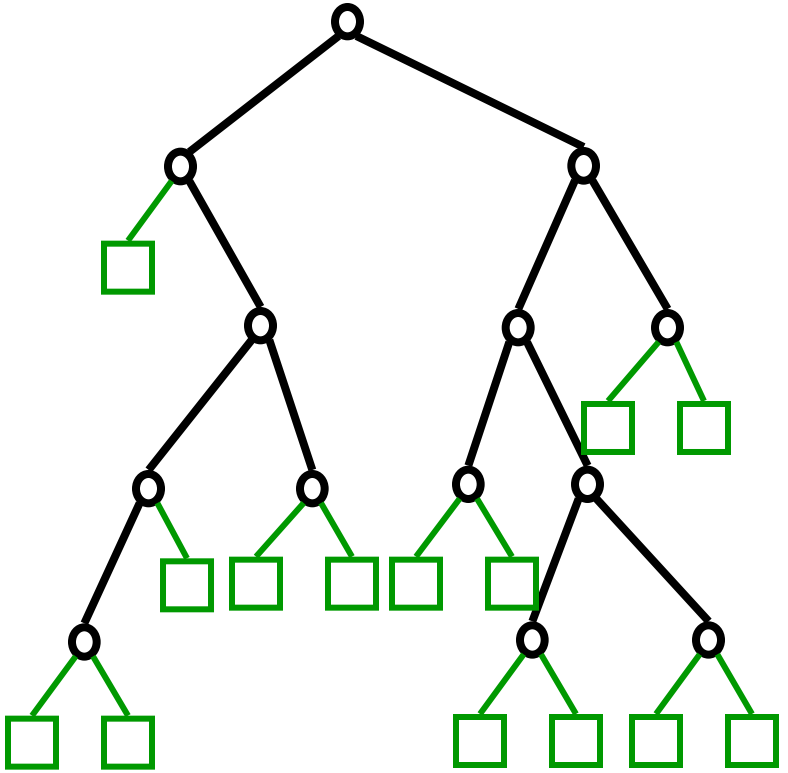
  Elke boom kan vol gemaakt worden door elke lege subboom (elk ontbrekend kind) te vervangen door een nieuwe knoop: deze speciale knopen vormen vervolgens de bladeren van de verkregen boom: externe knopen. In Schaum: extended binary tree
  (dit lijkt op het aangeven van de NULL-pointers in een pointer implementatie)

- compleet: alle nivo's zijn helemaal gevuld, op eventueel het onderste nivo na, en daar zitten de knopen zo ver mogelijk naar links aangesloten.
  (past bij een array implementatie, maar dat leert u later …)

extended tree

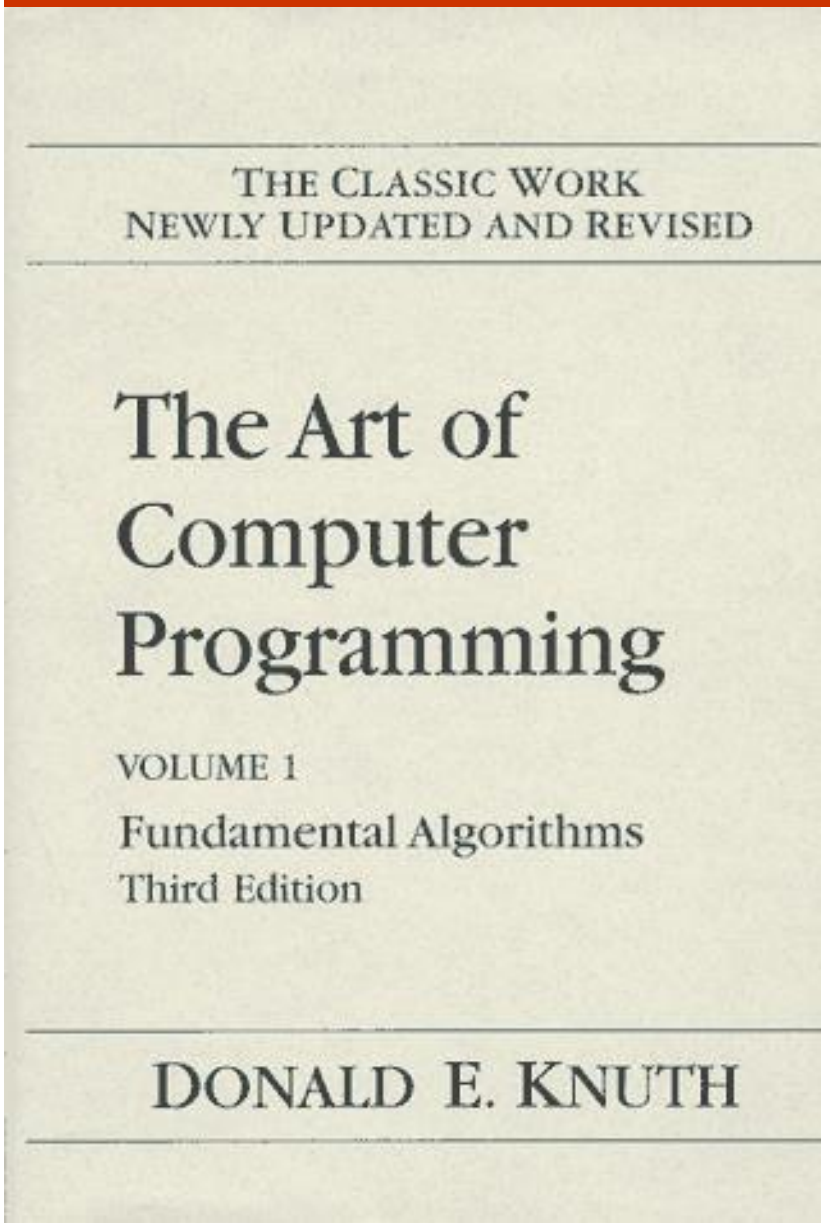


binaire boom

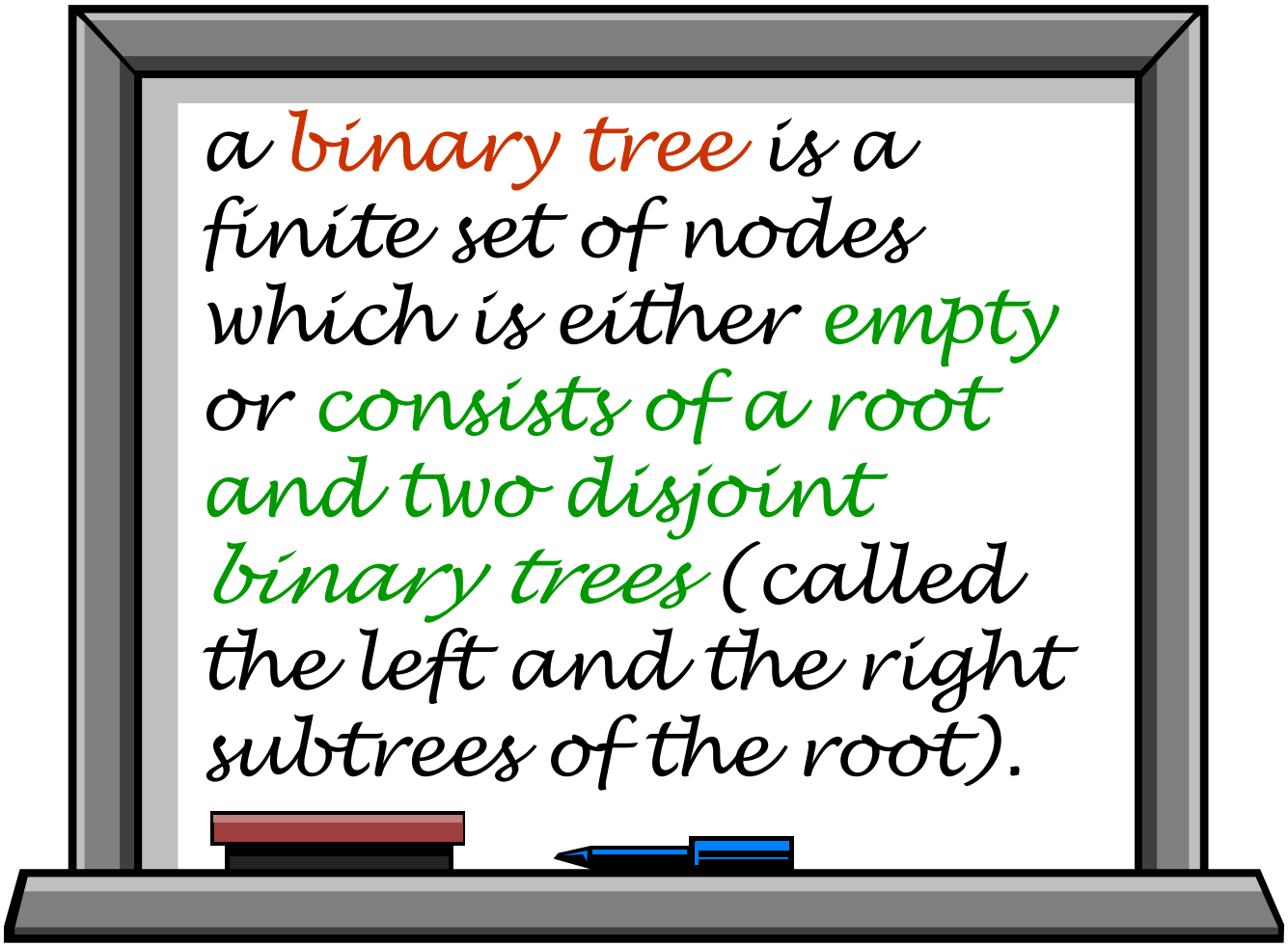


uitgebreide binaire boom
interne / externe knopen

Donald E. Knuth



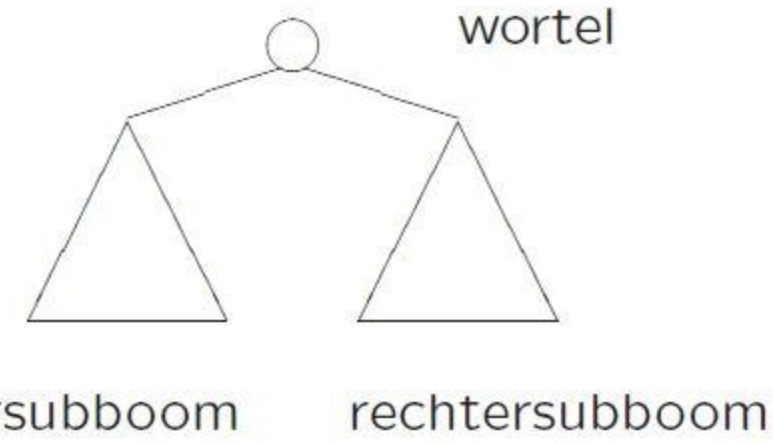
Knuth says:



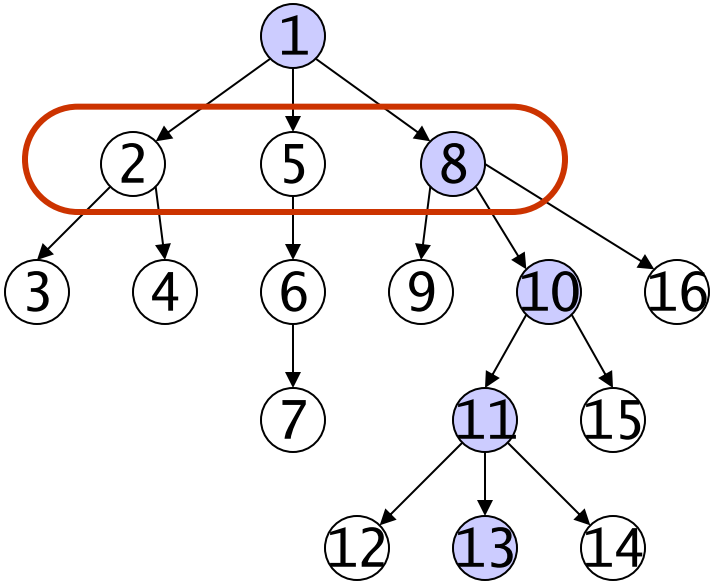
recursie!

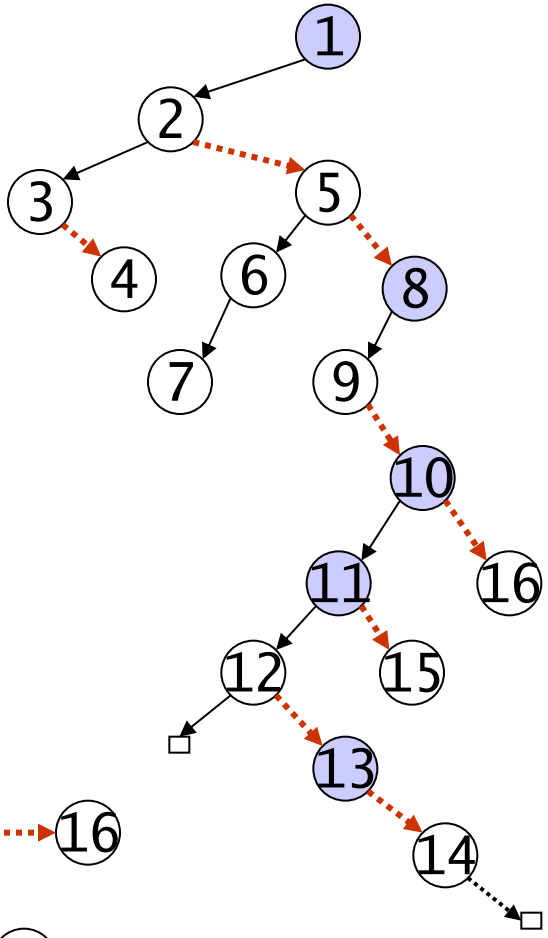
Algoritmiek 2016/02 **Binaire bomen: recursief**

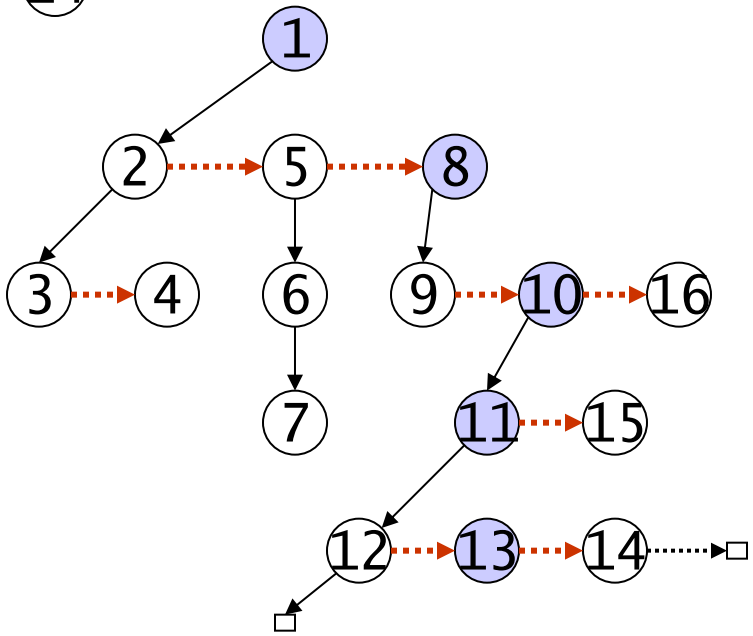
Recursieve definitie: een binaire boom is een eindige verzameling knopen die ofwel leeg is, ofwel bestaat uit een speciale knoop (de wortel) en twee disjuncte verzamelingen knopen die samen de rest van alle knopen vormen. Die knoopverzamelingen vormen beide ook weer een binaire boom: de linkersubboom en de rechtersubboom.



§10.9 eerste kind, rechter broer







1-1-correspondentie tussen geordende gewortelde bomen en binaire bomen waarvan de wortel alleen een linkerkind heeft

sibling

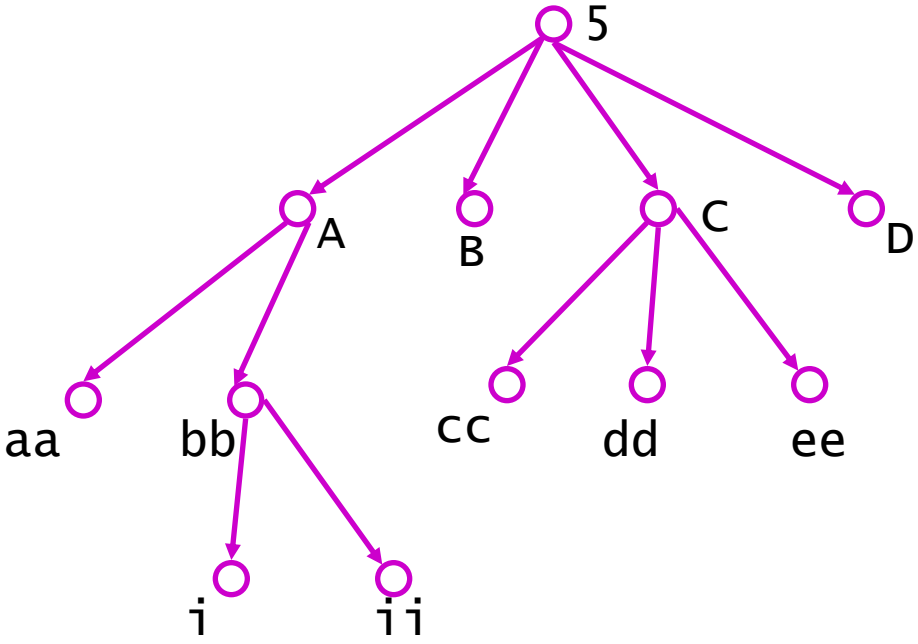
De eerste-kind rechter-broer representatie vormt elke boom om in een binaire boom, links en rechts krijgen dan die nieuwe betekenis van kind en broer (resp.)

Opmerking. Jammer genoeg kennen wij in het Nederlands geen onzijdig woord voor ‘sibling’, dus vandaar de broer. Sorry.

bomen: knopen ordenen

p.238 Polish notation
p.240-241 traversing binary trees

knopen ordenen



- 5
  - A
    - aa
    - bb
      - i
      - ii
  - B
  - C
    - cc
    - dd
    - ee
  - D

- aa
  - i
  - ii
- bb
- A
- B
  - cc
  - dd
  - ee
- C
- D
- 5

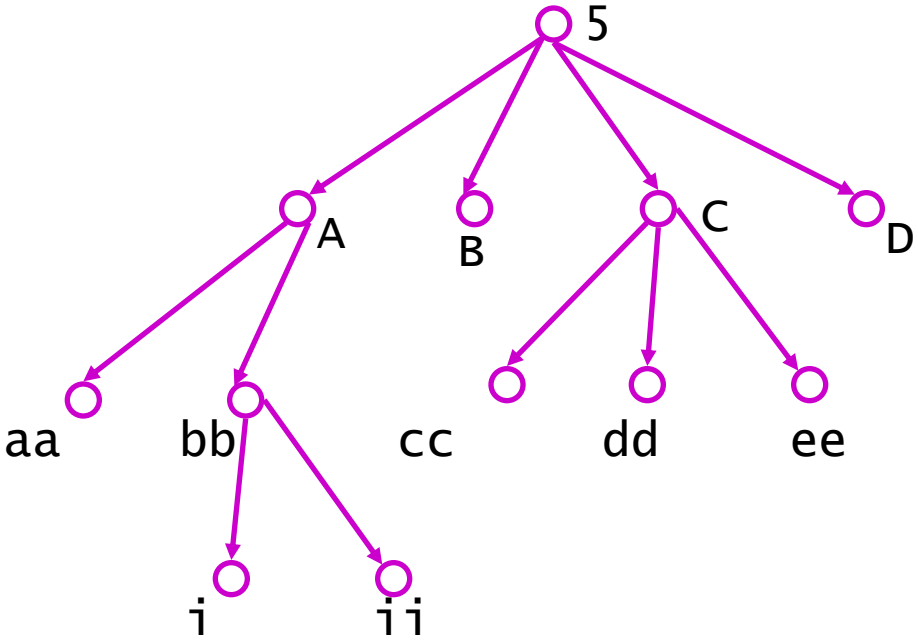
ordening van de knopen

↕

omloop door de boom

voor geordende gewortelde bomen of binaire bomen

knopen ordenen



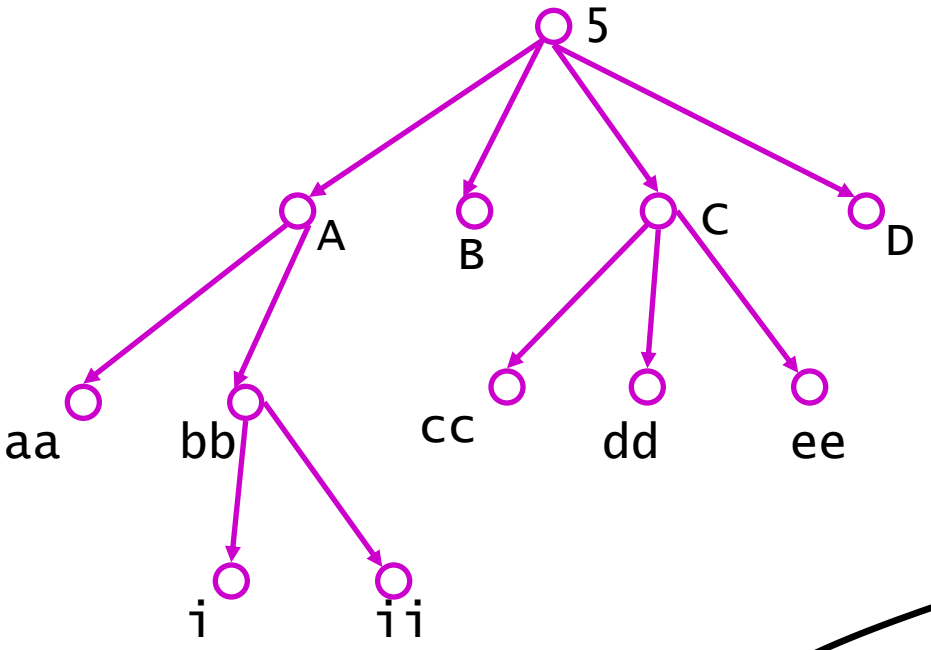
- 5
  - A
    - aa
    - bb
      - i
      - ii
  - B
  - C
    - cc
    - dd
    - ee
  - D

- aa
  - i
  - ii
- bb
- A
- B
  - cc
  - dd
  - ee
- C
- D
- 5

inhoudsopgave:
eerst hoofdstuk,
dan onderdelen

eerst onderdelen
dan hoofdstuk

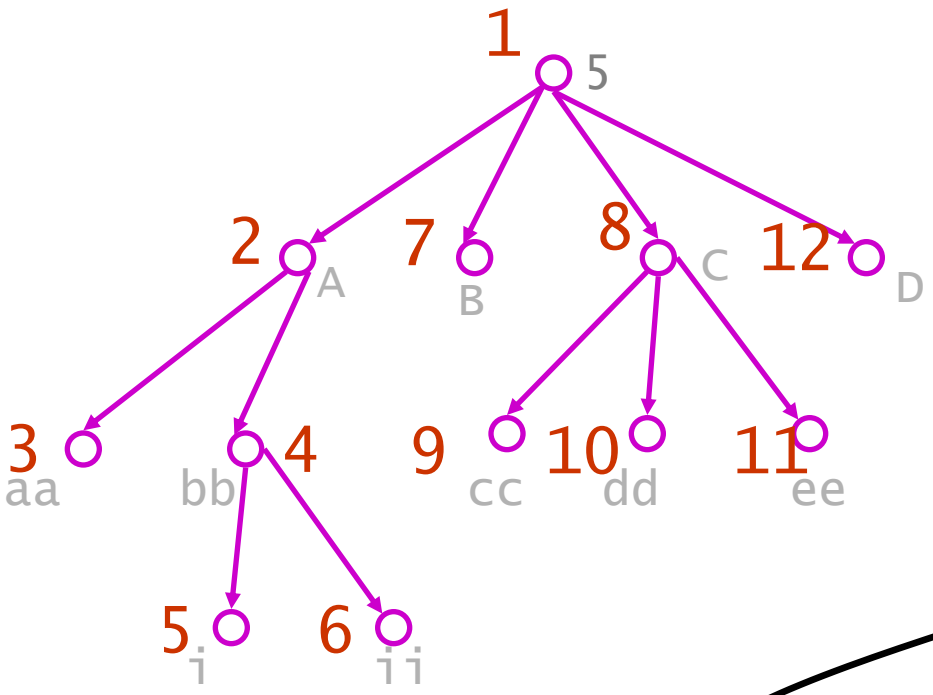
preorde



- 5
  - A
    - aa
    - bb
      - i
      - ii
  - B
  - C
    - cc
    - dd
    - ee
  - D

‘eerst knoop, dan (subbomen van) kinderen’

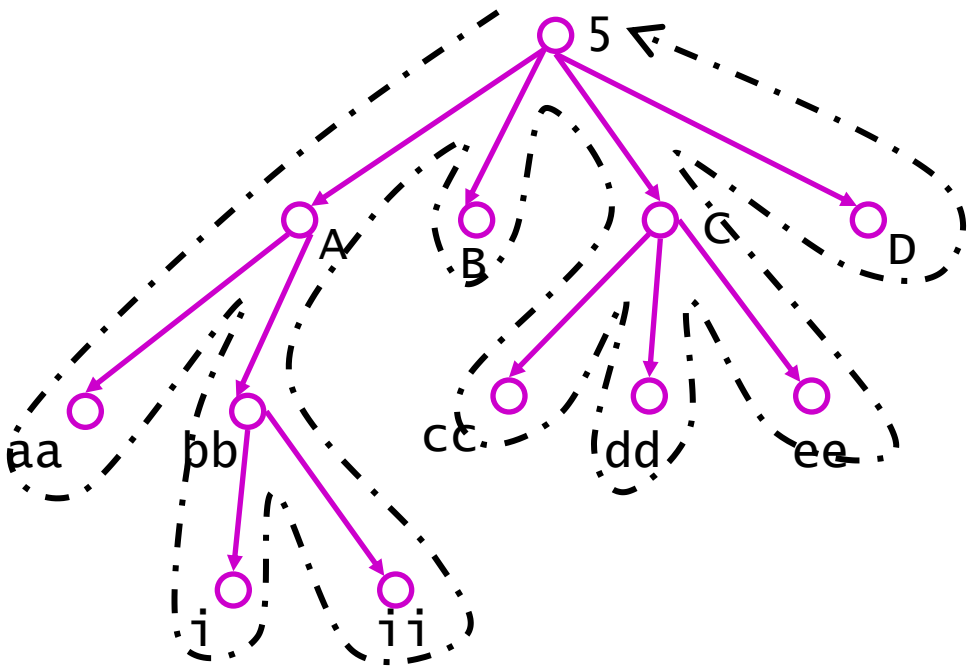
preorde



nummering

1 5
2 • A
3 • aa
4 • bb
5 •i
6 •ii
7 • B
8 • C
9 • cc
10 • dd
11 • ee
12 • D

omloopmethode: preorde

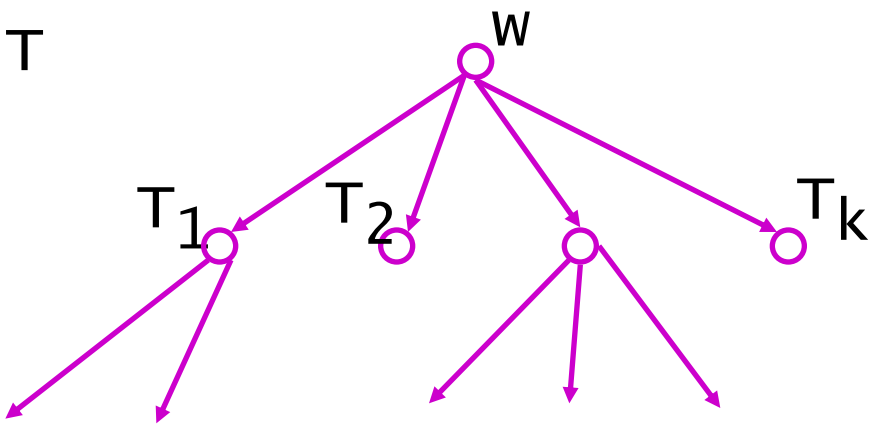


- 5
  - A
    - aa
    - bb
      - i
      - ii
  - B
  - C
    - cc
    - dd
    - ee
  - D

Omloopmethode

eerste bezoek

preorde



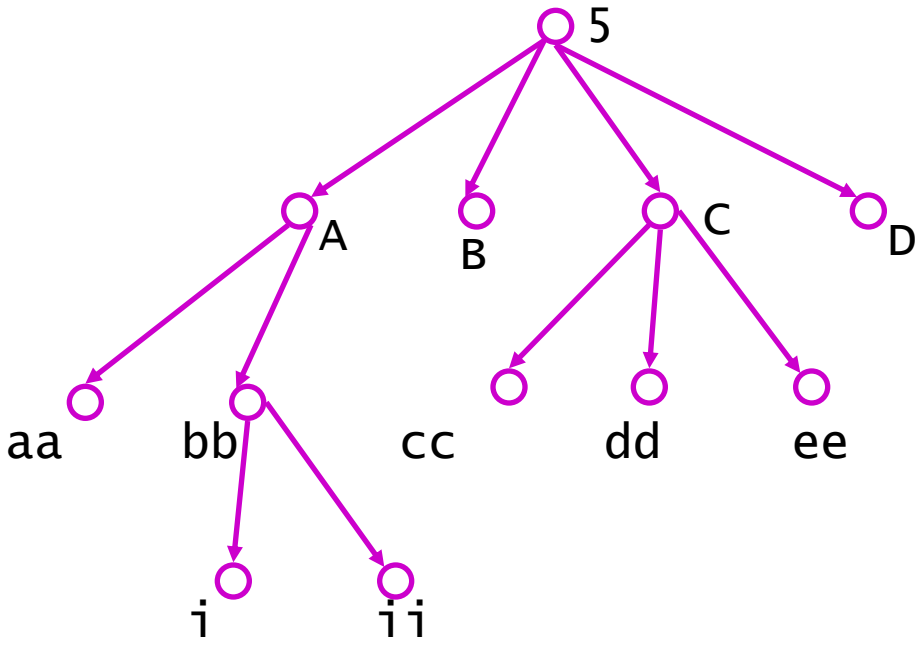
recursieve definitie

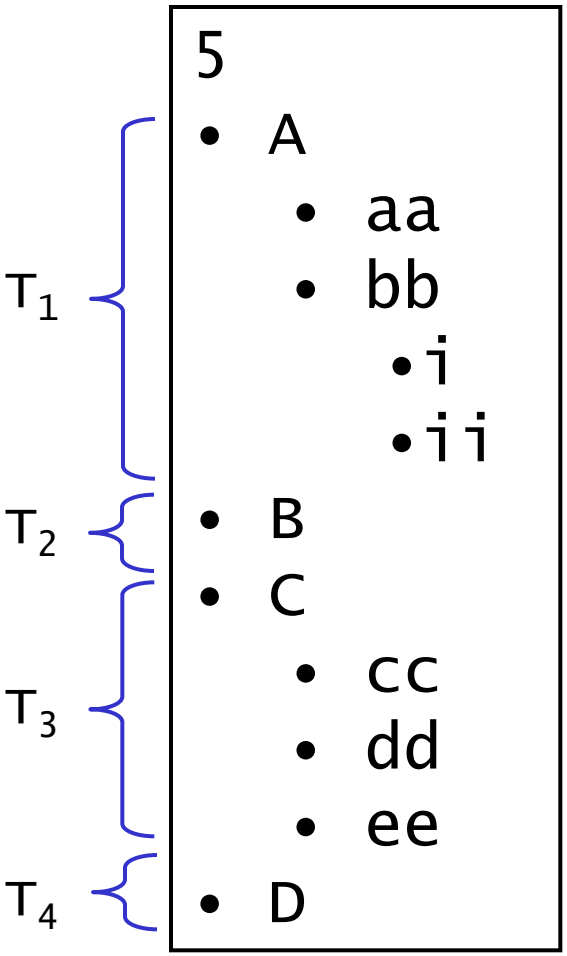
boom T
deelbomen T_1, T_2, …, T_k van wortel

$pre(T)$ = wortel(T),
$pre(T_1)$, $pre(T_2)$, …, $pre(T_k)$

‘eerst knoop bezoeken, dan (subbomen van) kinderen*’ *op *dezelfde* manier

preorde

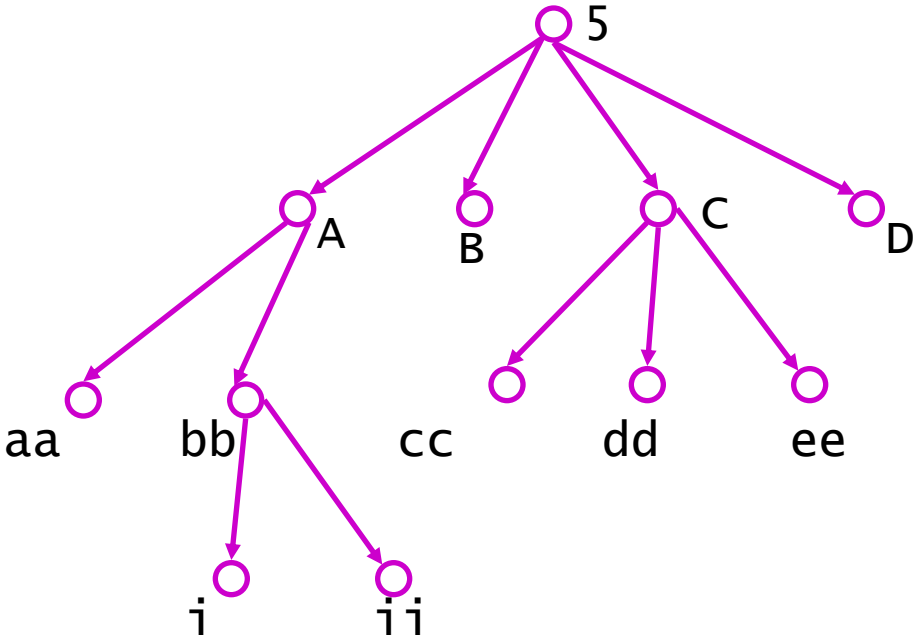




pre-ordening

$pre(T) = wortel(T), pre(T_1), pre(T_2), \ldots, pre(T_k)$

postorde

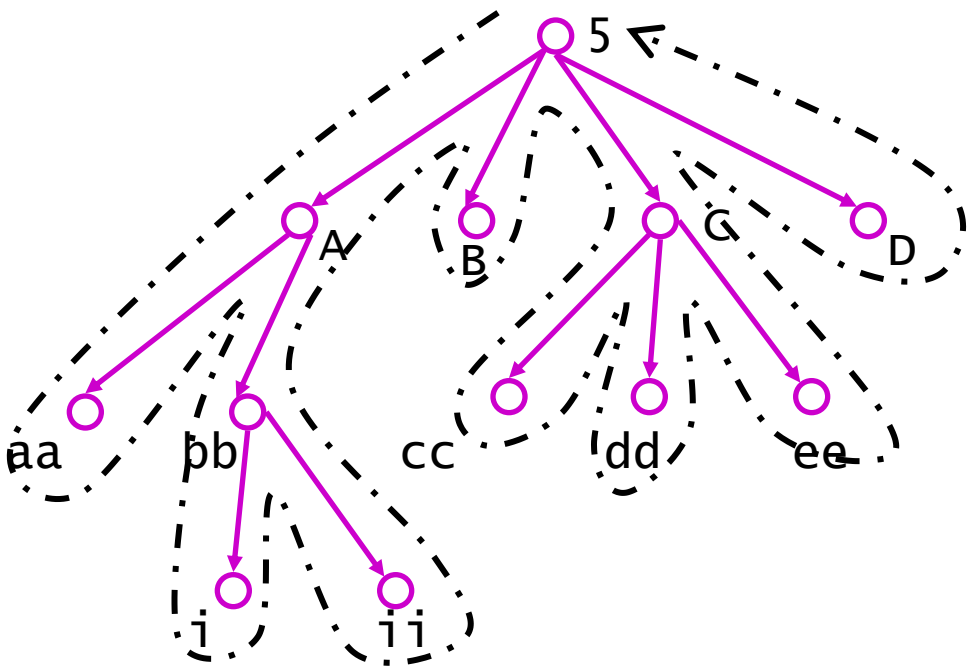


- aa
  - i
  - ii
- bb
- A
- B
  - cc
  - dd
  - ee
- C
- D

5

‘eerst (subbomen van) kinderen* bezoeken, dan knoop’ *op *dezelfde* manier

omloopmethode: postorde



- aa
  - i
  - ii
- bb
- A
- B
  - cc
  - dd
  - ee
- C
- D

5

Omloopmethode

derde bezoek

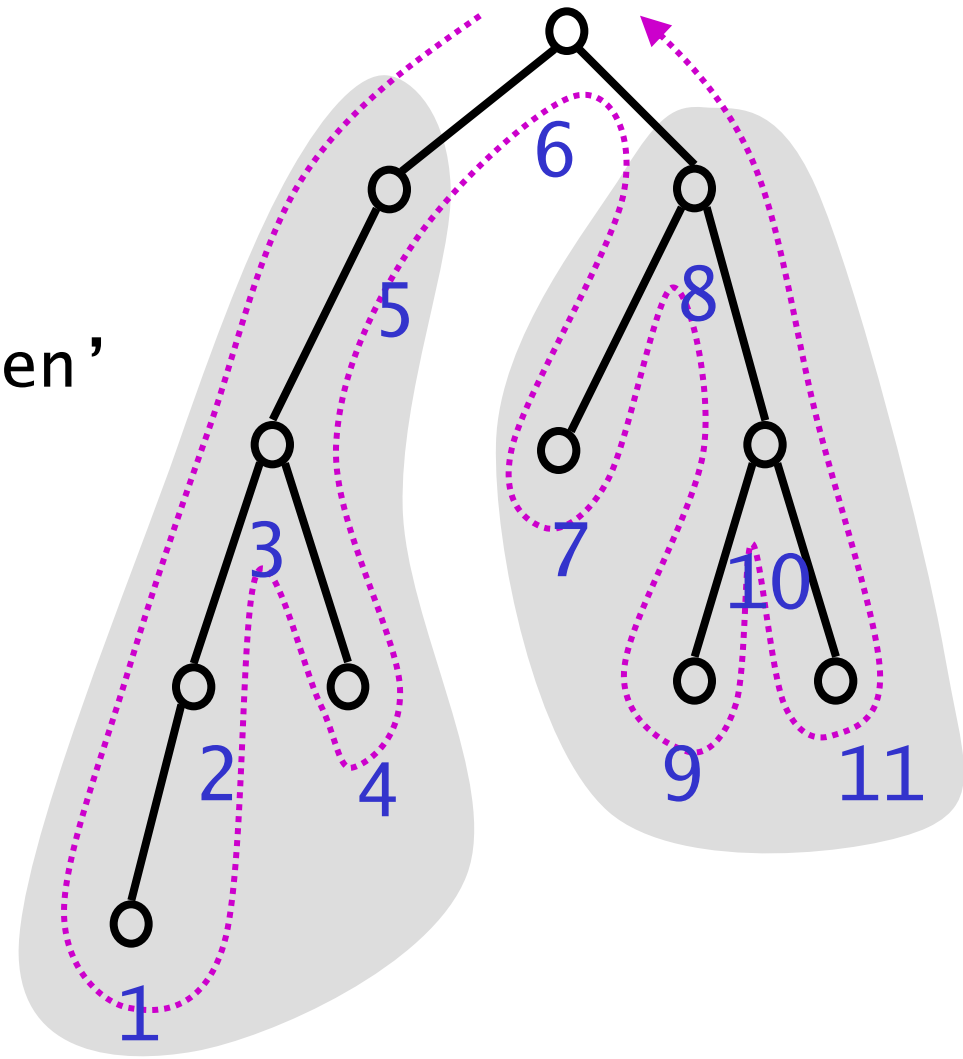
symmetrische ordening

bij binaire bomen

‘knoop tussen
(subbomen van) kinderen’

tweede bezoek

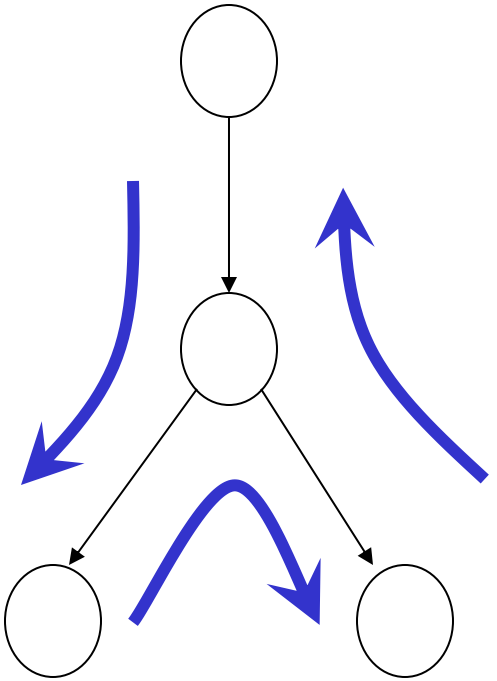
Omloopmethode



$symm(T) = symm(T_L), wortel(T), symm(T_R)$

wandelingen

(1)
preorde
WLR



(3)
postorde
LRW

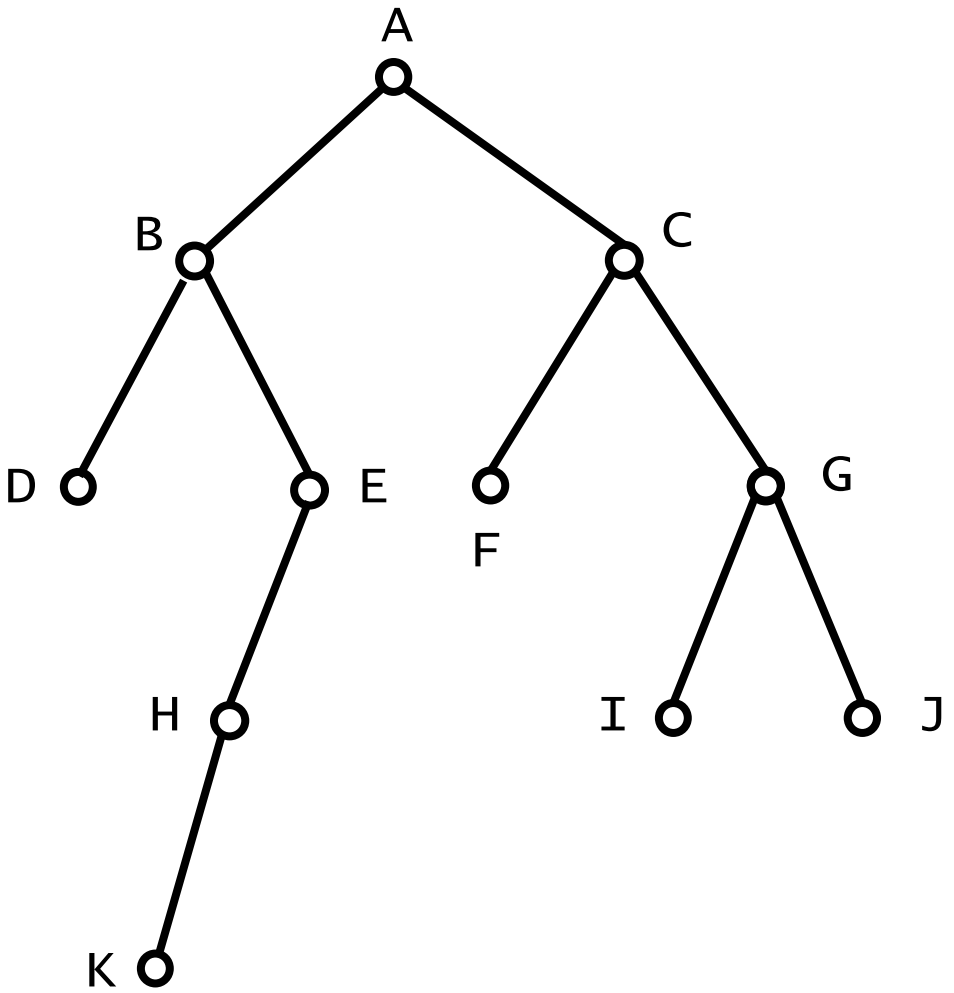
(2)
symmetrisch
LWR

(bij binaire bomen)

wandelingen

Bij de omloopmethode zijn er voor binaire bomen drie natuurlijke momenten om (systematisch) een knoop te ‘bezoeken’, bij de eerste, tweede, of derde keer dat we een knoop tegenkomen.
Dat leidt dan tot pre-orde, symmetrische, en post-orde wandelingen.

sommetje



Geef de inhoud van de knopen in

(i) WLR-volgorde
(ii) LRW-volgorde
(iii) LWR-volgorde

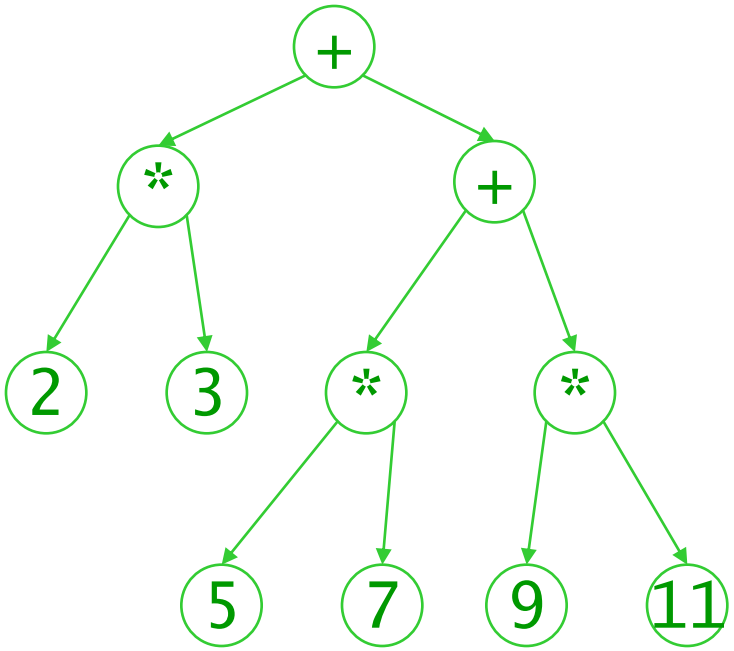
expressies

Een (algebraïsche) expressie die alleen binaire operaties gebruikt kan worden weergegeven door een *volle* binaire boom -> expressieboom

De Poolse notatie of prefix-notatie voor (algebraïsche) expressies (bedacht in 1920) correspondeert met een preorde-wandeling door de expressieboom. Haakjes zijn dan niet nodig, in tegenstelling tot bij de gebruikelijke notatie (infix-notatie), zoals (2+3)*((5*7)+(9*11)).
De infix-notatie correspondeert met een symmetrische wandeling (inorder in het Engels) met haakjes.

expressies

interne knopen: operatoren
bladeren: waardes



Hier zijn de operatoren binair. De boom is een *volle* binaire boom

omloopmethode

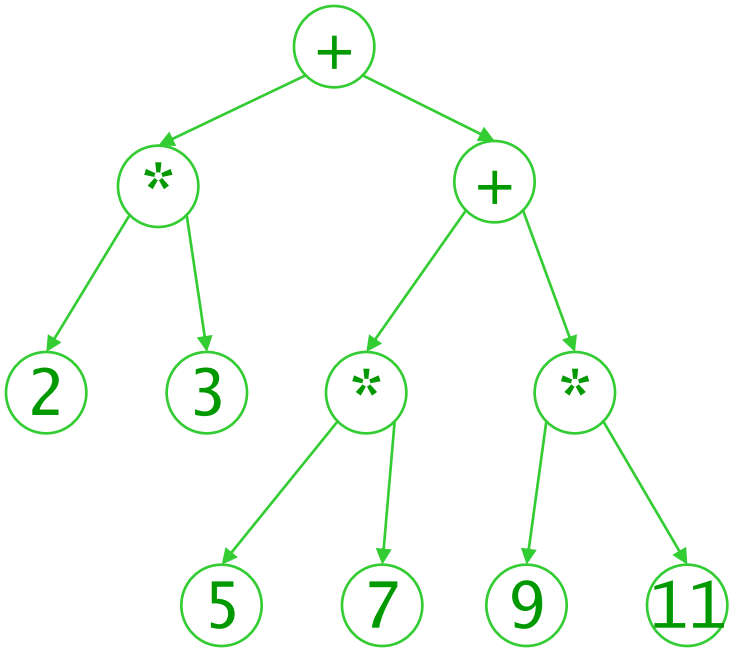
infix, volledig gehaakt
((2*3)+((5*7)+(9*11)))

prefix *Polish notation*
+ * 2 3 + * 5 7 * 9 11

postfix *reverse Polish*
2 3 * 5 7 * 9 11 * + +

zonder haakjes!
'ariteit' bekend

expressies



infix, volledig gehaakt
((2*3)+((5*7)+(9*11)))

haakjes ivm volgorde
evt. voorrangsregels

prefix *Polish notation*
+ * 2 3 + * 5 7 * 9 11
postfix
2 3 * 5 7 * 9 11 * + +

géén haakjes (echt niet nodig!)

recept

prefix *Polish notation*

+ * 2 3 + * 5 7 * 9 11

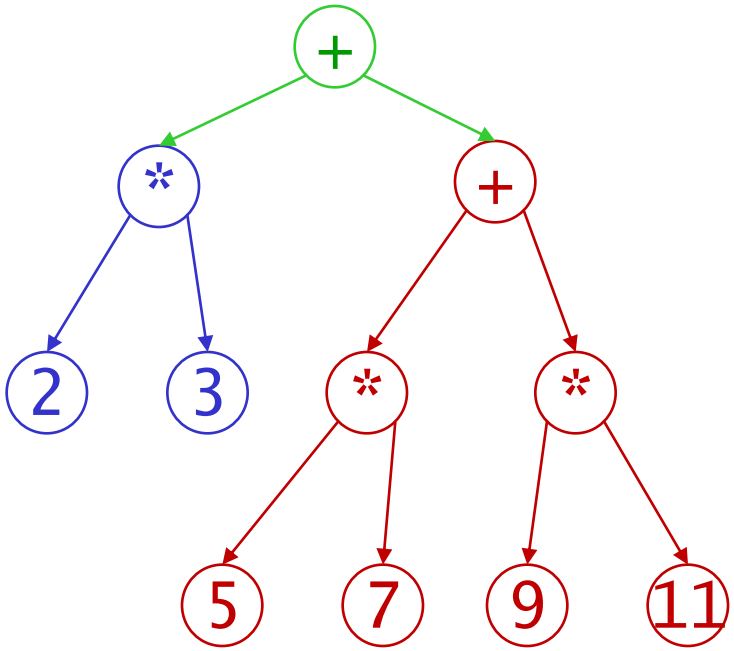
van prefix naar infix:

herhaal

vervang @ x y

(d.w.z. operator + twee waardes)

door (x @ y)



+ * 2 3 + * 5 7 * 9 11

+(2*3)+(5*7)(9*11)

+(2*3)((5*7)+(9*11))

((2*3)+((5*7)+(9*11)))

van prefix naar infix

Het kan natuurlijk ook door eerst uit de prefixnotatie de expressieboom te construeren, en daaruit vervolgens de infix-notatie met haakjes af te leiden.

recept

postfix *Polish notation*

2 3 * 5 7 * 9 11 * + +

van postfix naar infix:

herhaal

vervang x y @

(d.w.z. operator + twee waardes)

door (x @ y)

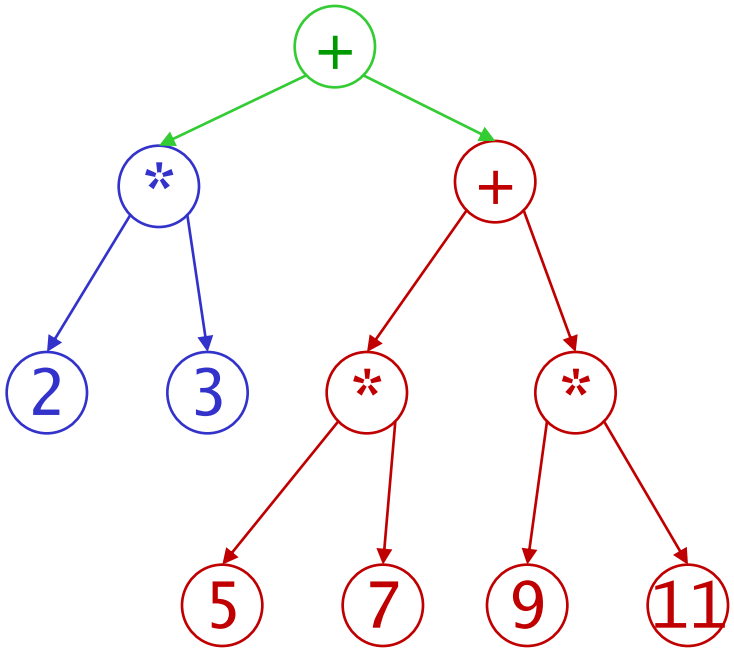
(dus door één nieuwe waarde aangegeven door haakjes

2 3 * 5 7 * 9 11 * + +

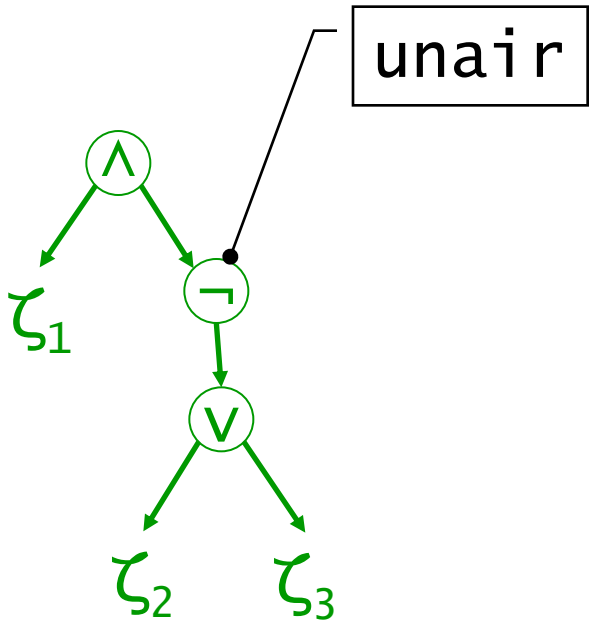
(2*3) (5*7) (9*11) + +

(2*3) ((5*7)+(9*11)) +

((2*3)+((5*7)+(9*11)))



expressies



omloopmethode

infix, waar nodig gehaakt

$\zeta_1 \wedge \neg(\zeta_2 \vee \zeta_3)$

prefix *Polish notation*

$\wedge\ \zeta_1\ \neg\ \vee\ \zeta_2\ \zeta_3$

postfix

$\zeta_1\ \zeta_2\ \zeta_3\ \vee\ \neg\ \wedge$

uniek te ontcijferen míts
ariteit bekend van de operatoren

recursieve functies bij bomen

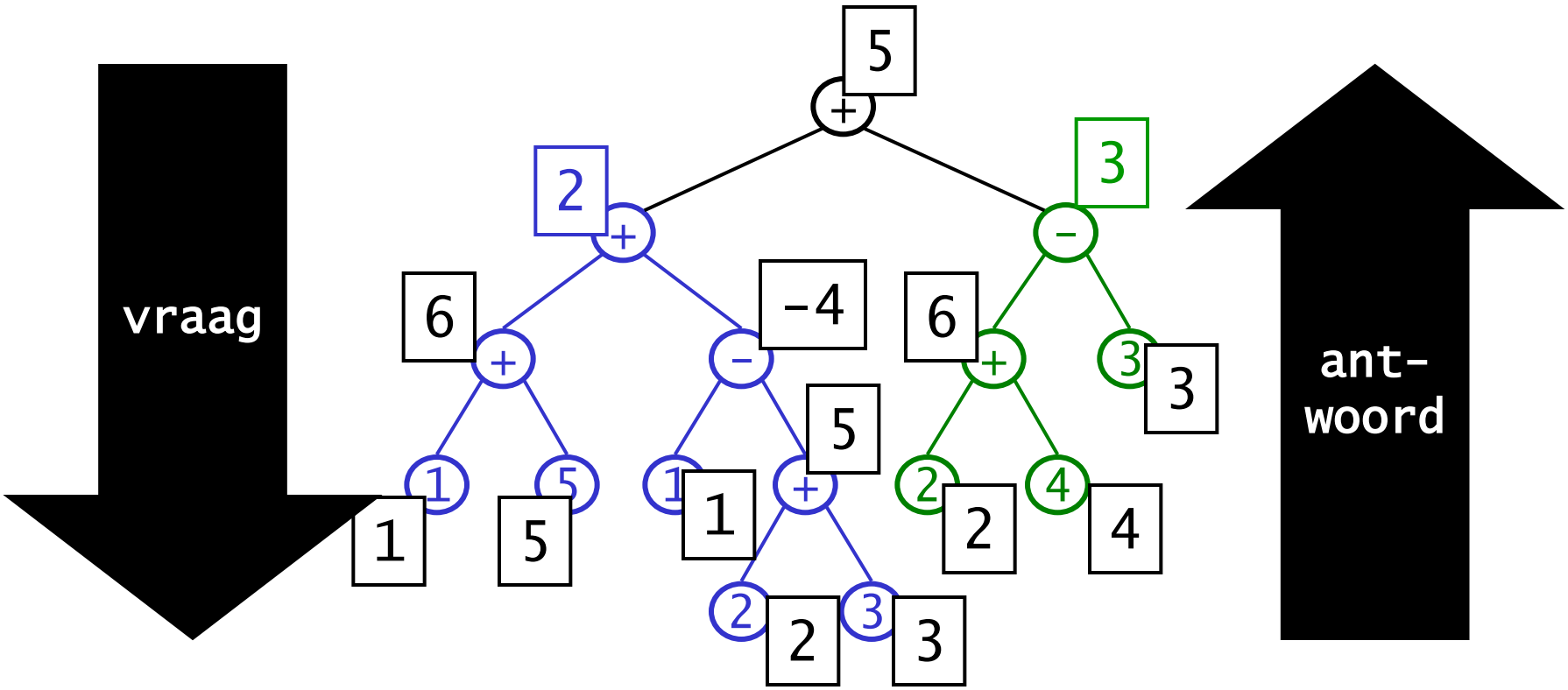
waarde expressie

basis

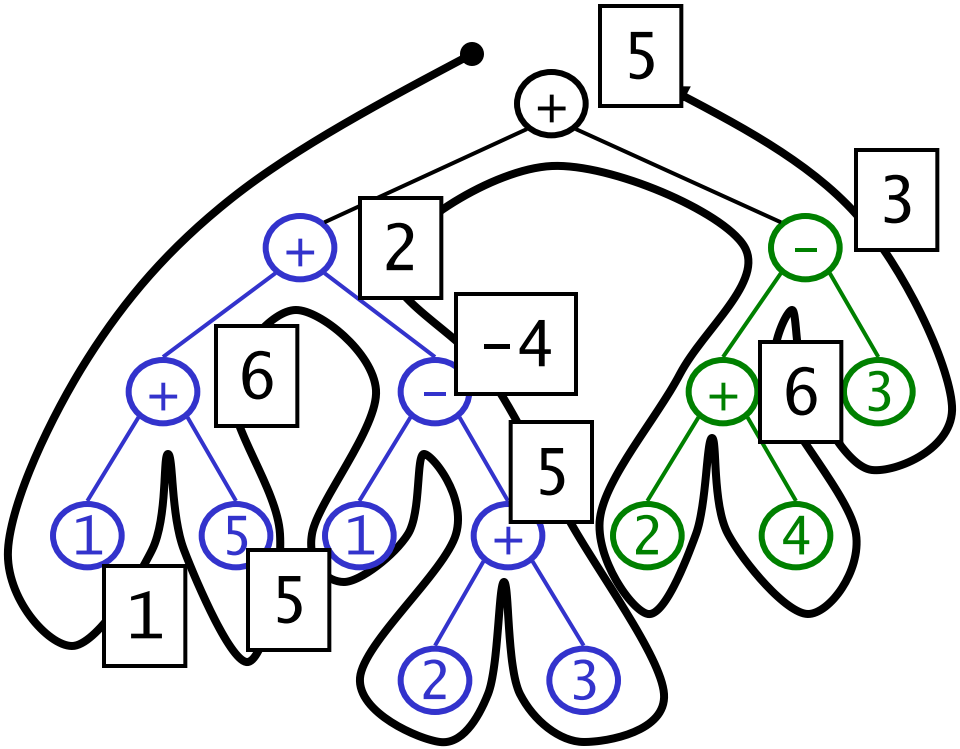
  blad ‘x’: f(blad) = getalswaarde x

recursie

  knoop ‘@’: f(knoop) = f(links) @ f(rechts)



evaluatie



postorde evaluatie

		6				5	-4	2			6			3	5		
1	5	+	1	2	3	+	-	+	2	4	+	3	-	+			

recursieve functies bij bomen

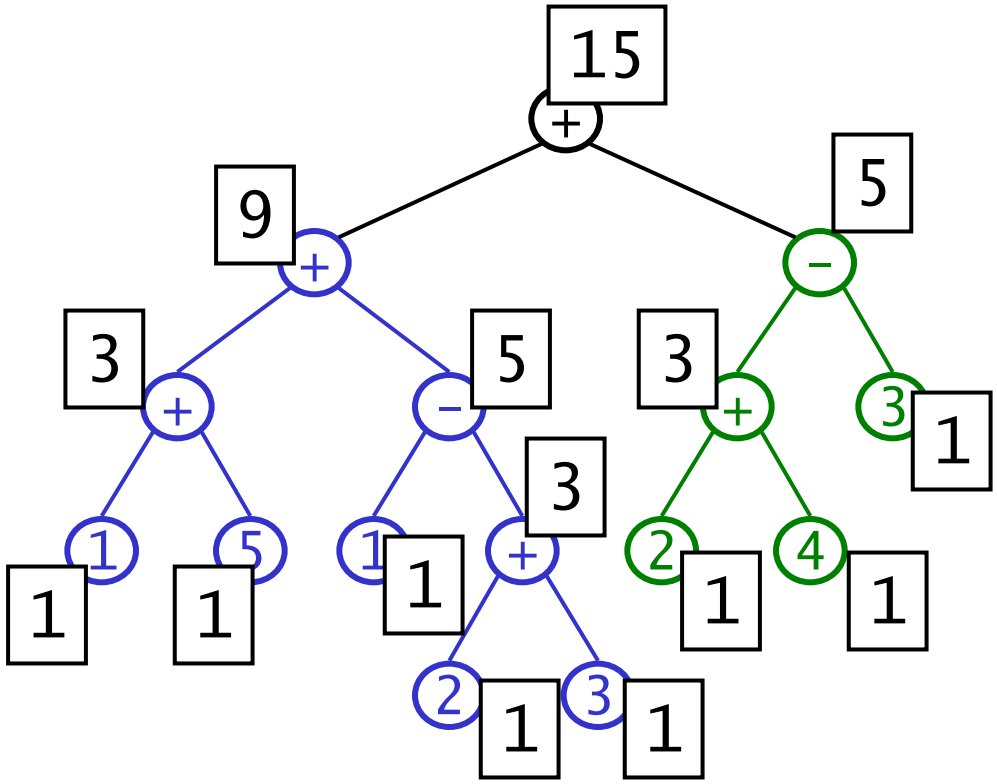
basis *aantal knopen*

blad f(blad) = 1

recursie

knoop f(knoop) = 1+f(links)+f(rechts)

opletten als een knoop geen linker- of rechterkind heeft



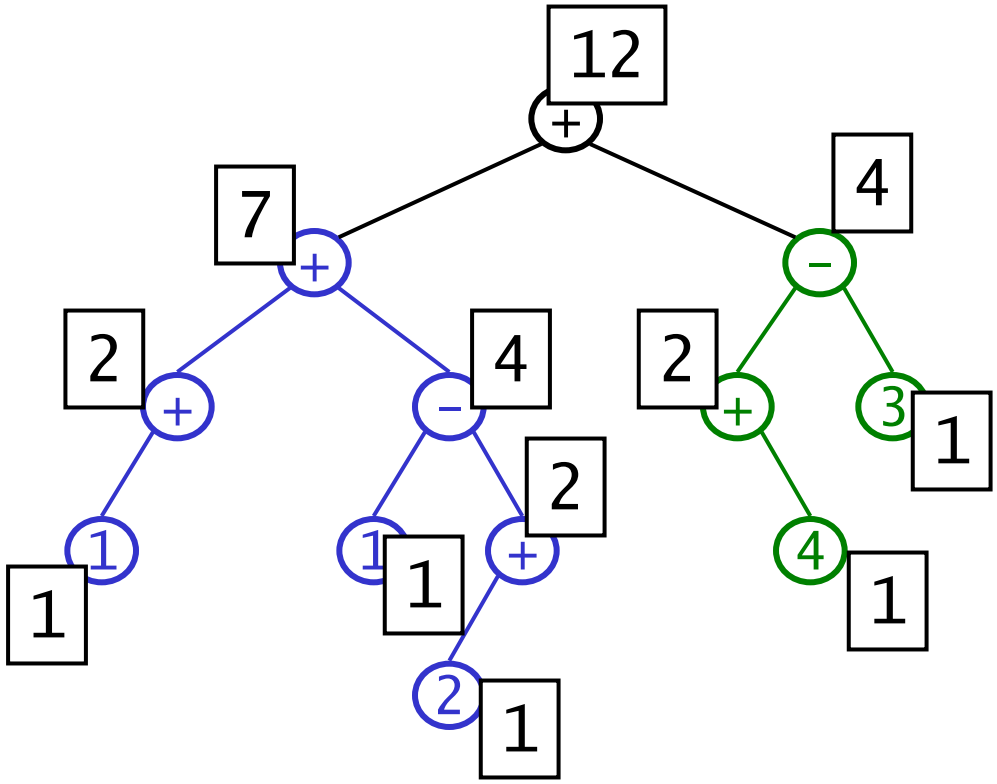
recursieve functies bij bomen

basis — *aantal knopen*

leeg f(leeg) = 0

recursie

knoop f(knoop) = 1+f(links)+f(rechts)



recursieve functies bij bomen

hoogte

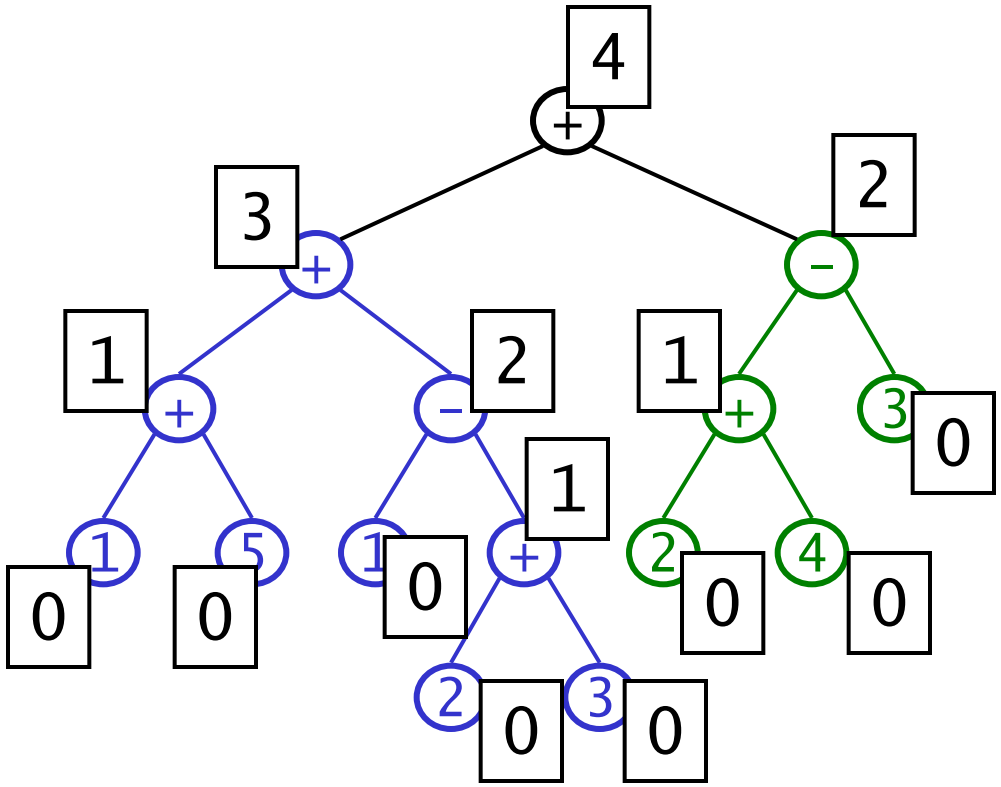
basis

blad h(blad) = 0

recursie

knoop h(knoop) = 1+max{h(links),h(rechts)}

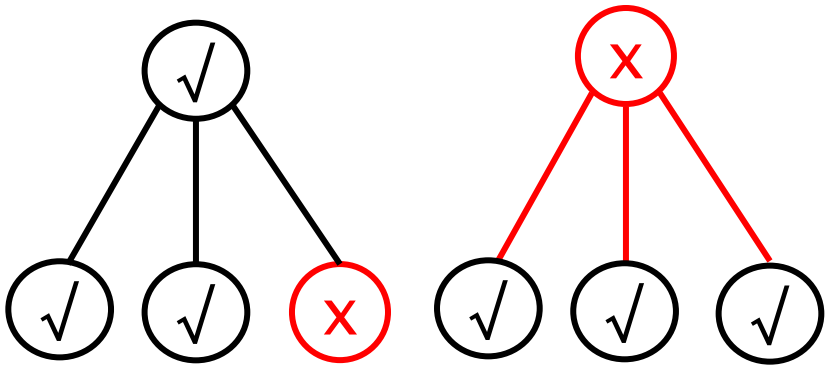
opletten als een knoop geen linker- of rechterkind heeft

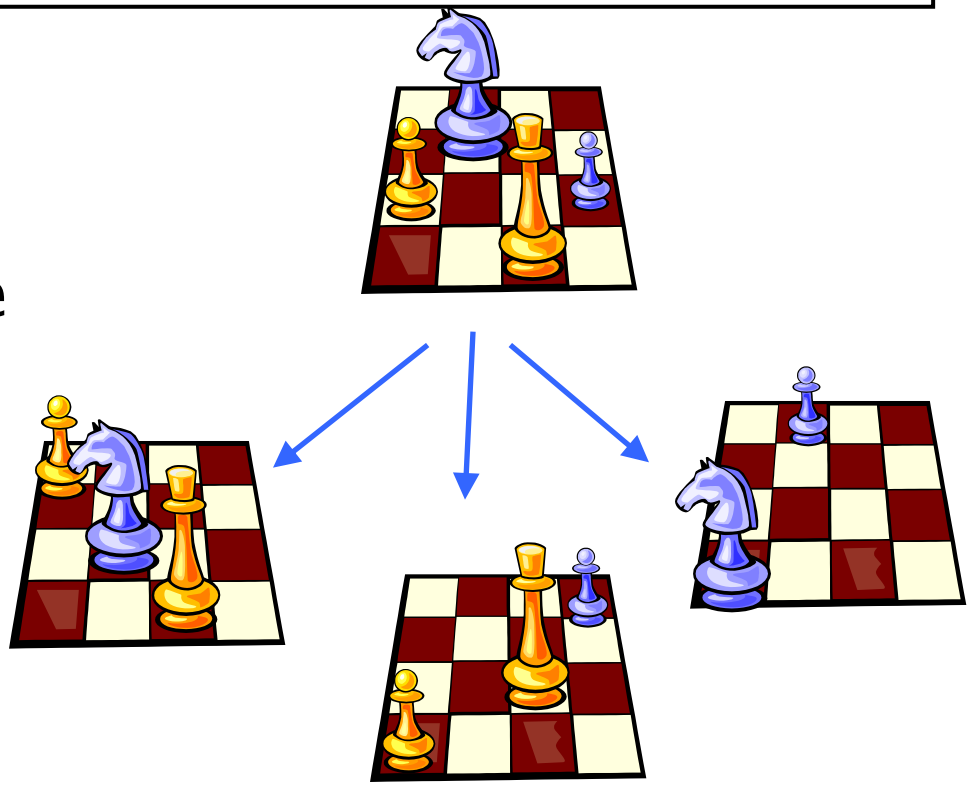


spel-evaluaties

basis
 blad: al of niet gewonnen (voor wie aan de beurt is
recursie
 knoop: winnend d.e.s.d.a. ∃ niet-winnend kind
       (verloren: ∀ kind winnend)

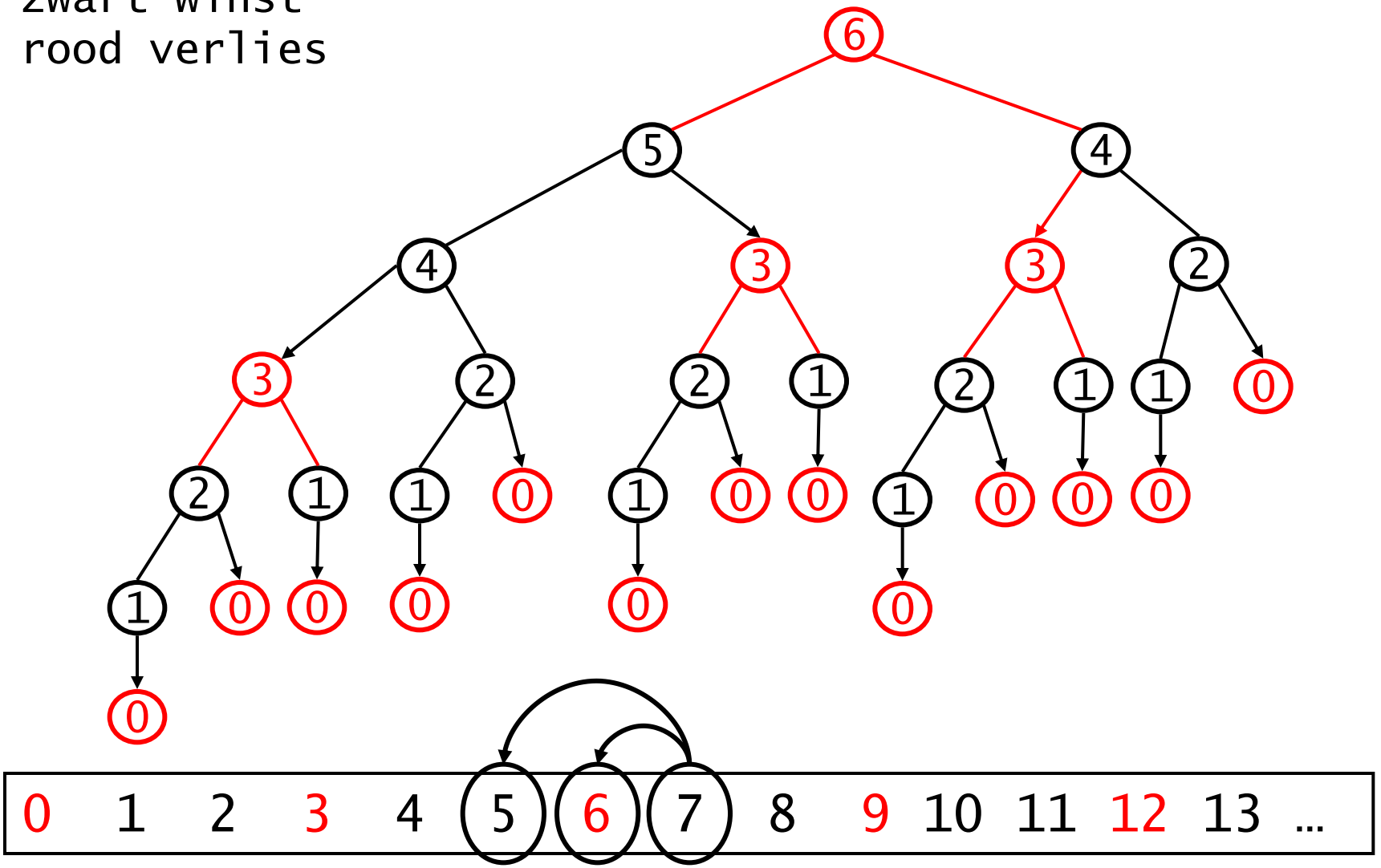
boom niet expliciet
te veel knopen:
    boom knotten +
    evaluatie-functie





Nim

zwart winst
rood verlies



(werk)college

College volgende week:
vrijdag 23 november,
9.15 – 11.00 in zaal 407-409 (Snellius) !

Dus: geen college op dinsdag, *wel college op vrijdag* !

Werkcollege deze week:
vrijdag 16 november,
9.00 – 10.45 in Snelliuszalen 402, 405