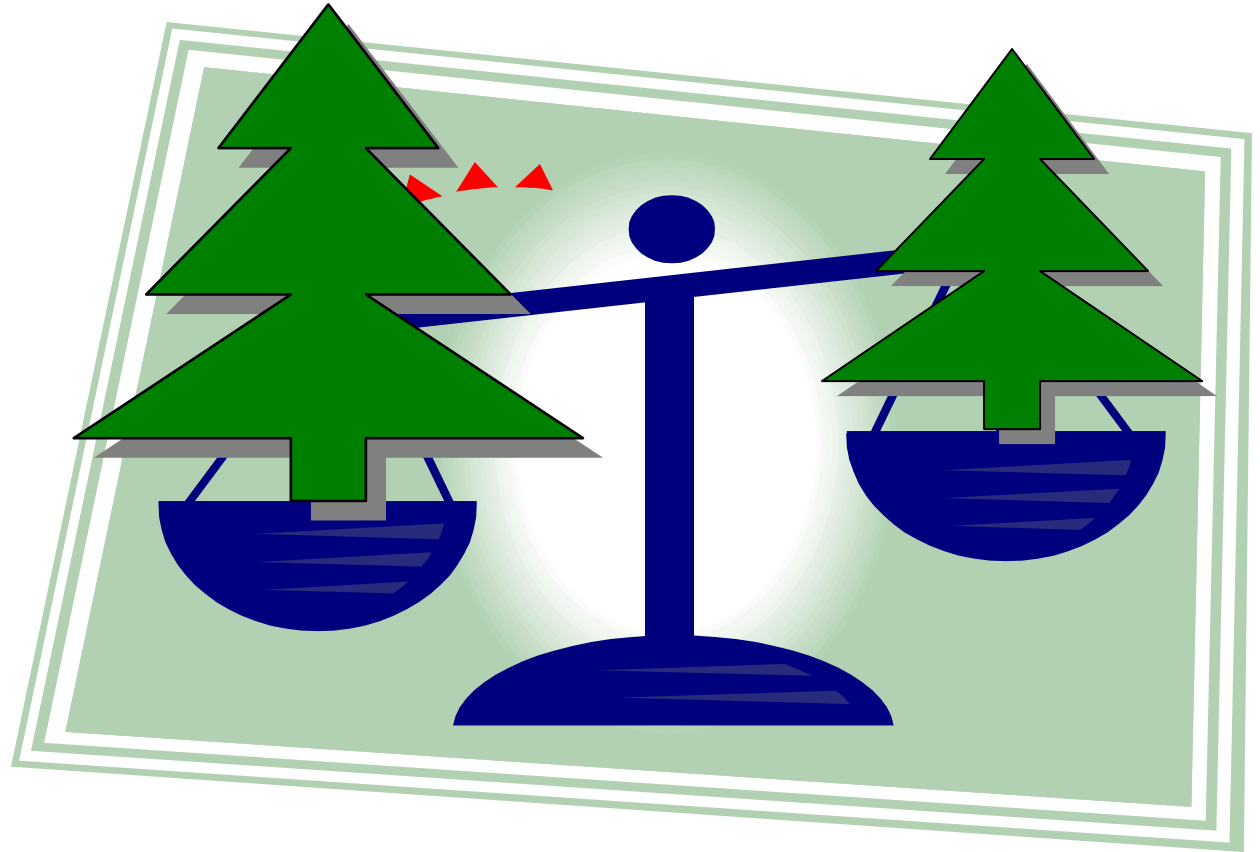


DATASTRUCTUREN

BALANCEREN VAN BOMEN



1

Dr. D.P. Huijsmans

26 september 2012

Universiteit Leiden, LIACS

BALANCEREN EN BINARY SEARCH

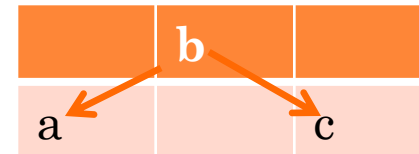
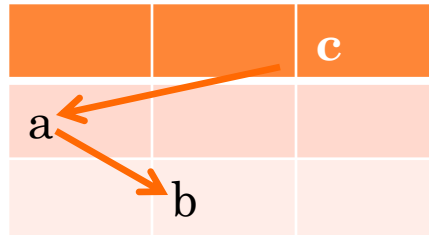
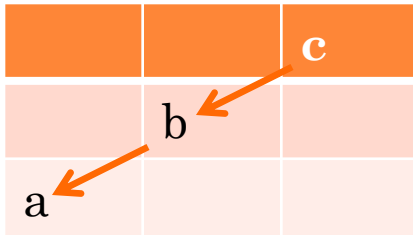
- Om $O(\log N)$ te kunnen zoeken moet de data gesorteerd zijn
- In een binary search tree (BST) InOrder doorlopen is de data gesorteerd toegankelijk
- De slechts denkbare BST realisatie na invoegen is een boom met allemaal lege linker of allemaal lege rechterknopen (backbone structuur)
- De complexiteit van zoeken bij een backbone BST is $O(N)$
- $O(\log N) \leq \text{Complexiteit zoeken BST} \leq O(N)$
- Om in een BST in $O(\log N)$ te kunnen zoeken moet deze boom maximaal gebalanceerd zijn

AANTAL BST'S MET ZELFDE INHOUD

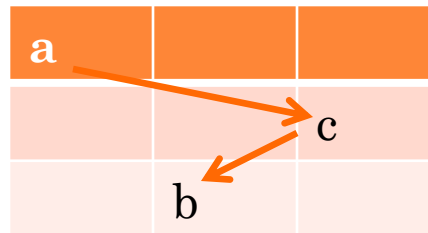
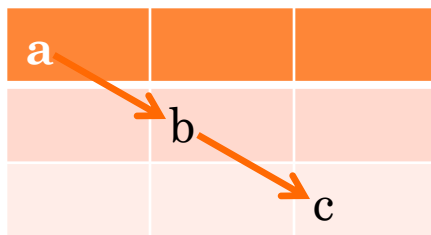
- Gesorteerd array met unieke gesorteerde waarden kent maar 1 verschijningsvorm:



- BST met 3 gesorteerde knopen kent al 5 vormen:



1 min pad vorm

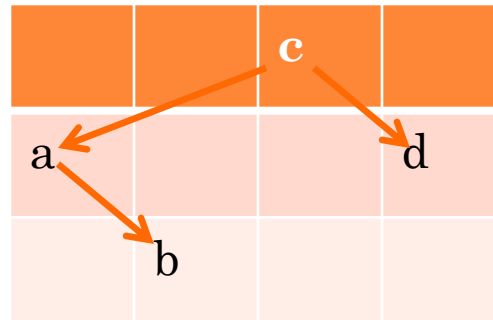
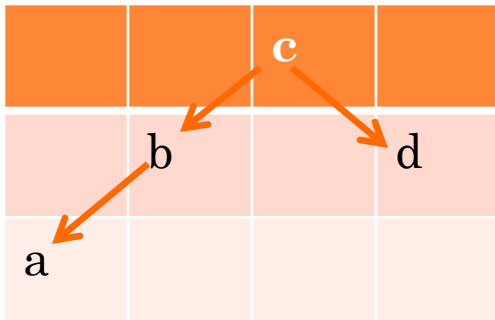
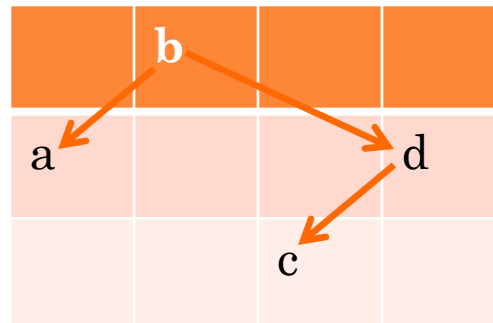
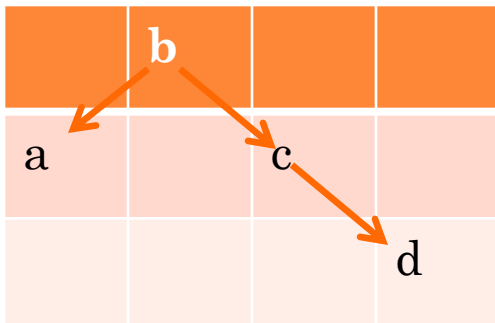


2 backbone vormen

AANTAL BST VERSCHIJNINGSVORMEN

Knopen(N)	N!	BST's	Min paden
1	1	1	1
2	2	2	2
3	6	5	1
4	24	14	4
5	120	42	6
6	720	132	4

DE 4 MIN PAD BST VORMEN VAN 4 KNOPEN



HOE CREËER IK EEN MIN PAD VORM VOOR MIJN BST BIJ INVOER?

- Bij toenemende N exponentiële toename aantal mogelijke BST's als waarden at random worden toegevoegd.
- Hoe te zorgen dat een min pad versie gevormd wordt en na toevoegingen en weglatingen ook blijft?
- Voor 1e inbreng zijn er methoden die dat bereiken en na wijzigingen methoden die:
 - Lokaal BST evenwichtiger maken: AVL bomen
 - Globaal de BST omzetten naar een beste balans: DSW algoritme

BST OPBOUWEN

VANUIT GESORTEERD ARRAY

- Een goed gebalanceerde BST kan makkelijk worden opgebouwd als alle waardes vooraf bekend zijn
- Zet de waardes in een array
- Sorteert het array
- Voeg elementen aan de BST toe door het gesorteerde array recursief met binaire zoekadressering te doorlopen

a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
							1							
			2								9			
	3				6				10				13	
4		5		7		8		11		12		14		15

BST OPBOUWEN

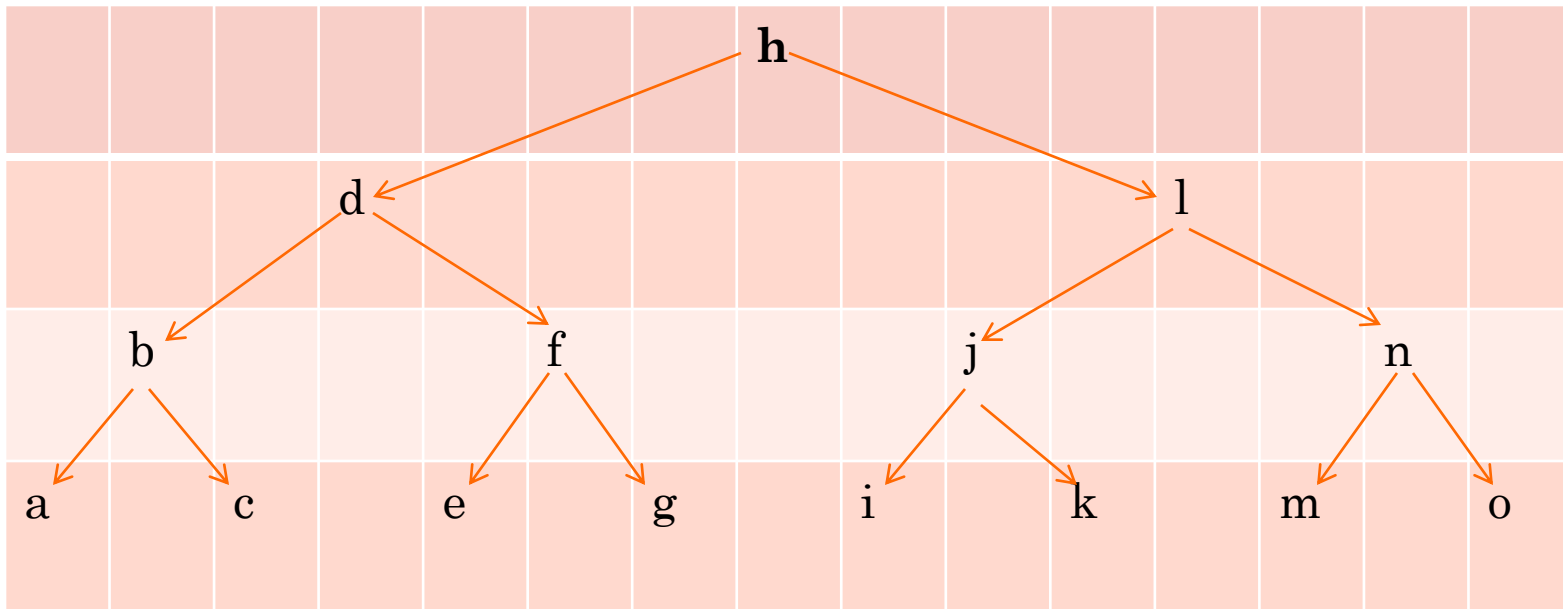
VANUIT GESORTEERD ARRAY-2

- Depth-first of niveau voor niveau toevoeging beide goed (maar 2 van de mogelijke gebalanceerde opbouw volgordes)

a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
							1							
			2								9			
	3				6				10				13	
4		5		7		8		11		12		14		15

a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
							1							
			2								3			
	4				5				6				7	
8		9		10		11		12		13		14		15

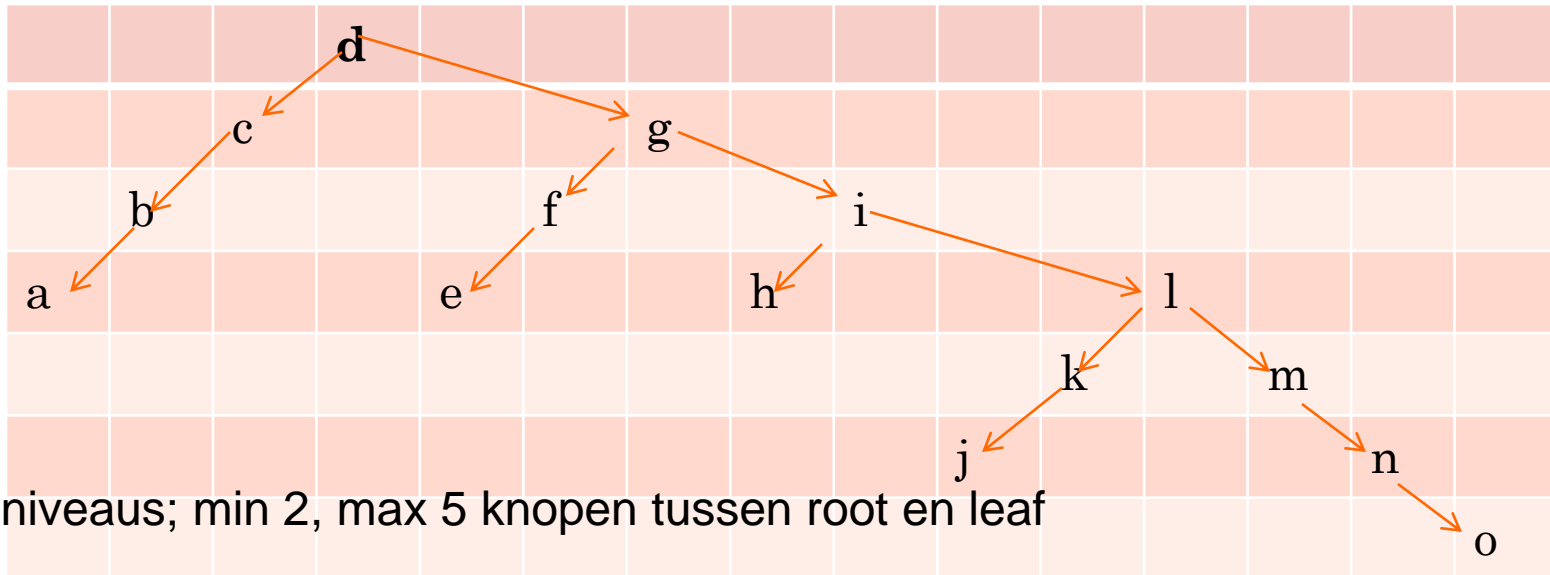
RESULTERENDE BST



4 niveaus; max 2 knopen tussen
root en leaf

HEROPBOUW SLECHT GEBALANCEERDE BOOM VIA GESORTEERD ARRAY

- Gesorteerd array kan worden opgebouwd door een bestaande BST (InOrder) te doorlopen en gepasseerde waarden in array weg te zetten.



- Array na InOrder doorloop:

a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
---	---	---	---	---	---	---	---	---	---	---	---	---	---	---

- Vanuit dit array BST gebalanceerd opbouwen:

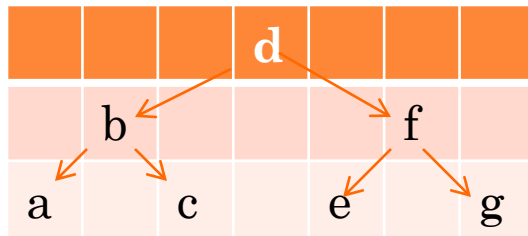
(HER)BALANCEREN VAN BST

- Hoe goed is de balans binnen een boom?
- Hoe karakteriseren we een optimale balans?
- Hoe goed is de balans onder een knoop?
- Hoe kunnen we de balans lokaal verbeteren?
- Hoe wordt de balans verstoord t.g.v. insert/delete?
- Kunnen we deze aanpassingen niet automatisch in de ADT opnemen?

HOE GOED IS DE BALANS BINNEN EEN BOOM?

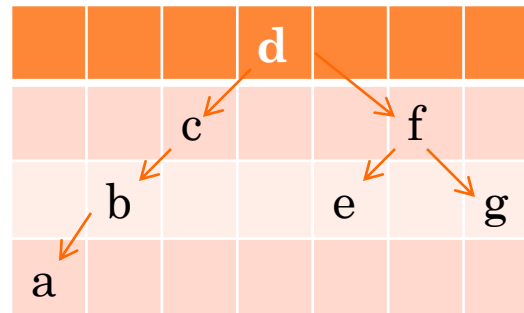
- Verschil in hoogte:
- Minimaal 0 alleen voor BST met 2^N-1 knopen
- Minimaal 1 voor BST's met ongelijk 2^N-1 knopen

Hoogte=3



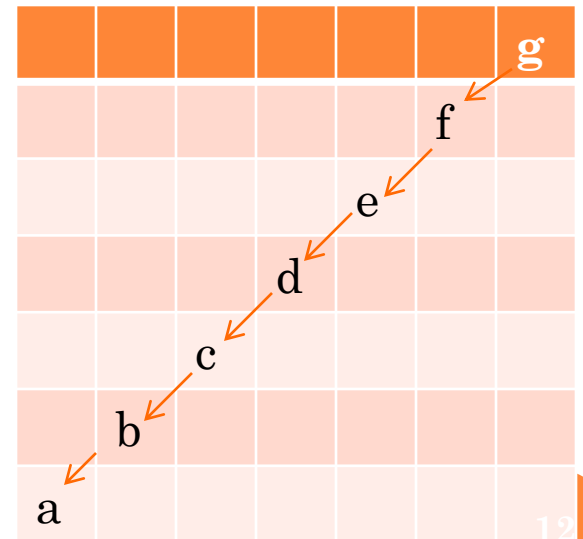
Bladeren op 1 niveau

Hoogte=4



Bladeren op 2 niveaus

Hoogte=7



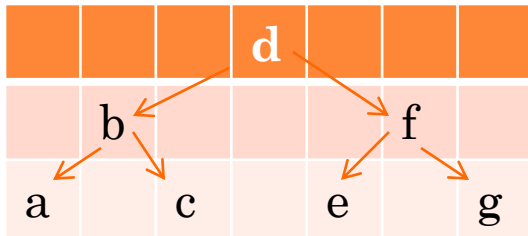
Blad op 1 niveau !

Kijken naar op hoeveel niveaus er
bladeren zijn is niet genoeg!

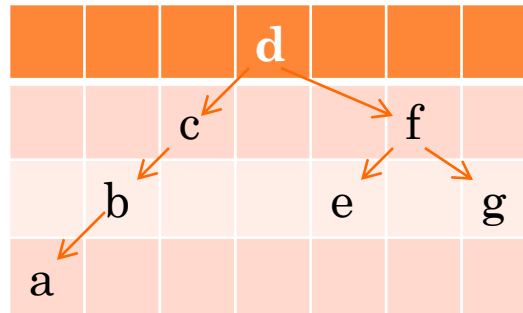
HOE GOED IS DE BALANS BINNEN EEN BOOM?

- Maximaal hoogteverschil voor subtrees van elke knoop:
- Minimaal 0 alleen voor BST met 2^N-1 knopen
- Minimaal 1 voor BST's met ongelijk 2^N-1 knopen
- Maximaal $N-1$ voor BST's met N knopen

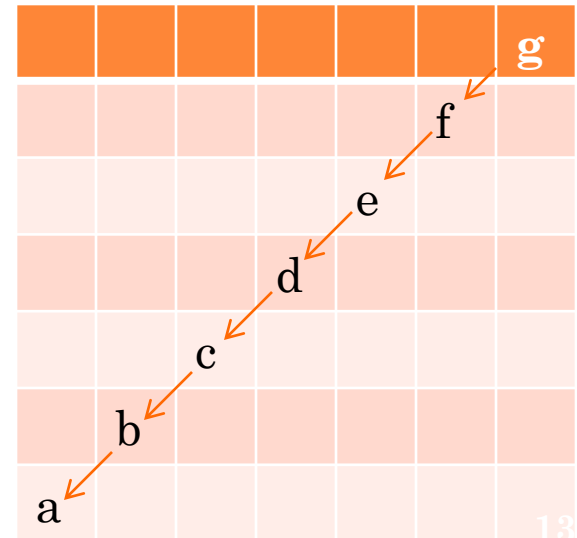
Max hoogteverschil=0



Max hoogteverschil=2



Max hoogteverschil=6

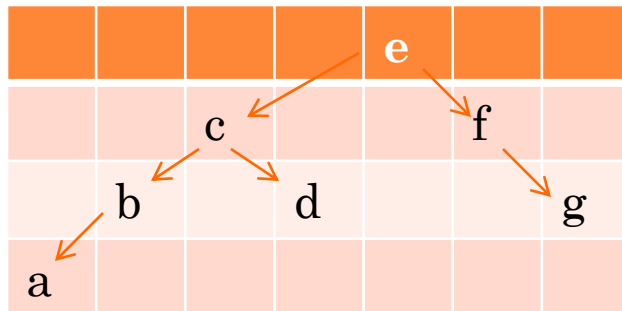


$$0 \leq \text{Maximaal hoogteverschil} \leq N-1$$

(HOOGTE) GEBALANCEERD (DEFINITIE)

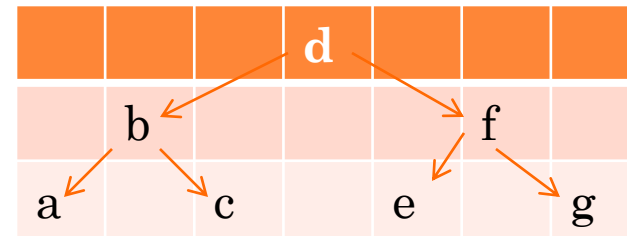
- Als: Maximaal hoogteverschil subtrees elke knoop 0/1
- Perfect gebalanceerd als tevens:
 - Alle bladeren op 1 of 2 niveaus
- Hoeft niet beste resultaat te zijn!!

Mogelijk AVL resultaat



Max hoogteverschil subtrees=1
Bladeren op 2 niveaus

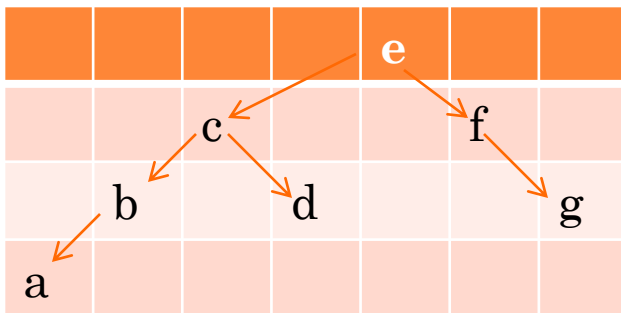
Zeker DSW resultaat



Max hoogteverschil subtrees =0
Bladeren op 1 niveau

LOCALE BALANS IN AVL BOOM

- Balans factor in elke knoop:
- Hoogte_rechter_subtree – hoogte_linker_subtree



Balansfactoren:

				-1		
		-1			1	
	-1		0			0
0						

TREE ALGORITMES VOOR HET BALANCEREN VAN BST'S

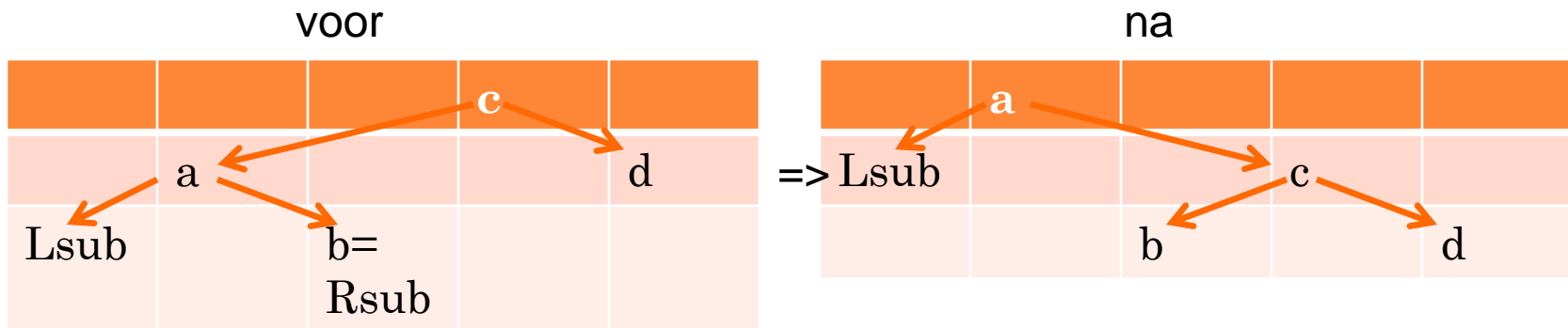
- Dure operatie om BST via array te balanceren $O(N)$
- Er is al vroeg gezocht naar wegen om een BST zonder tussenkomst van een array te balanceren op een $O(1)$ of $O(\log N)$ manier
- We bekijken vandaag:
 - - AVL trees
 - - DSW trees

PAUZE

AVL TREES

- Adel'son, Velskii en Landis (1962)
- Eigenschap: BST waarbij maximaal verschil in hoogte van subtrees 1 is
- Hoe verminder je de hoogte in een subtree?
- Via een zogenaamde rotatie
- Een rotatie is een lokale wijziging van de BST waarbij de (InOrder) volgorde behouden blijft maar lokaal de hoogte 1 vermindert (L/R) en 1 vermeerdert (R/L); het verandert dus de balancerings tussens de L en R subtrees onder een bepaalde knoop in de boom

RECHTS ROTATIE



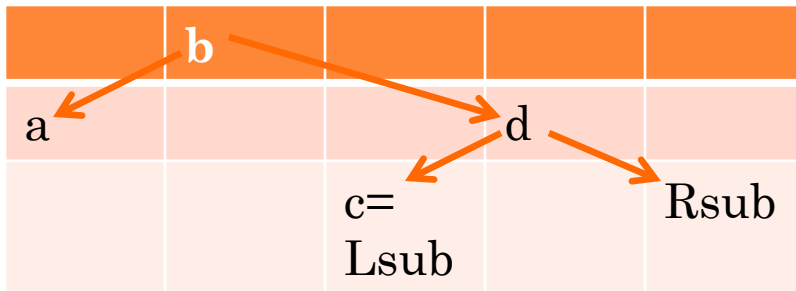
a en de linker subtree van a komt 1 niveau hoger;
 rechter subtree van a blijft op gelijke hoogte als linker subtree van c;
 c en rechter subtree c komen 1 niveau lager terecht



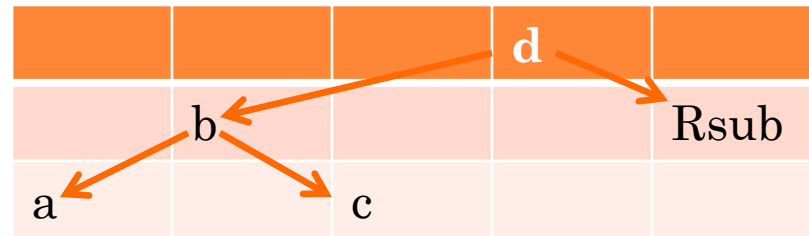
InOrder BST volgorde behouden

LINKS ROTATIE

voor



na



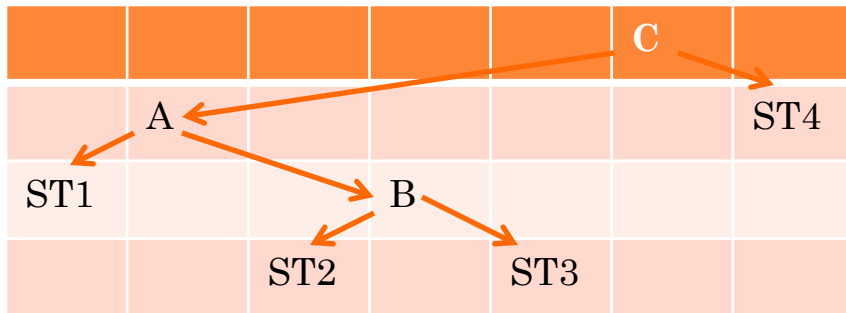
d en de rechter subtree van d komen 1 niveau hoger;
 linker subtree van d blijft op gelijke hoogte als rechter subtree van b;
 b en linker subtree a komen 1 niveau lager terecht



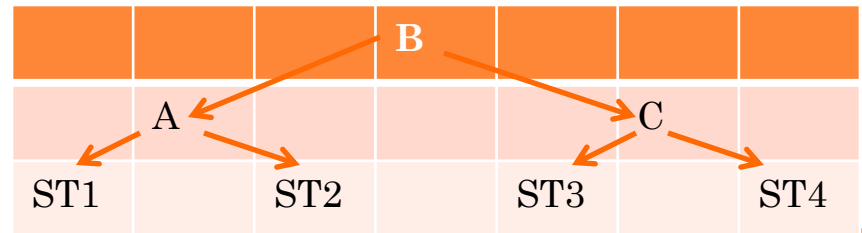
InOrder BST volgorde behouden

WELKE ROTATIES NODIG?

start



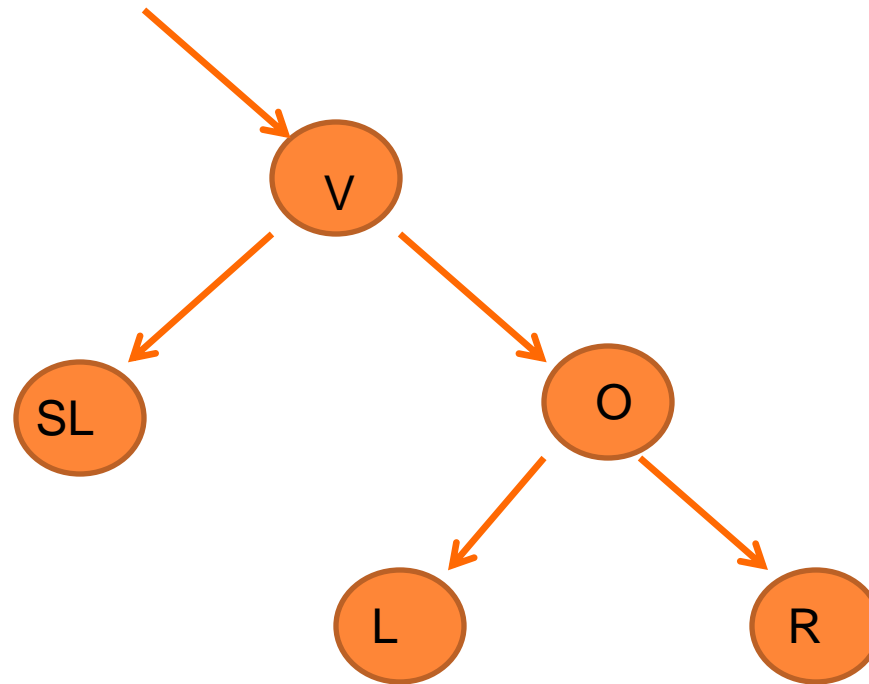
2xR en 1xL



DSW BALANCERING VAN EEN BST

- Day (1976) + Stout & Warren (1986)
- Maximaal gebalanceerd via louter rotaties
- Stap 1: roteren naar backbone structuur via rechts rotaties (tot alle linker subtrees leeg) in reverse order knoop volgorde (ROL)
- Stap 2: roteren naar maximaal gebalanceerd m.b.v. linksrotaties beginnend op het laagste nivo ($M=(N_{back}-1)/2$) steeds 2 opvolgende in backbone; herhalen met $M=(N_{back}-1)-M$ zolang $M>0$

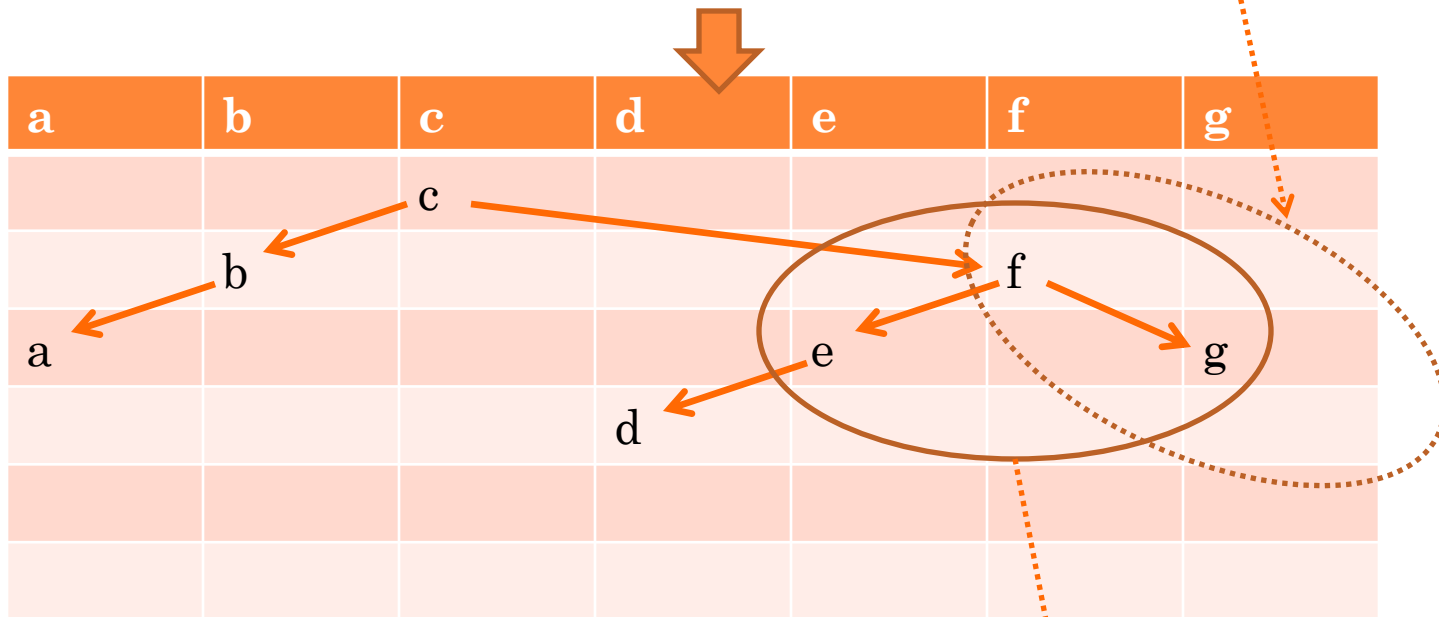
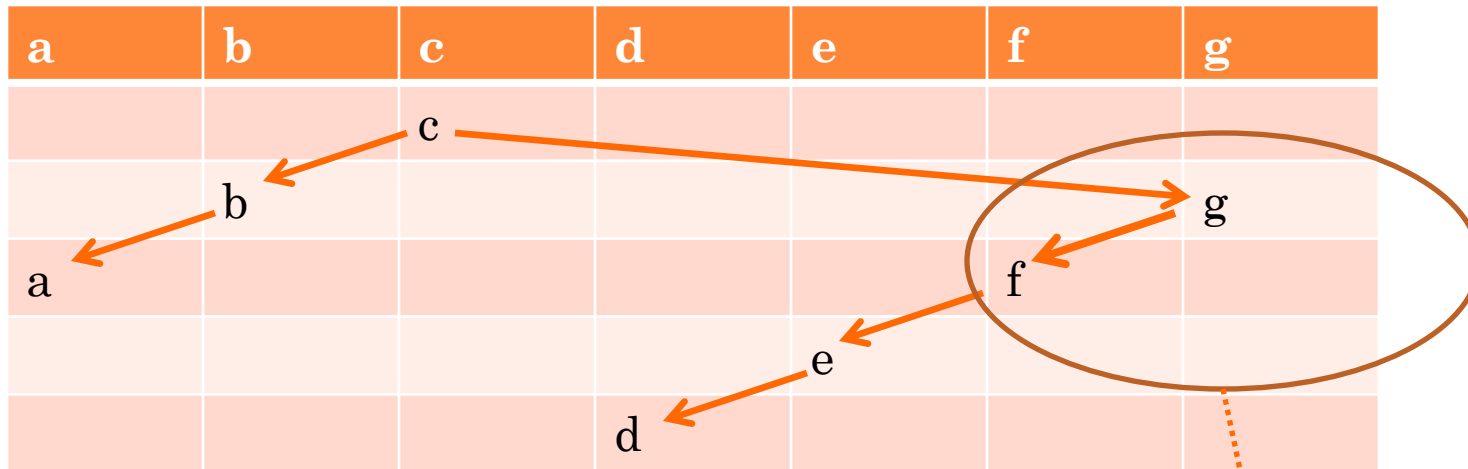
REVERSE INORDER PAD IN BST



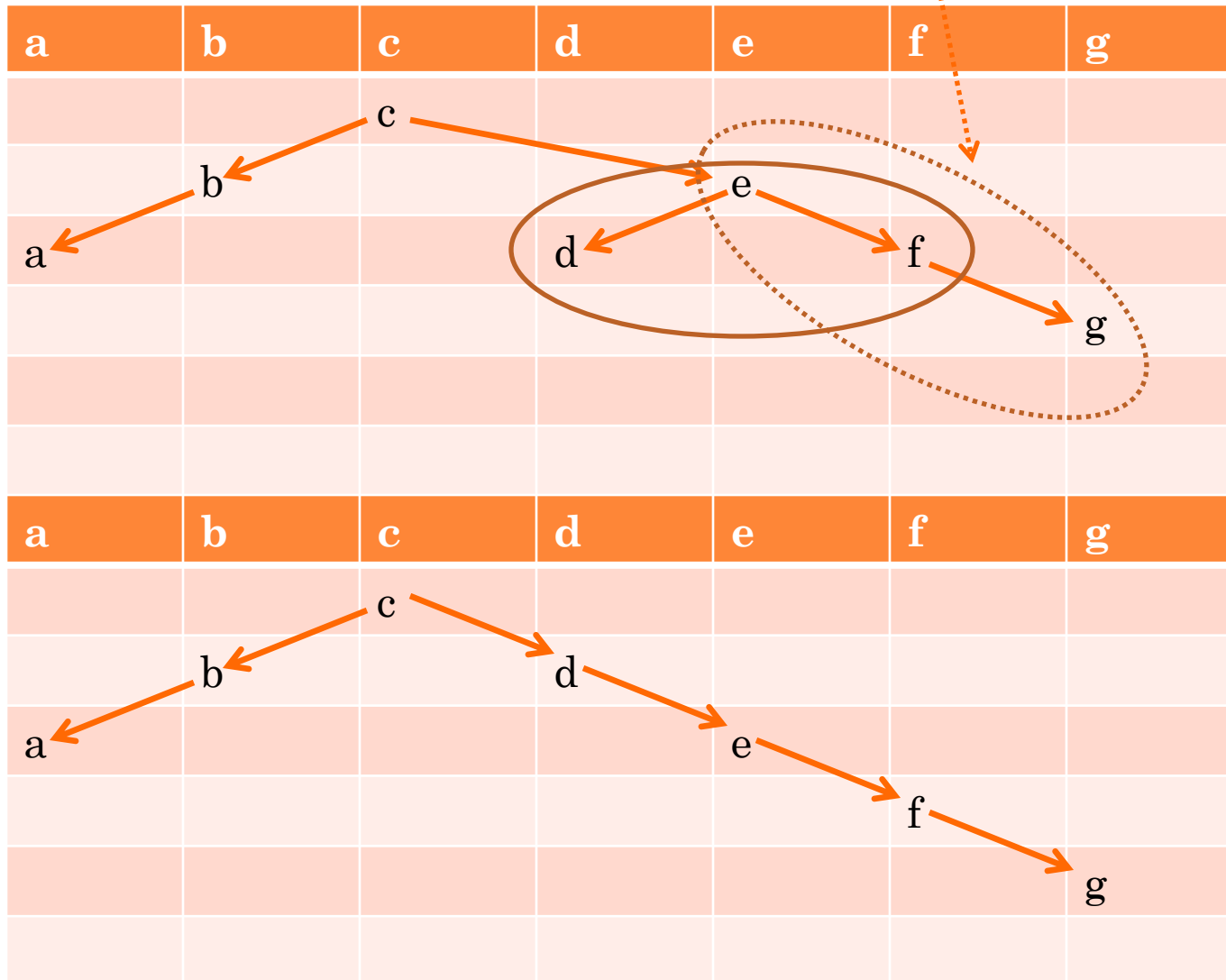
InOrder: ...SL,V,L,O,R

Reverse InOrder: R,O,L,V,SL,....

DSW VOORBEELD OP 7 KNOPEN BST



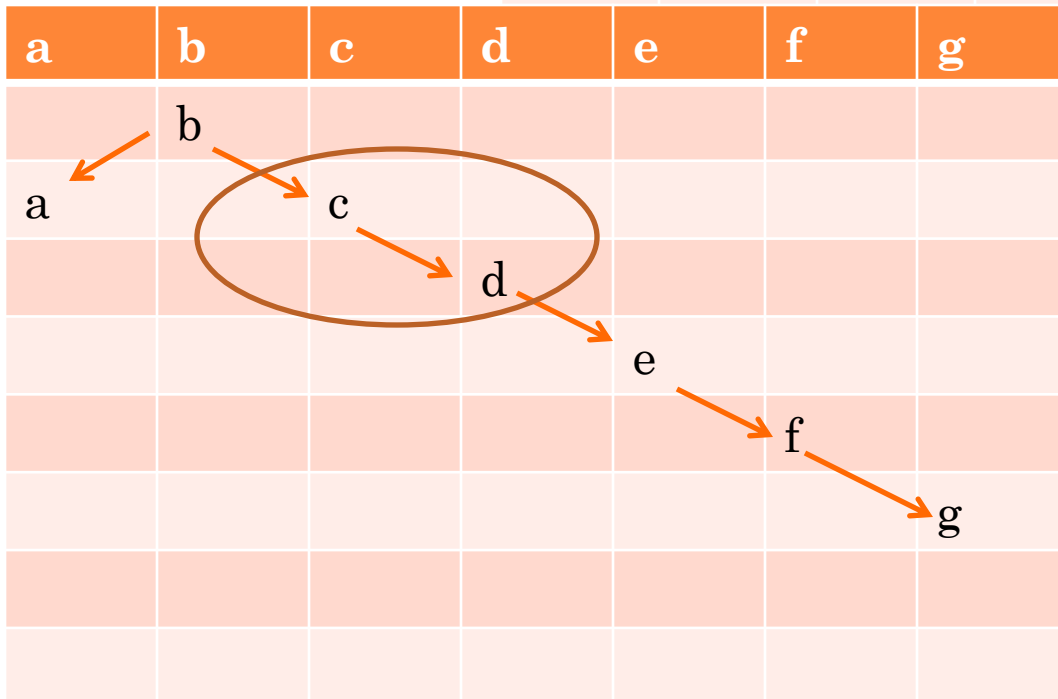
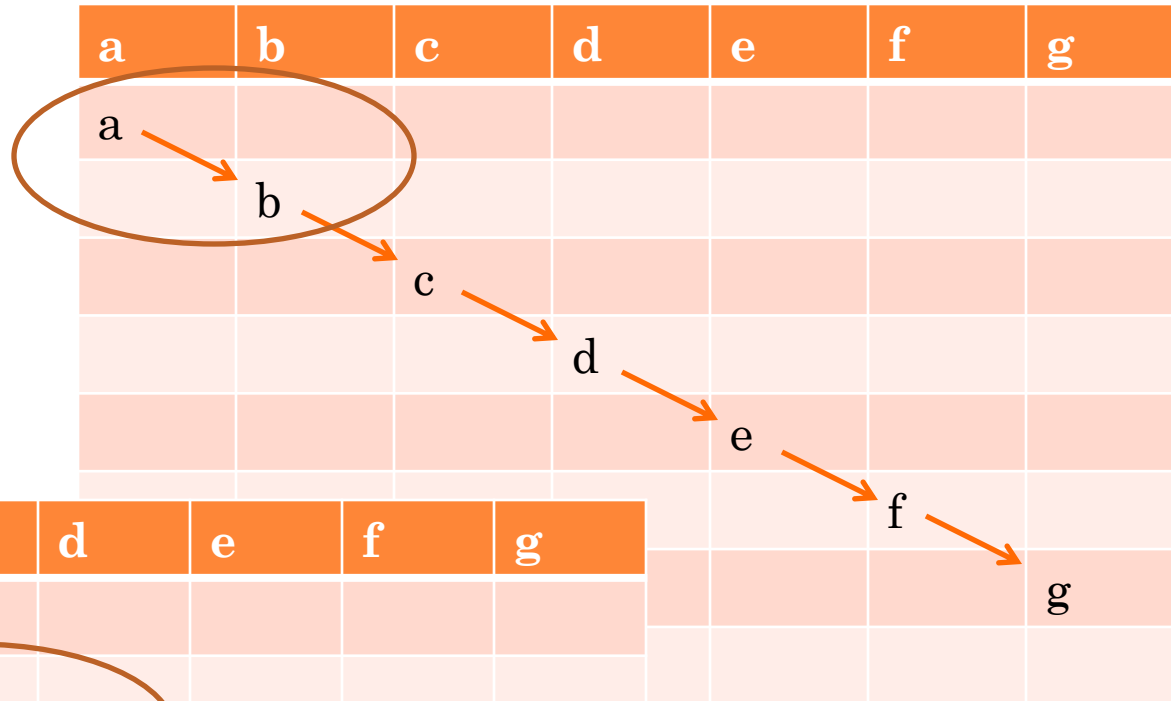
DSW VOORBEELD VERVOLG-1



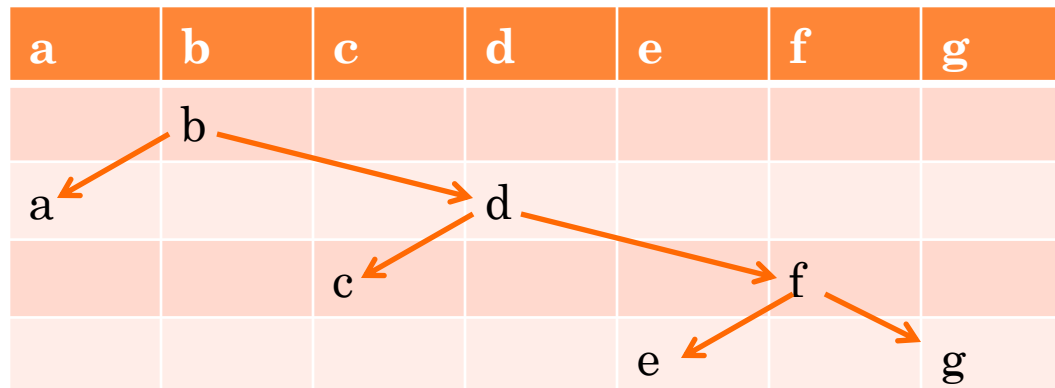
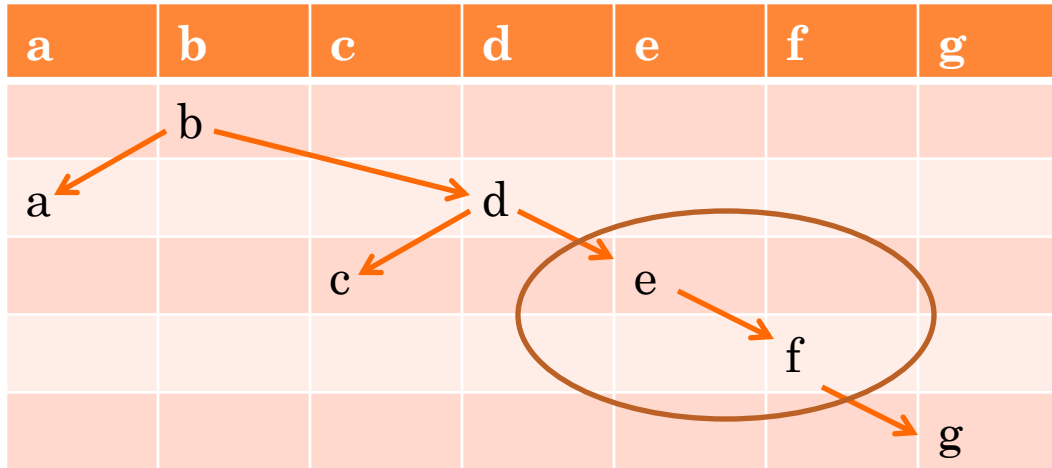
Doorgaan tot linker subtree voorste ook leeg -> backbone klaar

DSW VOORBEELD BACKBONE -> BALANCED

Start
compressie
fase

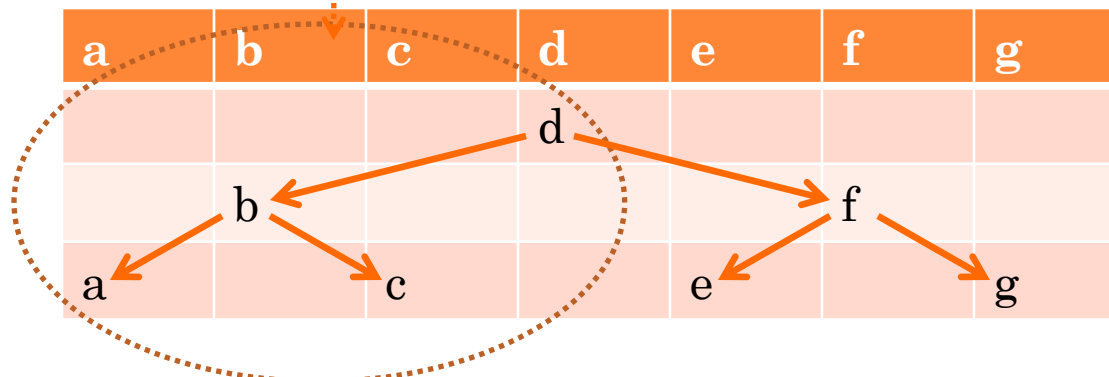
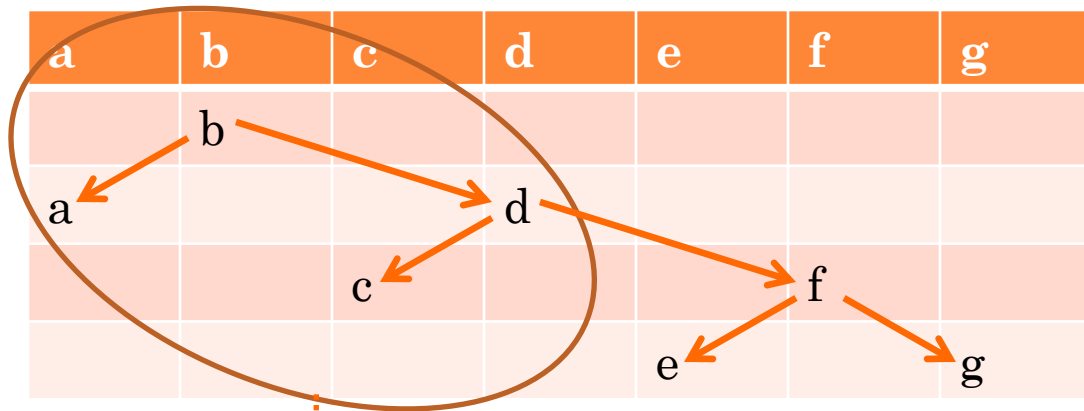


DSW VOORBEELD BACKBONE -> BALANCED 2



Hiermee laagste nivo afgewerkt

DSW VOORBEELD BACKBONE -> BALANCED 3



Hoogste nivo laatste linksrotatie -> klaar!

BST: THREADED VORM

VERSNELLEN INORDER AFLOPEN ZOEKPAD

- Variant van BST
- Oponthoud door nil-pointers tijdens doorloop
- Elke leaf heeft al 2 nil pointers
- Thread=zet adres InOrder opvolger in boom op plaats nil-pointer:
- Nil in linker child -> thread naar ouder knoop
- Nil in rechter child -> thread naar voorouder knoop waarvan huidige knoop verst in linker subtree zit
- Wat voor een adres in child pointer staat (pointer naar kind/ 1e niet lege InOrder opvolger) kan m.b.v. een extra bit onderscheiden worden

IS BINARY SEARCH SNELSTE ZOEKMETHODE?

- Vergelijk met zoeken in telefoongids
- Rekening houden met de verdeling over de plaatsnamen en het alfabet kies je een ingang meer voorin of meer achterin i.p.v. het midden
- Als je kennis hebt van de verdeling van de gesorteerde waardes kun je een betere gok doen dan steeds het midden
- De eerste stappen kun je na een aantal malen zoeken omzeilen m.b.v. een ingangstabel b.v. index waarbij de K of postcode 5000 begint
- Kan makkelijk enkele (begin)stappen uitsparen

OPGESLAGEN WAARDE EN ADRES VAN DEZE WAARDE

- Als onbekend is onder welk adres een waarde is opgeslagen helpt kennis dat de waardes gesorteerd zijn opgeslagen (binary search opzoek strategie mogelijk)
- Impliciet is er dan al iets van de waarde in het adres (index/key) doorgekomen
- Wat kun je doen als de op te bergen/op te roepen waarde expliciet samenhangt met het adres?

O(1) ZOEKMETHODEN IN DATA?

- Om in 1 keer een bepaalde key in data te vinden moet je data opslag structuren opzetten die een intern adres krijgen dat direct met de op te zoeken key samenhangt.
- Deze zogenaamde hash technieken komen later in dit college aan de orde
- Waardes hoeven dan zelfs niet gesorteerd te zijn