Een datastructuur met zoektijd $O(\log n)$ en constante update-tijd

J.H. van der Erf

RUU-CS-87-19
November 1987

Rijksuniversiteit Utrecht

Vakgroep informatica

Budapestlaan 6 3584 CD Utrecht
Corr. adres: Postbus 80.012 3508 TA Utrecht
Telefoon 030-53 1454
The Netherlands

INHOUD

Figuren

Een datastructuur met zoektijd O(log n) en constante update-tijd

J.H. van der Erf

Abstract

In deze scriptie wordt een datastructuur beschreven met zoektijd O(log n) en worst-case update-tijd O(1) wanneer de plaats bekend is waar de key toegevoegd resp. weggelaten moet worden.
Dit is een verbetering van reeds bekende oplossingen met zoektijd O(log n) en amortized update-tijd O(1) dan wel worst-case update-tijd $O(\log^* n)$.
De gepresenteerde methode gaat uit van een BB[α]-boom waarbij in de bladeren verzamelingen keys opgeslagen zijn ter grootte O(log n). De worst-case toevoeg-tijd O(1) wordt gerealiseerd door de gebruikelijke hoeveelheid herbalanceerwerk O(log n) te spreiden over Ω(log n) toekomstige toevoegingen die een enkele verzameling aankan alvorens zo'n verzameling gesplitst moet worden in gelijke helften hetgeen een insertie op de boom impliceert. De worst-case weglaat-tijd O(1) wordt gerealiseerd door middel van global-rebuilding.

keywords :
gebalanceerde binaire boom, BB[α]-boom, rotatie, bag, scanner, UNION-FIND-structuur, werkelijke resp. imaginaire balans, groeifactor, global rebuilding

HOOFDSTUK 0

Inleiding

Talrijke datastructuren zijn gebaseerd op gebalanceerde zoekbomen. Hun populariteit berust op de evenwichtige combinatie van worst-case zoektijd, nl. O(log n), en worst-case toevoeg- en weglaat-tijd, ook O(log n).
De toevoeging of verwijdering van een key bestaat normaliter uit de volgende 3 stappen :

i) de plaatsbepaling van de key in de boom. (O(log n))
ii) de eigenlijke toevoeging resp. verwijdering uit de structuur. (O(1))
iii) herbalancering van de boomstructuur opdat de zoektijd zo min mogelijk verslechterd. (O(log n))

In een aantal gevallen is de positie in de structuur waar een update plaats heeft bekend op het moment dat besloten wordt deze verandering uit te voeren. Zodoende blijft dan de eerste stap achterwege en komt de laatste stap voor verbetering in aanmerking waardoor de ondergrenzen voor de complexiteit van de updates niet meer gelden.

Harel en Lueker [3] geven een ingewikkeld algoritme waarmee een worst-case update-tijd O($\log^*$ n) bereikt wordt indien de positie waar de update plaats vindt bekend is. (Log^* is de inverse Ackerman-functie.)
Overmars [7] beschrijft een methode om te komen tot een amortized update-tijd O(1). Daartoe wordt uitgegaan van een gebalanceerde zoekboom waarbij in de bladeren *verzamelingen* keys ter grootte O(log n) worden opgeslagen (zie figuur 1). Wanneer zo'n verzameling te groot wordt, wordt deze gesplitst in 2 gelijke helften hetgeen een toevoeging op de boom betekent. Door de kosten te verdelen over de updates waarbij geen verzameling wordt gesplitst bereikt men een gemiddelde tijd O(1).
In deze scriptie wordt een dergelijke datastructuur gebruikt om te komen tot een worst-case update-tijd O(1) mits de positie van de update bekend is. Recentelijk is dit resultaat ook bereikt door Levcopoulos en Overmars [6], maar de daar gebruikte techniek is geheel anders dan hier toegepast wordt.

Bij de hier gebruikte methode worden niet alleen de kosten verdeeld, maar wordt de hoeveelheid herbalanceerwerk van de boom gespreid over toekomstige updates op beide helften van de gesplitste

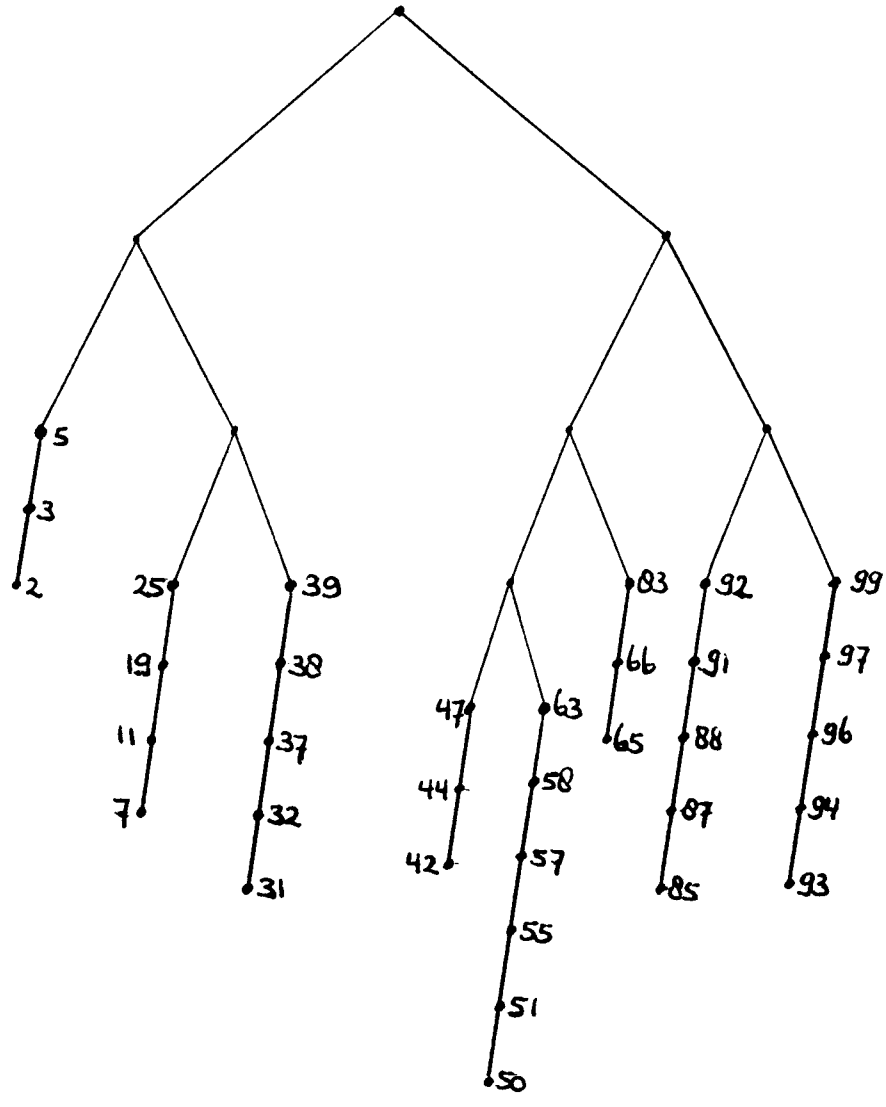
verzameling hetgeen in een worst-case tijd O(1) resulteert.
Om de werkwijze te vereenvoudigen worden in eerste instantie alleen toevoegingen op de datastructuur toegelaten. Later wordt de methode uitgebreid zodat ook verwijderingen toegestaan zijn. Bij dit laatste wordt gebruik gemaakt van de global-rebuilding-techniek.
Spreiding van het herbalanceerwerk over een aantal updates betekent dat er per gesplitste verzameling een wijzertje, scanner genaamd, bijgehouden dient te worden zodat bekend is hoever het herbalanceerwerk op het pad naar de wortel gevorderd is en men weet waar men bij de volgende update dit herbalanceerwerk dient te hervatten.

Met deze benaderingswijze gaan een aantal complicaties gepaard die betrekking hebben op

a) de aanwezigheid van meerdere scanners tegelijk in de boom.
   Een scanner wordt gegenereerd wanneer een bag te groot wordt en splitst. Bij splitsing van meerdere bags achtereen, houdt dit in dat er zich verschillende scanners tegelijk in de boom zullen bevinden.
b) de actualiteit van de balanswaarden van de interne knopen.
   Doordat het pad naar de wortel niet in een keer doorlopen wordt zullen hooggelegen knopen een balanswaarde bezitten die minder up-to-date is dan die van de laagst-gelegen knopen. Dit wordt nog versterkt door de aanwezigheid van meerdere scanners tegelijkertijd in de boom.
c) de verplaatsing van meerdere scanners ineens (o.a. bij rotaties).
   Wanneer bij een rotatie scanners, die staan bij de actieve knopen, verplaatst worden dan dient gegarandeerd te worden dat de informatie die deze scanners vertegenwoordigen intact blijft d.w.z. dat de balanswaarden van alle knopen die door een scanner bezocht horen te worden door die scanner precies één keer aangepast zullen worden.
d) de gebruikte rotaties en de functionele correctheid ervan.
   De rotaties zullen enigzins aangepast worden omdat bekende herbalanceringsalgoritmen uitgaan van exacte balanswaarden terwijl hier gewerkt wordt met balanswaarden met een zekere mate van onnauwkeurigheid.

Deze problemen zullen echter niet onoverkomelijk blijken.

Deze scriptie is als volgt opgebouwd.
In hoofdstuk 1 wordt een nauwkeuriger beeld gegeven van de voorgestelde aanpak en de problemen die daarbij optreden. Verder worden notaties, definities, aannames en andere vereiste voorkennis gegeven.
Hoofdstuk 2 gaat in op scanners. Aan de orde komen vragen als 'wat representeert een scanner precies', 'hoe worden verzamelingen scanners gerepresenteerd', 'hoe worden ze verplaatst'. Verder wordt aangegeven hoe het percentage scanners in een boom te begrenzen is.
Hoofdstuk 3 behandelt zaken betreffende de rotaties, zoals interpretatie van de balanswaarden i.v.m. de actualiteitsverschillen, aanpassing van de bekende herbalanceringsstrategie van Blum en Mehlhorn [1] en de correctheid ervan.
Vervolgens komt in hoofdstuk 4 de uitbreiding met deletions door toepassing van global rebuilding aan de orde.
Tenslotte geven we in hoofdstuk 5 enkele opmerkingen over gemaakte keuzen en conclusies omtrent het bereikte resultaat.
Als allerlaatste volgt een appendix met een Pascalprogramma dat ten behoeve van de leesbaarheid slechts toevoegingen aankan, alsmede een summiere toelichting bij de gebruikte procedures.



Figuur 1

binaire boom met in de bladeren verzamelingen keys

HOOFDSTUK 1

Oriëntatie

Om een beter beeld te krijgen van de oplossingswijze wordt eerst de oplossing van Overmars [7] beschreven waarbij de gemiddelde update-tijd O(1) is. Vervolgens wordt globaal de werkwijze beschreven om tot een worst-case insertie-tijd O(1) te komen. (De uitbreiding met deletions komt in hoofdstuk 4 aan bod.) Daarna wordt bekeken welke complicaties zich daarbij voordoen zodat vervolgens stapgewijs naar de uiteindelijke oplossing toegewerkt kan worden.

De door Overmars gepresenteerde methode, om tot een gemiddelde update-tijd O(1) te komen, luidt als volgt:
De gebruikte datastructuur is een aangepaste gebalanceerde zoekboom. In de bladeren van de boom worden niet de keys zelf opgeslagen maar *verzamelingen* keys, in het vervolg bags genaamd, ter grootte O(log n). Alle keys in bag B_i zijn kleiner dan de keys in bag B_{i+1} en de keys van een bag zijn geordend opgeslagen in een dubbel-gelinkte lijst. Iedere bag wordt in de boom gerepresenteerd door het grootste element. Deze bags zijn in eerste instantie $\lceil \log n \rceil$ groot en er wordt een grootte van hoogstens $2 \cdot \lceil \log n \rceil$ toegestaan.
Bij het zoeken naar een element wordt dus eerst de boom doorlopen en vervolgens de betreffende lijst zodat de zoektijd gegeven wordt door O(log n) + O(log n) = O(log n).
Bij een toevoeging wordt de key in bag B_i in O(1) tijd in de lijst geplaatst. Wanneer een lijst grootte $2 \cdot \lceil \log n \rceil$ bereikt wordt deze bag in twee gelijke helften gesplitst en wordt een bag aan de boom toegevoegd. De hoeveelheid werk komt voor rekening van de $> \lceil \log n \rceil$ toevoegingen die plaatshebben vanaf de laatste splitsing of constructie van bag B_i. Dit betekent een gemiddelde tijd van O(1) per insertie.
Om objecten te verwijderen wordt de werkwijze enigzins aangepast en wordt als volgt te werk gegaan.
Zij n_0 de grootte van de nieuwgebouwde structuur en laat N_i het aantal inserties zijn dat heeft plaatsgevonden. Er wordt toegestaan dat het aantal elementen in een bag hoogstens $2 \cdot \lceil \log (n_0 + N_i) \rceil$ is.
Een deletion van een element uit een bag wordt gerealiseerd door de key uit de dubbel-gelinkte lijst te verwijderen. Indien dit betekent dat de bag leeg raakt dan wordt *geen* (bag)deletion op de boom

uitgevoerd aangezien de hoeveelheid werk die daarmee gemoeid is veel meer dan O(1) bedraagt. Door na $\frac{1}{2}\cdot(n_0 + N_i)$ deletions de gehele structuur opnieuw te bouwen wordt gegarandeerd dat de baggrootte O(log n) is. Omdat het opnieuw bouwen van de gehele structuur O(n) kost en $n = \frac{1}{2}\cdot(n_0 + N_i)$ betekent dit gemiddeld O(1) per deletion. Een worst-case deletion-tijd O(1) is te bereiken door de opbouw van de nieuwe structuur te verdelen over een aantal deletions dat volgt. De worst-case insertie-tijd blijft echter Ω(log n).

Uitgaande van bovenbeschreven datastructuur ligt het voor de hand om de uitbreiding van een gebalanceerde boom met bags ter grootte O(log n) te gebruiken om de hoeveelheid herbalanceerwerk van O(log n) bij een update op een gebalanceerde boom te spreiden over de updates op een bag waardoor een worst-case update-tijd O(1) bereikt wordt.
Ook nu vereenvoudigen we ons probleem weer door in eerste instantie alleen inserties op de datastructuur toe te staan en de methode naderhand uit te breiden met deletions. Bij dit laatste wordt gebruik gemaakt van een techniek genaamd 'global rebuilding'. De hoofdgedachte bij global rebuilding is om verstoringen van de ideale gedaante van de datastructuur ten gevolge van updates (hier deletions) te tolereren zolang deze verstoring beperkt blijft. De oude structuur wordt vervangen door een tussentijds opgebouwde nieuwe, met een efficiëntere gedaante, zodra men kan verwachten dat de oorspronkelijke structuur te zeer van gedaante is verslechterd. (Zie hoofdstuk 4 voor een nauwkeuriger beschrijving.)

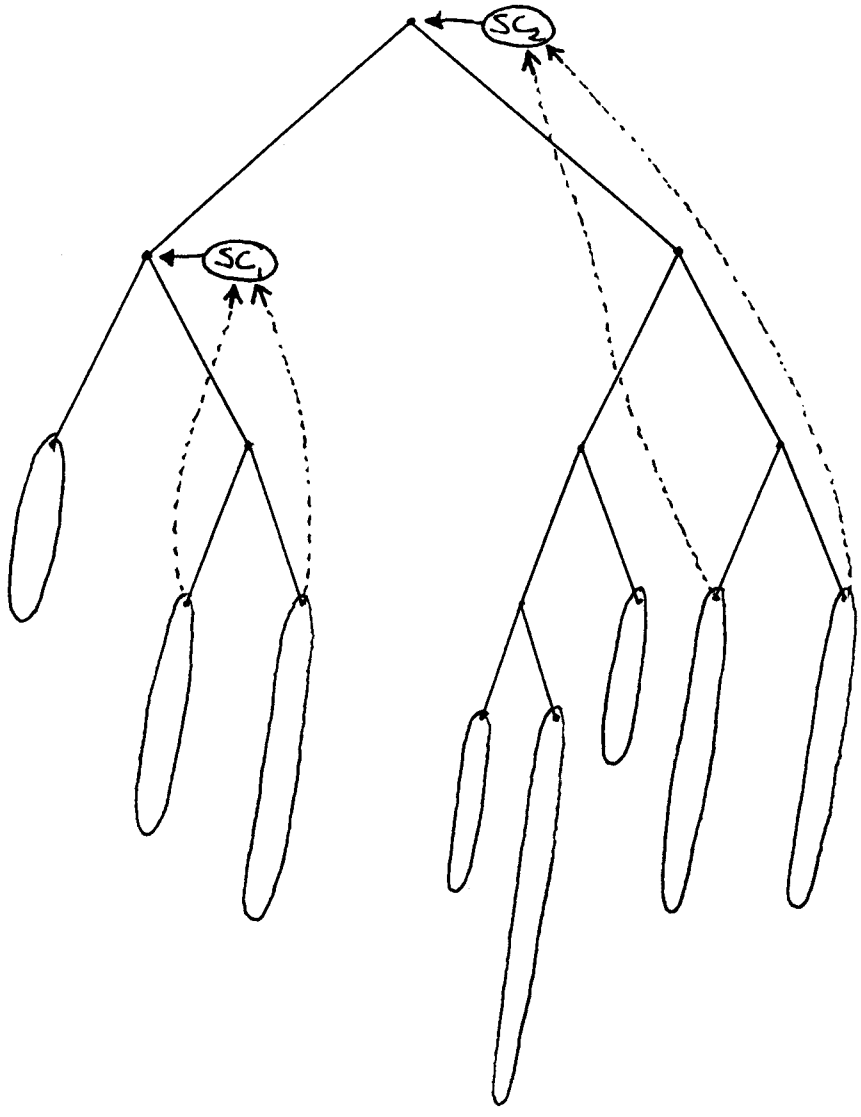
De benaderingswijze

We gaan uit van een gebalanceerde boom die uitgebreid is met bags ter grootte O(log n).
Zoeken in de datastructuur komt neer op het doorlopen van een pad in de boom gevolgd door positiebepaling van de key in de dubbel-gelinkte lijst. Hiermee is O(log n) + O(log n) = O(log n) tijd gemoeid.
Toevoeging van een key geschiedt nu door de key in de dubbel-gelinkte lijst te zetten en de grootte van de bag aan te passen. Dit kost O(1) werk. Indien de bag te groot wordt is het nodig om deze te splitsen in gelijke helften B_1 en B_2 hetgeen een bagtoevoeging in de boom impliceert. In plaats van de gebruikelijke handelswijze om daarbij het gehele pad naar de wortel ineens te herbalanceren, gaan we deze hoeveelheid werk verdelen over Ω(log n) toevoegingen op B_1 en B_2, die nodig zijn alvorens B_1 of B_2 zelf weer aan splitsing toe is, waardoor er per toekomstige insertie in B_1 of B_2 O(1) aan herbalanceerwerk uitgevoerd wordt. Verder kunnen die Ω(log n) toevoegingen gebruikt worden om per keer O(1) werk te verrichten aan de bepaling en het bijhouden van de middens van B_1 en B_2 zodat een splitsing van een bag ook slechts O(1) kost omdat het midden van de lijst niet meer bepaald hoeft te worden.

Spreiding van de hoeveelheid herbalanceerwerk over toekomstige toevoegingen aan beide deelbags betekent dat er bijgehouden dient te worden hoever men hiermee gevorderd is. Daartoe wordt bij de splitsing van bag B een wijzertje, scanner genaamd, gecreëerd die toegewezen wordt aan beide deelbags. Deze scanner wijst in de boom aan tot hoever de balanswaarden van knopen op het pad naar de wortel bijgewerkt zijn en waar men een volgende keer het herbalanceerwerk moet hervatten.
Bij de toevoeging van een key wordt dus eerst de key aan de dubbel-gelinkte lijst (bag) toegevoegd, daarna wordt de baggrootte opgehoogd, vervolgens wordt een constante hoeveelheid werk gedaan zodat op tijd het midden van de bag bekend is en tenslotte wordt eventueel de bag gesplitst of de aanwezige scanner een constante afstand verplaatst (d.w.z. O(1) herbalanceerwerk wordt verricht). In totaliteit betekent dit een worst-case insertie-tijd O(1).
Figuur 2 toont een boom met twee recentelijk gesplitste bags waarbij de deelbags een verwijzing naar de bijbehorende scanner bevatten.

Deze werkwijze om m.b.v. scanners de hoeveelheid herbalanceerwerk te spreiden over een aantal updates roept een aantal problemen op. Hier volgt ter oriëntatie een opsomming.

- Het pad naar de wortel wordt niet in één keer doorlopen en dus hebben hooggelegen knopen balanswaarden die minder up-to-date zijn dan laaggelegen knopen.
- Een scanner kan lange tijd stilstaan indien op de bijbehorende deelbags geen updates plaatshebben.



Figuur 2
boom met scanners

Omdat andere scanners zich daardoor niet laten beïnvloeden mag men zich afvragen of dit geen nadelige gevolgen heeft voor de balans van de gehele structuur na verloop van tijd.

- Er kunnen zich meerdere scanners bij één en dezelfde knoop bevinden. Er is daarom een geschikte datastructuur nodig voor *verzamelingen* scanners en wel zo dat aan het volgende voldaan is.
  a) Bij rotaties kunnen locale verzamelingen scanners in O(1) tijd verplaatst worden.
  b) Bij gewone updates op de bags kunnen scanners 'afzonderlijk' in O(1) tijd verplaatst worden.
- De aanwezigheid van meerdere scanners in de boom betekent dat bij een rotatie extra rekening dient te worden gehouden met de actualiteitsverschillen in balanswaarden van knopen die betrokken zijn bij de rotatie.
  a) Hoe wordt nu de keuze tussen een enkele of dubbele rotatie bepaald? (Bekende rotatiecriteria [1] zijn nu niet rechtstreeks toepasbaar.)
  b) Hoe wordt bij rotatie de consistentie van de structuur (i.h.b. de balanswaarden) behouden? (De balanswaarden zijn immers 'actualiteitsafhankelijk' en locale scanners worden gedurende een rotatie verschoven.)
- Indien *verzamelingen* scanners opgeschoven worden (bijvoorbeeld tijdens een rotatie) dan betekent dit voor de balanswaarden een verandering die groter is dan gebruikelijk. Men moet zich afvragen of de balansverstoring daarbij niet al te groot is opdat deze met een enkele of dubbele rotatie gecorrigeerd kan worden.

Als zoekboom is gekozen voor een BB[α]-boom. Het is niet uitgesloten dat ook andere boomsoorten voor een soortgelijke herbalanceringswijze in aanmerking komen. Hier is echter geen onderzoek naar gedaan.

De volgende secties van dit hoofdstuk zijn deels bedoeld als vluchtige kennismaking met hetgeen nog behandeld zal worden en deels als presentatie van benodigde kennis.

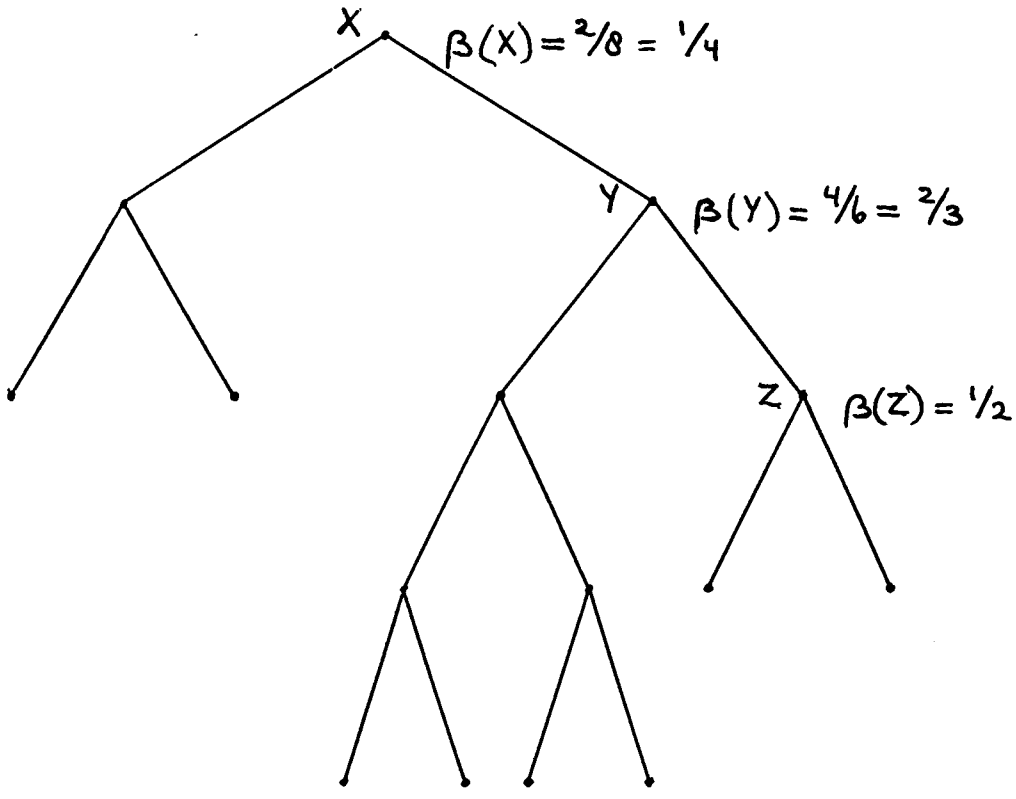
1.1. BB[α]-bomen

Een van de meest gebruikte datastructuren om grote hoeveelheden gegevens in op te slaan zijn gebalanceerde bomen. Hiervoor zijn meerdere redenen aan te geven. Gebalanceerde bomen zijn geschikt om gegevens *geordend* op te slaan op een manier dat de zoektijden vrij klein zijn, nl. O(log n) bij n elementen. Verder geldt dat gebalanceerde bomen een dynamisch karakter hebben dwz. ze zijn zeer geschikt voor verzamelingen die sterk aan veranderingen onderhevig zijn.
Er zijn allerlei soorten bomen die onderling sterk verschillen in de manier waarop de gegevens over de structuur verdeeld zijn. Zo variëert de graad van de interne knopen (dit is het aantal vertakkingen van een knoop dat toegestaan is) van boom tot boom. Verder bestaan bomen waarbij eisen worden gesteld aan het verschil in hoogte, de verhouding in het aantal keys in de deelbomen of het verschil in lengte tussen het langste en kortste pad.
BB[α]-bomen zijn binaire bomen, dwz. de graad van de interne knopen is 2, met als eigenschap dat voor iedere interne knoop geldt dat de fractie keys in de linkerdeelboom in het interval (α , 1− α) ligt (waarbij α gekozen is in het interval (0, 1/2)).
In sommige gevallen kiest men ervoor om bij àlle knopen (dus ook de interne) keys op te slaan, maar meestal worden de interne knopen alleen gebruikt voor het opslaan van tussenliggende waarden ten behoeve van het zoekalgoritme en worden keys in de bladeren opgeslagen. Hier wordt er van uitgegaan dat de keys alleen in de bladeren worden opgeslagen.
Volledigheidshalve volgt hier een definitie van een BB[α]-boom.

Zij T een binaire boom (met wortel T). Met T(K) wordt de subboom met wortel K bedoeld.
Het aantal bladeren in boom T(K) geven we aan met |K|. Dus |T|=n.
De linkerzoon van knoop K noteren we als K_L.

Definitie

De balans β(K) van knoop K wordt gedefinieerd als $|K_L|/|K|$.



Figuur 3
BB[1/4]-boom

Het is niet moeilijk in te zien dat geldt $0 \leq \beta(K) \leq 1$.

Definitie

Zij $\alpha \in (0, 1/2)$.

Een binaire boom T is een BB[α]-boom indien voor iedere interne knoop K geldt dat $\beta(K) \in (\alpha, 1-\alpha)$.

Dit betekent voor bijvoorbeeld BB[1/4]-bomen dat bij iedere interne knoop minstens 1/4 en maximaal 3/4 van het aantal keys zich bevindt in het linkergedeelte. (En dus geldt hetzelfde voor het aantal keys in het rechtergedeelte. Zie figuur 3.) In Blum en Mehlhorn [1] wordt aangegeven hoe men, uitgaande van de balanswaarden van de knoop en beide kinderen, de balans kan herstellen m.b.v. een enkele of dubbele rotatie.

1.2. Imaginaire en werkelijke balanswaarden

Aangezien de balanswaarden in de knopen niet overal even actueel zijn en er bovendien tijdens rotaties wellicht met locale scanner(set)s geschoven wordt ligt het voor de hand om een onderscheid te maken tussen imaginaire en werkelijke balanswaarden.

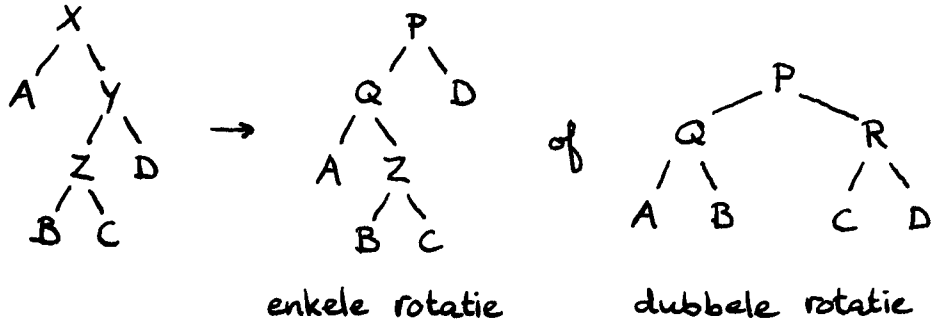
Onder een imaginaire balanswaarde wordt verstaan de balanswaarde zoals die bij de knoop bekend is en die door het algoritme bepaald wordt op grond van locaal opgeslagen waarden (lsize en rsize).
Onder de werkelijke balanswaarde van een knoop verstaan we de balans, zoals die in werkelijkheid is, van de totale subboom die opgeslagen is in het geheugen.

Door aan een aantal voorwaarden te voldoen kan er voor gezorgd worden dat het relatieve verschil tussen beide soorten balanswaarden klein genoeg is. De hoeveelheid data die bij een rotatie verplaatst wordt is dan nauwkeurig genoeg te benaderen uit de imaginaire balanswaarden. Een manier om dit te bereiken is ervoor te zorgen dat het percentage gesplitste bags in iedere deelboom binnen zekere grenzen blijft.
Wanneer we de boombladeren/bags in een (natuurlijk-geordende) dubbel-gelinkte lijst zetten is eenvoudig na te gaan of er een nabijgelegen bag recentelijk gesplitst is door te inspecteren of er een scanner aan zo'n bag gekoppeld is. Door splitsing van een bag uit te stellen indien er een bag binnen constante afstand m recent gesplitst is, is het percentage scanners (en dus het aantal gesplitste bags) te reduceren. Van uitstel tot splitsing komt geen afstel en de baggrootte blijft daarbij O(log n) wanneer de snelheid waarmee scanners verplaatst worden met een constante factor vermenigvuldigd wordt en een prioriteitsvolgorde opgelegd wordt aan dichtbij-elkaar-gelegen bags die wachten op toestemming om te splitsen.

1.3. Afkortingen, notaties

- α	:	van BB[α]; de nagestreefde balans
- $\alpha(X)$	:	de (opgeslagen) balans van knoop X
- $\alpha'(X)$	:	de α-waarde van X in werkelijkheid
- A, B, C, ...	:	wortels van subbomen (de passieve knopen)
- X, Y, Z, P, Q, R	:	actieve knopen
- T(X), T(A)	:	subboom met wortel X resp. A
- |A|, |Q|	:	het aantal bladeren in T(A) resp. T(Q)

Bij de rotaties benoemen we de knopen als volgt.



1.4. Knopen en scanners

Aan knopen zijn de volgende waarden bekend:

parent: een verwijzing naar de ouder van de knoop in de boom.

rank: de grootte van de subboom zoals die aan de knoop bekend is.

scan: Bij bladeren is dit een verwijzing naar de scanner die bij bagsplitsing aan de bag is toegewezen.
Bij interne knopen is dit een verwijzing naar de scannerset die staat bij deze knoop.

Verder kennen interne knopen nog de waarden:

key: een waarde ten behoeve van het zoekalgoritme.

lchild, rchild: een verwijzing naar beide kinderen.

lsize, rsize: de grootte van beide deelbomen voor zover de knoop hiervan op de hoogte is.
Merk op dat lsize resp. rsize *niet* noodzakelijk gelijk is aan de rank-waarde van lchild resp. rchild !!

Bladeren kennen nog de waarden:

size: het aantal keys in de desbetreffende bag. (Merk op dat de betekenis van 'lsize' en 'rsize' bij interne knopen geheel verschillend hiermee is.)

first: een verwijzing naar de lijst van elementen waaruit de bag bestaat.

mid: een pointer die op tijd verwijst naar het midden van de bag.

mn: het rangnummer in de lijst van het element waar mid naar wijst.

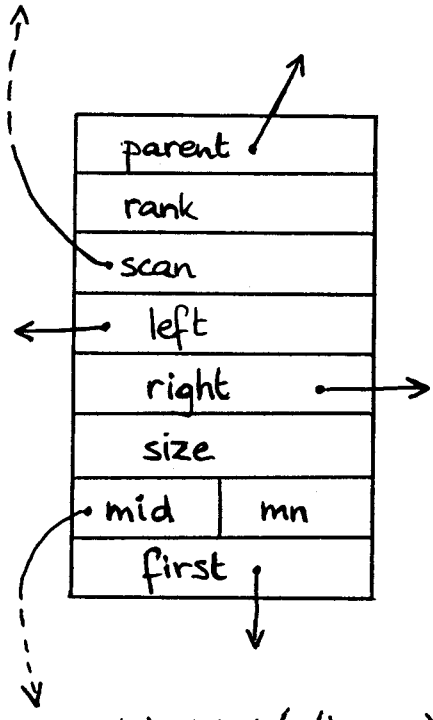
left, right: zijn pointers die wijzen naar bladeren in de boom met het doel snel de nabijgelegen bags te kunnen inspecteren. Left wijst naar de voorganger bij natuurlijke ordening op de bladeren. Right wijst naar de opvolger.

De imaginaire balans van de (interne) knoop wordt nu bepaald uit lsize en rsize (of rank). De werkelijke balans moet met een zekere mate van onnauwkeurigheid geschat worden aan de hand van de bekende imaginaire balanswaarde.

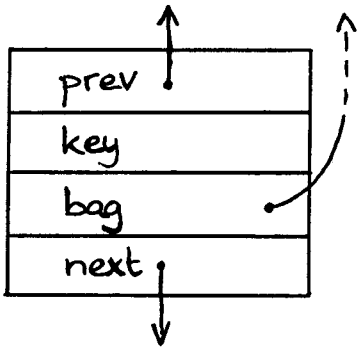
Een scanner kent, behalve een aantal waarden t.b.v. de datastructuur voor scannersets, minimaal de volgende waarden:

edge: de positie in de boom die aangeeft hoever het pad naar de wortel reeds doorlopen is.

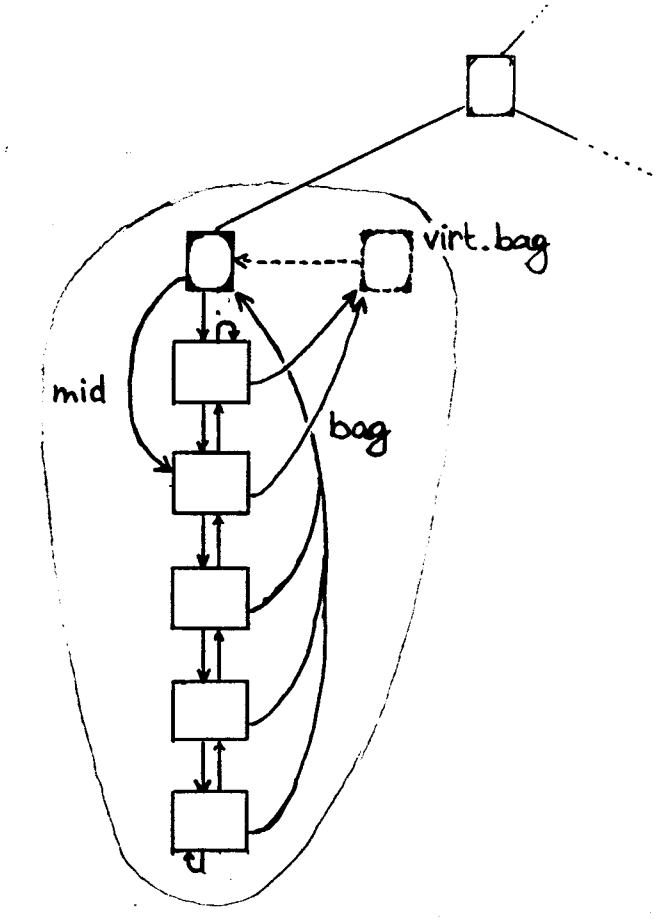
owner(s): een pointer naar de deelbags die bij splitsing ontstaan zijn en waarbij de scanner gecreëerd werd. Wanneer de scanner het gehele pad doorlopen heeft en merkt dat zijn functie vervuld is, wordt dit doorgegeven aan beide deelbags om een volgende splitsing mogelijk te maken.



(a) blad (/knoop)



(b) element



(c) bag

Figuur 4

1.5. Bags

Een bag bestaat uit een dubbel-gelinkte lijst van O(log n) elementen. Updates op de bags moeten in constante tijd te realiseren zijn en we gaan daarbij ervan uit dat de plaats van de update bekend is. Dit heeft tot gevolg dat een element de volgende waarden kent:

key: (spreekt voor zich.)

prev, next: wijzen naar een element met een hogere resp. lagere keywaarde (evt. dummy's).

bag: verwijst vanuit het element naar het boomblad dat de bag representeert zodat aanpassing van de size-, mid- en mn-waarde van de bag/knoop in constante tijd kan geschieden.

Aangezien splitsing van een bag eveneens in O(1) tijd dient te gebeuren gaan we als volgt te werk.
Na afloop van splitsing van bag B in bags B_1 en B_2 zullen we de mid-pointers van de deelbags (resp. de bladeren) B_1 en B_2 laten wijzen naar het begin van de lijsten. Het 'bag'-veld van het eerste element van zo'n lijst L_i laten we wijzen naar een virtuele bag V_i. Zo'n virtuele bag wordt bij splitsing van bag B_i de representant van een deelbag in de boom (die zal bestaan uit de eerste elementen van lijst L_i). De virtuele bag V_i laten we wijzen naar B_i zodat vanaf een willekeurig element binnen 2 stappen de echte bag te bereiken is. (Zie figuur 4.)
Bij iedere update van bag B_i verplaatsen we de mid-pointer m_i van B_i dan een stuk richting het midden (een constante hoeveelheid) waarbij meteen ervoor gezorgd wordt dat de 'bag'-velden van de elementen tot en met m_i wijzen naar V_i en de elementen na m_i blijven wijzen naar B_i. Dit betekent per update een constante hoeveelheid werk.
Indien nu na O(log n) updates op bag B_i deze weer aan splitsing toe is zal de mid-pointer wijzen naar het middelste element van de bag en wijzen de 'bag'-velden van de elementen uit de eerste helft van de lijst reeds naar de nieuw te vormen deelbag V_i zodat de uiteindelijke splitsing nog slechts O(1) tijd vergt.

1.6. Korte beschrijving van de werkwijze bij toevoegingen

De toevoeging van een element geschiedt globaal beschouwd als volgt.
Allereerst wordt het element toegevoegd in de betreffende bag, nadat eerst de positie van de toevoeging bepaald is.
Vervolgens wordt getracht de grootste lijst binnen constante afstand (m) te splitsen. Hierdoor wordt een locale prioriteitsvolgorde opgelegd op de bags die willen splitsen. Dit is mogelijk indien de bag groot genoeg is en bovendien de buren binnen afstand m recentelijk niet gesplitst zijn. Het laatste is het geval wanneer ze geen scanner meer 'bezitten'. Verder garandeert dit dat slechts een klein percentage bags splitst waardoor de hoeveelheid scanners niet te groot wordt.
Tenslotte worden de eventuele scanners van 4m+1 buren die rond de originele bag resp. de rechterhelft van de gesplitste bag liggen een constante afstand (p) omhoog verplaatst worden waarbij uiteraard de knopen geherbalanceerd worden voor zover nodig. Door p groot genoeg te kiezen zal blijken dat er geen onacceptabel oponthoud ontstaat bij het wachten van bags op toestemming om zelf te mogen splitsen.
Het zal duidelijk zijn dat op deze wijze de hoeveelheid herbalanceerwerk per toevoeging constant is. Dat ook de structuur in balans gehouden kan worden en dat er geen inconsistentie optreedt is niet evident.

1.7. Aannames (resultaten)

Hier volgen enkele resultaten opdat men vooraf reeds een indruk heeft van de groottes die vooralsnog vaag aangeduid worden.

- De boom die gebruikt wordt is een BB[α]-boom met *imaginaire* $\alpha = 1/4$.
  De herbalanceringsstrategie resp. het algoritme tracht de balanswaarden in de knopen in het interval (1/4, 3/4) te houden. Aangezien de informatie die hooggelegen knopen bezitten achterloopt bij die van lagergelegen knopen betekent dit dat de balanswaarden van de structuur,

zoals die opgeslagen is in het geheugen, in werkelijkheid iets kan afwijken van de balanswaarden zoals die aan de knopen bekend is.
Er zal aangetoond worden dat de werkelijke balans in het interval (14/59, 45/59) zal liggen.

- Bij n bladeren betekent dat voor de maximale padlengte (MPL) dat deze gelijk is aan trunc($2{,}56 \cdot {}^2\log n$) = trunc($3{,}69 \cdot \ln n$).

  bewijs: Zij gegeven een BB[α]-boom T waarvan de balans in iedere knoop α benadert m.a.w. T is een boom met zo weinig mogelijk bladeren bij zekere diepte d.
  Definieer een rij knopen $\{ k_i \}_{i=0..d}$ zdd. k_0 de wortel van T is en k_{i+1} is rechterzoon van k_i.
  Voor het aantal bladeren N_i in de subboom met wortel k_i geldt dan dat

  $$N_i = n, N_1 \le (1-\alpha)N_0,\ N_2 \le (1-\alpha)N_1 \le (1-\alpha)^2 N_0,\ \cdots,\ N_d \le (1-\alpha)^d N_0$$

  Omdat $N_d = 1$ volgt $1 \le (1-\alpha)^d N_0$ zodat $n = N_0 \ge \left\lceil \frac{1}{1-\alpha} \right\rceil^d$ en dus geldt $d \le {}^{\frac{1}{1-\alpha}}\log n = {}^{\frac{59}{14}}\log n = 2{,}56 \cdot {}^2\log n = 3{,}69 \cdot \ln n$.

- Een bag komt voor splitsing in aanmerking wanneer de grootte SBS (SplittingBagSize) = 6 · MaxPadLengte wordt bereikt. De grootte zal ten gevolge van het uitstellen van splitsing de waarde MBS (MaximalBagSize) = 8 · MaxPadLengte niet overschrijden.
- In verband met de onnauwkeurigheid van de opgeslagen balanswaarden is het noodzakelijk om een subboom geheel opnieuw op te bouwen i.p.v. te roteren wanneer blijkt dat de opgeslagen rankwaarde van een van de passieve knopen (in geval van de geplande rotatie) kleiner dan 16 blijkt. Dit komt doordat anders de grootte van de te verplaatsen deelboom niet nauwkeurig genoeg meer te schatten is om theoretische correctheid van de rotatie te kunnen waarborgen.

HOOFDSTUK 2

Scanners

Aangezien scanners een grote rol spelen bij de gepresenteerde opzet om een boom in balans te houden is het belangrijk dat men een goed beeld heeft van het hoe en waarom van scanners.
In dit hoofdstuk wordt eerst precies aangegeven wat een scanner voorstelt en hoe scanner(set)s gerepresenteerd en verplaatst worden binnen de gehele structuur. Vervolgens wordt aangetoond hoe het percentage scanners per deelboom begrensd kan worden zonder dat dit negatieve gevolgen heeft voor de vereiste ordegrenzen.
Aan het eind van dit hoofdstuk zal een samenvatting gegeven worden waarin de belangrijkste zaken betreffende scanners nog eens op een rijtje gezet worden.

2.8. De plaats van een scanner en zijn betekenis

Een scanner wordt gegenereerd bij het splitsen van een bag om aan te geven dat de boom een blad(/bag) meer heeft gekregen. De gegenereerde scanner behoort bij beide deelbags. Iedere toekomstige toevoeging op een van beide deelbags impliceert dat de scanner een stuk van het pad naar de wortel doorloopt en daarbij de bezochte knopen herbalanceert voor zover nodig. Wanneer de scanner zijn taak verricht heeft, wordt dit gemeld aan beide deelbags en is een volgende splitsing mogelijk. Doordat de bladeren geordend in een lijst worden opgeslagen hoeft aan de scanner in principe slechts 1 owner (bijvoorbeeld de linkerdeelbag) bekend te zijn.

Een scanner(set) staat bij een edge resp. de knoop die onderaan die edge hangt. (Dit is de meest logische correspondentie tussen een edge en een knoop.)
Indien een scanner bij een edge staat dan zijn alle knopen op het pad, van de gesplitste bag naar de wortel, die onder de betreffende edge liggen op de hoogte van de splitsing. Boven-de-edge-gelegen knopen zijn van de splitsing nog totaal niet op de hoogte.
Vanaf nu benoemen we een edge d.m.v. de bijbehorende knoop aangezien een knoop -practisch gezien- tastbaarder is dan een edge.

Behalve de gebruikelijke betekenis van een scanner bij een knoop is er 1 uitzonderingsgeval nl. de situatie dat de knoop een blad is.
Een scanner kan niet 2 knopen gelijktijdig bezoeken zodat de scanner bij creatie geplaatst wordt bij de vader van de twee deelbags. Het scan-veld dat bij interne knopen aanwijst welke scanner bij de knoop staat, kan zodoende bij bladeren gebruikt worden om aan te geven welke scanner aan de betreffende bag gekoppeld is. Als zodanig dient op deze positie de scanner dan ook nooit opgeschoven te worden ! Het is zeer belangrijk dat hiermee rekening wordt gehouden bij de implementatie van de rotaties waar locale scanners verplaatst worden.
Bij het splitsen van een bag wordt dan ook bij de bladknopen (de deelbags) naar de scanner *verwezen* terwijl de scanner zelf *geplaatst* wordt bij de gemeenschappelijke vader (een interne knoop) van beide bladknopen.

2.9. De representatie van scannersets

Aangezien een gecreëerde scanner niet in één ruk door de boom heenloopt (bij de x opeenvolgende acties) betekent dit dat er zich bij een interne knoop meerdere scanners kunnen ophouden.
In verband met de verplaatsing van scanners tijdens rotaties moet enerzijds op ieder moment bekend zijn of er een scanner (of meerdere) bij een edge/knoop staat en moet bij een knoop een verwijzing naar deze scanner(s) staan en anderzijds zullen alle scanners bij een knoop als één geheel (of hoogstens een eindig klein aantal gehelen) behandeld moeten kunnen worden.
De eis om een verzameling scanners bij een knoop als een groep te behandelen leidt tot de keuze om scanners op te slaan in een UNION-FIND-structuur. (Zie E.Horowitz en S.Sahni [5] voor een uitgebreide beschrijving van UNION-FIND-structuren.)
Een UNION-FIND-structuur stelt ons in staat om verzamelingen samen te voegen (UNION) in constante tijd en om snel te bepalen tot welke verzameling het element behoort (FIND). Het zal blijken dat hier vooral de UNION-eigenschap de belangrijkste is.
Bij een UNION-FIND-structuur kan men denken aan een boom met de volgende aanpassing:

1) Er zijn alleen verwijzingen naar boven (dwz. er zijn geen kind-pointers).
2) Er is geen beperking op de graad van de knopen dwz. het aantal 'kinderen' is onbegrensd. (Dit is min of meer een gevolg van de vorige aanpassing.)

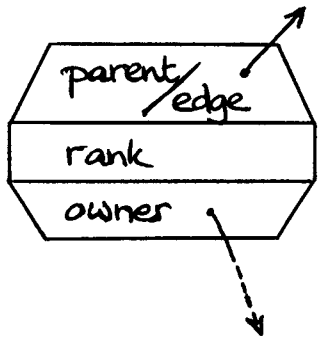
Een scanner is dan ook als volgt gedefinieerd:

```
type scanner = record
                 rank   : integer;
                 owner  : ptnode;
                 case isroot : boolean of
                   true : ( edge : ptnode );
                   false: ( parent : ptscanner )
               end;
```

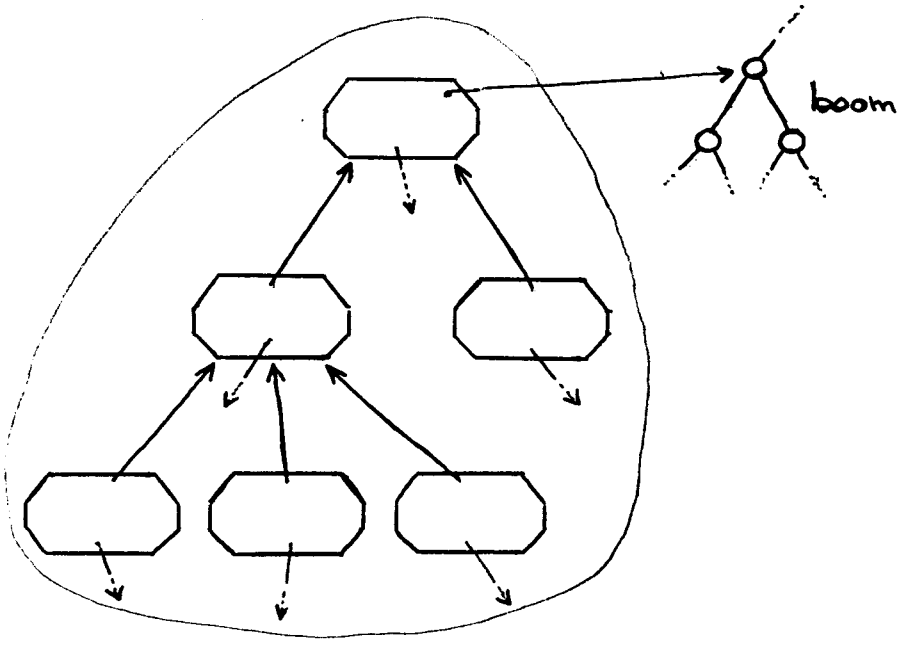
Hierbij geeft rank de grootte van de deel'boom' aan.
Indien een scanner de wortel van een UNION-FIND-structuur is (dwz. isroot = true) dan is het parent-veld vervangen door een pointer naar een knoop die aangeeft bij welke edge de scanners uit de UNION-FIND-structuur staan. (Zie figuur 5.) Bij de desbetreffende knoop (resp. edge) staat een verwijzing naar deze wortel en door het rank-veld van deze scanner te onderzoeken is dan meteen bekend hoeveel scanners er aanwezig zijn.

Union: Als we nu twee scannersets (UNION-FIND-structuren) A en B willen verenigen dan nemen we simpelweg de wortels van beide structuren, a en b genaamd, en maken a de vader van b door b niet meer 'root' te laten zijn en het parent-veld van b naar a te laten verwijzen. Daarbij wordt de rankwaarde van a opgehoogd. Hiermee is verzameling B in constante tijd toegevoegd aan verzameling A.

Find: Door de parent-velden van de scanners te volgen komt men bij de wortel van de UNION-FIND-structuur. Het edge-veld van deze wortel wijst dan de edge aan waar deze scanners bij



(a) scanner



(b) scannerset / UNION-FIND-structuur

Figuur 5

staan. Alhoewel deze actie in het slechtste geval lineaire tijd kost (bv. als alle 'knopen' van de UNION-FIND-structuur graad 1 hebben) is dit geen onoverkomelijk bezwaar aangezien het algoritme dat gebruikt wordt de scanners efficiënt verplaatst. Hoe dit gebeurt wordt beschreven in het volgende gedeelte.

2.10. De verplaatsing van scanners binnen de boom

In deze sectie wordt behandeld hoe de verplaatsing van scanners in zijn werk gaat in het geval dat een scanner door de boom loopt. De situatie waarbij scanners verplaatst worden t.g.v. rotaties komt in sectie 2.11 aan bod.

Bij het verplaatsen van scanners van een UNION-FIND-structuur gaan we als volgt te werk:

1) Als de scanner binnen de UNION-FIND-structuur zich op diepte 0 of 1 bevindt dan wordt deze scanner (en daarmee indirect alle daaronderhangende scanners) één edge omhoog (in de boom) verplaatst.
2) Als de scanner zich binnen de UNION-FIND-structuur op diepte >1 bevindt dan vindt een vorm van padcompressie plaats door de scanner binnen de UNION-FIND-structuur een plaats omhoog te schuiven (dwz. scanner^.parent := scanner^.parent^.parent). Merk op dat bij het verplaatsen van een scanner eigenlijk een hele verzameling scanners wordt verplaatst. Dit verschijnsel is inherent aan het gebruik van een UNION-FIND-structuur. Alhoewel er nadelen verbonden zijn aan deze werkwijze kent deze aanpak ook voordelen zoals o.a. het gegeven dat er een lichtelijke versnelling van informatieverplaatsing plaatsheeft.

Het plaatsen van een scanner(set) bij een andere (hogergelegen) edge geschiedt als volgt:

i) Indien er nog geen scanners bij die edge staan dan wordt de scanner(set) gewoon rechtstreeks bij deze edge gezet.
ii) Indien er wel reeds een scanner(set) bij de edge staat dan wordt deze scannerset uitgebreid door de wortel van de te verplaatsen scannerset kind te maken van de wortel van de reeds aanwezige scannerset (zie Union).

Tenslotte moet er ook nog voorzien zijn in het geval dat een scannerset de boom doorlopen heeft en deze dreigt te verlaten (in geval 1).

3) Wanneer een scanner de boom doorlopen heeft en deze dreigt te verlaten dan wordt deze scanner uit de gehele structuur gehaald door deze scanner los te koppelen van de owner(s). Verder wordt de geheugenruimte, die deze scanner beslaat, vrijgegeven indien zijn rank-waarde 1 is, dwz. er zijn geen scanners die naar deze scanner verwijzen. Als dit echter niet het geval is dan blijft de scanner voorlopig nog bestaan. Dit gebeurt omdat enerzijds de scanners in de desbetreffende UNION-FIND-structuur er nog niet van op de hoogte zijn dat ze de boom verlaten hebben en anderzijds omdat de geheugenruimte die deze scanner gebruikt nog niet vrijgegeven mag worden ivm. de scanners die onder de desbetreffende scanner hangen. Scanners die de boom reeds verlaten hebben maar zich hiervan nog niet bewust zijn komen hier pas achter als ze de desbetreffende UNION-FIND-structuur willen verlaten en ze merken dat de wortel de boom reeds verlaten heeft. Ook dan worden deze scanners weer losgekoppeld van de owners en er wordt gekeken of de scanner die de boom reeds verlaten had nu wel *gedispose*d kan worden (tesamen met de scanner die de boom nu ook verlaten heeft).

Vooruitgang

Zoals de titel van sectie 2.10 al aanduidt wordt hier de verplaatsing van scanners bij het doorlopen van de boom behandeld. Dit betekent dat er ook sprake moet zijn van vooruitgang.
De mate van vooruitgang wordt gemeten door de afstand van de scanner tot de wortel van de boom. Door als afstandsmaat te nemen 2 · de diepte van de scannerset in de boom + de diepte van de scanner in de UNION-FIND-structuur is vooruitgang bij scannerverplaatsing verzekerd.

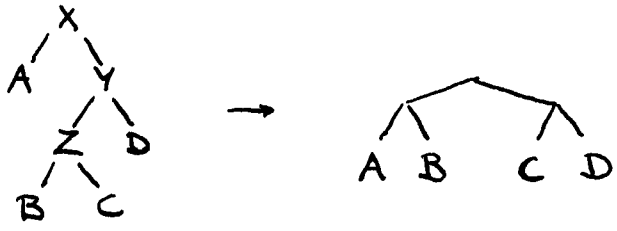
bewijs: Indien er een vorm van padcompressie plaatsvindt doordat de scanner binnen de UNION-FIND-structuur verplaatst wordt (geval 2)) dan neemt de afstand tot de wortel van de boom af

met 1.
In geval 1) waarbij de scanner binnen de boom opschuift maken we onderscheid tussen de situatie waarbij de scanner wortel is van een UNION-FIND-structuur en de situatie waarbij de scanner zich op diepte 1 binnen de UNION-FIND-structuur bevindt. (De mogelijkheid waarbij de scanner de gehele structuur verlaat (zie geval 3)) laten we buiten beschouwing omdat vooruitgang in dat geval voor zich spreekt.)

- De te verplaatsen scanner is wortel van de scannerset.
  In deze situatie neemt de diepte binnen de boom af met 1 en de diepte binnen een UNION-FIND-structuur neemt toe met 0 (in geval i)) resp. 1 (in geval ii)) zodat de afstand tot de wortel van de boom vermindert met 2 resp. 1 en er sprake is van vooruitgang.
- De te verplaatsen scanner bevindt zich op diepte 1 binnen de UNION-FIND-structuur.
  In deze situatie neemt de diepte binnen de boom af met 1 en de diepte binnen de UNION-FIND-structuur neemt af met 1 (in geval i)) resp. 0 (in geval ii)) zodat de afstand tot de wortel van de boom hierbij vermindert met 3 resp. 2 en er eveneens sprake is van vooruitgang.

2.11. Verplaatsing van scanners bij rotaties

Afgezien van de gebruikelijke noodzaak om scanners te verplaatsen (nl. om informatie door te geven) is er nog een situatie waarbij scanners verplaatst worden nl. rotaties. Aangezien hierbij grote hoeveelheden onafhankelijke informatie tegelijkertijd verplaatst worden is het belangrijk dat deze informatie niet onnodig door elkaar raakt en bovendien dient ervoor gezorgd te worden dat de informatie precies één keer wordt afgegeven op die plaatsen waar dit nodig is.
Het ligt voor de hand om scannersets bij een rotatie zo min mogelijk te verplaatsen aangezien er dan zo min mogelijk balansverschuivingen optreden en dit is onder meer belangrijk omdat de oorspronkelijke balanswaarden niet overal even actueel zijn. Een grote scannerverplaatsing zou wellicht kunnen leiden tot een te grote mate van onbalans, immers de te verplaatsen scanners moeten hun informatie ook nog afgeven. Wanneer we hiermee rekening trachten te houden doen zich enkele moeilijkheden voor waardoor van deze benadering moet worden afgezien.
Beschouw namelijk maar de situatie van een dubbele rotatie.

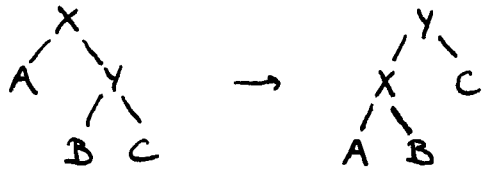


Omdat de deelboom met wortel B verplaatst wordt van de rechterkant naar de linkerkant dienen de van B afkomstige scanners die bij Y staan mee te gaan naar de linkerhelft (terwijl de van C afkomstige scanners in de rechterhelft blijven). De bij Y opgeslagen scanners dienen dus als verschillende groepen opgeslagen te worden.
Evenzo dienen de scanners bij X onderverdeeld te zijn in groepen, waaronder een groep van B afkomstige scanners en een groep van C afkomstige scanners immers na rotatie bevinden ze zich in linker- resp. rechterdeelboom. Na rotatie zullen deze groepen weer bij de locale wortel staan.
Op dat moment is de onderverdeling in groepen echter niet meer nauwkeurig genoeg immers voor een volgende dubbele rotatie moet bekend zijn welke scanners uit de linker- resp. rechterdeelboom van B resp. C komen. We kunnen nu wel in de beginsituatie een verdere onderverdeling van groepen scannersets maken maar de reeds gebruikte redenering toont aan dat we dan op hetzelfde probleem stuiten.
Dit probleem is ook niet op te lossen door de scanners bij X bij een rotatie te verplaatsen naar de vader van X omdat dan de balans van de vader van X verandert. In het geval dat deze balans hersteld moet worden, dmv. een rotatie, is een kettingreactie denkbaar en dat wensen we per sé niet.
We mogen dus concluderen dat we de scannerset bij een knoop niet mogen opdelen in groepen. Alle

locale scanners zullen bij een rotatie dan ook verplaatst worden naar de locale wortel X. De nieuwe balanswaarden worden dan volledig opnieuw bepaald vanuit de locale 'bladeren' (de passieve knopen). Alhoewel daarbij een heleboel scanners hun informatie tegelijkertijd moeten afgeven zal in het hoofdstuk over rotaties blijken dat de hierbij optredende balansverstoringen binnen de perken blijven in die zin dat men met een enkele of dubbele rotatie kan volstaan om de boom locaal in balans te houden.

Vooruitgang

De verplaatsing van alle locale scannersets naar de locale wortel houdt een lichtelijke versnelling van informatie/scanner-verplaatsing in.
Aangezien hier echter scanners verplaatst worden moet ook hier 'vooruitgang' van de scanners gegarandeerd blijven. Dit lijkt een trivialiteit maar in de praktijk schuilt er een addertje onder het gras. Daartoe bekijken we een enkelvoudige rotatie.



Indien we de scanners namelijk efficiënt denken te verplaatsen door de scanners van A,B,C en X toe te voegen aan die van Y dan kan het gebeuren dat de scanners van X in UNION-FIND-diepte met 1 toenemen. Dit betekent een onbedoelde achteruitgang resp. verwijdering van de wortel van de boom.
Om correct te zijn zullen eerst de scanners van (A,B,C en) Y bij die van X gevoegd moeten worden, om de scanners die dan bij X staan te verplaatsen naar Y (waar op dat moment geen scanners staan).
De afstand van de scanners bij X tot de wortel van de boom blijft dan gelijk. De afstand van de scanners bij A, B, C en Y tot de wortel van de boom neemt echter af aangezien de diepte in de boom met 1 of 2 vermindert en de UNION-FIND-diepte met 1 vermeerdert (waarbij de diepte in de boom dubbel telt). Omdat hier de verplaatsing t.g.v. een rotatie bekeken wordt volstaat het dat er geen sprake is van achteruitgang. De stilstand van de scanners bij X is daarom niet bezwaarlijk.
Voor de overige rotaties geldt hetzelfde voor wat betreft de scannerverplaatsing.

2.12. De noodzaak om het aantal scanners te beperken

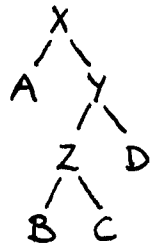
Het gebruik van een UNION-FIND-structuur betekent voor de scannerverplaatsing t.g.v. een update dat een groep scanners verplaatst wordt. Alle scanners geven daarbij gelijktijdig hun informatie af waardoor de balansverstoring groter is dan gebruikelijk. Het is daardoor twijfelachtig of de rotaties nog wel geschikt zijn om de balans te herstellen.
Een bijkomende reden hiervoor is dat de deelboom die verplaatst wordt (en de geschatte grootte hiervan) bepaald wordt op grond van locale balanswaarden. De opgeslagen balanswaarden zijn echter imaginair. Indien er nu veel locale scanners aanwezig zijn dan is -in het ongunstigste geval- het verschil tussen imaginaire en werkelijke balanswaarden relatief gezien nogal groot en de grootte van de te verplaatsen subboom uiterst moeilijk te taxeren.

Als concreet voorbeeld dient de volgende situatie:

$\alpha(X) = \alpha$

$\alpha(Y) = 1-\alpha$

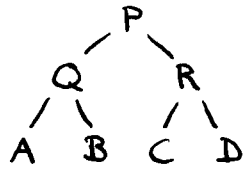
$\alpha(Z) = 1-\alpha$



Veronderstel dat er op dit moment geen scanners in de totale boom zijn. Zij |T| het totaal aantal bladeren van de gehele boom dwz. |T| = |A| + |B| + |C| + |D|.
Er geldt dan dat

$|A| = \alpha|T|$
$|B| = (1-\alpha)^3|T|$
$|C| = \alpha(1-\alpha)^2|T|$
$|D| = \alpha(1-\alpha)|T|$

Laat nu B (en C) 2 keer zo groot worden waarbij alle afgesplitste scanners zich bij B (resp. C) ophopen. X heeft dan nog geen notie van bovenstaande groei hebben.
Nu geldt $|A| = \alpha|T|$ en $|B| = 2(1-\alpha)^3|T|$. (Hierbij heeft |T| nog de oude waarde.)
Na een dubbele rotatie, die veroorzaakt is door groei van D en een hiervan afkomstige scanner die X bereikt heeft, bereiken we de volgende situatie:



Er geldt $\alpha < \alpha(Q) = \dfrac{\alpha}{\alpha + 2(1-\alpha)^3}$

$\Leftrightarrow \alpha + 2(1-\alpha)^3 < 1$
$\Leftrightarrow 2(1-\alpha)^2 < 1$
$\Leftrightarrow (1-\alpha)^2 < \frac{1}{2}$
$\Leftrightarrow 1-\alpha < \frac{1}{2}\sqrt{2}$
$\Leftrightarrow \alpha > 1-\frac{1}{2}\sqrt{2}$

Volgens Blum en Mehlhorn [1] moet echter gelden $\alpha \leq 1-\frac{1}{2}\sqrt{2}$ willen de rotaties correct werken. Maw. met een startconfiguratie als hierboven loopt men het risico dat met een dubbele rotatie nog teveel verplaatst wordt. Om de balans toch nog te kunnen herstellen door het gebruik van enkele en dubbele rotaties is het noodzakelijk om het percentage scanners per deelboom te begrenzen en te relateren aan de grootte van de betreffende deelboom.

2.13. Beperking van het percentage scanners in de praktijk

Om het percentage scanners per deelboom te in de hand te kunnen houden worden de boombladeren/bags geplaatst in een dubbel-gelinkte lijst die geordend is volgens de natuurlijke ordening op de keys. (Zie figuur 6.)
Splitsing van een voldoend grote bag (en creatie van een scanner) wordt nu alleen geoorloofd indien alle bags binnen afstand m, in de lijst met bags, niet recentelijk gesplitst zijn. Dit laatste is na te gaan door te inspecteren of er een scanner aan de bag is toegewezen. Door de variabele m te variëren is het percentage scanners per deelboom te begrenzen.
Deze voorwaarde om te mogen splitsen impliceert dat splitsing van een of meerdere bags soms een tijd

lang uitgesteld moet worden. Om de ordegrens voor de grootte van bags niet te overschrijden worden nog een aantal aanpassingen aangebracht hetgeen resulteert in de volgende werkwijze.

Oplossing

Laat een blad een scanner hebben dan zorgen we ervoor dat de m buren aan beide kanten geen scanners hebben (afgezien van zijn wederhelft bij splitsing).

Toevoegen van een element:

1) Voeg het element toe aan de bag.
2) Bepaal wie van de m linker- en m rechterburen plus de bag zelf het grootst is.
3) Splits deze bag indien dit is toegestaan op grond van scanners en size.
   (D.w.z. dat enerzijds de grootte voldoende moet zijn en dat anderzijds de bags binnen afstand m van deze grootste bag geen scanners bezitten. Dit na te gaan kost O(1) tijd doordat m vooraf vastgesteld wordt en dus constant is.)
4) Verplaats de scanners van de oorspronkelijke bag plus de 2m linker- en 2m rechterburen ieder een constante hoeveelheid p omhoog.
   (De bezochte knopen worden hierbij uiteraard geherbalanceerd.
   Merk op dat het overgrote deel van de buren geen scanner bezit zodat hiermee minder werk gemoeid is dan in eerste instantie lijkt. Maximaal zijn dit 4 scanners die ieder 2 keer p posities verplaatst worden.)

Opmerkingen

- Stap 2 dient om er voor te zorgen dat er een soort prioriteitsvolgorde wordt opgelegd aan de locale bags die voldoende groot zijn om voor splitsing in aanmerking te komen.
  Bovendien is splitsing van een bag niet meer afhankelijk van updates op de bag zelf.
- Stap 3 garandeert dat het percentage scanners per deelboom begrensd is.
- Bij stap 4 wordt er voor gezorgd dat alle scanners in de buurt voldoende regelmatig worden doorgeschoven opdat van uitstel om te splitsen geen afstel komt.
- Door p groot genoeg te kiezen zal blijken dat alles soepel verloopt.

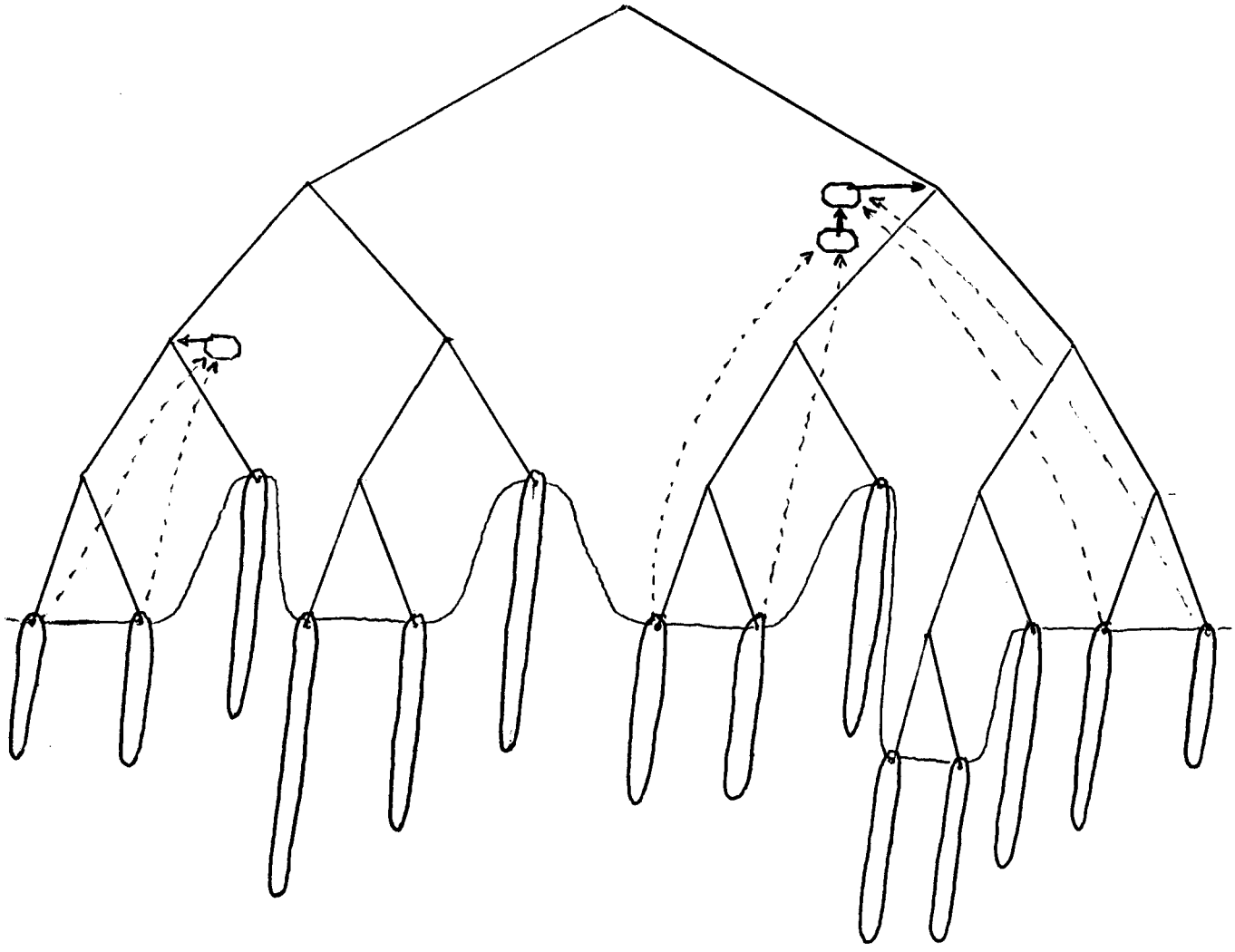
Correctheid van de werkwijze

Zij MPL de Maximum PadLengte. Laat de SplittingBagSize SBS gegeven worden door S·MPL en de MaximumBagSize MBS door M·MPL waarbij S en M nog nader te bepalen zijn.
Alvorens een bag werkelijk mag splitsen moeten er wellicht eerst de scanners van de 2m buren weggeduwd worden zodat $MBS = SBS + 2m \cdot \frac{MPL}{p} = (S + \frac{2m}{p}) \cdot MPL$.
Hierbij is er van uitgegaan dat de dichtbijgelegen buren na splitsing niet dermate hard aangroeien dat ze de splitsing van de eigenlijke bag alsnog kunnen verhinderen door opnieuw te splitsen.
Om dit te garanderen eisen we dat $SBS - \frac{1}{2} MBS \geq \frac{2m}{p} \cdot MPL$
m.a.w. $\left\{S - \frac{1}{2}(S + \frac{2m}{p})\right\} \cdot MPL \geq \frac{2m}{p} \cdot MPL$ waaruit volgt $\frac{1}{2} S - \frac{m}{p} \geq \frac{2m}{p}$
zodat moet gelden $S \geq \frac{6m}{p}$
Kiezen we nu p=m en S=6 (SBS=6·MPL) dan geldt MBS=8·MPL (M=8).
De benodigde tijd is O(m)+O(m)+O(mp)=O(1), want p en m worden vast gekozen, zoals eenvoudig is in te zien.

Samenvatting

Scanners worden gecreëerd bij splitsing van een bag en ze staan bij interne knopen.
Hogergelegen knopen zijn nog in het geheel niet op de hoogte van hun aanwezigheid resp. splitsing van

de corresponderende bag.
Scannersets worden gerepresenteerd dmv. een UNION-FIND-structuur vanwege het gemak waarmee verzamelingen scanners verplaatst en samengevoegd kunnen worden.
Het verplaatsen van een scanner impliceert daardoor dat een heel groepje scanners meeverplaatst wordt.
Bij een rotatie worden àlle scanners verplaatst die staan bij knopen die direct betrokken zijn bij de rotatie maw. de actieve knopen èn de wortels van de subbomen die bij de rotatie meedoen.
Verstoring van de balans wordt begrensd door er voor te zorgen dat een bag alleen mag splitsen (en een scanner creëren) indien dichtbij gelegen bags geen scanner bezitten (en dus niet recent gesplitst zijn). Hierdoor is de groei van deelbomen beperkt.
Door zeer regelmatig scanners te verplaatsen en daarbij de gepasseerde knopen te herbalanceren, wordt voorkómen dat splitsing van een bag te lang wordt uitgesteld.



Figuur 6
De complete uitgebreide BB[α]-boom

HOOFDSTUK 3

Rotaties

Naast scanners zijn rotaties (en de correctheid ervan) het tweede hoofditem. De nieuwe wijze waarop met bomen wordt omgegaan heeft tot gevolg dat er nieuwe condities gelden bij het herstellen van de balans.
Allereerst gaan we in op de groei van deelbomen en de aanpassing van de rotaties die daardoor nodig is.
Vervolgens wordt aangegeven hoe de imaginaire balanswaarden vertaald kunnen worden naar een interval waarbinnen de werkelijke balans zich bevindt.
Daarna wordt de herbalanceringsstrategie van Blum en Mehlhorn bijgesteld en de correctheid van de rotaties wordt bewezen.
Aan het eind van dit hoofdstuk wordt weer een samenvatting gegeven waarin de belangrijkste zaken op een rijtje worden gezet.

3.14. De groei van deelbomen en geconditioneerde rotaties

Bij de bepaling van de toe te passen rotatie maakt het bekende standaardalgoritme (van Blum en Mehlhorn [1]) gebruik van de op dat moment bekend-zijnde balanswaarden. In ons geval, waar wordt gewerkt met scanners, zijn deze waarden echter niet 100% actueel aangezien sommige nodes al informatie (van scanners afkomstig) bezitten die voor andere nodes nog totaal onbekend is.
Zodoende moet bij de interpretatie van de (imaginaire) balanswaarden rekening gehouden worden met eventuele groei van de deelbomen. Hiertoe introduceren we het begrip groeifactor. Dit is de mate waarin een boom (zonder scanners) kan groeien zonder dat dit aan de wortel van de boom bekend wordt. Formeler:

Definitie

Zij n de grootte van een boom zonder scanners is en n' de grootte van de boom nadat een maximaal aantal scanners is afgesplitst zonder dat de wortel van de boom daar notie van heeft.

De groeifactor $\gamma(n)$ wordt gedefiniëerd als n'/n.
(Onder de grootte van een boom wordt het aantal bladeren van die boom verstaan.)

Merk op dat de groeifactoren afhangen van de grootte van m (zie sectie 2.13). Immers wanneer m groot is dan kunnen er weinig bags splitsen omdat in bovenstaande definitie ervan wordt uitgegaan dat de wortel niets van de groei mag merken. Voor bomen kleiner dan m geldt echter dat $\gamma(n) = (n+1)/n$ immers alle bladeren bevinden zich onderling binnen afstand n en dus zeker binnen afstand m zodat splitsing van een bag splitsing van alle andere bags tegenhoudt zolang de scanner de boom niet verlaten heeft.

Om correctheid van de rotaties aan te kunnen tonen is het noodzakelijk dat de groeifactoren van de deelbomen relatief klein zijn.
Bij een groeifactor 2 geldt namelijk dat het interval waarbinnen de werkelijke balans moet liggen (bij schatting uit de imaginaire balanswaarde) te groot is om bruikbaar te zijn bij de correctheidsbewijzen van de rotaties.
Een bovengrens voor alle groeifactoren is minimaal 2 (als namelijk de 'boom' uit slechts 1 blad bestaat) en deze waarde is té groot om correctheid van de rotaties te kunnen aantonen. Om te kunnen garanderen dat de groeifactoren van deelbomen, die bij een rotatie betrokken zijn, voldoende klein zijn gaan we als volgt te werk.
Allereerst wordt op grond van de balanswaarden gekozen voor een enkele of dubbele rotatie. Daarna wordt gecontroleerd of de passieve knopen wortel zijn van deelbomen met een (vooraf vastgestelde) minimale grootte. Indien dit zo is wordt de rotatie uitgevoerd. Zo niet dan wordt er niet geroteerd maar bouwen we de hele boom onder de knoop die uit balans was opnieuw op. Dit laatste is alleen het geval bij subbomen die een zekere grootte niet te boven gaan en de hoeveelheid werk is zodoende naar boven begrensd door een constante.

Een practische bovengrens voor de groeifactoren

We kiezen m=27 d.w.z. een bag mag alleen splitsen indien er binnen afstand 27 in de lijst met boombladeren geen bag bevindt waaraan een scanner is toegewezen.
Door deze keuze zijn de groeifactoren eenduidig bepaald.

Stelling 3.1 : $n \geq 14 \Rightarrow \gamma(n) < 15/14$ (mits m=27)
bewijs :

De groeifactor $\gamma(n)$ wordt gegeven door

$$\left\{n+\left\lceil\frac{n}{28}\right\rceil\right\}/n \;=\; \left\{n+1+\left\lfloor\frac{n-1}{28}\right\rfloor\right\}/n$$

immers per 28-tal bags kan er een bag splitsen.
Aangezien deze formules niet erg handzaam zijn bij het rekenwerk moet er naar iets anders omgezien worden.

Bekijk eerst de grootte van een boom voor en na (maximale) groei.

$1 \rightarrow 2$	$m+2 \rightarrow m+4$	$2m+3 \rightarrow 2m+6$	
$2 \rightarrow 3$	$m+3 \rightarrow m+5$	$2m+4 \rightarrow 2m+7$	etc.
:	:	:	
:	:	:	
$m+1 \rightarrow m+2$	$2(m+1) \rightarrow 2m+4$	$3(m+1) \rightarrow 3m+6$	

Merk op dat de groeifactor binnen kolom k daalt immers $\gamma(p) = \frac{p+k}{p}$ en

$$\gamma(p) > \gamma(p+1) \Leftrightarrow \frac{p+k}{p} > \frac{p+1+k}{p+1}$$

$$\Leftrightarrow (p+k)(p+1) > p(p+k+1)$$

$$\Leftrightarrow p^2 + p(k+1) + k > p^2 + p(k+1) \qquad OK.$$

(P.S. Hier is k de *absolute* groei EN het nummer van de kolom.)

We interesseren ons nu nog voor de rij

$$1 \rightarrow 2 \qquad m+2 \rightarrow m+4 \qquad 2m+3 \rightarrow 2m+6 \qquad 3m+4 \rightarrow 3m+8 \qquad \text{etc.}$$

Ook hier daalt de groeifactor want $\gamma(pm+(p+1)) = \frac{pm+2(p+1)}{pm+(p+1)}$ en

$$\gamma(pm+(p+1)) > \gamma((p+1)m+(p+2))$$

$$\Leftrightarrow \frac{pm + 2(p+1)}{pm + (p+1)} > \frac{(p+1)m + 2(p+2)}{(p+1)m + (p+2)}$$

$$\Leftrightarrow \{pm+2(p+1)\}\cdot\{(p+1)m+(p+2)\} > \{pm+(p+1)\}\cdot\{(p+1)m+2(p+2)\}$$

$$\Leftrightarrow p(p+1)m^2 + p(p+2)m + 2(p+1)^2 \cdot m + 2(p+1)(p+2) >$$

$$> p(p+1)m^2 + 2p(p+2)m + (p+1)^2 \cdot m + 2(p+1)(p+2)$$

$$\Leftrightarrow (p+1)^2 > p(p+2)$$

$$\Leftrightarrow p^2 + 2p + 1 > p^2 + 2p \qquad OK.$$

Nu geldt, met m=27, (in de eerste kolom) $\gamma(15) = 16/15 < 15/14$ en (bij de tweede kolom) $\gamma(29) = 31/29 < 15/14$ zodat volgt $\gamma(n) < 15/14$ voor $n > 14$.

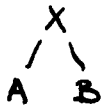
[einde bewijs]

Groeifactoren kleiner dan 15/14 zullen blijken te volstaan om de correctheid van de rotaties aan te tonen.
Indien de deelboom onder een passieve knoop minder dan 16 bedraagt dan was de grootte vóórdat er scanners bij stonden wellicht 14 of nog minder en is een groeifactor < 15/14 niet te garanderen. Daarom besluiten we tot het volledig herbouwen van de subboom in het geval dat deze situatie zich voordoet.

Bij de correctheidsbewijzen van de rotaties mogen we er nu van uitgaan dat de groeifactoren van de deelbomen kleiner zijn dan 15/14. Bij de berekeningen gebruiken we daarom als groeifactor $\gamma = 15/14$.

3.15. De relatie tussen de imaginaire en werkelijke balans

Alvorens de rotaties correct bewezen kunnen worden wordt eerst aangegeven hoe men de imaginaire balanswaarde moet vertalen naar een waarde voor de werkelijke balans. Omdat daarbij een schatting gegeven wordt, zal de opgeslagen balanswaarde vertaald worden naar een interval waarbinnen de werkelijke balans liggen moet.



Stelling 3.2 : Zij $\alpha(X) = p/q$ de imaginaire balans van knoop X.
Dan ligt de werkelijke balans van X in het interval $\left(\frac{14p}{15q - p} , \frac{15p}{14q + p} \right)$
bewijs :

Zij p de grootte van de linkerdeelboom van X en q de grootte van de boom onder X zoals deze aan X bekend zijn. (Aan het eind van het bewijs wordt aangetoond dat deze eis niet per sé noodzakelijk is.)
We zijn geïnteresseerd in de uiterste waarden die de werkelijke balans kan aannemen en daartoe gaan we ervan uit dat er bij X geen scanners staan en X geen notie heeft van groei.
Verder behoeven we slechts 2 gevallen te beschouwen nl.

a) het maximale aantal scanners staat bij B en deelboom T(A) is niet gegroeid.
b) het maximale aantal scanners staat bij A en deelboom T(B) is niet gegroeid.

De werkelijke balans van X is minimaal in geval a) en bedraagt

$$\frac{p}{p + 15/14 \cdot (q-p)} = \frac{14p}{15q - p}$$

De werkelijke balans van X is minimaal in geval b) en bedraagt

$$\frac{15/14 \cdot p}{15/14 \cdot p + (q-p)} = \frac{15p}{14q + p}$$

Merk tot slot op dat in teller en noemer van beide grenzen de factor g := ggd(p, q) (grootst gemene deler van p en q) weggedeeld kan worden zodat het geen verschil maakt of we bijvoorbeeld nemen p=200 en q=500 dan wel p=2 en q=5.

[einde bewijs]

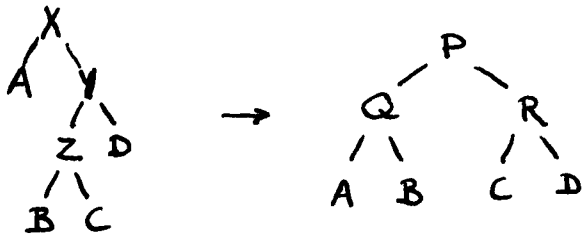
Met deze stelling zullen, bij het correct bewijzen van de rotaties, de opgeslagen balanswaarden vertaald worden naar intervallen voor de werkelijke balanswaarden. Daarna kan dan vervolgens de grootte van de deelbomen onder passieve knopen geschat worden.
We zullen hier gebruik maken van BB[α]-bomen waarbij de imaginaire balanswaarden in het interval (1/4, 3/4) gehouden zullen worden. Maken we gebruik van de bovenstaande stelling dan betekent dit voor de werkelijke balanswaarden dat deze minimaal $\frac{14 \cdot 1}{15 \cdot 4 - 1} = \frac{14}{59}$ en maximaal $\frac{15 \cdot 3}{14 \cdot 4 + 3} = \frac{45}{59}$ bedragen.

De bepaling van nieuwe (balans)waarden uit de oude waarden.

In sectie 2.11 is aangetoond dat er moeilijkheden verschijnen indien de locale scanners bij rotaties zo min mogelijk verschoven worden. Er is toen radicaal besloten om alle locale scanners bij de locale wortel te plaatsen.
Alhoewel daarbij indirect een sterke balansverstoring van de actieve nodes denkbaar is heeft deze massale scannerverplaatsing het voordeel dat er bij het bepalen van de nieuwe waarden (na een rotatie) geen rekening gehouden hoeft te worden met her en der verspreide scannersets en de daar aan gekoppelde verschillen in actualiteit van o.a. de balanswaarden. Het is daardoor eenvoudiger na te gaan of de rotaties hun beoogde effect hebben.
Voor wat betreft de nieuwe waarden van de actieve nodes behoeft nu geen omslachtig rekenwerk verricht te worden maar worden deze rechtstreeks bepaald uit de rankwaarden van de passieve nodes.

3.16. De grenswaarde van de balans tussen enkele en dubbele rotatie

Omdat ons algoritme werkt met imaginaire balanswaarden i.p.v. met exacte waarden is het te verwachten dat de keuze voor een enkele of dubbele rotatie bij een andere grenswaarde ligt dan Blum en Mehlhorn gebruiken [1].
Blum en Mehlhorn, in de eenvoudigste vorm, geven aan dat deze grenswaarde bij $C(\alpha) = 1/(2-\alpha)$ ligt als we de balans van de nodes in het interval $(\alpha , 1-\alpha)$ willen houden.
Deze grenswaarde is voor ons algoritme helaas niet bruikbaar. We bekijken daartoe het geval van een dubbele rotatie:



In het bijzonder nemen we de balans van R onder de loep.
$\alpha(R)$ is minimaal als $|C|$ minimaal en $|D|$ maximaal.
Daartoe veronderstellen we, voor willekeurige $0 < \alpha < \frac{1}{2}$, dat
$\alpha(Y) \approx C(\alpha) = 1/(2-\alpha)$ en $\alpha(Z) \approx 1-\alpha$,
resp. als bijbehorende werkelijke balanswaarden (zie 3.12)

$$\alpha'(Y) = \frac{14\cdot 1}{15\cdot(2-\alpha)-1} = \frac{14}{29-15\alpha}$$

en

$$\alpha'(Z) = \frac{15\cdot(1-\alpha)}{14\cdot 1 + (1-\alpha)} = \frac{15-15\alpha}{15-\alpha}$$

Nu geldt

$$\alpha(R) < \alpha \Leftrightarrow \frac{\alpha'(Y)\cdot\{1-\alpha'(Z)\}}{\alpha'(Y)\cdot\{1-\alpha'(Z)\} + \{1-\alpha'(Y)\}} < \alpha$$

$$\Leftrightarrow \alpha'(Y)\cdot\{1-\alpha'(Z)\} < \alpha\cdot\{1-\alpha'(Y)\cdot\alpha'(Z)\}$$

$$\Leftrightarrow \frac{14}{29-15\alpha}\cdot\frac{14}{15-\alpha} < \alpha\cdot\left\{1-\frac{14}{29-15\alpha}\cdot\frac{15-15\alpha}{15-\alpha}\right\}$$

$$\Leftrightarrow 14\cdot 14\alpha < \alpha\{(29-15\alpha)(15-\alpha) - 14(15-15\alpha)\}$$

$$\Leftrightarrow 196 > 15\alpha^2 - 44\alpha + 225$$

$$\Leftrightarrow 0 > 15\alpha^2 - 44\alpha + 29 \qquad (**)$$

Noem het rechterlid $f(\alpha)$.
Aangezien voor iedere toegestane waarde van α geldt $f'(\alpha) < 0$ is f strikt monotoon dalend.
Omdat verder $f(\frac{1}{2}) > 0$ vinden we een tegenspraak met (**) voor iedere waarde van α.
M.a.w. de waarde voor $C(\alpha)$ die gegeven wordt door Blum en Mehlhorn is niet bruikbaar.
We zullen aantonen dat de rotaties correct werken voor $\alpha = 1/4$ en $C(\alpha) = 13/21$.

3.17. Correctheid van de rotaties

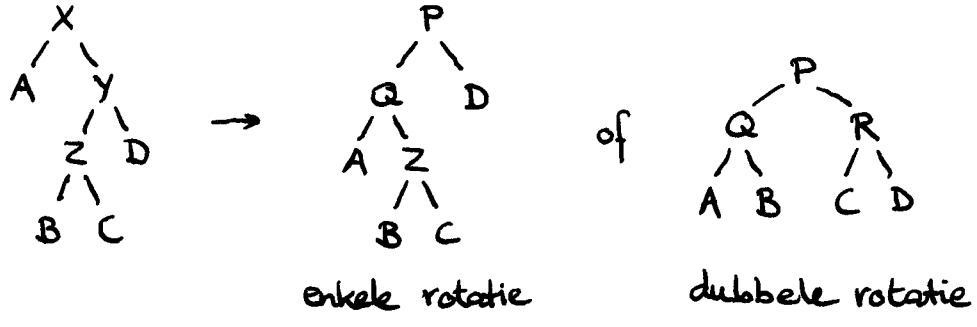
Voordat we overgaan tot het rekenwerk maken we de volgende observatie:
In de praktijk ligt de balans $\alpha(X)$ van X op het moment dat we gaan roteren buiten het interval $(\alpha, 1-\alpha)$. Dit bemoeilijkt het rekenwerk enigzins immers de overige balanswaarden liggen binnen het interval $(\alpha, 1-\alpha)$. Daarom gaan we *theoretisch* als volgt te werk:

Als je scanners omhoog wilt schuiven van Y naar X dan kijk je eerst of daardoor de balans van X buiten het interval $(\alpha, 1-\alpha)$ zou raken. Indien dit niet zo is dan verplaats je de scanners gewoon en hoeft dan niet te roteren. Als dit wel zo is dan ga je meteen roteren zonder eerst de scanners naar X door te schuiven. Bij de rotatie worden de scanners dan vanzelf wel verplaatst.

Merk op dat het uiteindelijke resultaat bij de praktische en de theoretische werkwijze exact hetzelfde is. We hebben nu echter als theoretisch voordeel dat de balans van X ook in het interval $(\alpha, 1-\alpha)$ ligt evenals de balans van Y en Z.

Bij het aantonen van de correctheid van de rotaties doen we alsof de intervalgrenzen worden aangenomen (ook al is dit in werkelijkheid niet het geval) omdat deze grenzen willekeurig dicht benaderd kunnen worden. Verder nemen we, zonder verlies van algemeenheid, aan dat de balans van X aanleiding geeft tot het uitvoeren van een linkerrotatie.

if $\alpha(Y) \leq C(\alpha)$
then single (left) rotation
else double (left) rotation



Zie de figuur voor de naamgeving van de locale nodes.

Om de rotaties correct te bewijzen worden eerst de imaginaire balanswaarden vertaald naar werkelijke balanswaarden, waarna de grootte van de deelbomen geschat kan worden.

Een imaginaire balans p/q betekent dat de werkelijke balans in het interval $\left[\frac{14p}{15q-p}, \frac{15p}{14q+p} \right]$ moet liggen (zie Stelling 3.2).
Voor de balanswaarden α, $C(\alpha)$ en $1-\alpha$ houdt dit de volgende 'vertaling' in:

$$\frac{1}{4} \to \left[\frac{14}{59}, \frac{15}{57}\right], \quad \frac{13}{21} \to \left[\frac{91}{151}, \frac{195}{307}\right], \quad \frac{3}{4} \to \left[\frac{42}{57}, \frac{45}{59}\right]$$

Enkele rotatie

M.a.w. $\alpha(Y) \in (\alpha, C(\alpha))$

i) $\alpha(P)$ is minimaal als A en Z minimaal en D maximaal
dwz. $\alpha(Y) = \alpha = 1/4$ resp. $\alpha'(X) = 14/59$ (minimaal) en $\alpha'(Y) = 14/59$ (minimaal)
Dan geldt $|A| = 14/59$, $|Z| = 45/59 \cdot 14/59$ en $|D| = 45/59 \cdot 45/59$

$$\alpha(P) = \frac{|A| + |Z|}{|A| + |Z| + |D|} = 0.418... > \frac{1}{4}$$

ii) $\alpha(P)$ is maximaal als A en Z maximaal en D minimaal
dwz. $\alpha(Y) = C(\alpha) = 13/21$ resp. $\alpha'(X) = 15/57$ (maximaal) en $\alpha'(Y) = 195/307$ (maximaal)
Dan geldt $|A| = 15/57$, $|Z| = 42/57 \cdot 195/307$ en $|D| = 42/57 \cdot 112/307$

$$\alpha(P) = \frac{|A| + |Z|}{|A| + |Z| + |D|} = 0.731... < \frac{3}{4}$$

iii) $\alpha(Q)$ is minimaal als A minimaal en Z maximaal
dwz. $\alpha(Y) = C(\alpha) = 13/21$ resp. $\alpha'(X) = 14/59$ (minimaal) en $\alpha'(Y) = 195/307$ (maximaal)
Dan geldt $|A| = 14/59$ en $|Z| = 45/59 \cdot 195/307$

$$\alpha(Q) = \frac{|A|}{|A| + |Z|} = 0.328... > \frac{1}{4}$$

iv) $\alpha(Q)$ is maximaal als A maximaal en Z minimaal
dwz. $\alpha(Y) = \alpha = 1/4$ resp. $\alpha'(X) = 15/57$ (maximaal) en $\alpha'(Y) = 14/59$ (minimaal)
Dan geldt $|A| = 15/57$ en $|Z| = 42/57 \cdot 14/59$

$$\alpha(Q) = \frac{|A|}{|A| + |Z|} = 0.600... < \frac{3}{4}$$

Dubbele rotatie

M.a.w $\alpha(Y) \in (C(\alpha), 1-\alpha)$

i) $\alpha(P)$ is minimaal als A en B minimaal en C en D maximaal
dwz. $\alpha(Y) = 13/21$ en $\alpha(Z) = 1/4$
resp. $\alpha'(X) = 14/59$ (minimaal), $\alpha'(Y) = 91/151$ (minimaal) en $\alpha'(Z) = 14/59$ (minimaal)
Dan geldt $|A| = 14/59$, $|B| = 45/59 \cdot 91/151 \cdot 14/59$,
$|C| = 45/59 \cdot 91/151 \cdot 45/59$ en $|D| = 45/59 \cdot 60/151$

$$\alpha(P) = \frac{|A| + |B|}{|A| + |B| + |C| + |D|} = 0.346... > \frac{1}{4}$$

ii) $\alpha(P)$ is maximaal als A en B maximaal en C en D minimaal
dwz. $\alpha(Y) = \alpha(Z) = 3/4$
resp. $\alpha'(X) = 15/57$ (maximaal), $\alpha'(Y) = \alpha'(Z) = 45/59$ (maximaal)
Dan geldt $|A| = 15/57$, $|B| = 42/57 \cdot 45/59 \cdot 45/59$,
$|C| = 42/57 \cdot 45/59 \cdot 14/59$ en $|D| = 42/57 \cdot 14/59$

$$\alpha(P) = \frac{|A| + |B|}{|A| + |B| + |C| + |D|} = 0.691... < \frac{3}{4}$$

iii) $\alpha(Q)$ is minimaal als A minimaal en B maximaal
dwz. $\alpha(Y) = \alpha(Z) = 3/4$
resp. $\alpha'(X) = 14/59$ (minimaal), $\alpha'(Y) = \alpha'(Z) = 45/59$ (maximaal)
Dan geldt $|A| = 14/59$ en $|B| = 45/59 \cdot 45/59 \cdot 45/59$

$$\alpha(Q) = \frac{|A|}{|A| + |B|} = 0.348... > \frac{1}{4}$$

iv) $\alpha(Q)$ is maximaal als A maximaal en B minimaal
dwz. $\alpha(Y) = 13/21$ en $\alpha(Z) = 1/4$
resp. $\alpha'(X) = 15/57$ (maximaal), $\alpha'(Y) = 91/151$ (minimaal) en $\alpha'(Z) = 14/59$ (minimaal)
Dan geldt $|A| = 15/57$, $|B| = 42/57 \cdot 91/151 \cdot 14/59$

$$\alpha(Q) = \frac{|A|}{|A| + |B|} = 0.714... < \frac{3}{4}$$

v) $\alpha(R)$ is minimaal als C minimaal en D maximaal
dwz. $\alpha(Y) = 13/21$ en $\alpha(Z) = 3/4$
resp. $\alpha'(Y) = 91/151$ (minimaal) en $\alpha'(Z) = 45/59$ (maximaal)
Dan geldt $|C| = \alpha'(X) \cdot 91/151 \cdot 14/59$ en $|D| = \alpha'(X) \cdot 60/151$

$$\alpha(R) = \frac{|C|}{|C| + |D|} = 0.264... > \frac{1}{4}$$

vi) $\alpha(R)$ is maximaal als C maximaal en D minimaal
dwz. $\alpha(Y) = 3/4$ en $\alpha(Z) = 1/4$
resp. $\alpha'(Y) = 45/59$ (maximaal) en $\alpha'(Z) = 14/59$ (minimaal)
Dan geldt $|C| = \alpha'(X) \cdot 45/59 \cdot 45/59$ en $|D| = \alpha'(X) \cdot 14/59$

$$\alpha(R) = \frac{|C|}{|C| + |D|} = 0.710... < \frac{3}{4}$$

Samenvatting

Indien een van de passieve nodes de wortel is van een kleine deelboom dan is de grootte van de te verplaatsen deelboom theoretisch dusdanig moeilijk te schatten dat er niet gegarandeerd kan worden dat de rotatie het beoogde effect bereikt. In dat geval wordt in plaats van de rotatie de hele subboom opnieuw opgebouwd. De hoeveelheid werk is daarbij gelijk aan O(1) omdat de grootte van de subboom naar boven toe begrensd is.
Wanneer de subboom echter groot genoeg is zorgt het voorgestelde invoegalgoritme ervoor dat de groeifactoren van de subbomen klein genoeg zijn om de werkelijke balanswaarden nauwkeurig genoeg te schatten op grond van de bekende (imaginaire) balanswaarden, waardoor de correctheid van de rotaties gewaarborgd wordt.
Bij het toevoegen van een element en de poging om een bag te splitsen wordt gekeken naar O(m) buren aan beide kanten. Het toevoegen wordt afgesloten met het verplaatsen van hooguit 4 scanners (van O(m) buren) en wel p posities voor iedere scanner.
De keuze van m bepaalt het percentage bags dat op ieder willekeurig moment gesplitst kan zijn. De keuze van p garandeert dat de gecreëerde scanners snel genoeg verdwijnen en niet voor opstopping zorgen.
Kiezen we p = m en SplittingBagSize = 6 · MaximumPadLengte dan is een ordelijk verloop verzekerd waarbij MaximumBagSize = 8 · MaximumPadlengte. Kiezen we verder m = 27 dan verrichten de rotaties hun werk naar wens mits de grootte van de deelbomen met als wortel een passieve node groter gelijk 16 is.
In het geval dat een verplaatste deelboom kleiner dan 16 is zal een rotatie niet-noodzakelijk het gewenste effect hebben en is het nodig de boom locaal geheel opnieuw op te bouwen.

Stelling 3.3 : Zij V een verzameling keys ter grootte n. Er bestaat een datastructuur voor V z.d.d. de zoektijd O(log n) bedraagt en waarop toevoegingen kunnen plaatsvinden in worst-case tijd O(1) mits de positie van de key bekend is.

HOOFDSTUK 4

Global rebuilding

Om de datastructuur ook geschikt te maken voor deletions wordt gebruik gemaakt van global-rebuilding (zie [8]). Daarbij wordt uitgegaan van een datastructuur die een ideale vorm bezit d.w.z. de verdeling van data over de structuur is dusdanig dat de toegang tot deze data zo efficiënt mogelijk kan geschieden. Bij updates wordt de vorm van de datastructuur niet aangepast om de efficiëntie te behouden maar staat men een zekere mate van inefficiëntie toe. Zulke updates waarbij de zoektijd, benodigde hoeveelheid geheugen en update-tijden niet meer dan een vaste factor slechter zijn dan bij een nieuw-gebouwde structuur worden zwakke updates genoemd.
Er zal gebruik gemaakt worden van zwakke deletions en sterke inserties. D.w.z. bij groei van de boom t.g.v. een bagsplitsing wordt de balans wel aangepast, maar wanneer een bag verwijderd wordt wordt de structuur niet geherbalanceerd.
Bij het verwijderen van een lege bag uit de boom wordt de vader van het betreffende boomblad vervangen door de andere zoon en bij dit boomblad geven we nu aan dat deze knoop een (virtuele) subboom representeert met grootte 2. Wanneer er meer van dergelijke bagdeletions plaatsvinden dan bestaan er boombladeren die aangeven dat ze een boom representeren met grootte >1. Bij splitsing van een bag die staat bij een dergelijk boomblad wordt de virtuele boom gedeeltelijk vervangen door een stuk echte boom (en dan is herbalancering van de knopen op het pad naar de wortel zelfs onnodig). Bij rotaties kan men delen van een virtuele boom verplaatsen door virtuele grootten aan te passen.

Voor zekere $0 < \lambda < 1$ en k_λ zijn op een nieuwe structuur met n elementen $\leq \lambda n$ zwakke deletions toelaatbaar alvorens de bovengenoemde criteria meer dan een factor k_λ slechter worden dan bij een nieuw-gebouwde structuur met m elementen.
Zij S een nieuw-gebouwde structuur met n_0 elementen. Na λn_0 zwakke deletions is S ongeschikt wegens de inefficiënte opslagvorm van de structuur. Daarom wordt na $\frac{1}{2}\lambda n_0$ zwakke deletions op S begonnen met de constructie van een nieuwe structuur S' uit de oude structuur. Laat S de grootte n_1 bereikt hebben. Door ervoor te zorgen dat de nieuwe structuur S' binnen $\frac{1}{6}\lambda n_1$ updates gereed is om de oude structuur S te vervangen, wordt gegarandeerd dat S vervangen wordt alvorens er in totaal λn_0

zwakke deletions op hebben plaatsgevonden.
De constructie van de nieuwe structuur S' geschiedt in twee fasen. (Deze twee fasen worden echter verdeeld over twee porties van $\frac{1}{12}\lambda n_1$ updates.) Tijdens de eerste fase wordt een nieuwe structuur gebouwd, die bestaat uit de keys uit de oude structuur van dat moment. Gedurende deze periode worden updates geplaatst in een buffer om later uitgevoerd te kunnen worden op de structuur die nu geconstrueerd wordt. Ten behoeve van queries zullen deze updates wel op de oude structuur uitgevoerd worden. Tijdens de tweede fase wordt de buffer geleegd door meer updates uit de buffer te verwerken dan erin worden geplaatst.

Om ervoor te zorgen dat S' op tijd klaar is om S te vervangen wordt per update op de gegevensstructuur de volgende hoeveelheid werk verricht:
- $k_\lambda U(n)$ werk wordt verricht aan S. (Hierbij staat $U(n)$ voor de hoeveelheid werk die een update met zich meebrengt.)
- $k_\lambda U(n_1 + \frac{1}{6}n_1) + P(n_1) / \frac{1}{6}\lambda n_1$ werk wordt verricht aan fases 1 en 2 van de constructie van S'. (Hierbij geeft $P(n)$ de hoeveelheid werk aan die gemoeid is met de bouw van een nieuwe datastructuur van n elementen.)

De grootheid die aangeeft hoeveel werk er wordt verricht aan de constructie van S' geeft hierbij alleen de *hoeveelheid* werk aan en niet de verdeling van taken.
Na $\frac{1}{6}\lambda n_1$ updates is dan in totaal $\frac{1}{6}\lambda n_1 \cdot k_\lambda U(n)$ werk verricht aan S en
$\frac{1}{6}\lambda n_1 \cdot k_\lambda U(n_1 + \frac{1}{6}n_1) + P(n_1) / \frac{1}{6}\lambda n_1$ aan het bouwen van S' en het uitvoeren van de gebufferde updates hierop. Alle werk dat verricht diende te worden is nu verricht zodat de buffer leeg is en S vervangen kan worden door S'.
Merk op dat deze $\frac{1}{6}\lambda n_1$ updates vallen binnen de toegestane $\frac{1}{2}\lambda n_1$ updates die op S' toegestaan zijn alvorens wellicht met de constructie van een nieuwe structuur S'' uit S' begonnen moet worden.

4.18. Toepassing van global rebuilding in de praktijk

Tijdens de constructiefase wordt enerzijds informatie uit de oude structuur S overgebracht naar de nieuwe structuur S' en anderzijds worden tevens op de oude structuur S nog updates uitgevoerd. Voor onze uitgebreidde BB[α]-bomen betekent dit dat bags niet zomaar gecopiëerd kunnen worden. In plaats daarvan moeten de keys uit de oude structuur S verzameld worden. Daarna worden de verzamelde keys dan in nieuwe bags geplaatst die dan op hun beurt in de nieuw te bouwen BB[α]-boom gezet worden.
De vervanging van S door S' is dus te verdelen in 3 fasen, namelijk
a) het verzamelen van de keys uit de structuur S
b) het plaatsen van deze keys in een (perfect gebalanceerde) BB[α]-boom met nieuwe bags
c) het verwerken van de updates in de buffer en het vrijgeven van de geheugenruimte die gebruikt wordt door de oude structuur S

Fase a

Het verzamelen van de keys gaat als volgt in zijn werk:
De keys worden van groot naar klein opgehaald en in een dubbel-gelinkte lijst van elementen gezet. Deze verzameling noemen we POOL. Daartoe houden we alleen een pointer, C genaamd, bij die wijst naar het element in de oude structuur dat het laatst is toegevoegd aan POOL. Wanneer het einde van een bag/lijst bereikt wordt kan via het bag-veld van het laatst bezochte element de bijbehorende bag/knoop bereikt worden zodat via het left-veld de volgende bag bezocht kan worden voor het collecteren van de keys/elementen. Bagsplitsing binnen de oude structuur is daarbij geen bezwaar.
Updates dienen alleen opgeslagen te worden in de buffer indien de betreffende key groter of gelijk is aan de laatst-bezochte key van het verzamel-proces, immers dan behoort de update-key tot de reeds verzamelde keys. Wanneer de update-key echter kleiner is dan de de key van pointer C dan hoeft de update niet in de buffer opgeslagen te worden omdat deze update dan nog automatisch meegenomen wordt.
Tijdens het verzamelen van keys wordt aan de elementen uit de oude structuur een verwijzing toegevoegd naar elementen uit de POOL met dezelfde key. Bij het opslaan van de updates in de buffer

kan dan meteen de positie van de update in de nieuwe structuur (dit is een element) opgeslagen worden. Een update uit de buffer kan dan in O(1) tijd plaatsvinden aangezien de positie bekend is.
Behalve de wijzer C wordt ook nog het aantal verzamelde keys bijgehouden ten behoeve van de bouwfase b).
Het spreekt voor zich dat deze fase O(n) tijd vergt.

Fase b

Uit de lijst POOL ter grootte O(n) dient nu de nieuwe datastructuur gebouwd te worden. Daartoe wordt allereerst POOL doorlopen en daarbij verdeeld in bags ter grootte O(log n). Deze bags worden geplaatst in een array. Daarna wordt m.b.v. dit array van bags een perfect gebalanceerde binaire boom gebouwd.
Beide stappen kosten O(n) tijd zodat de buffer eveneens O(n) groot zal zijn.

Fase c

Omdat aan de in de buffer opgeslagen update het element bekend is waar de update plaats heeft kan dit in O(1) tijd geschieden.
Bij de uitvoering van de gebufferde updates dient de opgeslagen positie nog mogelijk enigzins aangepast te worden. Elementen die in de oude structuur behoorden tot een en dezelfde bag kunnen namelijk door herverdeling van de keys over nieuwe bags terechtkomen in verschillende, maar naast elkaar gelegen bags. Aangezien bij de bepaling van de nieuwe update-positie geen rekening is gehouden met de key-scheiders in de interne knopen (van de nieuwe structuur) vereist dit een extra controle die in O(1) tijd te realiseren is wanneer men gebruik maakt van threads (zie [5]). Deze controle is alleen noodzakelijk indien de opgeslagen update-positie het kleinste dan wel grootste element van de bag is . De update-key dient dan vergeleken te worden met slechts één key-scheider en bij correctie wordt de update-positie dan het grootste element van de voorgaande bag resp. het kleinste element van de volgende bag.
Een uitwerking van deze wijze van uitbreiding met deletions wordt hier niet gegeven in verband met de grote hoeveelheden randgevallen die, bij implementatie, speciale aandacht verdienen (echter niet onoplosbaar zijn).

Stelling 4.1 : Zij V een verzameling keys ter grootte n. Er bestaat een datastructuur voor V z.d.d. de zoektijd O(log n) bedraagt en waarop inserties en deletions uitgevoerd kunnen worden in worst-case tijd O(1) mits de positie van de update bekend is.

HOOFDSTUK 5

Practische beschouwingen

De boomstructuur gebruikt imaginaire $\alpha = 1/4$. In verband met de correctheidsbewijzen voor de rotaties is voor een breukwaarde gekozen i.p.v. de waarde $1-\frac{1}{2}\sqrt{2}$ die Blum en Mehlhorn aangeven.
De resultaten uit 3.15 die de imaginaire balans vertalen naar de werkelijke balans tonen dat dan voor de werkelijke balans (van de boomstructuur in het geheugen) geldt $\alpha = 14/59$.
Voor de maximale padlengte volgt hieruit dat MPL = $2{,}56 \cdot {}^{2}\log n = 3{,}69 \cdot \ln n$.
Wat betreft de toegestane baggrootte moet gelden
splittingbagsize $SBS \geq \frac{6m}{p} \cdot MPL$ en
maximumbagsize $MBS = SBS + \frac{2m}{p} \cdot MPL$
De keuze van m wordt bepaald door eisen te stellen aan het percentage bags met een scanner (waardoor het aantal recent gesplitste bags relatief begrensd is).
De keuze van p is vrij. Kiest men p klein dan wordt de zoektijd van de gehele structuur slecht(er). Wanneer p daarentegen groot gekozen wordt dan moet men per toevoeging veel scanners verplaatsen en zal de hoeveelheid herbalanceerwerk navenant toenemen. De hoeveelheid werk blijft echter hoe dan ook O(1). Tamelijk willekeurig is genomen p=m zodat SBS=6·MPL en MBS=8·MPL.

De keuze voor $C(\alpha) = 13/21$ vloeide voort uit een poging om $C(\alpha)$ te bepalen zonder daarbij rekening te houden met het feit dat de opgeslagen balanswaarden gebaseerd zijn op verschillende nivo's van actualiteit. Het resulteerde in de eis $4/7 < C(\alpha) < 2/3$ zodat voor $C(\alpha) = 13/21$ genomen is.

5.19. Alternatieven voor de gemaakte keuzen

Voor de representatie van *scannersets* is de UNION-FIND-structuur uitermate geschikt omdat sets snel verenigd kunnen worden (hetgeen bij rotaties belangrijk is), de grootte snel bepaald kan worden en ze zich vanzelf weer oplossen (o.a. bij het verlaten van de boom) zodat er relatief weinig werk aan overhead nodig is.

Een klein nadeel is dat de geheugenruimte bij het verlaten van de boom niet per sé meteen vrijkomt maar de hoeveelheid geheugen die dit kost is beperkt immers het volgende geldt.
Zij n het aantal bladeren in een boom. Dan kunnen er in eerste instantie hoogstens n/14 scanners gecreëerd worden. Daarvan kan hooguit de helft geheugenruimte bezetten terwijl die niet meer 'actief' gebruikt wordt (dwz. de scanner is niet meer gekoppeld aan een blad/bag). De andere helft behoort wel nog bij een blad en hangt op UNION-FIND-diepte >1 waardoor de wortel nog niet ge-disposed mag worden.
De bladeren die losgekoppeld zijn van hun scanners (die in de eerstgenoemde helft zitten) kunnen dan wel opnieuw een scanner creëren maar het totaal aantal scanners dat niet meer bij een blad hoort maar wel wortel is van een UNION-FIND-structuur met grootte >2 (en dus op een passieve manier geheugen gebruikt) zal nooit aan kunnen groeien tot meer dan een aantal van n/14.

5.20. Practische bruikbaarheid van de scannerboom

De tot dusver ontwikkelde datastructuur en updatestrategie zijn gebaseerd op *theoretische* bruikbaarheid. Het is echter nog de vraag of de toepasbaarheid in de praktijk realistisch is. Immers wanneer de constanten die in de orde-notatie verstopt zitten gigantisch groot zijn dan is het praktische nut (vooralsnog) nihil. We hebben reeds gezien dat de zoektijd 9 keer zo groot kan worden doordat de maximale baggrootte 8 keer de padlengte bedraagt. Voor wat het herbalanceerwerk aangaat het volgende:
Aan het eind van iedere toevoeging worden scanners die behoren tot dichtbij gelegen buren verplaatst en wel maximaal 4 scanners ieder p posities omhoog. Dus van 4p knopen wordt de balans aangepast en zo nodig wordt er geherbalanceerd. Doordat p=m=27 worden zodoende 108 knopen bezocht. Een scannerboom begint dus pas zijn vruchten af te werpen indien het gemiddelde pad ruimschoots 100 lang is. Maw. het aantal opgeslagen gegevens ligt in de orde van $2^{100} = (2^{10})^{10} \approx (10^3)^{10} = 10^{30}$.
Aangezien de structuur en de toevoeghandelingen uitgebreider zijn dan gebruikelijk en bovendien weglatingen nog buiten beschouwing zijn gelaten is het nog zeer de vraag of het de moeite loont een gewone BB[α]-boom te verbeteren op de onderzochte wijze.

Referentielijst

1. N.Blum & K.Mehlhorn,On the average number of rebalancing operations in weight-balanced trees,Theoretical Computer Science 11 (1980) pag. 303-320.

2. D.Harel & G.Lueker,A data structure with moveable fingers and deletions,Techn.Rep.145,Dept.of ICS,University of California at Irvine,1979.

3. D.Harel,Fast updates of balanced trees with a garanteed time bound per update,manuscript,1979.

4. E.Horowitz & S.Sahni,Fundamentals of Data Structures (1976) pag. 248-257,Pitman

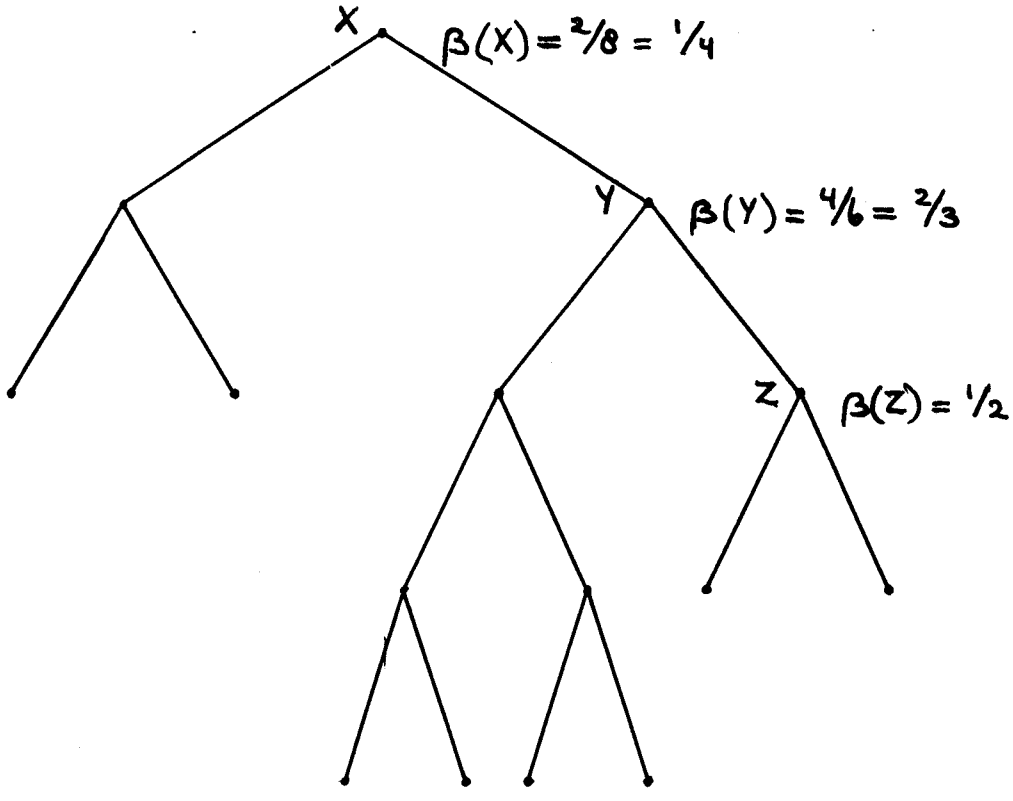
5. E.Horowitz & S.Sahni,Fundamentals of Data Structures (1976) pag. 239-243,Pitman

6. C.Levcopoulos & M.H.Overmars,A balanced search tree with O(1) worst-case update time,1987.

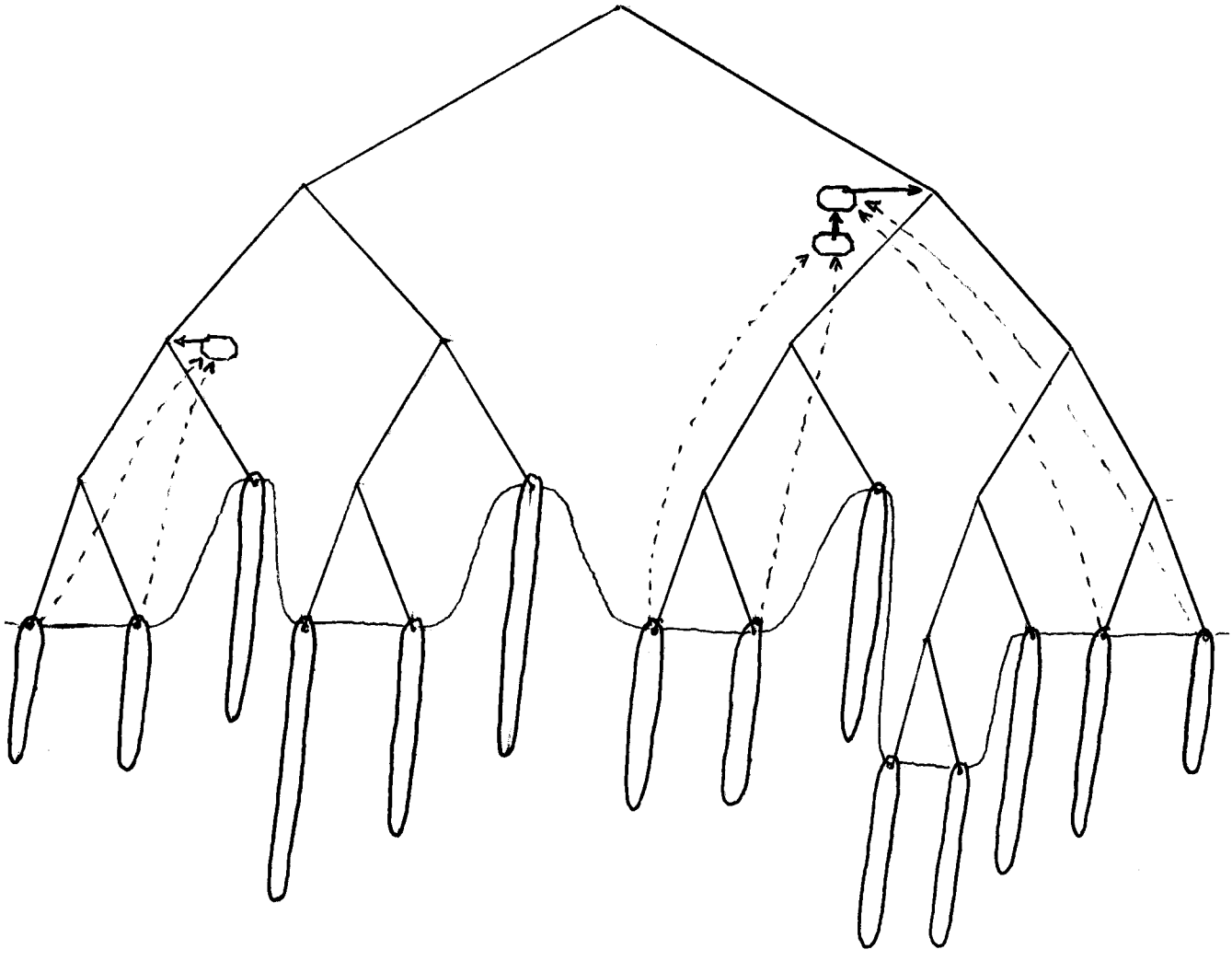
7. M.H.Overmars,An O(1) average time update scheme for balanced search trees,Department of Computer Science,University of Utrecht.

8. M.H.Overmars,The design of Dynamic Data Structures,Springer-Verlag Lecture Notes in Computer Science 156 pag. 67-72.

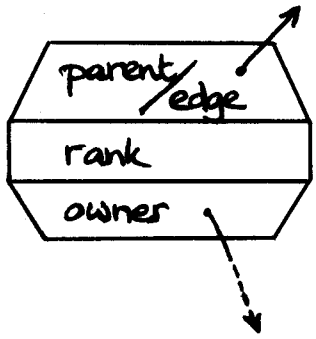
9. M.H.Overmars en J.van Leeuwen,Worst-case optimal insertion and deletion methods for decomposable searching methods,Inform.Proc.Lett.12 (1981) pag.168-173



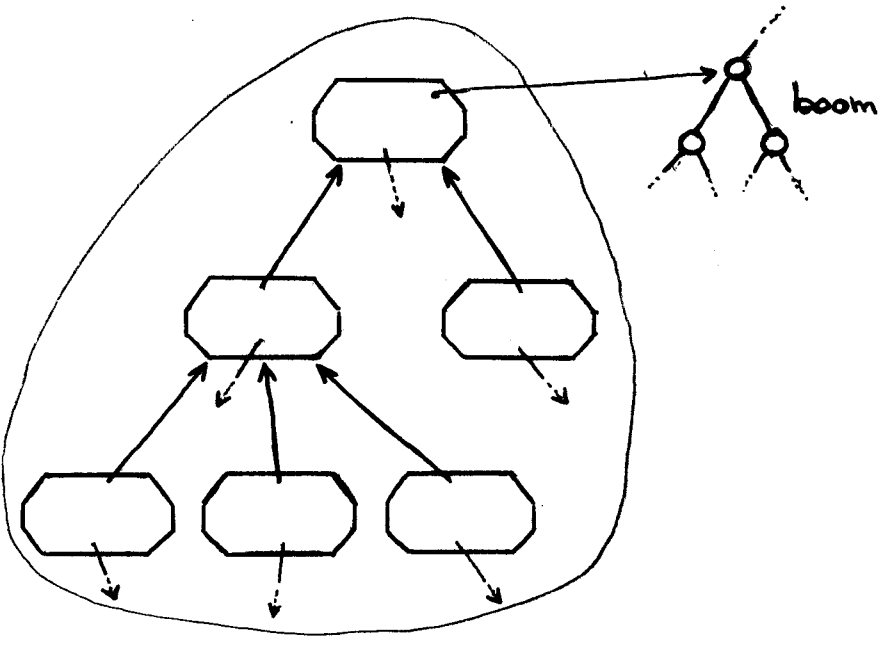
Figuur 3
BB[1/4]-boom



Figuur 6
De complete uitgebreide BB[α]-boom

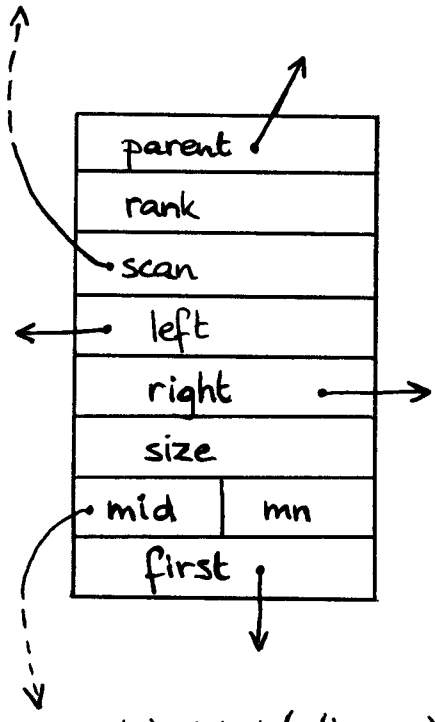


(a) scanner

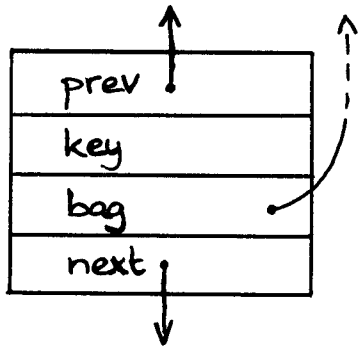


(b) scannerset / UNION-FIND-structuur

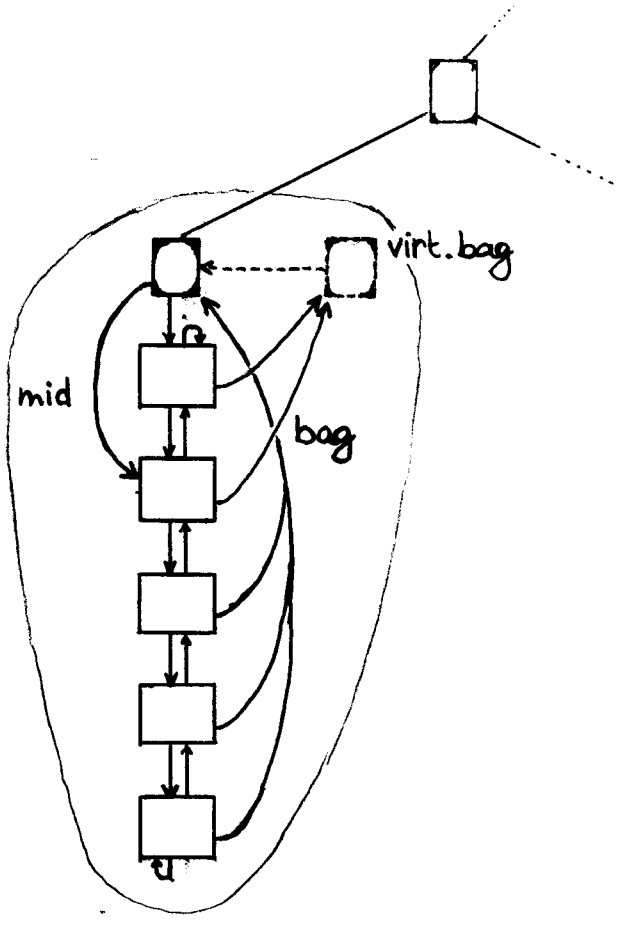
Figuur 5



(a) blad (/knoop)

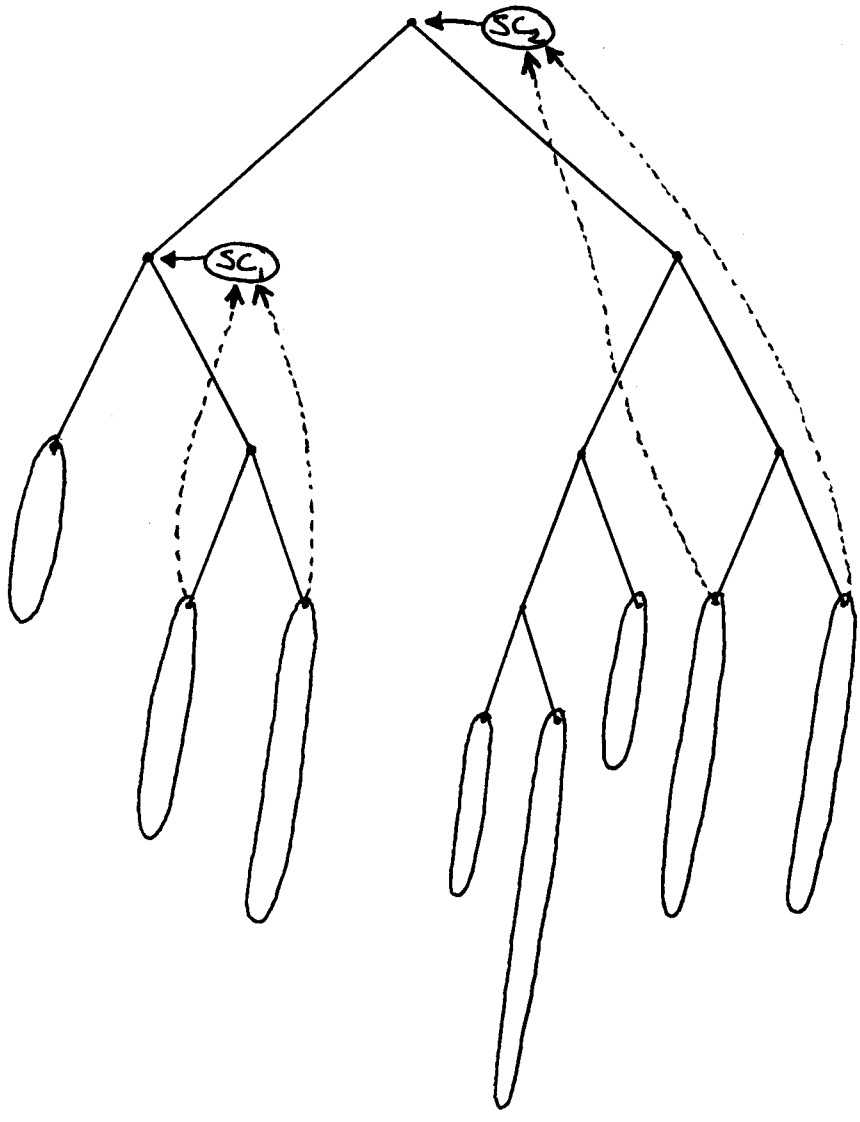


(b) element

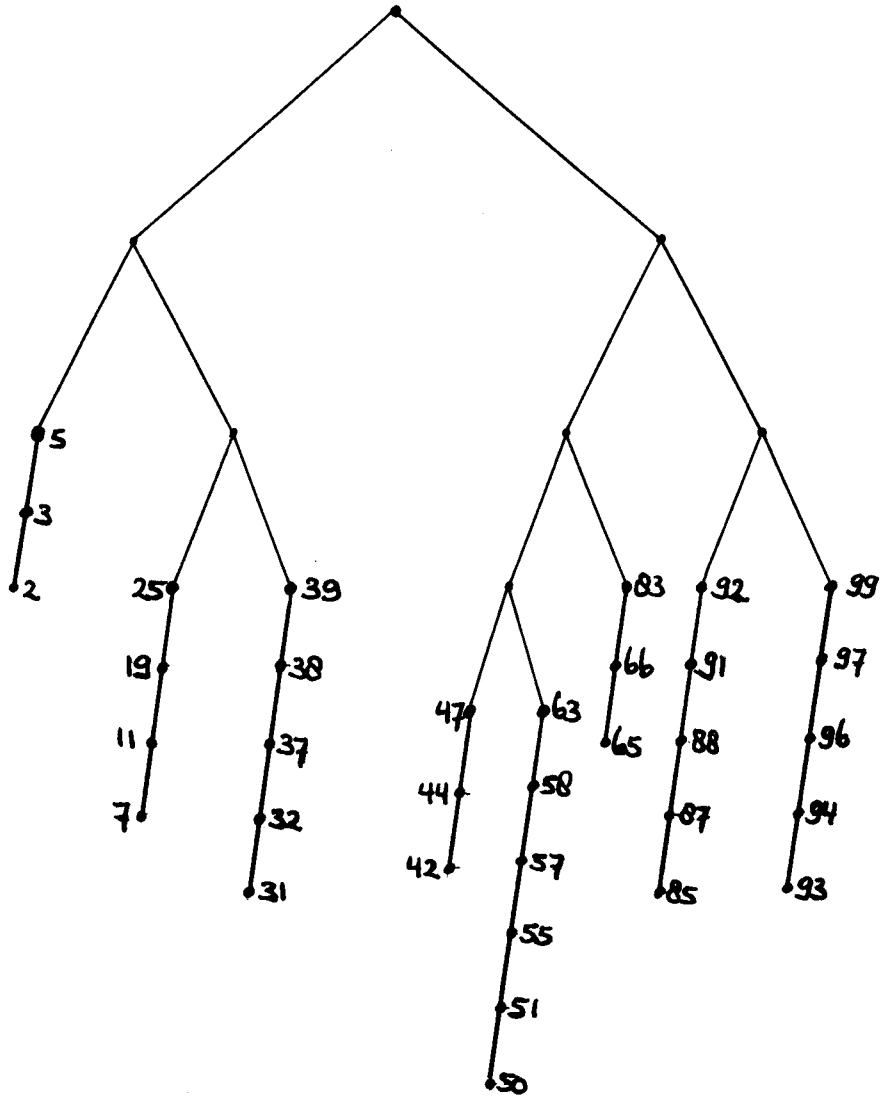


(c) bag

Figuur 4



Figuur 2
boom met scanners



Figuur 1

binaire boom met in de bladeren verzamelingen keys

Een datastructuur met zoektijd $O(\log n)$ en constante update-tijd

J.H. van der Erf

RUU-CS-87-19
November 1987

Rijksuniversiteit Utrecht

Vakgroep informatica

Budapestlaan 6 3584 CD Utrecht
Corr. adres: Postbus 80.012 3508 TA Utrecht
Telefoon 030-53 1454
The Netherlands

Een datastructuur met zoektijd $O(\log n)$ en constante update-tijd

J.H. van der Erf

Technical Report RUU-CS-87-19
November 1987

Department of Computer Science
University of Utrecht
P.O.Box 80.012
3508 TA Utrecht
the Netherlands