Algoritmen en Datastructuren II

Project Zeepbelbomen Deel 1

1 november 2015

Domien Van Steendam
2° Bachelor of Science : Informatica

---

1 – Theoretische vragen

VRAAG 1

Contradictie : Stel we hebben een zeepbelboom **B** met een bladzeepbel **z** die een kind in een andere zeepbel heeft.
Noem elke rij van zeepbellen tussen de wortelzeepbel **w** en een bladzeepbel van **B** een zeepbelnullpad (zeepbellen in volgorde van wortel naar bladzeepbel). Noem een kortste pad van alle zeepbel-nullpad **k** , dus waarvoor geldt dat er geen enkel zeepbel-nullpad bestaat met minder zeepbellen dan **k**.
Noem **d** het zeepbel-nullpad tussen **w** en **z**, deze bestaat zeker aangezien zeker 1 top in **z** een nullkind (lees “geen kind”) heeft. Zoniet zou **z** geen bladzeepbel zijn.
Nu zijn er 2 gevallen:

1) Het aantal zeepbellen in **d** is gelijk aan die van **k**. Noem **z’** de zeepbel die een top bevat dat kind is van een top in **z.** We definiëren nu het pad **d’** als een kortste pad van zeepbellen van **z’** naar de dichtstbijzijnde bladzeepbel. Maak nu een nieuw pad **d’’** = **d** + **d’.**
Nu bevat **d’** zeker 1 zeepbel, namelijk **z’**. Dus volgt uit bovenstaande gelijkheid dat **d’’ > d**. **B** is echter een zeepbelboom en er geldt dus steeds dat “elk pad van de wortelzeepbel **w** naar een bladzeepbel evenveel zeepbellen bevat.”. M.a.w. elke zeepbel-nullpad is even lang (waarbij de zeepbellen als elementen genomen worden).
Deze eigenschap geldt niet voor **d’’ en d** en dus kan **B** geen zeepbelboom zijn.

2) Het aantal zeepbellen in **d** is niet gelijk aan die van **k**. Aangezien **B** een zeepbelboom is, dan geldt “elk pad van de wortelzeepbel **w** naar een bladzeepbel bevat evenveel zeepbellen.”. Dit geldt niet voor **d** en **k**. **B** kan dus geen zeepbelboom zijn, wat tegen onze veronderstelling is.

Er kan dus geen zeepbelboom geconstrueerd worden die een bladzeepbel heeft die een kind in een andere zeepbel heeft.

Qed.

VRAAG 2 A

Voor binaire bomen is de maximale diepte log n , met n het aantal toppen. Hoe dieper de boom , hoe meer toppen er moeten zijn.

We kunnen dit vorige toepassen op een zeepbelboom (waarbij we “toppen” vervangen door “zeepbellen”) waarbij we kunnen zeggen : “Hoe dieper de zeepbelboom, hoe meer zeepbellen er moeten zijn”. Beschouw nu n toppen. We zullen met deze toppen nu een zeepbelboom maken door de toppen een binaire boom te maken en ze vervolgens in zeepbellen te stoppen. We willen echter dat de diepte (in zeepbellen) maximaal is, m.a.w. we proberen zoveel mogelijk zeepbellen te tekenen en de eigenschappen van de zeepbelboom te behouden.

We hebben een vast aantal toppen zitten en een vaste limiet voor het aantal toppen binnen één zeepbel (k) en we willen zoveel mogelijk zeepbellen hebben.

Nu maken we met de n toppen (en dus ook n zeepbellen) een binaire boom **B** die zo dicht mogelijk bij een perfect gebalanceerde binaire boom aanleunt. (Als n kan geschreven worden als “2^a+1” (a is een natuurlijk getal) dan zal deze boom precies perfect gebalanceerd zijn). De binaire boom **B** heeft nu een perfect gebalanceerd deel (zie zwarte driehoek in figuur)en eventueel een extra (niet-compleet) niveau met minstens 1 top (zie grijze cirkels in figuur). De diepte van deze boom is O(log n).

De beste oplossing om zoveel mogelijk zeepbellen te hebben is door elke top in een unieke zeepbel te stoppen, m.a.w. elke zeepbel bevat precies 1 top. Aangezien k minstens 2 is, wordt de limiet dus niet overschreden. Echter zullen de grijze toppen (zie figuur) er nu voor zorgen dat het nullpad niet steeds even lang is. In dat geval sluit je de grijze toppen aan bij de zeepbel van hun ouder. In die zeepbellen zitten er nu 2 toppen.
Dit kan steeds zonder de limiet te overschrijden, k is minstens 2.

B voldoet nu aan alle eigenschappen van een zeepbelboom. Als we elke zeepbel nu beschouwen als 1 top in een binaire boom **B’,** dan is **B’** perfect gebalanceerd en heeft dus diepte log(n’) met n’ het aantal zeepbellen. **B’** heeft dezelfde vorm als de zwarte driehoek op de figuur. De grijze toppen hun rol zijn hier dus uitgeschakeld voor zolang ze niet een niveau volledig opvullen. Als m nu het niveau van de grijze toppen voorstelt, dan zijn er 2^m verschillende zeepbelbomen bomen die na de operatie “Als we elke zeepbel nu beschouwen als 1 top in een binaire boom **B’”.** Namelijk een zeepbelboom met 0,1,…,m-1 zeepbellen met 2 toppen. We moeten dus een formule hebben die de zeepbeldiepte van de zwarte driehoek gelijkstelt aan die van de andere m-1 equivalente zeepbelbomen.
We tellen hiervoor een 1 op bij het aantal toppen waardoor de zwarte driehoek in het zelfde niveau geplaatst wordt. Daar nemen we de log (dus log(n+1)) van en we bekomen het niveau van de grijze toppen. Echter horen die allemaal bij de zeepbellen van hun ouder. Die ouders zitten op het niveau erboven. Die ouders zijn komen overeen met de wortels van de bladzeepbellen. Dus ten slotte trekken we nog een 1 af van log(n+1) (= het niveau van de grijze toppen).

De maximale diepte van een k-zeepbelboom met n toppen is dus **log(n+1)-1 = Θ(log n)**.

Qed.

bv. n=7 , k=2 (k mag variëren aangezien de formule onafhankelijk is van k). log(7+1)-1
= 3-1 = 2.

VRAAG 2 B

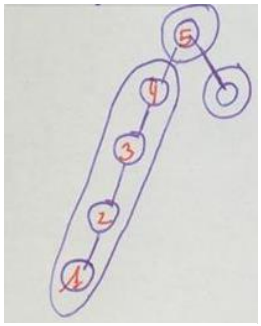
Ik bespreek kort de werkwijze en we zullen daarna alles in een formule gieten. Alle gebruikte lemma's worden onderaan het globale bewijs bewezen.

Werkwijze:

Voor n toppen zullen we een k-zeepbelboom construeren met een maximale diepte in toppen. In mijn werkwijze werken we van onder naar boven.
Hiervoor gaan we als volgt te werk:
Maak een eerste zeepbel door k toppen toe te voegen die op 1 lijn liggen. Als k groter of gelijk is aan n zijn we natuurlijk al klaar. De diepte is op dat moment dan n-1, maar dit is triviaal.
Om nu een extra top toe te voegen op diezelfde lijn , hebben we 2 toppen nodig. Namelijk 1 voor op de lijn en 1 als rechterkind om de zeepbelstructuur te behouden.
Ik ga voortaan elk zo'n top met zijn rechterdeelboom een element noemen. Alle toppen in die rechterdeelboom zitten apart in een zeepbel. Zo zal het zeepbelpad overal in balans zijn.

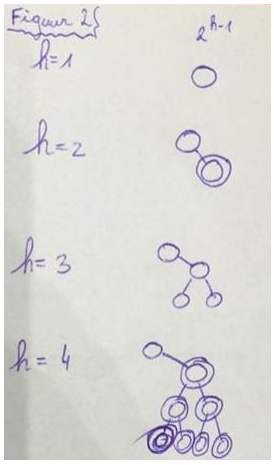
Per zeepbel kunnen we dus k elementen toevoegen.
Als het volgend element toegevoegd wordt , komt die in een nieuwe zeepbel erboven.

Op figuur 1 kan je een visualisatie zien van hoe ik te werk ga voor k=4.
(De rode cijfers staat voor de volgorde waarin ik de elementen heb toegevoegd zodat het duidelijk is. Dit zijn dus **niet** de sleutelwaardes).



Figuur 2

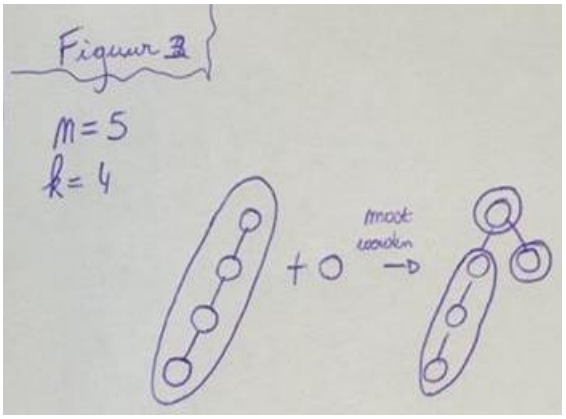
Definieer de hoogte **h** als de hoogte in zeepbellen van onder naar boven en startend bij **1**.
De eerste zeepbel links onderaan is dus hoogte 1, zijn ouderzeepbel bevindt zich op hoogte 2, … enz.
Lemma 1 (Bewijs van dit lemma bevindt zich onder het globale bewijs):
Het aantal toppen die je nodig hebt voor 1 element toe toevoegen in een zeepbel op hoogte **h** is 2^{h-1}.
De voor elementen in de eerste zeepbel (h=1) heb je 1 top nodig, voor de tweede zeepbel 2, dan 4 , dan 8 enzovoort. In figuur 2 worden de elementen weergegeven voor enkel hoogte 1 t.e.m. 4. De rechterdeelboom is steeds een perfect gebalanceerde boom.



Figuur 1

We voegen zoals hierboven dus steeds elementen toe tot en met we niet genoeg toppen meer hebben om een nieuw element te vormen. Als we dit punt bereiken zitten we dus met een overschot van toppen. Als h de hoogte is van de zeepbel waar het volgende element zou moeten komen dan zal de overschot strikt kleiner zijn dan 2^{h-1}.
Anders zou je wel nog een nieuw element kunnen vormen.
We willen een boom met n toppen dus zal deze overschot nog een plaats moeten krijgen.
Om dit op te lossen voegen we deze rest- toppen toe in de bladzeepbellen waar nog plaats is. Dit kan dus al niet de zeepbel links onderaan zijn, want die zit al vol. Voor elke andere bladzeepbel zijn er k-1 plaatsen voor toppen in te stoppen.

In Lemma 2 zullen we bewijzen dat er steeds genoeg plaatsen zijn voor alle toppen van onze rest.
Er is 1 uitzondering: Als n=k+1 dan zijn er (nog) geen niet-volle bladzeepbellen. Dan plaats je de extra top rechts van de wortel, en de wortel zit alleen in een zeepbel. Alle andere toppen zitten in dezelfde zeepbel (Zie figuur 3). Nu heb je wel een niet-volle bladzeepbel.
Voor n > k+1 ⇔ n >= k+2 kan het bovenstaande algoritme perfect uitgevoerd worden, want dan heb je vanaf de tweede zeepbel zeker 2 toppen waarvan 1 dan een bladzeepbel wordt.



Figuur 3 : Uitzondering , n=k+1

We zullen nu a.d.h.v. deze werkwijze een formule opstellen waarmee de diepte van de toppen op de rechte lijn berekend kan worden.

Formule:

We gaan eerst het aantal toppen berekenen die op de rechte lijn (uiterst links van de boom) liggen. De diepte is dan dit aantal -1, want de diepte is het aantal bogen. (Net zoals er n-1 gaten/tussenschotten zijn tussen n bomen langs een rijweg).

Lemma 3: De hoogte **h** (op de manier zoals ik ze hierboven gedefinieerd heb) van de zeepbelboom geconstrueerd zoals hierboven is $\lfloor \log_2 \lceil \frac{n}{k} \rceil \rfloor + 1$.

We weten dat alle zeepbellen onder de bovenste zeepbel volledig gevuld zijn. Daardoor kunnen we a.d.h.v. de hoogte **h** gemakkelijk uitrekenen hoeveel toppen er gebruikt zijn (onder de bovenste zeepbel). Per zeepbel hebben we intern **k** toppen. Er zijn dus minstens $(\mathrm{h}-1)*\mathrm{k}$ toppen op de lijn.

Ten slotte moeten de toppen in de bovenste zeepbel berekend worden. Van deze zeepbel hebben we echter geen garantie dat deze volledig gevuld is. De meest eenvoudige manier om deze toppen te berekenen leek mij als volgt:

1. Kijk hoeveel toppen er nog te verdelen zijn op het niveau van de bovenste zeepbel.

2. Deel dit aantal door 2^{h-1} (= aantal toppen dat vereist is voor een element te vormen op hoogte h). De (gehele quotiënt) is dan het aantal elementen dat we nog kunnen toevoegen. (Elk element draagt bij tot verhoging van de diepte met 1 eenheid).
*De rest*k representeert dan het aantal rest-toppen, maar die brengen niet bij tot de diepte aangezien die niet op de rechte lijn geplaatst worden. We weten volgens Lemma 2 wel dat we die rest-toppen steeds "kwijt" kunnen met behoudens van de zeepbel-structuur.*

Om het aantal resterende toppen voor de bovenste zeepbel te berekenen trekken we het aantal gebruikte toppen t.e.m. de tweede-onderste zeepbel af van **n.** Het aantal gebruikte toppen voor een zeepbel op hoogte h kennen we, namelijk $k*2^{h-1}$. We maken de sommatie van t.e.m. de tweede-onderste zeepbel die zich bevindt op hoogte h-1.

Dus $\sum_{i=1}^{h-1} k * 2^{i-1} = (Zie\ discrete\ Wiskunde)\ k * (2^{(h-1-1)+1} - 1) = k * (2^{h-1} - 1)$ is het aantal al geplaatste toppen als we aan de bovenste zeepbel (hoogte h) aangekomen zijn.

Het aantal elementen in de bovenste zeepbel is dan (zie puntje 2) $\lfloor\frac{n-k*(2^{h-1}-1)}{2^{h-1}}\rfloor$.

De diepte van de volledige boom is bijgevolg $(\mathrm{h}-1)*\mathrm{k}+\left\lfloor\frac{n-k*(2^{h-1}-1)}{2^{h-1}}\right\rfloor-1$, voor n!=k+1. Indien n= k+1 , dan is diepte n-2. (zie Figuur 3)

Bewijzen van de vermelde lemma's:

Lemma 1: Het aantal toppen die je nodig hebt voor 1 element toe toevoegen in een zeepbel op hoogte **h** is 2^{h-1}.

Bewijs : De rechterdeelboom van een element (op zeepbel-hoogte h) moet h-1 niveaus (≠dieptes) hebben opdat de zeepbelbalancering behouden blijft. De toppen in die rechterdeelboom zitten stuk voor stuk in een aparte zeepbel dus elke nieuwe top is ook een nieuwe zeepbel. De diepte in toppen van deze rechterdeelboom is dus gelijk aan de diepte in zeepbellen en dat is (h-1)-1= h-2. De rechterdeelboom is moet in onze werkwijze steeds een perfect gebalanceerde boom zijn. We weten uit DA1 het verband tussen de diepte **d** en het aantal toppen **n** in een perfect gebalanceerde boom, namelijk $n = 2^{d+1} - 1.$
We hebben voor onze rechterdeelboom voor een element op zeepbel-hoogte h precies

$2^{(h-2)+1} - 1 = 2^{h-1} - 1$ toppen nodig.

We tellen ten slotte de top waaraan de rechterdeelboom hangt erbij en dan bekomen we 2^{h-1}.

Qed.

Lemma 2: Bij het construeren van een zeepbelboom zoals beschreven in de werkwijze hierboven, zullen we steeds de rest-toppen kunnen plaatsen zonder dat de zeepbelboom-structuur verstrooid worden.

Bewijs :

(In dit bewijs gebruik ik het begrip 'hoogte' zoals ik ze in mijn werkwijze gedefinieerd heb).

Stel nu dat we ,bij het construeren van een boom zoals in mijn werkwijze, geen elementen meer kunnen toevoegen op hoogte $\boldsymbol{h}$ (is ≥3 , anders zouden we geen rest hebben en voor h=2 voegen we de toe aan de rechterdeelboom van de top eronder) en er $\boldsymbol{r}$ rest-toppen overblijven. Ik ga aantonen dat het aantal plaatsen in de bladzeepbellen steeds groter is dan het aantal rest-toppen. We weten dat de rechterdeelbomen van elk element perfect gebalanceerd is en dus dat alle bladzeepbellen (in die rechterdeelboom) zich op hetzelfde niveau bevinden. Uit DA1 weten we tevens dat voor perfect gebalanceerde bomen van diepte $\boldsymbol{L}$ er precies 2^L bladeren zijn. Voor een element van zeepbel-hoogte $\boldsymbol{h}$' heeft de rechterdeelboom $\boldsymbol{h}' - \mathbf{1}$ niveaus (diepte = $\boldsymbol{h}' - \mathbf{2}$) en dus $2^{h'-2}$ bladeren.
Aangezien elke top en dus ook elk blad alleen zit in zijn eigen zeepbel, kunnen we voor elk blad precies $k-1$ rest-toppen toevoegen.
Een element op hoogte $\boldsymbol{h}$' heeft dus plaats voor precies $(k-1)*2^{h'-2}$ rest-toppen.
Voor een zeepbel op hoogte $\boldsymbol{h}$', uitgenomen de bovenste, zijn er zeker dan $k*(k-1)*2^{h'-2}$.
Er zijn , uitgenomen de bovenste, $h-1$ zeepbellen.

We tellen alle bladeren nu bij elkaar op. We beginnen echter met zeepbelhoogte 2, want de onderste heeft natuurlijk geen (externe) bladeren. We hebben dan

$$\sum_{h'=2}^{h-1} k*(k-1)*2^{h'-2} = k*(k-1)*\sum_{h'=2}^{h-1} 2^{h'-2} = k*(k-1)*(2^{h-2}-1)$$

Aangezien we niet genoeg toppen hebben voor een nieuw element op hoogte $\boldsymbol{h}$ **,** is $\boldsymbol{r} < 2^{h-1}$.
Als we kunnen aantonen dat ons aantal plaatsen groter dan of gelijk aan dit laatste getal is, dan is dit lemma bewezen.
We bewijzen dit d.m.v. inductie op h, $h \geq 3$ $zoals\ hierboven\ al\ vermeld\ is$:

$$k*(k-1)*(2^{h-2}-1) \geq 2^{h-1}$$

Het aantal mogelijke plaatsen voor de rest-toppen , bij het construeren van een zeepbelboom volgens bovenstaande werkwijze, is dus steeds groter dan of gelijk aan het aantal toppen in de rest.

Qed.

Lemma 3: De hoogte **h** (op de manier zoals ik ze hierboven gedefinieerd heb) van de zeepbelboom geconstrueerd zoals hierboven is $\lfloor \log_2 \lceil \frac{n}{k} \rceil \rfloor + 1$.

Bewijs:

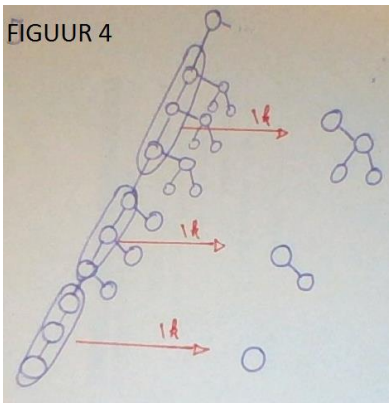
Beschouw een k-zeepbelboom geconstrueerd zoals hier boven met n toppen.

Bekijk de deling $\frac{n}{k}$ als $\frac{n_0+n_1+\ldots+n_h}{k}$, met n_i het aantal gebruikte toppen voor de zeepbel op hoogte i te vullen (dus ook de toppen in de rechterdeelbomen van elk element in zeepbel i). h is de hoogte van de bovenste zeepbel.

De deling $\frac{n}{k}$ kunnen we voorstellen alsof elke (volle) zeepbel afgebeeld wordt op een zeepbel met 1 element, want elke volle zeepbel bevat k elementen(≠toppen) en als men het delen per zeepbel bekijkt dan zal er 1 element per zeepbel overblijven.

Indien de bovenste zeepbel geen k elementen bezit, kunnen we de deling moeilijk visualiseren volgens bovenstaande opvatting. In dat geval zou de bovenste zeepbel geprojecteerd worden op een deel van een element (bv. 1/3 van een element), wat niet gedefinieerd is. Hoedanook zal de rest nog steeds strikt groter dan 0 zijn, want er zit zeker 1 element in de bovenste zeepbel. Zoniet zou deze leeg zijn en de bovenste zeepbel niet kunnen zijn. Deze bovenste zeepbel moeten we natuurlijk meetellen en dat is waarom we $\frac{n}{k}$ steeds afronden naar boven zodat we het beschouwen als een zeepbel met 1 element, want vanaf er 1 element in zit, wordt deze meegeteld als extra bel. Op deze manier gaat de bovenste zeepbel niet “verloren”. In figuur 4 wordt deze projectie in beeld gebracht.

FIGUUR 4



We hebben na de projectie h nieuwe elementen, voor elke uiterst linkse zeepbel precies één. De elementen verbinden we nu langs links en we bekomen een nieuwe perfect gebalanceerde boom. Dit is zo omdat een element van zeepbelhoogte h een verdubbeling is van een element van zeepbelhoogte

h-1 die dan aan elkaar gehangen worden door de wortel van de ene kind te maken van de andere. De diepte wordt in dat geval vergroot met 1.

Tevens is elke top in deze nieuwe boom een aparte zeepbel, want van elke oorspronkelijke zeepbel behoedden we na de projectie 1 element van die zeepbel. In onze werkwijze hierboven zit elke top in een element in een aparte zeepbel, we hebben dus een binaire boom. Alle oorspronkelijk uiterst linkse zeepbellen worden nu gerepresenteerd door 1 top. De diepte in toppen in deze boom is dus gelijk aan de diepte in zeepbellen in onze oorspronkelijke zeepbelboom. In een perfect gebalanceerde binaire boom is de diepte $\lfloor \log_2 m \rfloor$, $m\ is\ het\ aantal\ toppen$.

Bijgevolg is de $hoogte\ (=\ diepte\ +\ 1)$ dan $\lfloor \log_2 \lceil \frac{n}{k} \rceil \rfloor + 1$.

Qed.

2 – Implementatie

Mijn zeepbelboom wordt voorgesteld door een normale binaire zoekboom. De nodes bevatten echter een extra veld die refereert naar een instantie van de MijnZeepbel-klasse die de Zeepbel-interface implementeert. De zeepbellen worden onderling uit elkaar gehaald door middel van een uniek getal startend vanaf 1. Telkens een nieuwe zeepbel aangemaakt wordt , zal deze een niet eerder gebruikt getal toegekend krijgen die opgeslagen worden in een veld van de zeepbel-instantie. Dit getal wordt in mijn implementatie ook wel eens de kleur van de zeepbel genoemd. (Aangezien elke zeepbelinstantie een uniek getal heeft, kan er gewoon met '==' vergeleken worden en hoeft er dus geen 'equals()' overschreven te worden).

Ik bespreek per balanceringsvariant enkel het toevoegen(add) het verwijderen(remove). De ideeën van de andere bewerkingen zijn duidelijk uit hun implementatie en spreken voor zich.
Indien sommige methoden, klassen of andere functionaleiten niet duidelijk zouden zijn uit het verslag, raad ik aan om de documentatie bij de corresponderende methodes/klassen eens door te nemen.

Testen en experimenten staan in de map "Test".

Balanceringsvariant 1 – Zo weinig mogelijke wijzingen aan de onderliggende binaire zoekboom.

Toevoegen:

Eerst wordt de nieuwe sleutel gewoon toegevoegd als nieuwe sleutel en deel van dezelfde zeepbel van zijn ouder. Als die zeepbel nu overvol zit , wordt er geherbalanceerd. Het herbalanceren gebeurt steeds door de wortel van de zeepbel naar boven te duwen (ongeacht of die plaats heeft of niet).

(met "wortel" hieronder bedoel ik steeds de wortel van de zeepbel waarover ik spreek). Als de wortel en zijn 2 kinderen tot dezelfde zeepbel behoren, dan kan de wortel eenvoudig geduwd worden naar de bovenste zeepbel, analoog aan het voorbeeld in de opgave. Als de 2 kinderen en de wortel oorsponkelijk in dezelfde zeepbel zaten, is er nu een probleem want de zeepbel is niet meer samenhangend. Dit wordt opgelost door de linkse deelboom (binnen dezelfde zeepbel) van de oorspronkelijke wortel in een gloednieuwe zeepbel te stoppen. (De rechtse kant mag onveranderd blijven). Deze operatie gebruikt de zeepbelKeysiterator. Dit kost O(k log k), want die overloopt de toppen in die zeepbel van klein naar groot. Dit hoeft niet in volgorde te gebeuren dus kan dit sneller door de breedte-eerst te doorlopen. Dit zou dan O(k) kosten.

Stel nu dat de wortel slechts 1 kind heeft of zijn 2 kinderen in verschillende zeepbellen zitten, dan kunnen we bovenstaande niet uitvoeren. Voor deze 2 gevallen kunnen we hetzelfde algoritme toepassen.

Mijn oplossing hiervoor is als volgt:

We kijken eerst of er een zigzag-structuur te vinden is in deze zeepbel, t.t.z. ofdat het linkerkind een rechterkind heeft of het rechterkind een linkerkind heeft. Indien dit zo is, wordt de gepaste dubbele rotatie uitgevoerd. Indien dit niet zo is, liggen de wortel, zijn kind (van dezelfde zeepbel) en zijn

kleinkind op 1 rechte lijn. Hierop kunnen we een eenvoudige enkele rotatie op uitvoeren. Deze oplossing kan volledig in constante tijd uitgevoerd worden, zowel de checks als de rotatie zelf.

Nu heeft deze nieuwe wortel 2 kinderen en passen we hierop dezelfde procedure als in het bovenstaande geval toe.

Ten slotte kijken we of de zeepbel erboven (lees “de ouderzeepbel”) nu overvol zit. Indien dit zo is, passen we volledig dezelfde procedure toe als hierboven, maar met de wortel van deze ouderzeepbel. Dit stopt gegarandeerd , want we gaan steeds dichter bij de wortel van de volledige zeepbelboom. Het speciaal geval is dus indien de boom-wortel de wortel is die moet geduwd worden. In dat geval wordt deze een gloednieuwe zeepbel met zichzelf als het enige element in deze zeepbel. Er kan maximaal O(log n) keer gebalanceerd worden aangezien de maximale zeepbeldiepte O(log n) is (zie theorie vraag 2a).

Verwijderen:

Geimplementeerd maar geen tijd meer gehad om mijn algoritme neer te schrijven.

Balanceringsvariant 2 – Steeds de middelste sleutel naar boven duwen.

Toevoegen:

Indien een zeepbel na toevoeging van sleutel **s** overvol is, dan worden alle sleutels (zie maakGesorteerdeArray(…)) in een gesorteerde array gestopt met behulp van de zeepbelNodesIterator . Dit gebeurt in O(n log(n)) tijd. Vervolgens wordt uit de middelste top als nieuwe wortel van de zeepbel aangezien. De linker- en rechterhelft worden elk een aparte zeepbel die zo goed mogelijk gebalanceerd zal zijn (er wordt gestreefd naar een perfecte balancering).
De middelste sleutel wordt vervolgens naar de zeepbel erboven geduwd. Indien er geen is , wordt deze een gloednieuwe zeepbel.
Merk op dat deze procedure enkel voor een bladzeepbel uitgevoerd kan worden, en dus ook hoogstens 1 keer uitgevoerd wordt bij toevoeging van een nieuwe sleutel.
Indien de zeepbel erboven ook vol is, wordt de procedure van balanceringsvariant 1 recursief tot aan de wortel uitgevoerd. (In de opgave is niet gezegd dat dit verboden is). Dus O(log n * log (n-1)) = O(log n * log n). De totale complexiteit is dus O(n log(n) + log n * log n) = O(n log n)

Verwijderen:

Zelfde techniek als variant 1.

Balanceringsvariant 3 – Zoveel mogelijk sleutel naar boven duwen.

Toevoegen:

Geimplementeerd maar geen tijd meer gehad om mijn algoritme neer te schrijven.

Verwijderen:

Zelfde techniek als variant 1.

3 - experimenten

Hiervoor werd gebruikt gemaakt van experimenten.class in de map “test.

Er zijn 2 functies:

“EenTweeDrieEnz” voegt de rij 1,…,amount toe aan de zeepbelboom in die volgorde.

De tweede functie maakt gebruik van random gegenereerde arrays van integers. Die getallen worden dan in dezelfde volgorde toegevoegd aan de zeepbelboom.

De resultaten heb ik opgeslagen in het meegegeven Excel bestand, namelijk “experimenten.xlsx”.