Universiteit van Groningen

Gevorderde Algoritmen en Datastructuren

door

Jos van der Til & Rene Zuidhof

December 17, 2010

Contents

Chapter 1

Introductie

Een deel van de cursus Gevorderde Algoritmen en Datastructuren is een praticum, deze bestaat uit twee opdrachten. De eerste opdracht staat in dit document beschreven. Het doel van deze opdracht luidt: *"ervaring opdoen met het analyseren en vergelijken van zoekalgoritmen. De opdracht sluit aan bij Chapter 3 van het leerboek Algorithm Design van Goodrich en Tamassia. Het is de bedoeling om twee programma's te ontwikkelen, waarmee je kunt bepalen welke datastructuur beter is om een doorzoekbaar geordend lexicon (searchable ordered dictionary) te implementeren: een conventionele binaire zoekboom, of een van de volgende alternatieve datastructuren: AVL-boom, (2,4)-boom, splay tree, skip list. De programma's dienen een tekstbestand woord voor woord te lezen, waarbij elk nieuw woord wordt toegevoegd aan de datastructuur. Elk woord komt dus slechts ´e´enmaal voor in de datastructuur."*

Een aantal van deze datastructuren staat beschreven in dit document, naast de beschrijving van de datastructuur is ook de uitkomst van de analyses die zijn uitgevoerd beschreven. In deze analyses is gekeken naar het aantal stringvergelijkingen, het aantal toekenningen in de code en de tijd die nodig is bij bepaalde handelingen.

Als eerste zal in hoofdstuk 2 de conventionele binaire zoekboom behandeld worden. Hierna wordt gekeken naar gebalanceerde bomen, namelijk de AVL-boom (hoofdstuk 4) en de (2,4)-boom (hoofdstuk 5). Als laatste wordt in hoofdstuk 6 een alternatieve datastructuur behandeld, de skip list.

Verwachting..... ....

Chapter 2

Binaire zoekboom

Introductie Een van de simpelste zoekbomen is de binaire zoekboom. Elke interne knoop heeft twee kinderen waarvan de linkerkant lager is en de rechterkant hoger is dan de waarde in de interne knoop.

Zoeken Bij het zoeken naar een waarde k in een binaire boom wordt begonnen in de root knoop. Als k kleiner is dan de waarde in de knoop p wordt verder gezocht in het linkerkind van p. Wanneer k groter is dan de waarde in p wordt verder gezocht in het rechterkind van p. Dit wordt gedaan tot een externe knoop is bereikt of wanneer k gevonden is.

Toevoegen Met behulp van bovenstaande zoek methode wordt gezocht naar de toe te voegen waarde k. Wanneer de gevonden waarde p een interne knoop is wordt er direct gestopt met toevoegen, k zit namelijk al in de boom. Wanneer p een externe knoop is wordt deze vervangen door k en worden twee externe kinderen toegevoegd aan k.

Analyse

Chapter 3

Gebalanceerde bomen

Ongebalanceerd De in hoofdstuk 2 behandelde binaire boom is ongebalanceerd. Dit houdt in dat de boom aan een kant erg veel waardes kan hebben en aan de andere kant niks (in het slechtste geval). Hierdoor zal de zoektijd behoorlijk toenemen.

Gebalanceerd Om de problemen van een ongebalanceerde boom te voorkomen zijn een aantal bomen bedacht die er voor zorgen dat een boom altijd gebalanceerd is. Een aanal van deze bomen zijn de AVL-boom, de (2,4)-boom en de Splay-boom. De eerste twee van deze bomen zullen in hoofdstuk 4 en 5 behandeld worden.

Chapter 4

AVL-boom

Chapter 5

(2,4)-boom

Intro De tweede gebalanceerde boom die behanded wordt is de (2,4)-boom. Alle interne knopen van deze boom hebben één tot drie keys, het aantal kinderen van een knoop is het aantal keys + 1. Dit betekent dus dat elke knoop 2, 3 of 4 kinderen heeft, vandaar de naam (2,4)-boom (soms ook (2,3,4)-boom genoemd). Een andere eigenschap van deze boom is dat de diepte van alle externe knopen gelijk is. Deze eigenschappen zorgen ervoor dat de hoogte van een (2,4)-boom met n items $\Theta(logn)$ is. Deze eigenschappen worden bewaard door na elke toevoeging of verwijdering te kijken of de boom gebalanceerd moet worden. De boom moet gebalanceerd worden wanneer een knoop in de boom geen keys (woorden) meer bevat of wanneer een knoop meer dan drie keys bevat.

Balanceren Omdat bij deze analyse alleen gekeken wordt naar het toevoegen van keys zal de boom alleen gebalanceerd worden wanneer een knoop meer dan drie keys bevat. Wanneer een knoop meer dan drie keys bevat (overflow), zal de boom gebalanceerd moeten worden. Dit wordt gedaan door de derde waarde van de knoop met meer dan drie keys toe te voegen aan zijn parent. De eerste twee keys worden dan het kind aan de linkerkant van toegevoegde key in de parent en de vierde waarde het kind aan de rechterkant.

Zoeken Bij het searchen van een key zal begonnen worden bij de root knoop. Als de knoop niet de gezochte key bevat zal gekeken worden in het kind voor de key die lager is dan de invoer. Dit wordt gedaan tot een externe knoop is bereikt of tot een knoop is bereikt die de invoer bevat.

Toevoegen Bij het toevoegen zal eerst gezocht worden op de bovenstaande manier. Wanneer een knoop wordt gevonden die de key al bevat zal gestopt worden met toevoegen. Als dit niet het geval is, we hebben dan dus een externe knoop, zal de waarde toegevoegd worden aan de parent. Hierna zal de balance methode aangeroepen worden om te kijken of de boom ongebalanceerd is en balanceren wanneer dit het geval is.

Analyse

Chapter 6

Skip List

Intro Naast bomen zijn er ook andere manieren om zoekalgoritmen te implementeren. In dit hoofdsuk zal een van deze structuren behandeld worden, namelijk de skip list. Een skip list bestaat uit een aantal lijsten waarvan elke lijst een subset van de lijst onder hem bevat. De onderste lijst bevat alles waardes van de boom. Elke lijst bevat in ieder geval de waardes -∞ en ∞. Hierdoor zullen alle waardes in de lijst tussen deze twee waardes in zitten.

Zoeken Bij het zoeken naar een waarde k in een skiplist wordt begonnen in de meest linkse waarde in de bovenste lijst, deze waarde noemen we p. Hierna zal gekeken worden in de waarde onder p, als deze niet leeg is wordt dit de nieuwe waarde van p. Vervolgens wordt gekeken naar de waardes aan de rechterkant van p. Er wordt naar rechts gezocht tot er een waarde gevonden wordt hoger dan k, p wordt nu de waarde voor de laatst gevonde waarde. Dit wordt gedaan tot de gezochte waarde wordt gevonden. Wanneer k niet in de skip list zit wordt de hoogst mogelijke waarde voor k gereturned.

Toevoegen Bij het toevoegen van waarde k wordt eerst gezocht naar k op de manier zoals hierboven beschreven is. Als de gevonden waarde p gelijk is aan k zal direct gestopt worden, de waarde zit namelijk al in de skip list. Wanneer deze niet gelijk is zal de waarde toegevoegd worden na p. Hierna zal k toegevoegd worden zolang een random getal tussen de 0 en 1 lager is dan 0.5.

Verwijderen Bij het verwijderen van een waarde k zal eerst gezocht worden op deze waarde. Als de gevonden waarde p niet gelijk is aan k zal direct gestopt worden, k zit niet in de skip list. Wanneer k en p wel gelijk zijn zal p verwijderd worden samen met alle waardes boven p.

Analyse

Chapter 7

Conclusie

List of Figures