

Algoritmen en Datastructuren

Lucien Sina

9.12.2024

Inhoudsopgave

1	Inleiding	5
2	Algoritmen en hun Analyse	7
2.1	Abstractieniveaus van Probleembeschrijvingen	7
2.1.1	Voorbeeld: Lidmaatschapstest in een Verzameling	7
2.2	Beoordeling van de Kwaliteit van een Algoritme	9
2.2.1	Correctheid	9
2.2.2	Begrijpelijkheid	9
2.2.3	Eenvoud van Implementatie	9
2.2.4	Efficiëntie (Uitvoeringstijd en Geheugengebruik)	9
2.2.5	Afwegingen tussen Criteria	9
2.3	Uitvoeringstijd en Geheugengebruik van Algoritmen	10
2.3.1	Metten van de Uitvoeringstijd	10
2.4	Constructie van een Looptijdfunctie	11
2.4.1	Inleiding	11
2.4.2	Abstractie van Uitvoeringstijd	12
2.4.3	Case Study: Zoeken in een Array	13
2.4.4	Gebruik van O-Notatie	13
2.4.5	Oefeningen en Oplossingen	14
2.4.6	Samenvatting	15
2.5	Analyse van Controlestructuren	15
2.5.1	Elementaire Operaties en Sequenties	15
2.5.2	Looptijden van Basisbesturingselementen	16
2.6	Het Verbeteren van een Algoritme	17
2.6.1	Tijdcomplexiteitsanalyse	18
2.6.2	Verbeterd Algoritme met Vroege Terminatie	18
2.6.3	Tijdcomplexiteitsanalyse	18
2.6.4	Geoptimaliseerd Algoritme met Binaire Zoekopdracht	19
2.6.5	Tijdcomplexiteitsanalyse	19
2.6.6	Demonstratie	20
2.7	Algemene O-notatie	20
2.7.1	Definities	20
2.7.2	Relaties Tussen Functies	20
2.7.3	Voorbeeld: Vergelijken van $\log n$ en $\sqrt{n}$	22

2.7.4	Voorbeeld: Runtime van <code>contains</code> en <code>contains₂</code>	22
3	Algebra's en abstracte datatypen	25
3.1	Contexten	25
3.2	Abstractieniveaus	25
3.2.1	Algebra als Datatype	25
3.2.2	Formaliseren van Datatypen en Algebra's	26
3.3	Algebra's en Datatypen	26
3.3.1	Voorbeeld Handtekening	27
3.3.2	Twee Benaderingen voor Specificatie	27
3.4	Oefening: Algebra voor een Teller	30
3.5	Polymorfe en Monomorfe Datatypes	31
4	Basiskoncepten	33
4.1	Algoritme	33
4.1.1	Belangrijke Concepten	34
4.2	Definities van Algebra, ADT, en Model	35
4.3	Oefeningen en Oplossingen	36
4.3.1	Oefening 1.1: Algoritme	36
4.3.2	Oefening 1.2: Signatuur, Algebra, en ADT	36
4.3.3	Oefening 1.3: Datatypes en Gegevensstructuren	37
4.3.4	Oefening 1.4: Monomorfe en Polymorfe ADT's	37
4.3.5	Oefening 1.5: Implementatie van Gegevensstructuren	38
5	Programmeren en Gegevensstructuren	39
5.1	Constructie van Gegevensstructuren	39
5.2	Programmeertalen	40
5.3	Gegevenstypes in Java	40
5.3.1	Primitieve Gegevenstypes	40
5.4	Primitieve Gegevenstypes in Java	41
5.5	Arrays	42
5.5.1	Voorbeeld	42
5.5.2	Waardebereik van een Array Type	43
5.5.3	Adresberekening	43
5.5.4	Efficiëntie van Array Toegang	43
5.5.5	Arrays als Representatietools	43
5.5.6	Voorbeeld: Itereren door een Array	43
5.5.7	Specifieke Array Bereiken	43
5.5.8	Java-Specifiek Gedrag	44
5.6	Klassen	44
5.6.1	Structuur van een Klasse-definitie	44
5.6.2	Methoden in Klassen	45
5.6.3	Realiseren van Algebra of Abstracte Gegevenstypes	45
5.7	Records	45
5.7.1	Definitie van Records	46
5.7.2	Voorbeelden van Record Definities	46

5.7.3	Operaties op Records	47
5.7.4	Waardebereik van een Recordtype	47
5.7.5	Representatie en Adresberekening	47
5.7.6	Toegang in Constante Tijd	47
6	Dynamische Gegevensstructuren	49
6.1	Pointertypes	49
6.1.1	Definitie van Pointertypes	49
6.1.2	Declaratie en Gebruik van Pointervariabelen	50
6.1.3	Grafische Weergave van Pointers	50
6.1.4	Belang van Pointers in Dynamische Gegevensstructuren	50
6.2	Dereferencing en Garbage Collection	51
6.3	Referentietypen in Java	53
6.3.1	Toewijzingen	53
6.4	Methode-aanroepen in Java	54
7	Aanvullende datastructuren	57
7.1	Enumeratie Types	57
7.2	Subbereik Types	59
7.3	Verzamelingen	60
8	Elementaire Datatypes	63
8.1	Lijsten	63
8.1.1	Lijsten met Eerste, Rest, VoegToe en Concateneer	63
8.2	Lijsten met Expliciete Posities	64
9	Lijstimplementaties	69
9.1	Dubbel-Gelinkte Lijst	69
9.2	Implementatie van Dubbel-Gelinkte Lijsten	70
9.2.1	Een Lege Lijst Maken	70
9.2.2	Toegang tot de Eerste Positie	71
9.2.3	Invoegen van Elementen	71
9.2.4	Lijsten Samenvoegen	72
9.2.5	Elementen Zoeken	73
9.2.6	Elementen Verwijderen	73
9.3	Eenvoudige Gelinkte Lijst	73
9.4	Lijsten in Arrays	75
10	Stapel	79
10.1	Specificatie van een Stack	80
10.2	Stack Implementatie	80
11	Queues	83
11.1	Queue-operaties en uitbreidingen	83
11.2	Algebraïsche specificatie van queues	83
11.3	Queue-handtekening	84
11.3.1	Diagramvoorstelling	84

11.4 Toepassingen van Queues	84
11.5 Implementatie van Queues in Java	84
11.5.1 Implementatie van een Queue met een Cyclische Array	85
11.5.2 Implementatie van een Queue met Twee Stacks	86
11.5.3 Vergelijking van de Twee Implementaties	87
12 Mappings	89
12.1 Kenmerken van Mappings	89
12.2 Mappings met Eindig en Scalaire Domeinen	89
12.3 Grote of Spaarzame Domeinen	90
12.4 Samenvatting van Mapping-Implementaties	91
13 Binaire Bomen	93
13.1 Definitie	93
13.1.1 Binaire Bomen	94
13.1.2 Bladeren en Deelbomen	94
13.1.3 Knopen en Hun Classificatie	95
13.1.4 Paden, Voorouders en Afstammelingen	95
13.1.5 Subtrees en Padlengtes	96
13.1.6 Hoogte en Diepte	96
13.2 Kenmerken van Binaire Bomen	97
13.2.1 Maximale Hoogte van een Binaire Boom	97
13.2.2 Minimale Hoogte van een Binaire Boom	98
13.2.3 Exacte Minimale Hoogte van een Binaire Boom	98
13.2.4 Relatie Tussen Bladeren en Interne Knoten	99
13.2.5 Samenvatting van Kenmerken	100
13.3 Boomalgebra	100
13.3.1 Soorten	100
13.3.2 Operaties	100
13.3.3 Setdefinitie	100
13.3.4 Functiedefinities	100
13.3.5 Recursieve Structuren	101
13.3.6 Boomdoorlopen	101
13.3.7 Oefening: Berekening van de Hoogte van een Boom	102
13.4 Binaire Boom Java Implementaties	102
13.4.1 Met Pointers	102
13.4.2 Met Array Inbedding	103
13.4.3 Complexiteit Vergelijking	105
14 Algemene Bomen	107
14.1 Definitie	107
14.1.1 Illustratie	107
14.2 Implementaties van Algemene Bomen	109
14.2.1 Implementatie met Arrays	109
14.2.2 Implementatie met Binaire Bomen	109

15 Oefeningen	111
16 Eindwoorden	123
16.1 Aanbevolen Boeken door Lucien Sina	123
16.1.1 Logica	123
16.1.2 Programmeren	124
16.2 Literatuur	124

© 2025 door Lucien Sina. Alle rechten voorbehouden.

ISBN: 9789403831749

Gedrukt door Bookmundo

Voorwoord

Efficiënte algoritmen en datastructuren vormen het hart van de informatica en zijn de basis voor het oplossen van complexe problemen binnen en buiten de programmeerwereld. Het vermogen om algoritmen te ontwerpen, analyseren en implementeren is essentieel om de uiteenlopende uitdagingen aan te pakken waarmee programmeurs worden geconfronteerd. Dit boek heeft als doel lezers de nodige tools en inzichten aan te reiken om zich in dit boeiende vakgebied te kunnen bewegen.

Om de kernproblemen binnen de informatica aan te pakken, moeten programmeurs leren om nieuwe algoritmen te creëren door bestaande op een vaardige manier te combineren. Even belangrijk is het vermogen om deze algoritmen te analyseren op het gebied van uitvoeringstijd en geheugengebruik. Datastructuren, die informatie organiseren om efficiënte algoritmische bewerkingen mogelijk te maken, vormen een ander cruciaal aandachtspunt van dit boek.

Er zal een duidelijk onderscheid worden gemaakt tussen **datatypen**—de abstracte, specificatiegerichte kijk op het organiseren van gegevens—en **datastructuren**, die de concrete implementaties van deze datatypen vertegenwoordigen. Lezers zullen ontdekken dat één datatype meerdere implementaties kan hebben, elk met zijn eigen voor- en nadelen. Door deze verschillen te verkennen, helpt dit boek bij het ontwikkelen van een diepgaand begrip van hoe men efficiënte oplossingen voor uiteenlopende situaties kan kiezen en ontwerpen.

Om optimaal van dit boek te profiteren, is een basisvaardigheid in programmeren aan te bevelen. Bekendheid met een taal als Java zal bijzonder nuttig zijn. Voor lezers die nog basiskennis van programmeren nodig hebben, verwijs ik graag naar mijn eerdere boek over programmeren, waarin de essentiële concepten voor effectief coderen worden geïntroduceerd. Hoewel enige wiskundige vaardigheid de begripsvorming kan bevorderen, heb ik er bewust voor gekozen om alle benodigde wiskundige concepten in dit boek te behandelen.

Om het leren te versterken, bevat dit boek talrijke oefeningen, compleet met oplossingen. Deze oefeningen zijn bedoeld om de theoretische concepten te verankeren en praktische ervaring op te doen met implementaties.

Of je nu student bent, programmeur, of gewoon nieuwsgierig naar algoritmen en datastructuren, dit boek is zo opgebouwd dat het je stap voor stap door de materie leidt. Ik hoop dat het een waardevolle bron zal zijn tijdens jouw ontdekkingsreis binnen dit essentiële onderdeel van de informatica.

Hoofdstuk 1

Inleiding

Algoritmen werken op datastructuren, en datastructuren bevatten algoritmen als componenten. Daardoor zijn algoritmen en datastructuren onlosmakelijk met elkaar verbonden. In dit boek streven we ernaar om algoritmen en datastructuren te classificeren binnen de context van verwante concepten zoals functies, procedures, abstracte datatypen, datatypen, algebra's, typen, klassen en modules.

Op het meest abstracte niveau gebruiken we de wiskunde om de specificaties van algoritmen en datastructuren te formaliseren. Een algoritme realiseert een functie, die als specificatie voor dat algoritme dient. Evenzo is een algoritme zelf een specificatie van een procedure, functie of methode die uiteindelijk geïmplementeerd moet worden.

Het is belangrijk te beseffen dat algoritmen doorgaans niet als computerprogramma's worden gepresenteerd. In plaats daarvan worden ze beschreven op een hoger, menselijk leesbaar niveau om de communicatie te vergemakkelijken. Een computerprogramma *kan* echter wel dienen als een manier om een algoritme te beschrijven. Met andere woorden: een computerprogramma **is** een algoritme, maar de beschrijving van een algoritme is meestal niet beperkt tot een computerprogramma.

In het eerste hoofdstuk richten we ons op de analyse van algoritmen, waarbij we specifiek hun uitvoeringstijd en geheugengebruik onder de loep nemen.

Aan de kant van datastructuren beginnen we met abstracte concepten zoals **abstracte datatypen** en **algebra's**, die concrete datatypen definiëren. Een datastructuur is een implementatie van een algebra of een abstract datatype op algoritmisch niveau. Datastructuren kunnen op hun beurt worden geïmplementeerd in programmeertalen. Op programmeerniveau komen we belangrijke concepten tegen zoals datatypen, klassen en modules.

Deze gestructureerde benadering stelt ons in staat om de kloof te overbruggen tussen hoogabstracte concepten en hun concrete implementaties in programmeercode, wat leidt tot een omvattend begrip van algoritmen en datastructuren.

