

Cuvânt Înainte

Prezenta lucrare cuprinde prelegerile susținute de autor în cadrul cursului de “Structuri de Date și Algoritmi”, în fața anului IAA al Facultății de Automatică și Calculatoare a Universității POLITEHNICA din București. Lucrarea prezintă studenților câteva metode de dezvoltare a aplicațiilor de mari dimensiuni cum ar fi: dezvoltarea modulară a aplicațiilor, programarea generică, mecanisme de abstractizare procedurală sau de abstractizare a datelor. Sunt puse în evidență și mecanisme puse la dispoziția programatorului de limbajul C++ în scopul elaborării unor aplicații robuste, generale de mari dimensiuni.

Lucrarea prezintă principalele tipuri de date abstracte lista, stiva, coada, arborele, graful precum și unele metode generale de rezolvare a problemelor cum ar fi: DIVIDE ET IMPERA, backtracking sau programarea dinamică. În elaborarea algoritmilor de manipulare a acestor structuri de date trebuie avută în vedere stabilirea unor justificări asupra corectitudinii acestor algoritmi precum și analiza eficienței fiecărui astfel de algoritm. Unele aspecte ale domeniului de studiu al structurilor de date și algoritmilor, cum ar fi: studiul corectitudinii algoritmilor, demonstrarea riguroasă a corectitudinii bazată pe tehnici de semantică axiomatică au fost tratate de savanți iluștri ca Floyd, Hoare, Dijkstra sau Gries. Această direcție de studiu ar putea fi aprofundată în continuare de cititor.

Exemplele prezentate în cadrul fiecărui capitol au grade de dificultate diferite de la exemple simple, clasice și până la probleme propuse la concursurile de programare; sunt prezentate și câteva probleme pe care cititorul este invitat să încerce să le rezolve singur. O aceeași aplicație poate fi rezolvată în moduri diferite, uneori diferențele între cantitatea de resurse (spațiu de memorie, timp de execuție) necesare diverselor soluții sunt uriașe. Toate aceste aspecte trebuie însușite în profunzime de studenții de astăzi pentru că numai astfel ei vor putea deveni specialiștii competenți, profesioniștii de mâine.

Capitolul 1 descrie principiile care stau la baza dezvoltării aplicațiilor de mari dimensiuni. Este prezentat modul de a descompune în module o aplicație C precum și o serie de mecanisme de abstractizare a datelor specifice limbajului C++.

Capitolul 2 se referă la analiza algoritmilor. Sunt descrise modele de calcul, notații referitoare la analiza comportării asimptotice a algoritmilor, analiza implementării tipurilor abstracte de date, analiza amortizată și sunt date exemple care să ilustreze diverse procedee de analiză algoritmică.

În capitolul 3 sunt prezentate operațiile caracteristice tipului de date abstracte LISTA precum și diverse implementări ale acestui TDA: implementări bazate pe diverse structuri de memorare, implementări iterative sau recursive. Se descriu modalități de implementare a listelor generice atât în limbajul C cât și în limbajul C++ folosind șabloane.

Capitolele 4 și 5 prezintă două metode generale de rezolvare a problemelor: metoda *Divide et Impera* și metoda *căutării cu revenire în spațiul soluțiilor (backtracking)*.

Exemplele semnificative sunt însoțite de analiza detaliată a algoritmilor și sunt prezentate în ordinea crescătoare a nivelului de dificultate.

În capitolul 6 sunt descrise tipurile de date stivă și coadă împreună cu două aplicații semnificative care se referă la evaluarea expresiilor aritmetice și respectiv la simularea funcționării sistemelor cu evenimente discrete.

O structură de date utilizată frecvent într-o gamă largă de aplicații (datorită faptului că permite căutarea foarte eficientă în masive de date) este tabela de dispersie; ea este descrisă în capitolul 7.

Capitolul 8 este (alături de capitolul referitor la grafuri) unul din cele mai extinse din prezenta lucrare. El se referă la arbori, structuri de date ce permit elaborarea unor algoritmi deosebit de eficienți și utilizați în cele mai diferite situații. Sunt prezentați arborii binari de căutare, arborii AVL, arborii de căutare în spații multidimensionale și arborii parțial ordonați.

Capitolul 9 referitor la grafuri înfățișează principalele modalități de reprezentare precum și câteva categorii de aplicații relevante cum sunt: sortarea topologică, determinarea punctelor de articulație, metoda lui Kuhn pentru determinarea unei împerecheri optimale într-un graf bipartit. Soluția lui Floyd (bazată pe metoda programării dinamice) la problema determinării drumurilor minime între oricare două vârfuri ale unui graf face trecerea către metoda programării dinamice prezentată în capitolul 10.

Parcursul lucrării necesită cunoștințe preliminare de programare în limbajul C. Lucrarea poate fi folosită de toți cei interesați (elevi, studenți, specialiști) de îmbunătățirea calității programelor pe care le elaborează, de înțelegerea profundă a principalelor structuri de date și tehnici de programare ce pot fi folosite în dezvoltarea aplicațiilor complexe. Autorul recomandă cititorilor și în special studenților – valoroși – ai Facultății de Automatică și Calculatoare să nu considere această lucrare decât un început și să folosească bibliografia prezentată, pentru aprofundarea cunoștințelor în acest vast și fundamental domeniu de studiu.

Marius Dorian ZAHARIA
marius.zaharia@cs.pub.ro

CUPRINS

Cuvânt Înainte Foreword		iv
Capitolul 1.	Dezvoltarea modulară a aplicațiilor software complexe Modular development of complex software applications	1
1.1	Dezvoltarea de aplicații mari (proiecte) în limbajul C (Large projects development using the C programming language)	2
1.2	Abstractizarea datelor. Mecanisme de abstractizare (încapsulare, descompunere în module) puse la dispoziție de limbajul C++. Data abstraction. Mechanisms for abstract data specification in C++ programming language.	6
1.2.1	O aplicație – Trasarea curbelor autoasemenea folosind grafica turtle. An application – Drawing self-similar curves using turtle graphics	6
1.2.2	O aplicație – simularea funcționării unui CD player/recorder. An application – Simulation of a CD player/recorder	14
Capitolul 2.	Elemente de teoria analizei algoritmilor. Algorithms Analysis.	19
2.1	Notății folosite în analiza asimptotică a algoritmilor Some basic notions used in asymptotic analysis of algorithms	21
2.2	Exemple de analiză a unor algoritmi.Examples of algorithm analysis	22
2.3	Analiza implementării unui tip de date abstracte. Analiza amortizată. Analysis of the implementation of an abstract data type. Amortized analysis.	24
2.4	Complexitatea problemelor de calcul. Computational complexity.	25
2.4.1	Determinarea unei margini inferioare a timpului de execuție a unei probleme. Methods to find lower bounds of the running time of an algorithm.	26
Capitolul 3.	Tipul de date abstracte listă. The Abstract Data Type (ADT) List.	29
3.1	Operații caracteristice. Abstract operations.	29
3.2	Implementare cu variabile dinamice și legături implicite. Implementation using implicit references and dynamic allocation.	30

3.3	Implementare cu tablouri alocate static și legături explicite Implementations using explicit references and static arrays.	31
3.4	Implementări ce folosesc pointeri și variabile dinamice. Implementations using pointers and dynamic allocation.	33
3.5	Implementări ce folosesc funcții recursive. Implementations using recursive functions	36
3.6	Genericitate. Implementarea listelor generice. Implementations of generic lists in C.	36
3.7	Implementarea de aplicații generice în limbajul C++ Implementing generic applications in C++	40
3.7.1	Realizarea aplicațiilor generice folosind șabloane (templeți) Generic C++ applications using templates.	44
3.8	O aplicație – Tipul abstract de date multime. An application - the abstract data type “set”.	51
3.9	Clase container în C++. Container classes in C++.	52
Capitolul 4.	Metoda DIVIDE ET IMPERA. Divide and Conquer	55
4.1	Caracteristicile metodei. Main characteristics of this algorithm design method.	55
4.2	Primele exemple. First examples.	55
4.2.1	Problema turnurilor din Hanoi Towers of Hanoi.	55
4.7.2	Determinarea elementului maxim dintr-un vector. Maximum element of a vector.	57
4.7.3	Metoda de sortare Quicksort. Quicksort.	58
4.3	Analiza pe caz general a problemelor soluționabile prin metoda DIVIDE ET IMPERA General case analysis of Divide and conquer solutions	61
4.4	Alte exemple . Other examples.	54
4.4.1	Sortarea prin interclasare Mergesort.	64
4.4.2	Determinarea înfășurătorii convexe a unui set de puncte Convex Hull	65
Capitolul 5.	Metoda căutării cu revenire în spațiul soluțiilor (backtracking)	71
5.1	Schema generală de rezolvare backtracking. General scheme of backtracking solution	71
5.2	Un exemplu – problema celor n regine. The n queens problem	72
5.2.1	Soluția problemei celor n regine prin metoda forței brute. Brute force approach to solve the n queens problem	74
5.3	Analiza algoritmilor backtracking Analysis of backtracking based solutions	75
5.4	Problema rucsacului în spațiul discret The Knapsack problem.	76

5.4.1	Soluție cu tuple de lungime fixă Knapsack problem solution with fixed size tuples	77
5.4.2	Soluție cu tuple de lungime variabilă. Knapsack problem solution with variable size tuples.	80
5.5	O aplicație – Războiul stelelor. An application – Star trek “romulan battle”.	81
Capitolul 6.	Tipurile de date abstracte stiva și coada The abstract data types stack and queue	85
6.1	Stiva – Operații caracteristice. Diverse implementări The Stack ADT abstract operations. Different implementation methods.	85
6.2	O aplicație – Evaluarea expresiilor aritmetice. An application – expression evaluation.	87
6.3	Coada – Operații caracteristice. Diverse implementări The Queue ADT - abstract operations. Different implementation types.	92
6.4	Utilizarea stivei și cozii în aplicații generice C++. Biblioteca STL. Using Stack and Queue ADTs in generic applications (mplemented with STL)	95
6.5	O aplicație – Simularea funcționării sistemelor cu evenimente discrete . An application - Discrete events simulation.	97
Capitolul 7.	Tabele de dispersie Hash tables	103
7.1	Definiții Definitions	103
7.2	Metode de rezolvare a coliziunilor. Solving colisions.	103
7.3	Tabele asociative în limbajul C++. Implementing associative arrays in C++.	105
Capitolul 8.	Arbori Trees	109
8.1	Terminologie caracteristică Definitions.	109
8.2	Modalități de reprezentare a arborilor. Different data structures used for tree representation	110
8.2.1	Reprezentarea unui arbore binar cu celule alocate dynamic. Binary trees represented with dynamically allocated cells	110
8.2.2	Reprezentarea cu tablouri Arrays based tree representations.	111
8.2.3	Reprezentarea cu legături fiu-tată. Representations that use references son-parent.	112
8.2.4	Reprezentarea cu liste List based representations	113
8.2.5	Reprezentarea “fiu stâng-frate drept” “Left son”- “Right sibling” representation	114
8.2.6	Arbori binari înșiruiți Threaded binary trees	114
8.3	Parcurgerea arborilor Tree traversal	115
8.4	Generarea și afișarea unui arbore binar Building and printing a binary tree	117

8.5	Arbori binari de căutare. Binary search trees .	119
8.6	Arbori AVL AVL trees	125
8.7	Generalizări ale arborilor binari de căutare. Căutări în spații multidimensionale. Efficient search in multidimensional dataspace s. Generalizations of BSTs .	130
8.7.1	Arborele de domenii Domain tree	130
8.7.2	Arborele de segmente Interval tree	131
8.8	Arbori parțial ordonați. Heap ordered trees	135
8.8.1	Heap-uri binare Binary heaps	135
8.8.2	Operații cu arbori parțial ordonați. Abstract operations on heaps	136
8.8.3	Analiza operațiilor de manipulare a arborilor parțial ordonați Analysis of heap operations .	139
Capitolul 9.	Grafuri Graphs	140
9.1	Terminologie caracteristică Definitions	140
9.2	Graful ca tip abstract de date The graph seen as an abstract data type .	141
9.3	Modalități de reprezentare a grafurilor Graphs representations	141
9.3.1	Reprezentarea prin matrice de adiacență Adjacency matrix	141
9.3.2	Reprezentarea prin liste de adiacență Representation based on adjacency lists	142
9.3.3	Reprezentarea secvențială Sequential representation	143
9.4	Modalități de parcurgere a grafurilor Graph traversal methods	144
9.4.1	Parcurgerea în adâncime Depth-first traversal	144
9.4.2	Parcurgerea în lățime Breadth-first traversal	145
9.5	Sortarea topologică Topological sort	146
9.6	Arbori caracteristici ai unui graf Trees associated to a graph	149
9.6.1	Algoritmul lui Prim Prim's algorithm (MST minimum cost spanning tree)	149
9.6.2	Algoritmul lui Kruskal Kruskal's algorithm (MST)	156
9.6.2.1	Tipul de date abstracte colecție de mulțimi disjuncte The abstract data type "disjoint sets"	157
9.7	Puncte de articulație într-un graf neorientat. Finding articulation points in an undirected graph	159
9.8	Împerecheri maximale într-un graf bipartit neetichetat Maximal matchings in a bipartite graph .	162
9.9	Determinarea drumurilor optime într-un graf. Optimal path algorithms	168
9.9.1	Drumul minim într-un graf neorientat Shortest path in an undirected graph	169
9.9.2	Algoritmul lui Dijkstra Dijkstra's shortest path algorithm	170
9.9.3	Algoritmul Bellman-Kalaba de determinare a unui	174

	drum optim Optimal path with Bellman's algorithm	
9.9.3.1	Studiu de caz. Problema “pereii mătăietee”. Case study “The mellow pear problem”	175
9.9.4	Determinarea drumurilor minime pentru orice pereche de vârfuri (sursă, destinație). Algoritmul Floyd-Warshall All pairs shortest path algorithm (Floyd-Warshall)	178
Capitolul 10.	Metoda programării dinamice Dynamic programming.	180
10.1	Descrierea metodei Method description	180
10.2	Distanța între două șiruri de caractere Distance measure between two strings	180
10.3	Problema comis-voiajorului. Euristici de rezolvare bazate pe metoda programării dinamice The traveling salesman problem. Some heuristics based on dynamic programming .	183
Capitolul 11.	Probleme propuse Problems to solve	187
Bibliografie	Bibliography	192