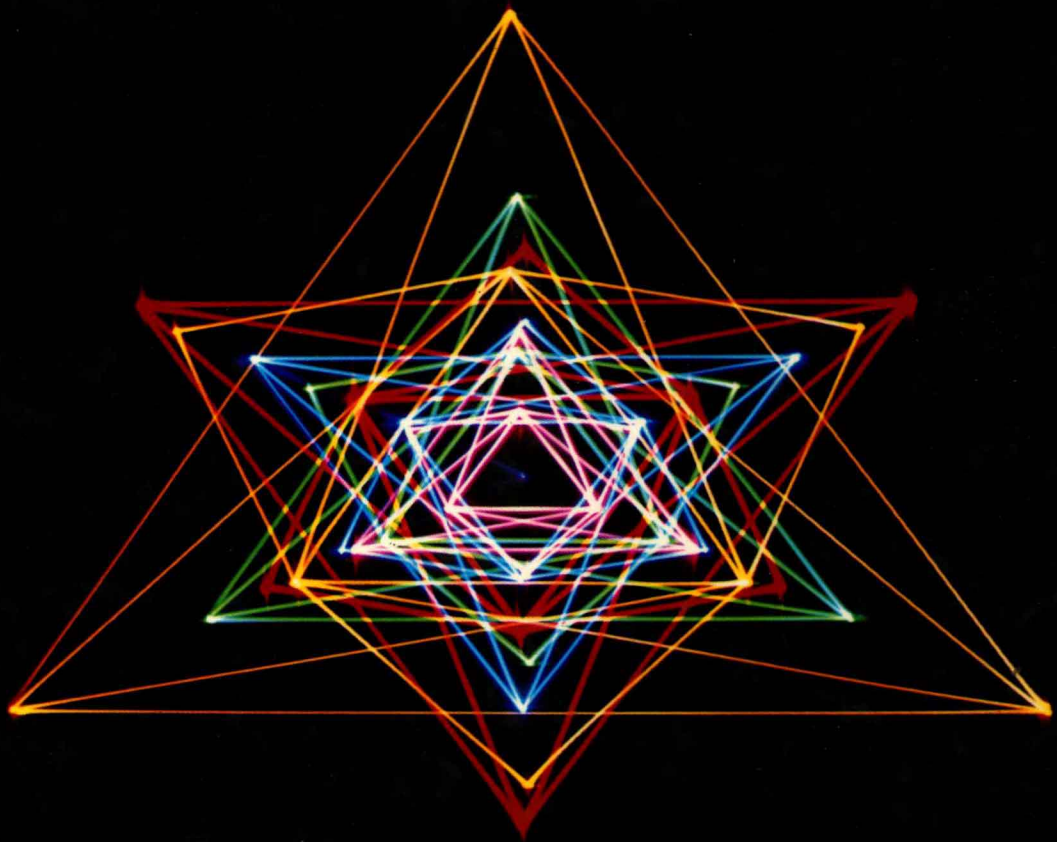


Introduction to Data Structures with Pascal



Thomas L. Naps
Bhagat Singh

Introduction to Data Structures with Pascal

Thomas L. Naps

Lawrence University

Bhagat Singh

University of Wisconsin Center System

WEST PUBLISHING COMPANY

St. Paul New York San Francisco Los Angeles

Copyediting: Martha S. Knutson
Design: Lucy Lesiak
Typesetting: Rolin Graphics
Cover Design: Paul Kemp
Cover Photograph: Floyd Rollefstad, Laser Fantasy Productions

COPYRIGHT ©1986 By WEST PUBLISHING COMPANY
50 West Kellogg Boulevard
P.O. Box 64526
St. Paul, MN 55164-1003

All rights reserved
Printed in the United States of America

Library of Congress Cataloging-In-Publication Data

Naps, Thomas L.
Introduction to data structures with PASCAL.

Includes index.

1. PASCAL (Computer program language) 2. Data structures (Computer science) I. Singh, Bhagat, 1940- II. Title.

QA76.73.P2N37 1986 005.7'3 85-22582
ISBN 0-314-93207-0

Introduction to Data Structures with Pascal

To
Joyce and Rosemary

Preface

With the rapid evolution of computer science, the study of data structures has found its way into the undergraduate curriculum as early as the sophomore year. Moreover, an increasingly diverse collection of students is studying this subject. Once solely the domain of hard-core computer science majors, data structures is now taken by business, economics, engineering, and mathematics majors who wish to enhance their chosen discipline with a strong background in computer science. Texts that emphasize a highly rigorous mathematical approach to data structures no longer are appropriate for this growing number of heterogeneous students. In *Introduction to Data Structures with Pascal*, we attempt to satisfy the needs of this new group of students by providing a text that emphasizes implementing and evaluating data structures in practical situations and avoids relying on an overly theoretical approach. The text is designed for anyone who has had a solid programming background in Pascal. Such a background should include a detailed grasp of procedures, functions, parameter passing, and arrays. To a lesser extent, some prior work with records and files is desirable although these topics are developed as needed in the text itself.

Introduction to Data Structures with Pascal represents an alternative to our earlier pseudocode-based *Introduction to Data Structures*. The development of such an alternative was motivated by a near 50-50 split which we found among the reviewers of our efforts. Though these reviewers were essentially in agreement as to topics covered and the order in which they were presented, there was a pronounced and irresolvable split between those who believed in a pseudocode approach and those who felt that seeing algorithms in the familiar Pascal language would make concepts easier for students. It is to this latter group that this version of the text is addressed.

In addition to providing a Pascal-specific orientation, we have also expanded our coverage of important software engineering concepts to help students as they begin using data structures in larger programming projects. Sections on Program Design Considerations appear in each chapter. Some of these sections (notably those in Chapter 3, Chapter 7, and Chapter 11) deal with the development of complete large-scale programs which use the data structures studied up to that point. Others touch upon design issues which are then further expanded in the exercises and programming problems.

Three new appendices have also been added for those students of Pascal who may need a bit more general experience as well as a thorough treatment of data structures. Appendix C presents a description of those testing, verification, and debugging strategies that are so essential for students developing large-scale programs. Appendix B on formal analysis of algorithms presents the basics of this topic. We have chosen to treat this issue in a separate appendix because we want the book to retain its intuitive, implementation-oriented approach. Appendix A discusses how various versions of Pascal implement direct access files—a topic which we consider essential despite its absence from the Pascal standard. A “real world” proof of the essential nature of this topic is that virtually all widely used implementations of Pascal have included procedures for direct access files in their repertoires. Appropriate references to all of these appendices are made throughout the text, making it easy for the student to merge this additional material with specific data structures topics. Depending upon the curriculum at your institution, this increased emphasis upon software engineering issues and the coverage of advanced topics in a Pascal-specific fashion may in fact make the text suitable for an advanced course in programming, as described in the C.S. II Course Guidelines recommended by the recent ACM Curriculum Task Force for C.S. II.

Further changes from the pseudocode version are less oriented toward Pascal but instead induced by feedback we have been receiving from the field. Exercises have been added in each chapter. The thematic Wing-and-a-Prayer Airlines problems proved popular, so we have included two other thematic problem lines running throughout the text—the Fly-By-Night Credit Company and the Bay Area Brawlers Professional Football Club. In Chapter 5, we have expanded our discussion of recursion, offering a more gradual introduction to this difficult topic. Discussions of algorithm efficiency in the body of the text itself have been isolated in separate sections for greater emphasis and easier reference. Finally, new “In the World of Applications” boxes have been developed for Chapters 4, 7, and 9 to replace old ones which some readers felt lacked a “real world” flavor.

However, despite these changes, the two major goals of this text remain unchanged from the pseudocode version:

1. The student must acquire an understanding of the algorithms that manipulate various data structures.
2. The student must learn to select from among data structures for a given application.

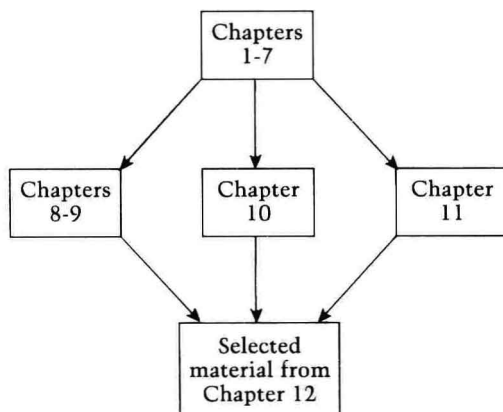
Relative to this second goal, data structures could be considered the toolbox of the computer scientist or data processor. Many jobs can be accomplished in more than one way; the student who understands data structures thoroughly is able to match the tool to the job most effectively.

Understanding of data structures is enhanced by pictures. For this reason, we have made ours a graphically oriented book. Throughout the text, algorithms written in Pascal are enhanced by "graphic documentation" to clarify the responsibilities of various segments of code. Our experience as instructors of data structures has so often indicated that a student's question about how to do something can be answered by the simple dictum to "draw a picture." Data structures are, after all, the programmer's way of implementing certain types of mental images within the computer. Every segment of a program is in some way responsible for maintaining or altering that structural image. Can there be a better way to document that segment than to picture what it is intended to do? In a certain sense, this graphic orientation embodies the essence of data abstraction—that the student understand the data structure itself rather than memorize a particular algorithm or the details of a particular implementation.

The order of topics presented in the book is relatively flexible, although the first seven chapters represent a core of fundamental material that should be covered before moving on to any of the topics in Chapters 8 through 12. Chapter 1 lays the groundwork for the rest of the book by specifying a rationale for data structures and developing those advanced features of Pascal (including records and pointer variables) that will be used throughout. The first data structure discussed in detail is the linked list, presented in Chapter 2. Initial implementation of linked lists in this chapter is achieved through the use of arrays instead of Pascal's `NEW` and `DISPOSE` procedures to allow the student to actually trace through the implementation. In Chapter 3, we consider strings in detail as an example of an application in which pointers, indices, and linked lists can be used effectively. Chapter 4 presents two special types of lists: queues and stacks, and Chapter 5 follows up on this theme by looking at two critically important applications of stacks: the parsing of arithmetic expressions and recursion. Chapters 6 and 7 cover in detail what we consider to be the most versatile of all data structures—the tree. We discuss both binary and general trees, along with such variations as threading and height-balancing.

Although the first seven chapters of this book should be studied consecutively, there is considerable leeway in the order in which the remaining topics may be covered. Some instructors may find it possible to cover only selected topics from Chapters 8 through 12, and these chapters are connected loosely enough to allow picking and choosing. Chapter 8 fully describes multidimensional and sparse matrices, which serves as a convenient introduction to the adjacency matrix representation of graphs and networks, the topic of Chapter 9. Chapters 8 and 9 thus form a cohesive unit. However, it is possible to proceed directly from Chapter 7 to either Chapter 10 (Sorting) or Chapter 11 (Search Methods). Chapter 12, Data Structures and Data Management, presents a nice wrapup for a data structures course by describing how a variety of data structures can be applied in the areas of memory management and database management. Because not all instructors will find time in a semester to cover the material in Chapter 12, we have written it such that it lends itself to independent study. For those students going on into systems programming, the material in Chapter 12 on memory management algorithms used by operating systems should prove particularly valuable. For the more business-oriented student, the database management section is more suitable. The following flowchart summarizes the possible sequences in which chapters can be covered.

Additional features of the book include "In the World of Applications . . ." boxes, which emphasize how data structures are used in a variety of settings; chapter exercises and programming problems; a glossary; and an appendix of solutions to odd-numbered exercises. The Instructor's Manual provides useful hints on how to present the material in each chapter, solutions to even-numbered exercises, and additional ideas for programming problems and projects. Transparency masters that can be used to illustrate key concepts are available to adopters from the publisher. A list of figures from the text available as transparency masters is in the Instructor's Manual.



Acknowledgements

Although work on this text was greatly smoothed by the previous work we had done for the pseudocode version, a large number of people have nonetheless played key roles in its development. Of course, our students never cease to amaze us with the clever ideas and insights they continually offer. Jerry Westby, our editor, always had another (and another) innovative idea to try. Our work with production editor Peggy Adams and the staff at West Publishing proved to be a pleasure. Thanks go also to the large group of skilled reviewers who offered constructive criticism on our efforts:

John Rezac, Johnson County Community College
Andrew Bernat, University of Texas at El Paso
Michael Henry, West Virginia University
Charles Williams, Georgia State University
W. Bruce Croft, University of Massachusetts at Amherst
Robert Holloway, University of Wisconsin at Madison
Tamisra Sanyal, Monroe Community College
Martha Hansard, Georgia State University
Barent Johnson, University of Wisconsin at Oshkosh

Finally, our most sincere gratitude is directed to our wives and families. The warmth and freshness they provide keep our lives nicely unstructured.

Index

- Abstract syntax trees, 199
- Acyclic graph, 259, 260
- ADD, procedure, 87, 92
- ADDREC, procedure, 357, 358
- ADDTOTREE, procedure, 280
- Adelson-Velskii, G. M., 185, 199
- Adjacency matrix, 249
- Air Force space missions, 427
- ALCOR Pascal, 417
- Algorithms, 5–21, 418
- All, function, 280
- ALLOCATE, procedure, 370
- Analysis of the Bubble Sort, 424
- Ancestor, 144
- Arc, 247
- Array implementation of a stack, 98
- Arrays, 1, 2, 27–29, 83–91, 98–99
- Artificial intelligence, 4, 246
- Assignment, string, 54, 75
- Asymptotic Measurements, 421
- Available list, 368
- Average case, 421
- AVERAGE, procedure, 31
- AVL method, 186
- AVL rotation, 186
- AVL trees, 185
- AVLINSERT, procedure, 195

- B-tree index, 345
- B-tree indexing, 346–348
 - efficiency of, 348–350
- B-tree insertions, 349–350
- B-tree of order n , 346
- B-trees, 346
- Backtracking, 106, 132–133
- Backward arcs, 259
- Backward link, 43
- Balance factor, 185
- Band matrices, 227
- Basic, 53
- Basic-plus, 61
- Batch processing, 83
- Best case, 421
- Best fit strategy, 411
- Big O, 421
- Binary buddies, 370
- Binary buddy system, 373–377, 409
- Binary representation—general trees, 201

- Binary search, 318
 - efficiency of, 321, 424
- Binary tree, 145, 147, 301
 - indexing, 345
 - search, 322
 - traversals, 152
- BINARYSEARCH, procedure, 320
- Blank padding, 55, 56
- Blink, pointer, 69, 70
- Block, 47
- Blocked queue, 94
- Blocking, 307, 340
- Boundary folding, 325
- Boundary tag buddies, 370, 381, 409
- Boundary tag method, 381
- Branch, 143
- Breadth-first search, 263
- Breadth-first spanning forest, 263
- Bubble sort, 6–12, 288
 - efficiency of, 8, 9
- Bucket, 338
- Bucket hashing, 338
- Buddy blocks, 370
- Buddy systems, 370
- BUILDTREE, procedure, 170
- Buying decisions, 4

- CALL, 122
- CASE, statement, 75
- Chained pointer database, 394
- Character strings, 53
- Check buddies, reference, 372
- Child nodes, 143
- Circular implementation of a queue, 89–91
- Circular linked list, 42
- Cluster, 73
- COALESCE, procedure, 371, 372
- Coalesced, 37
- COBOL, 53
- CODASYL model, 403–406
- Codd, E. F., 406
- Collision processing, 328–340
- Collisions, 323, 335
- Column coordinates, 222
- Column major, 224
- Compaction, 66, 67, 366
- Compiler design, 96, 160, 198
- Complex network, 399, 400, 403

- Computer system, 365
- Computerworld, 301
- CONCAT_FL, procedure, 57, 58
- Concatenation, string, 54, 56–58, 63, 73
- Conformant arrays, 309
- Cross arcs, 259
- Cybernetics, 246
- Cycle, 248
- Cylinder, 342

- Data abstraction, 230, 394
- Data duplication problem, 386
- Data item, 1
- Data structures, uses of, 4
- Data systems, languages, 404
- Database, 4, 365, 386
 - designers, 403
 - management, 206, 263, 365, 386
- Database management system (DBMS), 403
- Datamation, 394
- Dec's VAX Pascal, 416
- Degree, 248
- DELETE, procedure, 77
- DELETENODE, procedure, 38–41, 48
- DELETENODEDOUBLE, procedure, 44, 45
- Deleting from a queue, 84–86, 93
- Deleting in a linked list, 37–40
- Deletion algorithm for binary tree, 164
- Deletion, string, 54
- Delimiter, 109
- Depth of a tree, 148
- Depth-first search, 257
- Depth-first spanning forest, 259
- DFSEARCH, procedure, 262
- Digit or Character extraction, 326
- Digraph, 246
- Dijkstra, E. W., 269
- Diminishing increment sort, 294
- Direct access, 314
- Directed edges, 246
- Directed graph, 246
- Directed path, 247
- Directory, 342, 343, 344
- Disk layout, 339
- Disk pack, 342
- DISPOSE, procedure, 32, 39
- Division remainder technique, 326, 328
- Dope vector, 225, 228
- Dot notation, 19
- Double arrow arcs, 259
- Doubly linked circular list, 43, 69
- Driver procedures, 428
- Dummy header, 37, 40, 42, 100
- Dummy root node, 179
- Dynamic memory allocation, 17–21, 30, 61, 63, 91, 100

- Edges, 246
- Edgweight, 252
- Efficiency of the binary tree, 168
- Efficiency-Generalized dope vector, 234
- Efficient algorithm, 5, 420
- Equivalence classes, 428, 429
- EVALUATEPOSTFIX, procedure, 114, 115
- External sorting, 286, 307

- Factorial expressions, 116
- FACTORIAL, function, 118
- Fibonacci buddies, 370
 - buddy systems, 377–381, 409
 - numbers, 380
 - schemes, 384
 - sequence, 377
 - systems, 377
- Fields, 11, 386
- FIFO, 82
- File organization, 314
- File sorting, 307
- FIND, command, 416
- FINDMIN, procedure, 280
- FINDPATH, procedure, 255
- Finiteness, 418, 419
- First-in-first-out (FIFO), 82, 103
- Fixed length, 366
- Fixed length string method, 54–60
- Flink, pointer, 69, 70
- Floyd, R. W., 274
- Folding, 325
- FORTTRAN, 53, 224, 324
- Forward link, 43
- Fragmentation, 366
- Fragments, 366
- Front, pointer, 82–95, 100, 103
- Full, 162
- Full tree, 177
- Functionally cohesive, 24
- Fundamental criterion of efficiency, 421

- Garbage collection, 63, 66–67, 365, 409
- GARBAGECOLLECT, procedure, 371
- Gaussian elimination, 245
- Genealogical tree, 142
- General efficiency of sort routines, 287
- General graph, 246
- General trees, 200
- Generalized dope vector method, 227
- Generic, 5
- GET, command, 416
- GETNODE, procedure, 32–34, 37, 39, 43, 48
- Global, 10
- Global variables, 134
- Graph, 246
- Graphic documentation, 7

- Hamilton, Sir William Rowan, 252
- HANOI, procedure, 121, 122
- HASH, function, 357
- Hashfile, program, 357–359
- Hashing function, 322
 - construction of, 323–328
- Hashing subscripts, 241
- Hashing techniques, 323–328
- Head pointer, 30
- Heap, 18
- Heap sort, 301–306
 - efficiency of, 305
- HEAPSORT, procedure, 304
- Height of a tree, 185
- Height-balanced binary trees, 185
- Height-balancing, 176, 177
- Hierarchical, 142, 146
 - Database Mgmt. Systems, 146
 - model, 403–405
- Hinds, J. A., 381

- IMS (Information Management System), 404
- Incidence matrix, 249
- Indegree, 248
- Index, 340
- Index tables, 61–64
- Indexed search techniques, 340–355
- Indexed sequential access Method, 341
- Indexed sequential search technique, 341–345, 425
- Infix notation, 107, 147, 156
- INFIXPRIORITY, function, 110
- Information management systems, 146
- INITIALIZE, procedure, 33
- Inorder predecessor, 178
- Inorder successor, 178
- Inorder threads, 178
- Inorder traversal, 147, 156, 178, 179
- INORDER_STACK_TRAVERSAL, procedure, 157
- INORDERTRAV, procedure, 156
- Input, 418, 419
- INSERT, procedure, 76
- INSERT_FL, procedure, 59, 60
- INSERT_LL, procedure, 70–72
- INSERT_WI, procedure, 64–65, 68
- Inserting in a linked list, 32, 33, 36
- Inserting in a queue, 84, 85, 87, 92, 95
- Insertion rule, 161, 182
- Insertion sort, 288
 - efficiency of, 290
- INSERTION, procedure, 289
- Insertion, string, 54, 63–66, 70
- Insertions into a threaded tree, 182
- INSERTNODE, procedure, 36, 37–39, 42, 48
- INSERTNODEDOUBLE, procedure, 44, 45
- INSERTTHREADED, procedure, 184
- INSPECT, procedure, 358, 359

- Internal sorting, 286
- Intersection records, 405
- INTOPOST, procedure, 112, 113
- Intrinsically Slow Access Method, 345
- Inverted files, 392, 397
- ISAM, acronym, 341

- Key, 314
- Key-to-address transformation, 322
- Kirchoff's laws, 235
- Kroehnke, D., 389
- kth Fibonacci sequence, 381

- Landis, Y. M., 185, 199
- Last-in-first-out (LIFO), 82, 103
- Le Chatelier, Henri Louis, 235
- Leaf node, 144
- Left buddy count, 380
- Left subscript major form, 226
- Left subtree, 145
- LEFTOFLEFT, procedure, 188
- Length of a path, 250
- LENGTH, function, 55, 56
- Level of a tree, 143
- LIFO, 82, 103
- Linear collision processing, 328–331
- Linear hashing, efficiency of, 331
- Linear representation, 147, 148
- Linear sequence of memory locations, 223
- LINEARHASH, procedure, 329, 330
- Link field, 30
- Linked hashing, efficiency of, 337, 338
- Linked lists, 27, 28, 91–93, 222
 - implementation of queue, 89, 91–93
 - implementation, stack, 101
 - method, 69–74
- Linked method-collision processing, 335–337
- Linked representation, 147, 149, 150, 152
- LINKEDHASH, procedure, 337
- LIST, procedure, 78
- LOADTREE, procedure, 211
- Local variables, 134
- Logical module, 6
- Logical order, 30
- Lset, 61–63

- Management of data, 365
- Many-to-many relationship, 264, 400
- Mapping function, 224
- Matrix, 222
- Matrix multiplication, 251
- Maurer & Williams, authors, 325
- Members, the, 404
- Memory stack, 97

- Memory waste, 55, 58
- Menu, 75
- Merge sort, 307–308
- Meyers, Glenford, 431
- Micro assembler, 97
- Minimum spanning tree, 252, 265
- MINPATHFLOYD, procedure, 274
- MINSPAN, procedure, 268, 280
- MINSPAND, program, 280
- Mod, operator, 89–91, 103
- Modular design, 75
- Modular testing, 428
- Multi-linked list, 46, 47
- Multidimensional arrays, 224
- Multilink database, 396
- Multilink files, 392, 394–397
- Murphy's Law, 6, 328

- Network, 252, 264, 399
- NEW, procedure, 18, 32, 39, 100
- Newton Method, 419
- Nil, 40, 42
- Node, 30, 246
- Non-unique secondary key, 390
- NONRECURSIVEHANOI, procedure, 128–130
- Null, 30, 73

- One-to-many relationship, 254, 388, 389, 392, 396, 400, 401, 403, 404, 407
- One-to-one relationship, 388
- Operating system, 94, 252
- Optimal index, 425
- Order of a function, 421
- Ordered list, 28
- Ordering, 286
- Ordering property for binary trees, 160, 177, 182
- Otherwise, statement, 75
- Outdegree, 248
- Output, 418, 420
- Overflow area, 335
- Owner, the, 404

- Palindrome, program, 104, 135–136
- Parallel fields, 315
- Parent node, 143
- Parenthesizing expressions, 108
- Parse trees, 199
- Parsing, 107, 109–111
- Pascal record declaration, 11, 13
- Pattern matching, 54, 70
- Physical order, 30
- Pivot node, 186, 192
- PL/1, 53

- PLACEQUEEN, procedure, 134
- Pointer sort, 18–20, 29
- Pointer variables, 18–20
- Pointers, 14, 149
- Polynomial, 51, 52
- Pop, 100, 103, 110, 113, 118, 124, 127, 130, 154
- POP, procedure, 100
- Popping, stack, 82–83, 96
- Postern of Fate, novel, 73
- Postfix notation, 107, 109, 147, 159
- Postorder traversal, 147, 159, 204
- POSTORDERTRAV, procedure, 159
- Preciseness, 418, 419
- Prefix notation, 108, 109, 147, 152
- Preorder traversal, 147, 152–153, 257
- PREORDER_STACK_TRAVERSAL, procedure, 154, 155
- PREORDERTRAV, procedure, 153
- Primary key, 366, 390
- Prime hash area, 335
- Principle of Le Chatelier, 235
- Priority queues, 94–95
- Program design, 427
- Proportional, 9
- Push, 97, 98, 103, 110, 113, 118, 120, 124, 154
- PUSH, procedure, 100
- Pushing, stack, 82–83, 97
- Put, command, 416, 417
- PUT, procedure, 233

- Quadratic and Rehashing methods, 332, 333
 - efficiency, 333, 334
- Queue, 82–89, 222
 - conditions, 85, 91, 94
 - examples, 84–93
- Quick sort, 295
 - efficiency of, 299
 - recursive, 299
- QUICKSORT, procedure, 297
- QUICKSORTRECURSIVE, procedure, 299

- Random number generator, 324
- Randomized storage, 324
- Rank, 225
- Read-write head, 339
- Ready queue, 94
- Rear, pointer, 82–93, 103
- Record, 2, 386
- Recursion, 97, 115–118, 120, 122–123, 127, 130, 137, 144, 257
- Recursive call, 97
- Relational DBMS, 394
- Relational algebra, 406
- Relational calculus, 406

- Relational model, 403, 406
- Relations, 406
- REMOVE, procedure, 86, 92–93
- Reset, command, 415, 416
- RETURNNODE, procedure, 32, 35, 37, 39
- Reverse Polish notation, 107
- REVERSE, procedure, 135
- Right subscript major form, 226
- Right subtree, 145
- RIGHTOFLLEFT, procedure, 190
- Root, 143
- Root node, 143
- Row coordinates, 222
- Row major, 224
- Runtime, 4

- Scheduling, 83–95, 100, 246, 252
- Schneyer, R., 431
- Search efficiencies, Summary table, 361
- Search node, 257
- Secondary key processing, 390
- Sectors, 339
- Seeds, 324
- Seek, command, 415, 416, 417
- Segment, 307
- Selection sort, 290
 - efficiency of, 291
- SELECTION, procedure, 290–291
- Semaphores, 94
- Sequential search, 314, 317, 339
 - efficiency of, 318
- SEQUENTIALSEARCH, procedure, 317
- Shell sort, 292–294
 - efficiency of, 294
- Shell, program, 309
- SHELLSORT, procedure, 294–295, 309–310
- Shift folding, 325
- Shortest path algorithm, 268
- SHORTPATH, procedure, 273
- Siblings, 143
- Simple network, 399, 403
- Simulation, 4
- Single arrow arcs, 259
- Singly linked circular list, 42
- Singly linked list, 30, 52
- Singly-dimensioned arrays, 222
- Sink node, 248
- SNOBOL, 53
- Software engineering, 24
- Software system, 24
- Sorting, 6–23, 29, 286
- Sorting algorithms, 286
- Sorting methods comparison table, 311
- Source node, 248
- Spanning forest, 399

- Sparse matrices, 224, 249
- SPARSE, function, 231
- Stack, 82, 96, 107, 109–111, 113, 115, 118, 120, 122–127, 222
- Stack frame, 118–119, 123
- Stack, examples, 83, 98, 101
- STACKPRIORITY, function, 110
- Static memory allocation, 17, 30
- String handling, 53
- String manipulation tools, 54
 - table, 79
- String overflow, 58, 59
- String, storage techniques, 54
- Strongly connected, 247
- Stub routines, 428
- Subroutine call, 95
- Substring operations, 54, 70, 72
- Subtree, 144
- Symbol table, 224
- Symmetric matrices, 250
- Symmetric structures, 146
- Synonyms, 323
- System memory allocation, 18, 19

- Tail recursion, 120
- Target, 314
- Teague, author, 325
- Ternary representation—general tree, 204
- Ternary tree, 204
- TERNARYPREORDER, procedure, 208, 212
- Test cases, 428
- Text oracle, 428
- Text editor, 76–78, 81
- Text manipulation, uses of, 53
- Texted, program, 76–79
- THREADEDINORDER, procedure, 181
- Threading, 150, 176
- Top, pointer, 82, 96–103
- Topological order, 261, 276
- TOPOLOGICALORDER, procedure, 277
- Tracks, 339
- Traveling salesman problem, 252
- Traversing, 146–147, 152
- Tree, 142, 222, 246
- Tree arcs, 259
- Trie indexing, 350–354
 - efficiency of, 354–355
 - insertions, 354, 355
- Two-dimensional array, 222

- UCSD and TURBO Pascal, 416
- Unbounded set, 421
- Undirected graph, 246
- Uniqueness, 418, 419

Unix operating system, 163
UNSTACKC, procedure, 101
Upper triangular matrix, 276
Usable workspace, 66

Vectors, 27
Vertices, 252
Volatile list, 321

WALKDOWN, procedure, 304
Weakly connected, 247
Weighted edges, 252, 264
Workspace/index table method, 61–69
Worst case, 421
Wrap around, queue, 89, 90

Contents

Preface xv

1

Data Structures—An Overview 1

- 1-1 Introductory Considerations 1
- 1-2 Algorithms for Data Structures 5
- 1-3 An Example of Algorithm Development in Pascal 6
 - Initial Version of Bubble Sort, 6
 - Enhancing Efficiency of the Bubble Sort, 9
- In the World of Applications . . . 23
- Program Design Considerations 24
- SUMMARY 25
- KEY TERMS 25
- EXERCISES 26
- PROGRAMMING PROBLEMS 26

2

Linked Lists 27

- 2-1 Introductory Considerations 27
- 2-2 Arrays 27
- 2-3 Linked Lists 28
 - Singly Linked Lists, 30
 - Insertions and Deletions, 32
- 2-4 Variations on Linked List Structures 40
 - Dummy Headers, 40
- In the World of Applications . . . 41
 - Circular Linked Lists, 42
 - Doubly Linked Circular List, 43
- Program Design Considerations 47
- SUMMARY 49
- KEY TERMS 49
- EXERCISES 50
- PROGRAMMING PROBLEMS 50