Data Structures and Algorithms using Python

Efficiently using data structures to collect, organize, and retrieve information is one of the core abilities modern computer engineers are expected to have. Python currently is one of the most popular programming languages, and as such, it has become vital for students to understand this concept in this language.

This student-friendly textbook provides a complete view of data structures and algorithms using the Python programming language, striking a balance between theory and practical application. All major algorithms have been discussed and analysed in detail, and the corresponding codes in Python have been provided. Diagrams and examples have been extensively used for better understanding. Running time complexities are also discussed for each algorithm, allowing the student to better understand how to select the appropriate one.

The book has been written with both undergraduate and graduate students in mind. Each chapter ends with a large number of problems, including multiple choice questions, to help consolidate the knowledge gained. This will also be helpful with competitive examinations for engineering in India such as GATE and NET. As such, the book will be a vital resource for students as well as professionals who are looking for a handbook on data structures in Python.

Subrata Saha is Head of the Department of Computer Applications at Techno India Hooghly, West Bengal. He has more than 20 years of teaching experience in various subjects in computer science and engineering, including data structures and algorithms, programming in C, C++, Java, and Python, basic computation, operating systems, and so on. He has previously published Basic Computations and Programming with C with the Cambridge University Press in 2016.
Data Structures and Algorithms using Python

Subrata Saha
To my beloved wife, 
Ms Sriparna Saha, and my princess, Miss Shreya Saha, whose constant support motivated me to write this book.
## Contents

**Preface**  
xxi  

**Acknowledgments**  
xxiii  

1. DATA STRUCTURE PRELIMINARIES  
1  

1.1 Concept of Data Type  
1.1.1 Primitive data type  
1.1.2 User defined data type  
1.1.3 Abstract data type  
1

1.2 What Is Data Structure?  
2

1.3 Definition and Brief Description of Various Data Structures  
1.3.1 Array  
1.3.2 Linked list  
1.3.3 Stack  
1.3.4 Queue  
1.3.5 Graph  
1.3.6 Tree  
1.3.7 Heap  
2

1.4 Data Structures versus Data Types  
9

1.5 Operations on Data Structures  
10

*Data Structure Preliminaries At a Glance*  
11

*Multiple Choice Questions*  
11

*Review Exercises*  
14
2. INTRODUCTION TO ALGORITHM

2.1 What Is an Algorithm? 15
2.2 Importance of an Algorithm 16
2.3 Different Approaches to Designing an Algorithm 16
2.4 Algorithm Design Tools: Flowchart and Pseudocode 16
  2.4.1 Flowchart 17
  2.4.2 Pseudocode 19
2.5 Control Structures Used in an Algorithm 20
2.6 Time and space complexity 24
2.7 Best Case, Worst Case, Average Case Time Complexity 24
2.8 Time–Space Trade-off 25
2.9 Frequency Count and Its Importance 25
2.10 Analyzing Algorithms 27
  2.10.1 Big O Notation 27
  2.10.2 \( \Omega \) (Omega) Notation 29
  2.10.3 \( \Theta \) (Theta) Notation 30
  2.10.4 Other Useful Notations 30
2.11 Divide and Conquer Strategy 31
2.12 Dynamic Programming 32
2.13 Greedy Method 34

Introduction to Algorithm At a glance 35

Multiple Choice Questions 36

Review Exercises 39

3. ARRAY

3.1 Definition 41
3.2 Creating an Array 41
3.3 Accessing Elements of an Array 45
3.4 Operations on an Array 48
  3.4.1 Adding Elements to an Array 48
  3.4.2 Removing Elements from an Array 49
  3.4.3 Slicing of an Array 51
  3.4.4 Searching Element in an Array 53
  3.4.5 Updating Elements in an Array 54
  3.4.6 Concatenation of Arrays 55
### Contents

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.4.7 Multiplication or Repetition on Array</td>
<td>56</td>
</tr>
<tr>
<td>3.5 Representation of Polynomials</td>
<td>57</td>
</tr>
<tr>
<td>3.6 Two Dimensional Array</td>
<td>59</td>
</tr>
<tr>
<td>3.7 Creation of a Two Dimensional Array</td>
<td>59</td>
</tr>
<tr>
<td>3.8 Accessing Elements of a Two Dimensional Array</td>
<td>62</td>
</tr>
<tr>
<td>3.9 Representation of a Two Dimensional Array in Memory</td>
<td>63</td>
</tr>
<tr>
<td>3.9.1 Row Major Representation</td>
<td>63</td>
</tr>
<tr>
<td>3.9.2 Column Major Representation</td>
<td>64</td>
</tr>
<tr>
<td>3.10 Operations on Two Dimensional Arrays</td>
<td>65</td>
</tr>
<tr>
<td>3.10.1 Matrix Addition</td>
<td>65</td>
</tr>
<tr>
<td>3.10.2 Matrix Multiplication</td>
<td>66</td>
</tr>
<tr>
<td>3.10.3 Transpose of a Matrix</td>
<td>66</td>
</tr>
<tr>
<td>3.10.4 Slicing of a Matrix</td>
<td>66</td>
</tr>
<tr>
<td>3.11 Sparse Matrix</td>
<td>68</td>
</tr>
<tr>
<td>3.12 Programming Examples</td>
<td>74</td>
</tr>
<tr>
<td><strong>Array at a Glance</strong></td>
<td>79</td>
</tr>
<tr>
<td><strong>Multiple Choice Questions</strong></td>
<td>79</td>
</tr>
<tr>
<td><strong>Review Exercises</strong></td>
<td>81</td>
</tr>
<tr>
<td><strong>Problems for Programming</strong></td>
<td>84</td>
</tr>
</tbody>
</table>

#### 4. PYTHON DATA STRUCTURES

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>4.1 Lists</td>
<td>85</td>
</tr>
<tr>
<td>4.1.1 Creating a List</td>
<td>85</td>
</tr>
<tr>
<td>4.1.2 Accessing List Elements</td>
<td>86</td>
</tr>
<tr>
<td>4.1.3 Operations on a List</td>
<td>88</td>
</tr>
<tr>
<td>4.1.3.1 Adding Elements to a List</td>
<td>88</td>
</tr>
<tr>
<td>4.1.3.2 Removing Elements from a List</td>
<td>89</td>
</tr>
<tr>
<td>4.1.3.3 Slicing of a List</td>
<td>91</td>
</tr>
<tr>
<td>4.1.3.4 Searching Element in a List</td>
<td>93</td>
</tr>
<tr>
<td>4.1.3.5 Updating Elements in a List</td>
<td>94</td>
</tr>
<tr>
<td>4.1.3.6 Concatenation of Lists</td>
<td>95</td>
</tr>
<tr>
<td>4.1.3.7 Multiplication or Repetition of List</td>
<td>96</td>
</tr>
<tr>
<td>4.1.4 Nested List</td>
<td>97</td>
</tr>
<tr>
<td>4.1.5 List Functions</td>
<td>97</td>
</tr>
<tr>
<td>4.1.6 List Methods</td>
<td>98</td>
</tr>
</tbody>
</table>
4.1.7 Looping in a List  
4.1.8 List vs. Array  
4.2 Tuples  
4.2.1 Creating a Tuple  
4.2.2 Accessing Tuple Elements  
4.2.3 Operations on Tuple  
4.2.4 Nested Tuples  
4.2.5 Tuple Methods  
4.2.6 Looping in a Tuple  
4.3 Sets  
4.3.1 Creating a Set  
4.3.2 Operations on a Set  
4.3.2.1 Adding Elements to a Set  
4.3.2.2 Removing Elements from a Set  
4.3.2.3 Searching an Element in a Set  
4.3.3 Set Methods  
4.3.4 Frozenset  
4.4 Dictionaries  
4.4.1 Creating a Dictionary  
4.4.2 Accessing Values in a Dictionary  
4.4.3 Operations on a Dictionary  
4.4.3.1 Adding and Modifying an Element in a Dictionary  
4.4.3.2 Removing an Element from a Dictionary  
4.4.3.3 Membership Test in a Dictionary  
4.4.4 Looping in a Dictionary  
4.4.5 Nested Dictionaries  
4.4.6 Dictionary Methods  
4.5 Comparative Study  
4.6 Programming Examples  

*Python Data Structures at a Glance*  
*Multiple Choice Questions*  
*Review Exercises*  
*Problems for Programming*
7. LINKED LIST 207

7.1 Definition 208
7.2 Advantages of a Linked List 208
7.3 Types of Linked Lists 208
7.4 Implementing a Singly Linked List 210
7.5 Operations on Singly Linked List 211
   7.5.1 Creating a Singly Linked List 212
   7.5.2 Displaying a Singly Linked List 213
   7.5.3 Inserting a New Element in a Singly Linked List 214
      7.5.3.1 Inserting an Element at the Beginning of a List 215
      7.5.3.2 Inserting an Element at the End of a List 216
      7.5.3.3 Inserting a Node after a Specified Node 217
      7.5.3.4 Inserting a Node before a Specified Node 219
   7.5.4 Deleting a Node from a Singly Linked List 220
      7.5.4.1 Deleting the First Node 220
      7.5.4.2 Deleting the Last Node 221
      7.5.4.3 Deleting any Intermediate Node 222
7.6 Applications of a Singly Linked List 228
7.7 Implementing a Circular Singly Linked List 233
7.8 Operations on a Circular Singly Linked List 233
   7.8.1 Creating a Circular Singly Linked List 234
   7.8.2 Displaying a Circular Singly Linked List 235
   7.8.3 Inserting a New Element in a Circular Linked List 236
      7.8.3.1 Inserting an Element at the Beginning of a Circular Linked List 237
      7.8.3.2 Inserting an Element at the End of a List 239
      7.8.3.3 Inserting a Node after a Specified Node 240
      7.8.3.4 Inserting a Node before a Specified Node 242
   7.8.4 Deleting a Node from a Circular Singly Linked List 243
      7.8.4.1 Deleting the First Node 244
      7.8.4.2 Deleting the Last Node 245
      7.8.4.3 Deleting any Intermediate Node 246
7.9 Applications of a Circular Singly Linked List 253
7.10 Implementing a Doubly Linked List 255
7.11 Operations on a Doubly Linked List 256
7.11.1 Inserting an Element at the Beginning of a Doubly Linked List 257
7.11.2 Inserting an Element at the End of a Doubly Linked List 258
7.11.3 Inserting a Node at a Specified Position in a Doubly Linked List 259
7.11.4 Deleting a Node from a Doubly Linked List 261
  7.11.4.1 Deleting the First Node 261
  7.11.4.2 Deleting the Last Node 262
  7.11.4.3 Deleting any Intermediate Node 263
7.12 Implementation of a Circular Doubly Linked List 270
7.13 Operations on a Circular Doubly Linked List 271
  7.13.1 Inserting an Element at the Beginning of a Circular Doubly Linked List 271
  7.13.2 Inserting an Element at the End of a Circular Doubly Linked List 273
  7.13.3 Deleting the First Node from a Circular Doubly Linked List 274
  7.13.4 Deleting the Last Node from a Circular Doubly Linked List 276
7.14 Header Linked List 283
7.15 Advantages of a Header Linked List 296
7.16 Disadvantages of a Linked List 296
7.17 Programming Examples 297
Linked List at a Glance 303
Multiple Choice Questions 304
Review Exercises 308
Problems for Programming 309

8. STACK 311
  8.1 Definitions and Concept 311
  8.2 Operations Associated with Stacks 312
  8.3 Representation of a Stack 312
    8.3.1 Array Representation 312
    8.3.2 Python List Representation 314
    8.3.3 Linked Representation 317
  8.4 Multiple Stacks 321
  8.5 Applications of a Stack 321
    8.5.1 Parenthesis Checking Problem 322
    8.5.2 Conversion and Evaluation of Different Arithmetic Expressions 325
      8.5.2.1 Different Notations of Arithmetic Expressions 325
      8.5.2.2 Conversion of an Infix Expression into a Postfix Expression 327
Contents

8.5.2.3 Evaluation of a Postfix Expression 331
8.5.2.4 Conversion of a Postfix Expression into an Infix Expression 334
8.5.2.5 Conversion of an Infix Expression into a Prefix Expression 335
8.5.3 Reversing any Sequence 339
8.5.4 Recursion 341

Stack at a Glance 343
Multiple Choice Questions 344
Review Exercises 348
Problems for Programming 349

9. QUEUE 351
9.1 Definitions and Concept 351
9.2 Operations Associated with Queues 352
9.3 Representation of a Queue 352
9.3.1 Array Representation of a Queue 353
9.3.2 Circular Queue 360
9.3.2.1 Operations on a Circular Queue 361
9.3.3 Python List Representation of a Queue 369
9.3.4 Linked Representation of a Queue 371
9.3.4.1 Using a Header List 371
9.3.4.2 Using a Single Circular Linked List with a Single Tail Pointer 375
9.4 Multiple Queues 377
9.5 Special Queues 378
9.5.1 DEQue 378
9.5.2 Priority Queue 381
9.6 Applications of a Queue 384
Queue at a Glance 385
Multiple Choice Questions 385
Review Exercises 387
Problems for Programming 388

10. TREES 391
10.1 Definition and Concept 391
10.2 Terminology 392
10.3 Types of Trees 393
10.3.1 General Tree 393
10.3.2 Forest 393
10.3.3 Binary Tree 394
10.3.4 Strictly Binary Tree 394
10.3.5 Complete Binary Tree 395
10.3.6 Full Binary Tree 395
10.3.7 Extended Binary Tree 396
10.3.8 Binary Search Tree (BST) 396
10.3.9 Expression Tree 397
10.3.10 Tournament Tree 398

10.4 Representation of a Binary Tree 398
10.4.1 Array Representation of a Binary Tree 398
10.4.2 Linked list Representation of a Binary Tree 399

10.5 Binary Tree Traversal 400
10.5.1 Preorder Traversal of a Binary Tree 401
10.5.2 Inorder Traversal of a Binary Tree 403
10.5.3 Postorder Traversal of a Binary Tree 404
10.5.4 Level Order Traversal of a Binary Tree 406

10.6 Construction of a Binary Tree from the Traversal Path 406
10.7 Conversion of a General Tree to a Binary Tree 408

10.8 Binary Search Tree (BST) 410

10.9 Operations on a Binary Search Tree – Recursive and Non-recursive 412
10.9.1 Insertion of a New Node in a Binary Search Tree 412
10.9.2 Searching a Node in a Binary Search Tree 414
10.9.3 Traversing a Binary Search Tree 415
10.9.4 Deletion of a Node from a Binary Search Tree 415
10.9.5 Find the Largest Node from a Binary Search Tree 420
10.9.6 Finding the Smallest Node from a Binary Search Tree 421
10.9.7 Counting the Total Number of Nodes in a Binary Search Tree 422
10.9.8 Counting the Number of External Nodes in a Binary Search Tree 422
10.9.9 Counting the Number of Internal Nodes in a Binary Search Tree 423
10.9.10 Finding the Height of a Binary Search Tree 423
10.9.11 Finding the Mirror Image of a Binary Search Tree 424

10.10 Threaded Binary Tree 436
10.10.1 Representation of a Threaded Binary Tree 439
# Contents

10.10.2 Operations on an Inorder Threaded Binary Tree  
  10.10.2.1 Inorder Traversal of an Inorder Threaded Binary Tree  
  10.10.2.2 Inserting a New Node in an Inorder Threaded Binary Search Tree  
  10.10.2.3 Deletion of a Node from an Inorder Threaded Binary Search Tree  

10.11 AVL Tree  
  10.11.1 Operations on an AVL Tree  
    10.11.1.1 Inserting a Node in an AVL Tree  
    10.11.1.2 Deleting a Node from an AVL Tree  

10.12 Red–Black Tree  
  10.12.1 Inserting a New Node in a Red–Black Tree  
  10.12.2 Deleting a Node from a Red–Black Tree  

10.13 Huffman Coding  

10.14 M-way Search Trees  

10.15 B Tree  
  10.15.1 Inserting a New Element in a B Tree  
  10.15.2 Deleting Elements from a B Tree  
  10.15.3 Searching an Element from a B Tree  

10.16 B+ Tree  
  10.16.1 Inserting a New Element in a B+ Tree  
  10.16.2 Deleting Elements from a B+ Tree  

10.17 B* Tree  

10.18 2–3 Tree  

10.19 Trie Tree  

Strings at a Glance  

Multiple Choice Questions  

Review Exercises  

Problems for Programming  

11. HEAP  

11.1 Definition and Concept  

11.2 Representation of a Heap in Memory  

11.3 Operations on a Heap  
  11.3.1 Inserting a New Element in a Heap  

11.4 Heapify  

11.5 Heap Sort  

11.6 Priority Queue  

11.7 Applications of Heaps  

11.8 Splay Trees  

Strings at a Glance  

Multiple Choice Questions  

Review Exercises  

Problems for Programming  


<table>
<thead>
<tr>
<th>Contents</th>
</tr>
</thead>
<tbody>
<tr>
<td>11.3.2 Deleting an Element from a Heap</td>
</tr>
<tr>
<td>11.4 Applications of Heap</td>
</tr>
<tr>
<td>11.4.1 Implementing a Priority Queue Using Heap</td>
</tr>
<tr>
<td>Heap at a Glance</td>
</tr>
<tr>
<td>Multiple Choice Questions</td>
</tr>
<tr>
<td>Review Exercises</td>
</tr>
<tr>
<td>Problems for Programming</td>
</tr>
<tr>
<td><strong>12. GRAPH</strong></td>
</tr>
<tr>
<td>12.1 Definition and Concept</td>
</tr>
<tr>
<td>12.2 Terminology</td>
</tr>
<tr>
<td>12.3 Representation of a Graph</td>
</tr>
<tr>
<td>12.3.1 Adjacency Matrix Representation</td>
</tr>
<tr>
<td>12.3.2 Incidence Matrix Representation</td>
</tr>
<tr>
<td>12.3.3 Adjacency List Representation</td>
</tr>
<tr>
<td>12.3.4 Adjacency Multi-list Representation</td>
</tr>
<tr>
<td>12.4 Operations on a Graph</td>
</tr>
<tr>
<td>12.4.1 Insertion Operation</td>
</tr>
<tr>
<td>12.4.2 Deletion Operation</td>
</tr>
<tr>
<td>12.4.3 Graph Traversal</td>
</tr>
<tr>
<td>12.4.3.1 Breadth First Search Algorithm</td>
</tr>
<tr>
<td>12.4.3.2 Depth First Search Algorithm</td>
</tr>
<tr>
<td>12.5 Minimum Spanning Tree</td>
</tr>
<tr>
<td>12.5.1 Prim’s Algorithm</td>
</tr>
<tr>
<td>12.5.2 Kruskal’s Algorithm</td>
</tr>
<tr>
<td>12.6 Shortest Path Algorithm</td>
</tr>
<tr>
<td>12.6.1 Within a Given Source and Destination</td>
</tr>
<tr>
<td>12.6.2 Among All Pairs of Vertices</td>
</tr>
<tr>
<td>12.7 Applications of Graph</td>
</tr>
<tr>
<td>Graph at a Glance</td>
</tr>
<tr>
<td>Multiple Choice Questions</td>
</tr>
<tr>
<td>Review Exercises</td>
</tr>
<tr>
<td>Problems for Programming</td>
</tr>
</tbody>
</table>
13. SEARCHING AND SORTING  

13.1 Introduction to Searching  
13.1.1 Linear Search  
13.1.2 Binary Search  
13.1.3 Interpolation Search  

13.2 Introduction to Sorting  
13.2.1 Bubble Sort  
13.2.2 Selection Sort  
13.2.3 Insertion Sort  
13.2.4 Quick Sort  
13.2.5 Merge Sort  
13.2.6 Heap Sort  
13.2.7 Radix Sort  
13.2.8 Shell Sort  

13.3 Comparison of Different Sorting Algorithms  

13.4 Concept of Internal and External Sorting  

Searching and Sorting at a Glance  

Multiple Choice Questions  
Review Exercises  
Programming Exercises  

14. HASHING  

14.1 Definitions and Concept  
14.2 Hash Functions  
14.2.1 Division Method  
14.2.2 Multiplication Method  
14.2.3 Mid-square Method  
14.2.4 Folding Method  
14.2.5 Length Dependent Method  
14.2.6 Digit Analysis Method  

14.3 Collision Resolution Technique  
14.3.1 Open Addressing  
14.3.1.1 Linear Probing  
14.3.1.2 Quadratic Probing
14.3.1.3 Double Hashing 617

14.3.2 Chaining 620

14.4 Rehashing 623

14.5 Applications of Hashing 623

Hashing at a Glance 623

Multiple Choice Questions 624

Review Exercises 627

Appendix: Answers of Multiple Choice Questions 629

Index 633
In computer science and engineering, data structure and algorithm are two very important parts. Data structure is the logical representation of data, so that insertion, deletion, and retrieval can be done efficiently, and an algorithm is the step-by-step procedure to solve any problem. By studying different data structures, we are able to know their merits and demerits, which enriches our knowledge and our ability to apply the appropriate data structures at proper places when we try to write new applications. Studying different standard algorithms provides us much knowledge about solving new problems. Both data structures and algorithms are interrelated and are complementary to each other. By studying both data structures and algorithms, we may acquire a solid foundation of writing good code. This comprehensive knowledge helps to understand new frameworks as well.

With the studying of data structures and algorithms, it is very important to implement them using proper languages. Several books have been written on this topic using the C language. But today Python has become very popular because of its features such as being easy, open source, object oriented, portable, multi-threaded, extensive libraries, embeddable, etc. Hence, in this book data structures and algorithms are implemented using Python.

**About the Book**

This book is written to serve the purposes of a textbook for undergraduate courses of computer science and information technology and for undergraduate and postgraduate courses of computer application where data structure is one of the subjects in the syllabus. In this book different data structures and algorithms are discussed in a lucid manner so that students can understand the concept easily. All the relevant data structures and their operations are discussed with diagrams and examples for better understanding. After discussing relevant algorithms in detail, the algorithmic representation and the corresponding code in Python are given. Various programming examples along with new
problems for practice and a set of MCQ questions at the end of each chapter increase the self-learning process of the students.

The salient features of this book are:

- This book is written in very simple English for better understanding the complex concepts.
- In order to make the presentation visually interactive for students, neat labelled diagrams are provided wherever necessary. For each topic, explanations are clear and concise, avoiding verbosity as much as possible.
- Each topic is discussed in detail with proper examples.
- Algorithms are presented with algorithmic representation as well as with the corresponding code in Python.
- Complexity analysis is discussed for almost all problems discussed in this book and in very lucid manner for better understanding.
- A large number of solved programming examples.
- MCQ questions and their solutions.
Acknowledgments

I thank my beloved students for their constant motivation which made me write this book. My students are extremely fond of and fascinated by my ‘teaching from the ground up’ style and their demand inspired me to write this book. My heartiest thanks to them.

I would like to thank my student Sayandip Naskar for drawing the figures of stack and queue.

My heartiest thanks to my family members and friends for their constant support, encouragement, and unconditional love, which helped to conclude the book.

Finally, I would like to thank all the reviewers of this book for their critical comments and suggestions. I convey my sincere gratitude to Mr Agnibesh Das and the entire editing team at Cambridge University Press, India, for their great work.