1. Introduction

1.1. Objectives

1.2. Getting Started

1.3. What Is Computer Science?

1.4. What Is Programming?

1.5. Why Study Data Structures and Abstract Data Types?

1.6. Why Study Algorithms?

1.7. Review of Basic Python

1.8. Getting Started with Data

1.8.1. Built-in Atomic Data Types

1.8.2. Built-in Collection Data Types

1.9. Input and Output

1.9.1. String Formatting

1.10. Control Structures

1.11. Exception Handling

1.12. Defining Functions

1.13. Object-Oriented Programming in Python: Defining Classes

1.13.1. A Fraction Class

1.13.2. Inheritance: Logic Gates and Circuits

1.14. Summary

1.15. Key Terms

1.16. Discussion Questions

1.17. Programming Exercises

2. Analysis

2.1. Objectives

2.2. What Is Algorithm Analysis?

2.3. Big-O Notation

2.4. An Anagram Detection Example

2.4.1. Solution 1: Checking Off

2.4.2. Solution 2: Sort and Compare

2.4.3. Solution 3: Brute Force

2.4.4. Solution 4: Count and Compare

2.5. Performance of Python Data Structures

2.6. Lists

2.7. Dictionaries

2.8. Summary

2.9. Key Terms

2.10. Discussion Questions

2.11. Programming Exercises

3. Basic Data Structures

3.1. Objectives

3.2. What Are Linear Structures?

3.3. What is a Stack?

3.4. The Stack Abstract Data Type

3.5. Implementing a Stack in Python

3.6. Simple Balanced Parentheses

3.7. Balanced Symbols (A General Case)

3.8. Converting Decimal Numbers to Binary Numbers

3.9. Infix, Prefix and Postfix Expressions

3.9.1. Conversion of Infix Expressions to Prefix and Postfix

3.9.2. General Infix-to-Postfix Conversion

3.9.3. Postfix Evaluation

3.10. What Is a Queue?

3.11. The Queue Abstract Data Type

3.12. Implementing a Queue in Python

3.13. Simulation: Hot Potato

3.14. Simulation: Printing Tasks

3.14.1. Main Simulation Steps

3.14.2. Python Implementation

3.14.3. Discussion

3.15. What Is a Deque?

3.16. The Deque Abstract Data Type

3.17. Implementing a Deque in Python

3.18. Palindrome-Checker

3.19. Lists

3.20. The Unordered List Abstract Data Type

3.21. Implementing an Unordered List: Linked Lists

3.21.1. The Node Class

3.21.2. The Unordered List Class

3.22. The Ordered List Abstract Data Type

3.23. Implementing an Ordered List

3.23.1. Analysis of Linked Lists

3.24. Summary

3.25. Key Terms

3.26. Discussion Questions

3.27. Programming Exercises

4. Recursion

4.1. Objectives

4.2. What Is Recursion?

4.3. Calculating the Sum of a List of Numbers

4.4. The Three Laws of Recursion

4.5. Converting an Integer to a String in Any Base

4.6. Stack Frames: Implementing Recursion

4.7. Introduction: Visualizing Recursion

4.8. Sierpinski Triangle

4.9. Complex Recursive Problems

4.10. Tower of Hanoi

4.11. Exploring a Maze

4.12. Dynamic Programming

4.13. Summary

4.14. Key Terms

4.15. Discussion Questions

4.16. Glossary

4.17. Programming Exercises

5. Sorting and Searching

5.1. Objectives

5.2. Searching

5.3. The Sequential Search

5.3.1. Analysis of Sequential Search

5.4. The Binary Search

5.4.1. Analysis of Binary Search

5.5. Hashing

5.5.1. Hash Functions

5.5.2. Collision Resolution

5.5.3. Implementing the Map Abstract Data Type

5.5.4. Analysis of Hashing

5.6. Sorting

5.7. The Bubble Sort

5.8. The Selection Sort

5.9. The Insertion Sort

5.10. The Shell Sort

5.11. The Merge Sort

5.12. The Quick Sort

5.13. Summary

5.14. Key Terms

5.15. Discussion Questions

5.16. Programming Exercises

6. Trees and Tree Algorithms

6.1. Objectives

6.2. Examples of Trees

6.3. Vocabulary and Definitions

6.4. List of Lists Representation

6.5. Nodes and References

6.6. Parse Tree

6.7. Tree Traversals

6.8. Priority Queues with Binary Heaps

6.9. Binary Heap Operations

6.10. Binary Heap Implementation

6.10.1. The Structure Property

6.10.2. The Heap Order Property

6.10.3. Heap Operations

6.11. Binary Search Trees

6.12. Search Tree Operations

6.13. Search Tree Implementation

6.14. Search Tree Analysis

6.15. Balanced Binary Search Trees

6.16. AVL Tree Performance

6.17. AVL Tree Implementation

6.18. Summary of Map ADT Implementations

6.19. Summary

6.20. Key Terms

6.21. Discussion Questions

6.22. Programming Exercises

7. Graphs and Graph Algorithms

7.1. Objectives

7.2. Vocabulary and Definitions

7.3. The Graph Abstract Data Type

7.4. An Adjacency Matrix

7.5. An Adjacency List

7.6. Implementation

7.7. The Word Ladder Problem

7.8. Building the Word Ladder Graph

7.9. Implementing Breadth First Search

7.10. Breadth First Search Analysis

7.11. The Knight’s Tour Problem

7.12. Building the Knight’s Tour Graph

7.13. Implementing Knight’s Tour

7.14. Knight’s Tour Analysis

7.15. General Depth First Search

7.16. Depth First Search Analysis

7.17. Topological Sorting

7.18. Strongly Connected Components

7.19. Shortest Path Problems

7.20. Dijkstra’s Algorithm

7.21. Analysis of Dijkstra’s Algorithm

7.22. Prim’s Spanning Tree Algorithm

7.23. Summary

7.24. Key Terms

7.25. Discussion Questions

7.26. Programming Exercises

What is a datastructure?

Classifying datastructures

Design patterns and datastructures

What is an algorithm?

Representing algorithms

Choosing the right algorithm

Measuring algorithm efficiency

Comparing algorithms with Big Oh