Chapter 0 Preface

Search the book

Go

Enter search terms or a module, class or function name.

0.1. How to Use this System
0.2. OpenDSA Project Content Status
0.2.1. Status Report
0.2.1.1. Configuring Your Own Book

Chapter 1 Introduction for Data Structures and Algorithms Courses

1.1. Data Structures and Algorithms
1.1.1. Data Structures and Algorithms
1.1.1.1. Introduction
1.1.1.2. A Philosophy of Data Structures
1.1.1.3. Selecting a Data Structure
1.1.1.4. Introduction Summary Questions
1.1.2. Some Software Engineering Topics

Chapter 2 Biographies

2.1. Spotlight: Carl Friedrich Gauss
2.1.1. Spotlight: Carl Friedrich Gauss
2.2. Spotlight: Francis Bacon
2.2.1. Spotlight: Francis Bacon

Chapter 3 Programming Tutorials

3.1. Command Line Basics
3.1.1. What’s A CLI?
3.2. Parsing Command Line Parameters In Your Progam
3.2.1. Parameters In Programming
3.3. Using Parameters in Eclipse
3.3.1. Using Parameters in Eclipse
3.4. Installing the Web-CAT Submission Plug-in for Eclipse
3.4.1. Introduction
3.4.2. Installing the Plug-in
3.4.2.1. Un-Installing an Old Plug-in
3.4.2.2. Installing the Plug-in
3.4.3. Installing student.jar
3.4.4. Installing formatting support
3.5. Common Debugging Methods
3.5.1. Common Debugging Methods
3.5.1.1. Print Debugging

3.5.1.2. Rubber Duck Debugging
3.5.1.3. Wolf Fence Debugging
3.5.1.4. Print Debugging vs Source Debugging
3.6. Debugging In Eclipse
3.6.1. Debugging In Eclipse
3.6.1.1. What Is A Debugger?
3.6.1.2. Debugging Terms
3.6.1.3. Debugging A Memory Pool
3.6.1.4. The Eclipse Console
3.6.1.5. Conditional Breakpoints
3.7. Reading Input (from Files or Otherwise)
3.7.1. The Scanner Class
3.8. Random Access Files In Java
3.8.1. Understanding File I/O
3.8.1.1. Using RandomAccessFile Class
3.9. JUnit Testing And You
3.9.1. Getting Started
3.9.1.1. Design Considerations
3.9.1.2. Meaningful Tests
3.9.1.3. More Information
3.10. Writing JUnit Tests
3.10.1. Writing JUnit Tests
3.11. Code Coverage In JUnit
3.11.1. Code Coverage In JUnit
3.11.1.1. Using EclEmma
3.12. Testing
3.12.1. Testing vs. Debugging
3.13. Testing for Code Coverage
3.13.1. Testing for Code Coverage
3.14. Another Example
3.14.1. Another Example
3.15. Bowling Example
3.15.1. Bowling Example

Chapter 4 Design I

4.1. Abstract Data Types
4.1.1. Abstract Data Types
4.2. Introduction to Object Oriented Programming
4.2.1. Introduction to Object Oriented Programming
4.2.1.1. Principles of Object Oriented Programming
4.2.1.2. Encapsulation
4.2.1.3. Inheritance
4.2.1.4. Polymorphism
4.2.1.5. Run-time Polymorphism
4.2.1.6. Compile-time Polymorphism
4.2.1.7. Abstraction

4.3. The Unified Modeling Language
4.3.1. The Unified Modeling Language
4.3.1.1. Static and Dynamic Diagrams
4.3.1.2. Class Diagrams
4.4. Software Development Processes
4.4.1. Software Development Processes
4.4.1.1. Waterfall Method
4.4.1.2. Rational Unified Process
4.4.1.3. Other Plan-Driven Approaches
4.4.1.4. eXtreme Programming (XP)
4.4.1.5. Crystal
4.4.1.6. Scrum
4.4.1.7. Feature-Driven Development (FDD)

Chapter 5 Introduction to Pointers in Java

5.1. Pointers Chapter Introduction
5.2. Basic References Part 1
5.2.1. Pointers and References
5.2.1.1. What is a Pointer?
5.2.1.2. What is a Reference?
5.2.2. Data Types in Java
5.2.3. Referencing and Dereferencing
5.2.4. changePointeeDataDirect
5.2.5. The Employee Class
5.2.6. Reference Assignments
5.2.7. circularList1
5.3. Basic References Part 2
5.3.1. Sharing
5.3.2. Shallow and Deep Copying
5.3.2.1. What does it mean to copy?
5.3.3. Shallow and Deep Comparing
5.3.4. Bad References
5.3.4.1. Why Are Bad Reference Bugs So Common?
5.4. Pointers Syntax
5.4.1. Syntax
5.4.1.1. Declaring a Reference Variable
5.4.1.2. Assigning a pointee to a reference
5.4.1.3. Dereference the reference
5.4.1.4. Example Reference Code
5.4.2. changeHeadPointer1
5.4.2.1. Reference Rules Summary
5.4.3. Java References vs Pointers
5.4.4. How Are References Implemented In The Machine?
5.5. Local Memory
5.5.1. Local Memory
5.5.1.1. Thanks For The Memory: Allocation and Deallocation

5.5.1.2. Local Parameters
5.5.1.3. Advantages Of Locals
5.5.1.4. Disadvantages Of Locals
5.5.1.5. Synonyms For “Local”
5.5.1.6. Local Memory Summary
5.5.1.7. How Does The Function Call Stack Work?
5.6. Heap Memory
5.6.1. Heap Memory
5.6.1.1. Allocation
5.6.1.2. Deallocation
5.6.1.3. Programming the Heap
5.6.1.4. A Heap Example
5.6.1.5. Arrays
5.7. Link Nodes
5.7.1. Link Nodes
5.8. Link Nodes Practice Exercises
5.8.1. createList
5.8.2. iterateOverList
5.8.3. changePointeeNext1
5.8.4. addNodeAtHead
5.9. Additional Practice Exercises
5.9.1. changeHeadPointer2
5.9.2. changeQPointer
5.9.3. changeRPointer
5.9.4. changePointeeDataInDirect
5.9.5. circularList2
5.9.6. circularList3
5.9.7. changePointeeNext2
5.9.8. findValueInList
5.9.9. addNodeAtTail
5.9.10. changeNodesOrder1
5.9.11. changeNodesOrder2

Chapter 6 Mathematical Background

6.1. Chapter Introduction
6.2. Sets and Relations
6.2.1. Set Notation
6.2.1.1. Relations
6.2.2. Equivalence Relations
6.2.3. Partial Orders
6.3. Miscellaneous Notation
6.4. Logarithms
6.4.1. Logarithms
6.5. Summations
6.5.1. Summations
6.6. Recurrence Relations

6.6.1. Recurrence Relations
6.7. Mathematical Proof Techniques
6.7.1. Mathematical Proof Techniques
6.7.1.1. Direct Proof
6.7.1.2. Proof by Contradiction
6.7.1.3. Proof by Mathematical Induction
6.8. Estimation
6.9. Chapter Summary Questions
6.9.1. Chapter Summary Questions

Chapter 7 Searching I

7.1. Searching in an Array
7.1.1. Searching in an Array
7.1.1.1. Sequential Search
7.1.1.2. Binary Search

Chapter 8 Algorithm Analysis

8.1. Chapter Introduction
8.2. Problems, Algorithms, and Programs
8.2.1. Problems, Algorithms, and Programs
8.2.1.1. Problems
8.2.1.2. Algorithms
8.2.1.3. Programs
8.2.1.4. Summary
8.2.1.5. Summary Questions
8.3. Comparing Algorithms
8.3.1. Comparing Algorithms
8.3.1.1. Introduction
8.3.1.2. Basic Operations and Input Size
8.3.1.3. Growth Rates
8.3.2. Growth Rates Ordering Exercise
8.4. Best, Worst, and Average Cases
8.4.1. Best, Worst, and Average Cases
8.5. Faster Computer, or Faster Algorithm?
8.5.1. Faster Computer, or Faster Algorithm?
8.6. Asymptotic Analysis and Upper Bounds
8.6.1. Asymptotic Analysis and Upper Bounds
8.6.1.1. Upper Bounds
8.6.1.2. Simplifying Rules
8.6.1.4. Summary
8.6.1.5. Practice Questions
8.7. Lower Bounds and Θ Notation
8.7.1. Lower Bounds and Theta Notation
8.7.1.1. Lower Bounds

8.7.1.2. Theta Notation
8.7.1.3. Classifying Functions
8.7.1.4. Summary Exercise
8.8. Calculating Program Running Time
8.8.1. Calculating Program Running Time
8.8.1.1. Case Study: Two Search Algorithms
8.8.1.2. Binary Search Practice Exercise
8.8.1.3. Analyzing Binary Search
8.8.2. Summary Exercise
8.9. Analyzing Problems
8.9.1. Analyzing Problems
8.10. Common Misunderstandings
8.10.1. Common Misunderstandings
8.11. Multiple Parameters
8.12. Space Bounds
8.13. Code Tuning and Empirical Analysis
8.13.1. Code Tuning and Empirical Analysis
8.13.1.1. Empirical Analysis
8.14. Algorithm Analysis Summary Exercises
8.14.1. Summary Exercise: CS2
8.15. Algorithm Analysis Summary Exercises
8.15.1. Summary Exercise: CS3

Chapter 9 Linear Structures

9.1. Chapter Introduction: Lists
9.2. The List ADT
9.2.1. The List ADT
9.2.1.1. Defining the ADT
9.2.2. List ADT Programming Exercise
9.3. Array-Based List Implementation
9.3.1. Array-Based List Implementation
9.3.1.1. Insert
9.3.1.2. Insert Practice Exericse
9.3.2. Append and Remove
9.3.2.1. Remove Practice Exericise
9.3.3. Array-based List Practice Questions
9.4. Linked Lists
9.4.1. Linked Lists
9.4.1.1. Why This Has Problems
9.4.1.2. A Better Solution
9.4.1.3. Linked List Implementation
9.4.2. Linked List Remove
9.5. Comparison of List Implementations
9.5.1. Space Comparison
9.5.2. Time Comparison
9.5.2.1. Practice Questions

- 9.6. Doubly Linked Lists
  - 9.6.1. Doubly Linked Lists
    - 9.6.1.1. Insert
    - 9.6.1.2. Append
    - 9.6.1.3. Remove
    - 9.6.1.4. Prev
    - 9.6.1.5. Mangling Pointers
- 9.7. List Element Implementations
  - 9.7.1. List Element Implementations
    - 9.7.1.1. Homogeneity
    - 9.7.1.2. Element Deletion
    - 9.7.1.3. Practice Questions
- 9.8. Stacks
  - 9.8.1. Stack Terminology and Implementation
    - 9.8.1.1. Array-Based Stacks
  - 9.8.2. Pop
- 9.9. Linked Stacks
  - 9.9.1. Linked Stack Implementation
    - 9.9.1.1. Linked Stack Push
  - 9.9.2. Linked Stack Pop
    - 9.9.2.1. Comparison of Array-Based and Linked Stacks
- 9.10. Freelists
  - 9.10.1. Freelists
- 9.11. Implementing Recursion
- 9.12. Queues
  - 9.12.1. Queue Terminology and Implementation
    - 9.12.1.1. Array-Based Queues
    - 9.12.1.2. The Circular Queue
    - 9.12.1.3. Array-based Queue Implementation
  - 9.12.2. Array-based Dequeue Practice
- 9.13. Linked Queues
  - 9.13.1. Linked Queues
  - 9.13.2. Linked Dequeue
  - 9.13.3. Comparison of Array-Based and Linked Queues
    - 9.13.3.1. Stack and Queue Summary Questions
- 9.14. Linear Structure Summary Exercises
  - 9.14.1. Practice Questions
  - 9.14.2. Chapter Review Questions

Chapter 10 Recursion

- 10.1. Introduction
  - 10.1.1. Introduction
- 10.2. Writing a recursive function
  - 10.2.1. Writing a recursive function
- 10.3. Code Completion Practice Exercises
  - 10.3.1. Introduction

10.3.2. Recursion Programming Exercise: Largest
10.3.3. Recursion Programming Exercise: Multiply
10.3.4. Recursion Programming Exercise: GCD
10.3.5. Recursion Programming Exercise: log
10.3.6. Recursion Programming Exercise: Cummulative Sum
10.3.7. Recursion Programming Exercise: Add odd values
10.3.8. Recursion Programming Exercise: Sum Of the Digits
10.3.9. Recursion Programming Exercise: Count Characters
10.4. Writing More Sophisticated Recursive Functions
10.5. Harder Code Completion Practice Exercises
10.5.1. Recursion Programming Exercise: Minimum of array
10.5.2. Recursion Programming Exercise: Is Reverse
10.5.3. Recursion Programming Exercise: Decimal to Binary
10.6. Writing Practice Exercises
10.6.1. Recursion Programming Exercise: Cannonballs
10.6.2. Recursion Programming Exercise: Check Palindrome
10.6.3. Recursion Programming Exercise: Subset Sum
10.6.4. Recursion Programming Exercise: Pascal Triangle
10.7. Tracing Recursive Code
10.7.1. Tracing Recursive Code
10.7.1.1. A Domino Analogy
10.7.1.2. Towers of Hanoi
10.8. Tracing Practice Exercises
10.8.1. Forward Flow Tracing Exercises
10.8.2. Backward Flow Tracing Exercises
10.8.3. Find Error Tracing Exercises
10.8.4. Two Recursive Calls Tracing Exercises
10.8.5. How Many Times Tracing Exercises
10.8.6. Harder Tracing Exercises
10.9. Summary Exercises
10.9.1. Summary Questions

Chapter 11 Design II

11.1. Design Patterns
11.1.1. Design Patterns
11.1.1.1. Flyweight
11.1.1.2. Visitor
11.1.1.3. Composite
11.1.1.4. Strategy
11.1.1.5. Summary Questions
11.2. Alternative List ADT Designs
11.3. Comparing Records
11.3.1. Comparing Records
11.4. The Dictionary ADT
11.4.1. The Dictionary ADT

Chapter 12 Binary Trees

- 12.1. Binary Trees Chapter Introduction
- 12.2. Binary Trees
  - 12.2.1. Definitions and Properties
  - 12.2.2. Practice Questions
- 12.3. Binary Tree as a Recursive Data Structure
  - 12.3.1. Binary Tree as a Recursive Data Structure
- 12.4. The Full Binary Tree Theorem
- 12.5. Binary Tree Traversals
  - 12.5.1. Binary Tree Traversals
    - 12.5.1.1. Preorder Traversal
    - 12.5.1.2. Postorder Traversal
    - 12.5.1.3. Inorder Traversal
    - 12.5.1.4. Implementation
  - 12.5.2. Postorder Traversal Practice
  - 12.5.3. Inorder Traversal Practice
  - 12.5.4. Summary Questions
- 12.6. Implementing Tree Traversals
  - 12.6.1. Implementing Tree Traversals
    - 12.6.1.1. Base Case
    - 12.6.1.2. The Recursive Call
  - 12.6.2. Binary Tree Increment By One Exercise
- 12.7. Information Flow in Recursive Functions
  - 12.7.1. Information Flow in Recursive Functions
    - 12.7.1.1. Local
    - 12.7.1.2. Passing Down Information
  - 12.7.2. Binary Tree Set Depth Exercise
  - 12.7.3. Collect-and-return
  - 12.7.4. Binary Tree Check Sum Exercise
  - 12.7.5. Binary Tree Leaf Nodes Count Exercise
  - 12.7.6. Binary Tree Sum Nodes Exercise
  - 12.7.7. Combining Information Flows
  - 12.7.8. Binary Tree Check Value Exercise
  - 12.7.9. Combination Problems
  - 12.7.10. Binary Tree Height Exercise
  - 12.7.11. Binary Tree Get Difference Exercise
  - 12.7.12. Binary Tree Has Path Sum Exercise
- 12.8. Binary Tree Node Implementations
  - 12.8.1. Binary Tree Node Implementations
- 12.9. Composite-based Expression Tree
  - 12.9.1. Composite-based Expression Tree
- 12.10. Binary Tree Space Requirements
  - 12.10.1. Binary Tree Space Requirements
- 12.11. Binary Search Trees
  - 12.11.1. Binary Search Tree Definition
    - 12.11.1.1. BST Search

- 12.11.2. BST Insert
- 12.11.3. BST Remove
- 12.11.4. BST Analysis
- 12.12. Dictionary Implementation Using a BST
- 12.13. Binary Tree Guided Information Flow
  - 12.13.1. Binary Tree Guided Information Flow
  - 12.13.2. Binary Search Tree Small Count Exercise
- 12.14. Multiple Binary Trees
  - 12.14.1. Mirror Image Binary Trees Exercise
  - 12.14.2. Same Binary Tree Exercise
  - 12.14.3. Structurally Identical Binary Trees Exercise
- 12.15. A Hard Information Flow Problem
- 12.16. Array Implementation for Complete Binary Trees
  - 12.16.1. Array Implementation for Complete Binary Trees
- 12.17. Heaps and Priority Queues
  - 12.17.1. Heaps and Priority Queues
  - 12.17.2. Building a Heap
  - 12.17.3. Removing from the heap or updating an object’s priority
  - 12.17.4. Priority Queues
- 12.18. Huffman Coding Trees
  - 12.18.1. Huffman Coding Trees
    - 12.18.1.1. Building Huffman Coding Trees
    - 12.18.1.2. Decoding
    - 12.18.1.3. How efficient is Huffman coding?
- 12.19. Trees versus Tries
  - 12.19.1. Trees versus Tries
- 12.20. Proof of Optimality for Huffman Coding
  - 12.20.1. Proof of Optimality for Huffman Coding
- 12.21. Binary Tree Chapter Summary
  - 12.21.1. Summary Questions

Chapter 13 Sorting

- 13.1. Chapter Introduction: Sorting
- 13.2. Sorting Terminology and Notation
  - 13.2.1. Sorting Terminology and Notation
- 13.3. Insertion Sort
  - 13.3.1. Insertion Sort
  - 13.3.2. Insertion Sort Analysis
- 13.4. Bubble Sort
  - 13.4.1. Bubble Sort
  - 13.4.2. Bubble Sort Analysis
- 13.5. Selection Sort
  - 13.5.1. Selection Sort
  - 13.5.2. Selection Sort Analysis
- 13.6. The Cost of Exchange Sorting
  - 13.6.1. The Cost of Exchange Sorting

- 13.6.2. Analysis
- 13.7. Optimizing Sort Algorithms with Code Tuning
  - 13.7.1. Code Tuning for Simple Sorting Algorithms
- 13.8. Shellsort
  - 13.8.1. Shellsort
  - 13.8.2. Putting It Together
  - 13.8.3. Shellsort Practice Exercise
  - 13.8.4. Optimizing Shellsort
  - 13.8.5. Shellsort Summary Questions
- 13.9. Mergesort Concepts
  - 13.9.1. Mergesort Concepts
  - 13.9.2. Mergesort Practice Exercise
- 13.10. Implementing Mergesort
  - 13.10.1. Implementing Mergesort
- 13.11. Quicksort
  - 13.11.1. Introduction
  - 13.11.2. Partition
  - 13.11.3. Putting It Together
  - 13.11.4. Quicksort Analysis
- 13.12. Heapsort
  - 13.12.1. Heapsort
  - 13.12.2. Heapsort Proficiency Practice
  - 13.12.3. Heapsort Analysis
- 13.13. Binsort
  - 13.13.1. Binsort
- 13.14. Radix Sort
  - 13.14.1. Radix Sort
  - 13.14.2. Array-based Radix Sort
    - 13.14.2.1. Radix Sort Analysis
- 13.15. An Empirical Comparison of Sorting Algorithms
  - 13.15.1. An Empirical Comparison of Sorting Algorithms
- 13.16. Lower Bounds for Sorting
  - 13.16.1. Lower Bounds for Sorting
- 13.17. Sorting Summary Exercises
  - 13.17.1. Sorting Summary Exercises

Chapter 14 File Processing

- 14.1. Chapter Introduction: File Processing
- 14.2. Primary versus Secondary Storage
- 14.3. Disk Drives
  - 14.3.1. Disk Drives
    - 14.3.1.1. Disk Drive Architecture
    - 14.3.1.2. Disk Access Costs
    - 14.3.1.3. Notes
- 14.4. Buffer Pools
  - 14.4.1. Buffer Pools

14.4.1.1. Replacement Strategies
14.4.1.2. The Dirty Bit
14.4.1.3. Implementing Buffer Pools
14.5. The Programmer's View of Files
14.6. External Sorting
14.6.1. External Sorting
14.6.1.1. Simple Approaches to External Sorting
14.6.1.2. Improving Performance
14.6.1.3. Replacement Selection
14.6.2. Multiway Merging
14.6.2.1. Empirical Results
14.6.2.2. Summary

Chapter 15 Hashing

15.1. Introduction
15.1.1. Introduction
15.2. Hash Function Principles
15.2.1. Hash Function Principles
15.3. Sample Hash Functions
15.3.1. Sample Hash Functions
15.3.1.1. Simple Mod Function
15.3.1.2. Binning
15.3.1.3. The Mid-Square Method
15.3.2. A Simple Hash Function for Strings
15.3.3. String Folding
15.3.4. Hash Function Practice
15.3.5. Hash Function Review Questions
15.4. Open Hashing
15.4.1. Open Hashing
15.5. Bucket Hashing
15.5.1. Bucket Hashing
15.5.2. An Alternate Approach
15.6. Collision Resolution
15.6.1. Collision Resolution
15.6.1.1. The Problem with Linear Probing
15.7. Improved Collision Resolution
15.7.1. Linear Probing by Steps
15.7.2. Pseudo-Random Probing
15.7.3. Quadratic Probing
15.7.4. Double Hashing
15.8. Analysis of Closed Hashing
15.8.1. Analysis of Closed Hashing
15.9. Deletion
15.9.1. Deletion
15.9.2. Hashing Deletion Summary Questions
15.10. Hashing Chapter Summary Exercises

15.10.1. Hashing Review

Chapter 16 Memory Management

16.1. Chapter Introduction: Memory Management
16.2. Dynamic Storage Allocation
16.2.1. Dynamic Storage Allocation
16.3. Sequential-Fit Methods
16.3.1. Sequential-Fit Methods
16.4. First Fit
16.4.1. First Fit
16.5. Circular First Fit
16.5.1. Circular First Fit
16.6. Best Fit
16.6.1. Best Fit
16.7. Worst Fit
16.7.1. Worst Fit
16.8. Sequential Fit Peformance
16.8.1. Sequential Fit Peformance
16.9. Other Memory Allocation Methods
16.9.1. Other Memory Allocation Methods
16.9.1.1. Buddy Methods
16.9.1.2. Other Methods
16.10. Failure Policies and Garbage Collection

Chapter 17 Indexing

17.1. Indexing Chapter Introduction
17.2. Linear Indexing
17.2.1. Linear Indexing
17.3. ISAM
17.4. Tree-based Indexing
17.4.1. Tree-based Indexing
17.5. 2-3 Trees
17.5.1. 2-3 Trees
17.6. B-Trees
17.6.1. B-Trees
17.6.1.1. B+ Trees
17.6.1.2. B-Tree Analysis
17.7. Indexing Summary Exercises
17.7.1. Indexing Summary

Chapter 18 General Trees

18.1. General Trees
18.1.1. General Trees

- 18.1.1.1. General Tree Definitions and Terminology
- 18.1.1.2. An ADT for General Tree Nodes
- 18.1.1.3. General Tree Traversals
- 18.2. Union/Find and the Parent Pointer Implementation
  - 18.2.1. The Union/Find Problem
    - 18.2.1.1. Parent Pointer Trees
    - 18.2.1.2. Equivalence Classes
    - 18.2.1.3. Weighted Union
    - 18.2.1.4. Path Compression
- 18.3. Sequential Tree Representations
  - 18.3.1. Sequential Tree Representations
  - 18.3.2. Alternative Sequential Representation
  - 18.3.3. Bit Vector Representation
  - 18.3.4. General Tree Sequential Representation

Chapter 19 Graphs

- 19.1. Graphs Chapter Introduction
  - 19.1.1. Graph Terminology and Implementation
    - 19.1.1.1. Graph Representations
  - 19.1.2. Graph Terminology Questions
- 19.2. Graph Implementations
- 19.3. Graph Traversals
  - 19.3.1. Graph Traversals
    - 19.3.1.1. Depth-First Search
  - 19.3.2. Breadth-First Search
- 19.4. Topological Sort
  - 19.4.1. Topological Sort
    - 19.4.1.1. Depth-first solution
    - 19.4.1.2. Queue-based Solution
- 19.5. Shortest-Paths Problems
  - 19.5.1. Shortest-Paths Problems
    - 19.5.1.1. Single-Source Shortest Paths
- 19.6. Minimal Cost Spanning Trees
  - 19.6.1. Minimal Cost Spanning Trees
    - 19.6.1.1. Prim’s Algorithm
    - 19.6.1.2. Prim’s Algorithm Alternative Implementation
- 19.7. Kruskal’s Algorithm
  - 19.7.1. Kruskal’s Algorithm
- 19.8. All-Pairs Shortest Paths

Chapter 20 Spatial Data Structures

- 20.1. Spatial Data Structures
  - 20.1.1. Spatial Data Structures
    - 20.1.1.1. Multdimensional Keys

20.2. The PR Quadtree
20.2.1. The PR Quadtree
20.3. KD Trees
20.3.1. KD Trees
20.4. The Bintree
20.4.1. The Bintree
20.4.1.1. Implementation Concerns
20.5. Other Spatial Data Structures

Chapter 21 Senior Algorithms Course

21.1. Data and Algorithm Analysis
21.1.1. Introduction
21.1.1.1. Prerequisites
21.1.1.2. What we will do
21.1.1.3. Process
21.2. An Introduction to Problem Solving
21.2.1. An Introduction to Problem Solving
21.2.1.1. Investigation and Argument
21.2.1.2. Heuristics for Problem Solving “In the Small”
21.2.1.3. Problem Solving “In the Large”
21.2.1.4. Pairs Problem Solving
21.2.1.5. Errors in Reasoning
21.2.1.6. References
21.3. Semester Overview
21.3.1. Semester Overview
21.3.1.1. Modeling Algorithm Cost
21.3.1.2. Big Issues
21.4. Introduction to Analyzing a Problem
21.4.1. Towers of Hanoi
21.4.1.1. The Model
21.4.1.2. Finding an Algorithm
21.4.1.3. Analysis of the Algorithm
21.4.1.4. Lower Bound for the Problem
21.4.1.5. New Models
21.4.1.6. Putting it all Together
21.5. Bounds Review
21.5.1. Bounds Review
21.5.1.1. Analyzing Problems
21.5.1.2. Modeling the Inputs
21.6. Growth Rates Review
21.6.1. Growth Rates Review
21.6.1.1. Asymptotic Notation
21.7. Summation Techniques
21.8. Solving Recurrence Relations
21.8.1. Solving Recurrence Relations
21.8.1.1. Estimating Upper and Lower Bounds

21.8.1.2. Expanding Recurrences
21.8.1.3. Divide-and-Conquer Recurrences
21.8.1.4. Average-Case Analysis of Quicksort

Chapter 22 Searching

22.1. Chapter Introduction: Search
22.2. Analyzing Search in Unsorted Lists
22.2.1. Analyzing Search in Unsorted Lists
22.2.1.1. Lower Bounds Proofs
22.3. Search in Sorted Arrays
22.3.1. Analysis
22.3.1.1. Jump Search
22.3.1.2. Binary Search
22.3.1.3. Lower Bounds Proof
22.3.1.4. Interpolation and Quadratic Binary Search
22.4. Self-Organizing Lists
22.4.1. Introduction
22.4.1.1. Frequency Count
22.4.2. Move to Front
22.4.3. Transpose
22.4.3.1. An Example
22.5. Bit Vectors for Representing Sets
22.6. Perfect Hashing
22.6.1. Perfect Hashing

Chapter 23 Lower Bounds

23.1. Finding the Maximum Value
23.1.1. Finding the Maximum Value
23.2. Adversarial Lower Bounds Proofs
23.3. State Space Lower Bounds Proofs
23.4. Finding the i th Best Element
23.5. Optimal Sorting

Chapter 24 Number Problems

24.1. Number Problems
24.1.1. Number Problems
24.1.1.1. Introduction
24.1.1.2. Exponentiation
24.1.1.3. Largest Common Factor
24.2. The Transformation Concept
24.2.1. The Transformation Concept: Integer Multiplication
24.2.2. Polynomial Multiplication
24.3. The Fast Fourier Transform

24.3.1. The Fast Fourier Transform

Chapter 25 Probabilistic Algorithms

25.1. Introduction to Probabilistic Algorithms
25.1.1. Probabilistic Algorithms
25.2. Finding Prime Numbers
25.3. Random Numbers
25.3.1. Random Numbers
25.4. Skip Lists
25.4.1. Skip Lists

Chapter 26 Search Structures

26.1. Balanced Trees
26.2. The AVL Tree
26.3. The Splay Tree
26.4. The Red-Black Tree
26.4.1. Exercise 1
26.4.2. Exercise 2

Chapter 27 Miscellaneous

27.1. The Sparse Matrix
27.2. Dynamic Programming
27.2.1. Dynamic Programming
27.2.2. Computing Fibonacci Numbers
27.2.3. The Knapsack Problem
27.2.4. Chained Matrix Multiplication
27.3. Amortized Analysis
27.4. 0/1 Knapsack Problem
27.4.1. 0/1 Knapsack Problem
27.4.1.1. Exercise 1
27.4.2. Exercise 2
27.4.3. Exercise 3
27.4.4. Exercise 4
27.5. Edit Distance
27.5.1. Edit Distance
27.5.2. Exercise 1
27.5.3. Exercise 2
27.6. KMP String Search Algorithm
27.6.1. KMP String Search Algorithm
27.7. Boyer-Moore String Search Algorithm
27.7.1. Boyer-Moore String Search Algorithm
27.8. Rabin-Karp String Search Algorithm [Draft]
27.8.1. Rabin-Karp String Search Algorithm [Draft]

27.9. General Tree Implementations
27.9.1. General Tree Implementations
27.9.1.1. Introduction
27.9.1.2. List of Children
27.9.1.3. The Left-Child/Right-Sibling Implementation
27.9.1.4. Dynamic Node Implementations
27.9.1.5. Dynamic Left-Child/Right-Sibling Implementation
27.10. K-ary Tree Implementations
27.10.1. K-ary Tree Implementations

Chapter 28 Limits to Computing

28.1. Limits to Computing
28.1.1. Limits to Computing
28.2. Reductions
28.2.1. Reductions
28.2.1.1. Example: The Pairing Problem
28.2.1.2. Reduction and Finding a Lower Bound
28.2.1.3. The Reduction Template
28.2.1.4. Two Multiplication Examples
28.2.1.5. Bounds Theorems
28.2.1.6. The Cost of Making a Simple Polygon
28.3. NP-Completeness
28.3.1. NP-Completeness
28.3.1.1. Hard Problems
28.3.1.2. The Theory of NP-Completeness
28.4. Circuit Satisfiability
28.4.1. Circuit Satisfiability
28.5. Formula Satisfiability
28.5.1. Formula Satisfiability
28.6. 3-CNF Satisfiability
28.6.1. 3-CNF Satisfiability
28.7. The Clique Problem
28.7.1. The Clique Problem
28.8. The Independent Set Problem
28.8.1. Independent Set
28.9. The Vertex Cover Problem
28.9.1. Vertex Cover
28.10. The Hamiltonian Cycle Problem
28.10.1. Hamiltonian Cycle
28.11. The Traveling Salesman Problem
28.11.1. Traveling Salesman
28.12. NP-Completeness Proofs
28.12.1. NP-Completeness Proofs
28.13. Reduction of Circuit SAT to SAT
28.13.1. Reduction of Circuit SAT to SAT
28.14. Reduction of SAT to 3-SAT

- 28.14.1. Reduction of SAT to 3-SAT
- 28.15. Reduction of 3-SAT to Clique
  - 28.15.1. Reduction of 3-SAT to Clique
- 28.16. Reduction of Clique to Independent Set
  - 28.16.1. Clique to Independent Set
- 28.17. Reduction of Independent Set to Vertex Cover
  - 28.17.1. Independent Set to Vertex Cover
- 28.18. Reduction of 3-SAT to Hamiltonian Cycle
  - 28.18.1. 3-SAT to Hamiltonian Cycle
- 28.19. Reduction of Hamiltonian Cycle to Traveling Salesman
  - 28.19.1. Hamiltonian Cycle to Traveling Salesman
- 28.20. Coping with NP-Complete Problems
- 28.21. Unsolveable Problems
  - 28.21.1. Unsolveable Problems
    - 28.21.1.1. Introduction
    - 28.21.1.2. Uncountability
    - 28.21.1.3. The Halting Problem Is Unsolvable
- 28.22. Turing Machines
  - 28.22.1. Turing Machines
    - 28.22.1.1. A General Model of Computation
    - 28.22.1.2. Interpreting Turing Machines
    - 28.22.1.3. Turing-Decideable vs. Turing-Acceptable Languages
    - 28.22.1.4. Making More Complicated Machines
    - 28.22.1.5. Turing Machine Extensions

Chapter 29 Appendix

- 29.1. Glossary
- 29.2. Bibliography

TODO List

Index

Search Page