DATA STRUCTURES AND ALGORITHMS SWE2001

Dr. Ramesh Ragala

February 13, 2022

- **Course Objectives:**
  - To understand the basic concepts of data structures and algorithms in various fields.
  - To learn sorting of and search data items.
  - To comprehend the necessity of time complexity in designing algorithms.
  - To design algorithms to solve real life problems.

- **Expected Course Outcome:** On completion of this course, student should be able to
  - Analyze and understanding stack operations and its applications in real world problems.
  - Understand the pros and cons of various queues and its operations.
  - Demonstrate linear data structures using dynamic arrays.
  - Evaluate algorithms and data structures in terms of time and memory complexity of basic operations
  - Understand, analyze and design sorting and searching algorithms.
  - Understand the importance of hashing.
  - Design non-linear data structure operations in real world problems.
  - Apply suitable data structures and algorithms for autonomous realization of simple programs or program parts.

MODULE - I – STACKS

- Introduction to Stack.
- Operations on Stack.
- Stack implementation using Arrays
- Applications of Stacks:
    - Balance of parenthesis in algebraic expressions
    - Converting expressions from infix to postfix or prefix form
    - Evaluating postfix or prefix form
    - Towers of Hanoi problem

MODULE - II – QUEUE

- Introduction to Queue.
- Operations on Queue.
- Circular Queue
- Queue implementation using Arrays
- Applications of Queue

Syallbus: MODULE - II – LINKED LIST

- Introduction to Linked List
- Single Linked List
- Double Linked List
- Circulat Linked List
- Operations on Linked Lists
- Stack implementation using Linked Lists
- Queue implementation using Linked Lists

Syallbus: MODULE - IV – ALGORITHM ANALYSIS

- Introduction to Algorithms
- Life Cycle of Algorthms
- Performance of Algorithm
  - Time Complexity
  - Space Complexity
- Growth Rate of Functions
- Asymptotic Notations
- Best Case, Average Case and Worest Case Analysis

Syallbus: MODULE - V – SORTING AND SEARCHING

- Sorting Techniques
  - Bubble Sort
  - Insertion Sort
  - Selection Sort
  - Radix Sort
  - Merge Sort
  - Quick Sort
  - Heap Sort
  - Shell Sort
- Searching
  - Linear Search
  - Binary Search
- Analysis of Sorting Techniques
- Analysis of Searching Algorithm

Syallbus: MODULE - VI – HASHING

- Introduction to Hashing
- Hash functions
- Open Hashing or Separate Chaining
- Closed Hashing
    - Linear Probing
    - Quadratic Probing
    - Double Hashing
    - Random Probing
    - Rehashing
    - Extendible Hashing

Syallbus: MODULE - VII – TREES AND GRAPHS

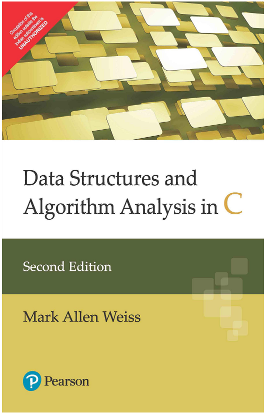
- Introduction to Trees
- Implementation of Tree
- Binary Tree Traversals
- Expression Tree
- Binary Search Tree
- AVL Tree
- Introduction to Graphs
- Graph Traversals
- Shortest Path Algortihm - Dijkstra's Algorithm

Syallbus: MODULE - VIII – CONTEMPORARY ISSUES

- Applications of Data structure in Industry
- Case Studies

Text Book

- Mark Allen Weiss, "Data structures and algorithm analysis in C", 2nd edition, Pearson education, 2013.



Circulation of this edition outside the Indian subcontinent is UNAUTHORIZED

Data Structures and Algorithm Analysis in C

Second Edition

Mark Allen Weiss

Pearson

Figure: Front cover of the book

Reference Book

- Debasis Samanta, "Classic data structures", PHI, 2^{nd} edition, 2014

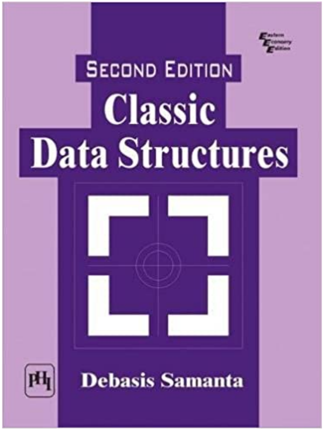
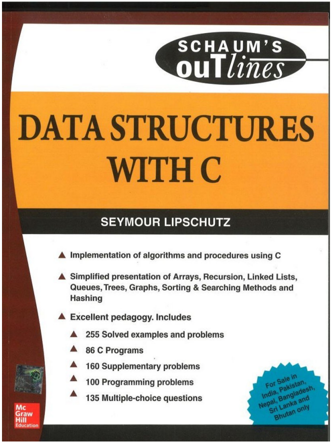


FIGURE: Front cover of the book

Reference Book

- Seymour Lipschutz "Data Structures by Schaum Series" 2^{nd} edition, TMH, 2013.



SCHAUM'S ouTlines

DATA STRUCTURES WITH C

SEYMOUR LIPSCHUTZ

- Implementation of algorithms and procedures using C
- Simplified presentation of Arrays, Recursion, Linked Lists, Queues, Trees, Graphs, Sorting & Searching Methods and Hashing
- Excellent pedagogy. Includes
  - 255 Solved examples and problems
  - 86 C Programs
  - 160 Supplementary problems
  - 100 Programming problems
  - 135 Multiple-choice questions

For Sale in India, Pakistan, Nepal, Bangladesh, Sri Lanka and Bhutan only

Mc Graw Hill Education

Figure: Front cover of the book

Reference Book

- Adam Drozdek, "Data structures and algorithms in C++", Cengage learning, 4^{th} edition, 2015.

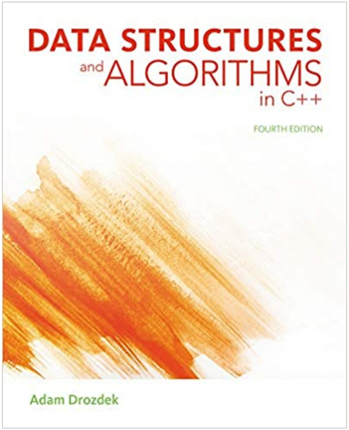


Figure: Front cover of the book

Reference Book

- Michael Goodrich, Roberto Tamassta, Michael H.GoldWasser "Data structures and algorithms in Java" 6^{th} edition, 2014.

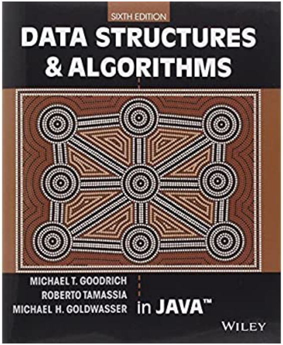


Figure: Front cover of the book

SWE2001 Theory Evaluation Procedur

Assessment	Marks
CAT - 1	15
CAT - 2	15
Quiz - 1	10
Quiz - 2	10
Oral Presentation	10
FAT	40
Total	100

Introduction to Algorithms

In general, An algorithm is a finite set of well-defined instructions that, if followed, accomplish a particular task.

In addition to this definition, all algorithms must follow five basic criteria or rules.

Introduction to Algorithms

In general, An algorithm is a finite set of well-defined instructions that, if followed, accomplish a particular task.
In addition to this definition, all algorithms must follow five basic criteria or rules.

- Input :

Introduction to Algorithms

In general, An algorithm is a finite set of well-defined instructions that, if followed, accomplish a particular task.

In addition to this definition, all algorithms must follow five basic criteria or rules.

- Input :
    - Zero or more quantities are externally supplied to the algorithm and the logic of that algorithm processes those quantities, are called inputs.

Introduction to Algorithms

In general, An algorithm is a finite set of well-defined instructions that, if followed, accomplish a particular task.
In addition to this definition, all algorithms must follow five basic criteria or rules.

- Input :
  - Zero or more quantities are externally supplied to the algorithm and the logic of that algorithm processes those quantities, are called inputs.
- Output:

Introduction to Algorithms

In general, An algorithm is a finite set of well-defined instructions that, if followed, accomplish a particular task.
In addition to this definition, all algorithms must follow five basic criteria or rules.

- Input :
  - Zero or more quantities are externally supplied to the algorithm and the logic of that algorithm processes those quantities, are called inputs.
- Output:
  - At least one quantity is produced after applying the logic on input quantities.
  - These results are called outputs.
  - Every algorithm should have at least one output.

Introduction to Algorithms

In general, An algorithm is a finite set of well-defined instructions that, if followed, accomplish a particular task.

In addition to this definition, all algorithms must follow five basic criteria or rules.

- Input :
  - Zero or more quantities are externally supplied to the algorithm and the logic of that algorithm processes those quantities, are called inputs.
- Output:
  - At least one quantity is produced after applying the logic on input quantities.
  - These results are called outputs.
  - Every algorithm should have at least one output.
- Definiteness :

Introduction to Algorithms

In general, An algorithm is a finite set of well-defined instructions that, if followed, accomplish a particular task.

In addition to this definition, all algorithms must follow five basic criteria or rules.

- Input :
  - Zero or more quantities are externally supplied to the algorithm and the logic of that algorithm processes those quantities, are called inputs.
- Output:
  - At least one quantity is produced after applying the logic on input quantities.
  - These results are called outputs.
  - Every algorithm should have at least one output.
- Definiteness :
  - Every instruction is clear and unambiguous.
  - The steps or instructions, that are used to describe algorithm or logic of algorithm should clearly specifies, what it going to do with out any confusion.

Introduction to Algorithms

In general, An algorithm is a finite set of well-defined instructions that, if followed, accomplish a particular task.
In addition to this definition, all algorithms must follow five basic criteria or rules.

- Input :
  - Zero or more quantities are externally supplied to the algorithm and the logic of that algorithm processes those quantities, are called inputs.
- Output:
  - At least one quantity is produced after applying the logic on input quantities.
  - These results are called outputs.
  - Every algorithm should have at least one output.
- Definiteness :
  - Every instruction is clear and unambiguous.
  - The steps or instructions, that are used to describe algorithm or logic of algorithm should clearly specifies, what it going to do with out any confusion.
  - To Achieve this criteria, algorithms are written in programming languages

Introduction to Algorithms

- Finiteness:

- Finiteness:
  - If we trace out the instructions of an algorithm, thus for all cases, the algorithm terminates after a finite amount of time.
  - i.e, Problem can be solved by algorithmically in a finite amount of time.
  - i.e algorithm can be terminated after finite number of steps.

- Finiteness:
  - If we trace out the instructions of an algorithm, thus for all cases, the algorithm terminates after a finite amount of time.
  - i.e, Problem can be solved by algorithmically in a finite amount of time.
  - i.e algorithm can be terminated after finite number of steps.
- Effectiveness:

Introduction to Algorithms

- Finiteness:
  - If we trace out the instructions of an algorithm, thus for all cases, the algorithm terminates after a finite amount of time.
  - i.e, Problem can be solved by algorithmically in a finite amount of time.
  - i.e algorithm can be terminated after finite number of steps.
- Effectiveness:
  - Every instruction must be very basic, so that it can be carried out, in principle, by a person using only pen and paper.
  - i.e each step of algorithm is simple, easy to understand, and also perform the processing in easy ways by anyone.
  - The operations of algorithm are not only definiteness, but also feasible.

Introduction to Algorithms

- Finiteness:
  - If we trace out the instructions of an algorithm, thus for all cases, the algorithm terminates after a finite amount of time.
  - i.e, Problem can be solved by algorithmically in a finite amount of time.
  - i.e algorithm can be terminated after finite number of steps.
- Effectiveness:
  - Every instruction must be very basic, so that it can be carried out, in principle, by a person using only pen and paper.
  - i.e each step of algorithm is simple, easy to understand, and also perform the processing in easy ways by anyone.
  - The operations of algorithm are not only definiteness, but also feasible.

Computational Procedure: satisfies definiteness and Effectiveness

Example: Operating System of Digital Computer

Definition:

" A data structure is a systematic way of organizing and accessing the data "

Life Cycle of Algorithm

it has four stages:

Life Cycle of Algorithm

it has four stages:

- How to Devise an Algorithm

Life Cycle of Algorithm

it has four stages:

- How to Devise an Algorithm
  - Knowledge on problem specification and user requirement
  - choose good algorithm strategies

Life Cycle of Algorithm

it has four stages:

- How to Devise an Algorithm
  - Knowledge on problem specification and user requirement
  - choose good algorithm strategies
- How to validate an algorithm

Life Cycle of Algorithm

it has four stages:

- How to Devise an Algorithm
  - Knowledge on problem specification and user requirement
  - choose good algorithm strategies
- How to validate an algorithm
  - Algorithm Validation (first Phase)
  - Program Verification or program proving (Second Phase)

Life Cycle of Algorithm

it has four stages:

- How to Devise an Algorithm
    - Knowledge on problem specification and user requirement
    - choose good algorithm strategies
- How to validate an algorithm
    - Algorithm Validation (first Phase)
    - Program Verification or program proving (Second Phase)
- How to analyse an algorithm

Life Cycle of Algorithm

it has four stages:

- How to Devise an Algorithm
  - Knowledge on problem specification and user requirement
  - choose good algorithm strategies
- How to validate an algorithm
  - Algorithm Validation (first Phase)
  - Program Verification or program proving (Second Phase)
- How to analyse an algorithm
  - performance

Life Cycle of Algorithm

it has four stages:

- How to Devise an Algorithm
  - Knowledge on problem specification and user requirement
  - choose good algorithm strategies
- How to validate an algorithm
  - Algorithm Validation (first Phase)
  - Program Verification or program proving (Second Phase)
- How to analyse an algorithm
  - performance
    - Time Complexity
    - Space Complexity

Life Cycle of Algorithm

it has four stages:

- How to Devise an Algorithm
    - Knowledge on problem specification and user requirement
    - choose good algorithm strategies
- How to validate an algorithm
    - Algorithm Validation (first Phase)
    - Program Verification or program proving (Second Phase)
- How to analyse an algorithm
    - performance
        - Time Complexity
        - Space Complexity
- How to test a Program

Life Cycle of Algorithm

it has four stages:

- How to Devise an Algorithm
  - Knowledge on problem specification and user requirement
  - choose good algorithm strategies
- How to validate an algorithm
  - Algorithm Validation (first Phase)
  - Program Verification or program proving (Second Phase)
- How to analyse an algorithm
  - performance
    - Time Complexity
    - Space Complexity
- How to test a Program
  - debugging

Algorithm Specifications

- Distinct Difference between algorithms and programs
- The algorithm is usually described in English language to ensure definiteness condition
- Some Other ways to describe algorithms:

Algorithm Specifications

- Distinct Difference between algorithms and programs
- The algorithm is usually described in English language to ensure definiteness condition
- Some Other ways to describe algorithms:
  - **Flow Charts**
    - It is used to represent the algorithm and algorithm flow control in graphical representation.
    - This method is not efficient and makes more complex for large algorithms.
  - **Pseudo Code**
    - It is a mixture of natural language and high – level programming constructs that describes the main ideas behind a generic implementation of a data structure or algorithm.
    - It is easy to read and understand
    - It should not resemble any particular programming language
    - The pseudo code is more compact than an equivalent actual software code fragment would be.

Pseudo Code Conventions

- Comments are begin with // and continue until the end of the line.
- Compound statement is represented by a block.
  Each block is indicated by matching braces only.
- Every statement is delimited by semicolon (;).
- Assigning a value to a variable done using assignment operator.
  variable := expression or variable
- It uses Boolean values (TRUE and FALSE), Logical Operators (AND, OR and NOT) and Relational Operators like $<$, $>$, $\leq$, $\geq$ and $==$.
- Elements of arrays can be accessed using subscripts braces and subscripts or indices
- READ and WRITE phases are used to specify the input and output of algorithm.

Pseudo Code Conventions

- It also uses break statement and return statement.
  - The break statement is used for force exit from loops.
  - The return statement with value is return from the specified method also exit from function it self.
- It also uses for, while and repeat-until looping statements.
- The while loop form:

```
while (condition) do
{
Statement - 1;
Statement - 2;
.
.
Statement - n;
}
```

Pseudo Code Conventions

- The for loop form:

```
for variable := value-1 to value-2 step STEP do
{
Statement - 1;
Statement - 2;
.
.
Statement - n;
}
```

- The repeat – until loop form:

```
repeat {
Statement - 1;
Statement - 2;
.
```

Pseudo Code Conventions

- It also uses conditional statements like IF-THEN block, IF-THEN-ELSE block, CASE etc.

- IF – THEN block form:

```
IF (condition) THEN
Statements;
```

- IF – THEN – ELSE block form:

```
IF (condition) THEN
Statements;
ELSE
Statements;
```

- CASE statement form:

```
CASE
{
: condition - 1 : statement - 1;
:condition - 2: statement - 2;
.
: condition - n : statement - n;
: Else : statement - n
}
```

Fundamentals of Algorithmic Problem Solving*

- Understanding the Problem
- Ascertaining the Capabilities of a Computational Device
- Choosing between Exact and Approximate Problem Solving
- Deciding on Appropriate Data Structures
- Algorithm Design Techniques
- Methods of Specifying an Algorithm
- Proving an Algorithm's Correctness
- Analyzing an Algorithm
- Coding an Algorithm

Fundamentals of Algorithmic Problem Solving*

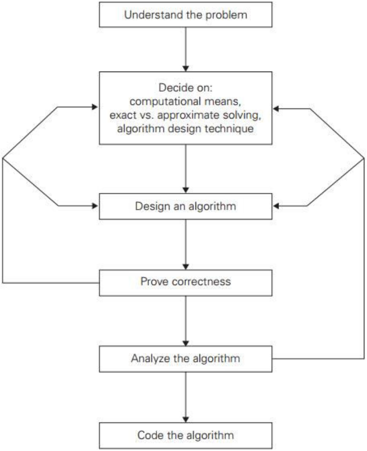


Figure: Algorithm design and analysis process

Performance Analysis

- To classify some good algorithms and data structures → a precise way to analyzing them.
- There are mainly two factors for judging algorithms that have a more direct relationship to performance.
  - Running time of algorithm and data structure operations
  - Space utilization for each operation of an algorithm
- There are two approaches to determine performance of a program.
  - Analytical Method
  - Experimental Method
- The performance evaluation can be loosely divided into major phases
  - Priori Estimate or Apriori Analysis or Perform Analysis
  - Posteriori Testing or Empirical Method or Performance Measurement
- The priori estimates is used to describe the task of estimating the time and space utilization of an algorithm during execution time. → This analysis is done at algorithmic level. → In this model, RAM (Random Access Machine) Model is used.