

# DATA STRUCTURES AND ALGORITHMS

## SWE2001

Dr. Ramesh Ragala

February 24, 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.

- 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

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

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

# SYALLBUS: MODULE - IV – ALGORITHMS ANALYSIS

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



- 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 Algorithm - Dijkstra's Algorithm

# SYALLBUS: MODULE - VIII – CONTEMPORARY ISSUES

- Applications of Data structure in Industry
- Case Studies

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

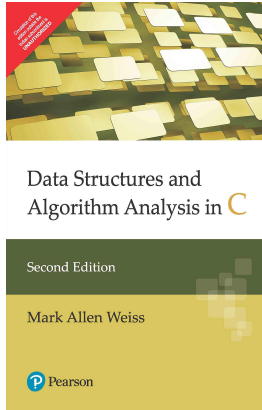


FIGURE: Front cover of the book

- Debasis Samanta, "Classic data structures", PHI, 2<sup>nd</sup> edition, 2014

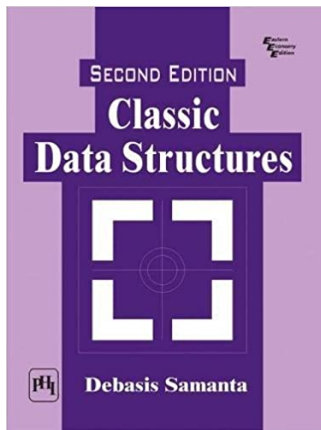


FIGURE: Front cover of the book

- Seymour Lipschutz "Data Structures by Schaum Series" 2<sup>nd</sup> edition, TMH, 2013.

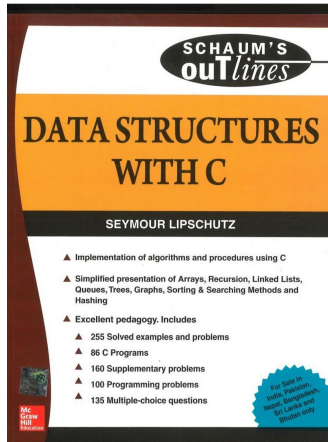
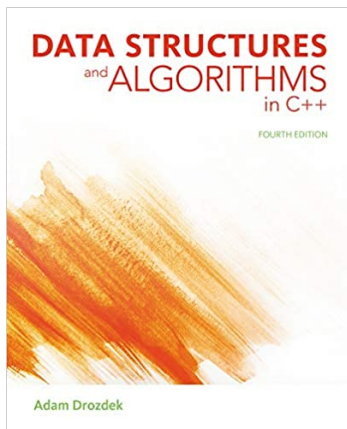


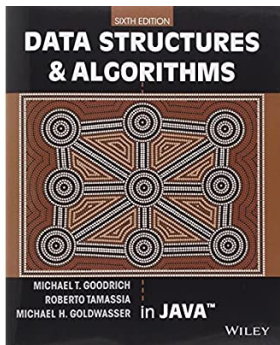
FIGURE: Front cover of the book

- Adam Drozdek, "Data structures and algorithms in C++", Cengage learning, 4<sup>th</sup> edition, 2015.



**FIGURE:** Front cover of the book

- Michael Goodrich, Roberto Tamassia, Michael H. Goldwasser "Data structures and algorithms in Java" 6<sup>th</sup> edition, 2014.



**FIGURE:** Front cover of the book



<b>Assessment</b>	<b>Marks</b>
CAT - 1	15
CAT - 2	15
Quiz - 1	10
Quiz - 2	10
Oral Presentation	10
FAT	40
<b>Total</b>	<b>100</b>

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.

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 :

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.

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:

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.

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 :

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.



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

- 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:

- 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.

- 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 ”

it has four stages:



it has four stages:

- How to Devise an Algorithm

it has four stages:

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

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

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)

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

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

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

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



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

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

- 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.

- 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 `<`, `>`, `≤`, `≥` 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.

- 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;
}
```

- 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;
.
}
```

- 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\*

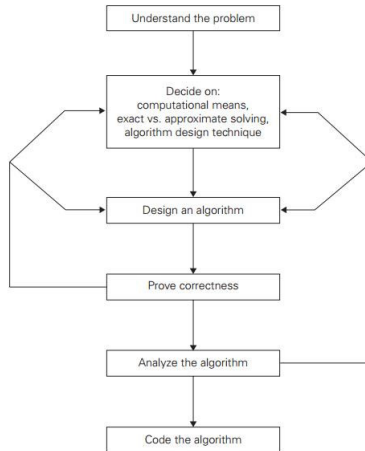


FIGURE: Algorithm design and analysis process