Comprehensive Notes on Data Structures and Algorithms

Your Name

March 20, 2025

Contents

1 Introduction to Data Structures and Algorithms

1.1 Basic Terminology

A good program is defined as one that:

- Runs correctly
- Is easy to read and understand
- Is easy to debug
- Is easy to modify

A program should run efficiently, executing in minimum time and using minimum memory space. Data management concepts are crucial for writing efficient programs.

1.2 Data Structures

A data structure is a group of data elements put together under one name, defining a particular way of storing and organizing data in a computer for efficient use.

1.3 Learning Objectives

This chapter discusses common data structures and algorithms, different approaches to designing algorithms, and notations for evaluating algorithm performance.

2 Introduction to Data Structures and Algorithms

Data structures are used in almost every program or software system. They are widely applied in areas such as compiler design, operating systems, statistical analysis packages, DBMS, numerical analysis, simulation, artificial intelligence, and graphics.

2.1 Elementary Data Structure Organization

Data structures are building blocks of a program. A record is a collection of data items, and a file is a collection of related records. Each record in a file may have a primary key that uniquely identifies it.

3 Classification of Data Structures

Data structures are categorized into primitive and non-primitive data structures. Non-primitive data structures are further classified into linear and non-linear data structures.

3.1 Primitive and Non-primitive Data Structures

Primitive data structures are fundamental data types supported by a programming language (e.g., integer, real, character, boolean). Non-primitive data structures are created using primitive data structures (e.g., linked lists, stacks, trees, graphs).

3.2 Linear and Non-linear Structures

Linear data structures store elements in sequential order (e.g., arrays, linked lists, stacks, queues). Non-linear data structures do not store elements in sequential order (e.g., trees, graphs).

4 Common Data Structures

4.1 Arrays

An array is a collection of similar data elements stored in consecutive memory locations. Arrays are declared using the syntax:

```
type name[size];
```

Arrays have limitations such as fixed size and contiguous memory storage.

4.2 Linked Lists

A linked list is a dynamic data structure where elements (nodes) form a sequential list. Each node contains data and a pointer to the next node. Linked lists allow easy insertion and deletion of elements.

4.3 Stacks

A stack is a linear data structure where insertion and deletion are done at one end (the top). Stacks follow the LIFO principle. Stacks can be implemented using arrays or linked lists.

4.4 Queues

A queue is a linear data structure where elements are added at one end (rear) and removed from the other end (front). Queues follow the FIFO principle. Queues can be implemented using arrays or linked lists.

4.5 Trees

A tree is a non-linear data structure consisting of nodes arranged in a hierarchical order. The simplest form of a tree is a binary tree, which has a root node and left and right subtrees.

4.6 Graphs

A graph is a non-linear data structure consisting of vertices and edges. Graphs represent complex relationships between nodes and are used in various applications like computer networks and social networks.

5 Operations on Data Structures

Common operations on data structures include:

- Traversing: Accessing each data item exactly once.
- Searching: Finding the location of data items that satisfy a given constraint.
- Inserting: Adding new data items to the data structure.
- Deleting: Removing data items from the data structure.
- Sorting: Arranging data items in a specific order.
- Merging: Combining two sorted lists into one sorted list.

6 Abstract Data Type (ADT)

An abstract data type (ADT) defines a data structure by focusing on what it does rather than how it does it. ADTs allow for separation of the use of a data structure from its implementation details.

7 Algorithms

An algorithm is a set of instructions that solve a problem. Algorithms are evaluated based on their time and space complexity. The efficiency of an algorithm is crucial for solving problems within resource constraints.

7.1 Different Approaches to Designing an Algorithm

There are two main approaches to designing algorithms: top-down and bottom-up. The top-down approach starts with a complex algorithm and decomposes it into modules, while the bottom-up approach starts with basic modules and builds up to higher-level modules.

7.2 Control Structures Used in Algorithms

Algorithms use control structures such as sequence, decision, and repetition to manage the flow of execution.

8 Time and Space Complexity

The efficiency of an algorithm is measured in terms of time complexity (running time) and space complexity (memory usage). Big O notation is commonly used to express the upper bound of an algorithm's complexity.

8.1 Big O Notation

Big O notation provides an upper bound for the complexity of an algorithm. It ignores constant factors and focuses on the dominant term in the complexity expression.

8.2 Other Useful Notations

Other notations include Omega () for lower bounds, Theta () for tight bounds, little o for non-asymptotically tight upper bounds, and little omega () for non-asymptotically tight lower bounds.

9 Examples and Exercises

9.1 Example 1: Swapping Two Values

```
void swap(int *a, int *b) {
    int temp = *a;
    *a = *b;
    *b = temp;
}
```

9.2 Example 2: Finding the Larger of Two Numbers

```
int findLarger(int a, int b) {
    if (a > b) return a;
    else return b;
}
```

9.3 Example 3: Checking if a Number is Even or Odd

```
void checkEvenOdd(int num) {
    if (num % 2 == 0) printf("Even\n");
    else printf("Odd\n");
}
```

9.4 Example 4: Printing the First N Natural Numbers

```c
void printNaturalNumbers(int n) {
    for (int i = 1; i <= n; i++) {
        printf("%d ", i);
    }
}
```

9.5 Example 5: Finding the Sum of the First N Natural Numbers

```c
int sumNaturalNumbers(int n) {
    int sum = 0;
    for (int i = 1; i <= n; i++) {
        sum += i;
    }
    return sum;
}
```

9.6 Multiple-choice Questions

1. Which data structure is defined as a collection of similar data elements?
    - (a) Arrays
    - (b) Linked lists
    - (c) Trees
    - (d) Graphs
2. The data structure used in hierarchical data model is
    - (a) Array
    - (b) Linked list
    - (c) Tree
    - (d) Graph
3. In a stack, insertion is done at
    - (a) Top
    - (b) Front
    - (c) Rear
    - (d) Mid
4. The position in a queue from which an element is deleted is called as
    - (a) Top
    - (b) Front

- (c) Rear
- (d) Mid

5. Which data structure has fixed size?
   - (a) Arrays
   - (b) Linked lists
   - (c) Trees
   - (d) Graphs
6. If TOP = MAX–1, then the stack is
   - (a) Empty
   - (b) Full
   - (c) Contains some data
   - (d) None of these
7. Which among the following is a LIFO data structure?
   - (a) Stacks
   - (b) Linked lists
   - (c) Queues
   - (d) Graphs
8. Which data structure is used to represent complex relationships between the nodes?
   - (a) Arrays
   - (b) Linked lists
   - (c) Trees
   - (d) Graphs
9. Examples of linear data structures include
   - (a) Arrays
   - (b) Stacks
   - (c) Queues
   - (d) All of these
10. The running time complexity of a linear time algorithm is given as
    - (a) O(1)
    - (b) O(n)
    - (c) O(n log n)
- (d) $O(n^2)$

11. Which notation provides a strict upper bound for f(n)?

- (a) Omega notation
- (b) Big O notation
- (c) Small o notation
- (d) Theta Notation

12. Which notation comprises a set of all functions h(n) that are greater than or equal to cg(n) for all values of n n0?
    - (a) Omega notation
    - (b) Big O notation
    - (c) Small o notation
    - (d) Theta Notation
13. Function in $o(n^2) notationis$

14. (a) $10n^2 (b) n^1.9$

15. (c) $n^2/100 (d) n^2$

9.7 True or False

16. Trees and graphs are examples of linear data structures. **False**
2. Queue is a FIFO data structure. **True**
3. Trees can represent any kind of complex relationship between the nodes. **False**
4. The average-case running time of an algorithm is an upper bound on the running time for any input. **False**
5. Array is an abstract data type. **False**
6. Array elements are stored in continuous memory locations. **True**
7. The pop operation adds an element to the top of a stack. **False**
8. Graphs have a purely parent-to-child relationship between their nodes. **False**
9. The worst-case running time of an algorithm is a lower bound on the running time for any input. **False**
10. In the top-down approach, we start with designing the most basic or concrete modules and then proceed towards designing higher-level modules. **False**
11. o(g(n)) comprises a set of all functions h(n) that are less than or equal to cg(n) for all values of n n0. **False**
12. Simply means the same as best case . **True**
13. Small omega notation provides an asymptotically tight bound for f(n). **False**
14. Theta notation provides a non-asymptotically tight lower bound for f(n). **False**

15. $n^3.001 (n^3)$.**True**

9.8 Fill in the Blanks

1. _____ is an arrangement of data either in the computer's memory or on the disk storage.
2. _____ are used to manipulate the data contained in various data structures.
3. In _____, the elements of a data structure are stored sequentially.
4. _____ of a variable specifies the set of values that the variable can take.
5. A tree is empty if _____.
6. Abstract means _____.
7. The time complexity of an algorithm is the running time given as a function of _____.
8. _____ analysis guarantees the average performance of each operation in the worst case.
9. The elements of an array are referenced by an _____.
10. _____ is used to store the address of the topmost element of a stack.
11. The _____ operation returns the value of the topmost element of a stack.
12. An overflow occurs when _____.
13. _____ is a FIFO data structure.
14. The elements in a queue are added at _____ and removed from _____.
15. If the elements of a data structure are stored sequentially, then it is a _____.
16. _____ is basically a set of instructions that solve a problem.
17. The number of machine instructions that a program executes during its execution is called its _____.
18. _____ specifies the expected behavior of an algorithm when an input is randomly drawn from a given distribution.
19. The running time complexity of a constant time algorithm is given as _____.
20. A complex algorithm is often divided into smaller units called _____.
21. _____ design approach starts by dividing the complex algorithm into one or more modules.
22. _____ case is when the array is sorted in reverse order.
23. _____ notation provides a tight lower bound for f(n).
24. The small o notation provides a _____ tight upper bound for f(n).
25. $540\text{n}^2 + 10$_____(n^2).