DATA STRUCTURES USING "C"

For

BCA Part-II (Session 2018-21) Students

BY

ANANT KUMAR

MCA, M. Phil, M. Tech.

Faculty Member

Department of Computer Science

J. D. Women's College, Patna

CHAPTER 1 - BASIC CONCEPTS

Introduction to Data Structure

Data structure is a representation of logical relationship existing between individual elements of data. In other words, a data structure defines a way of organizing all data items that considers not only the elements stored but also their relationship to each other. The term data structure is used to describe the way data is stored.

To develop a program of an algorithm we should select an appropriate data. Therefore, data structure is represented as:

Algorithm + Data structure = Program

A data structure is said to be *linear* if its elements form a sequence or a linear list like an array, stacks, queues and linked lists.

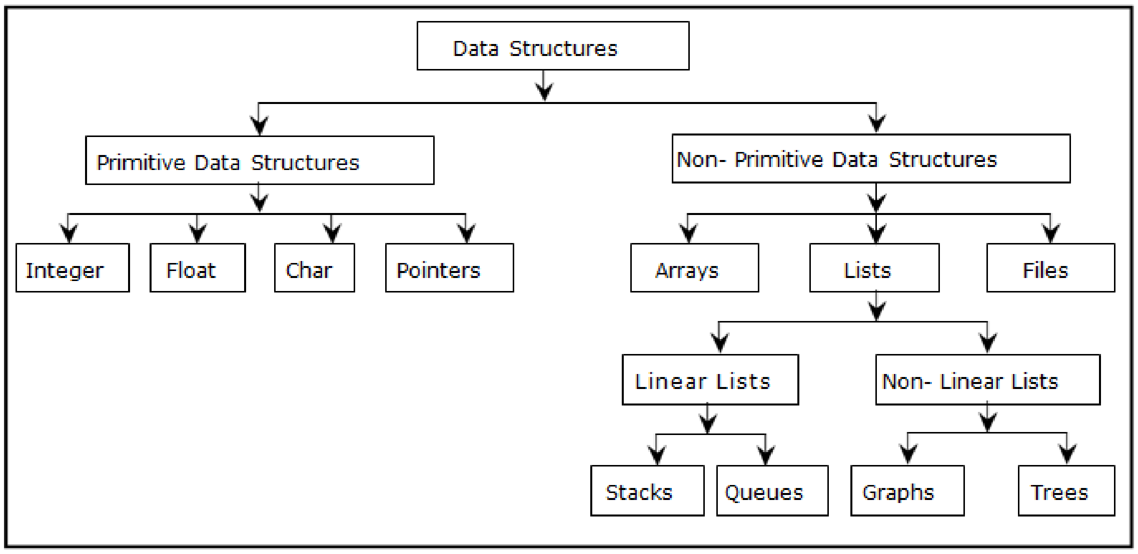
A data structure is said to be *nonlinear* if its elements form a hierarchical classification where, data items appear at various levels like Trees and Graphs.

Data structures are divided into two types:

- Primitive data structures.
- Non-primitive data structures.

Primitive Data Structures are the basic data structures that directly operate upon the machine instructions. They have different representations on different computers. Like Integers, floating point numbers, character constants, string constants and pointers.

Non-primitive data structures are more complicated data structures and are derived from primitive data structures. They emphasize on grouping same or different data items with relationship between each data item. Like Arrays, lists and files.



Classification of Data Structures

Advantages and disadvantages of data structure

Advantages:-

- It allows easier processing of data.
- It allows information stored on disk very efficiently.
- It provides management of databases like indexing with the help of hash tables and arrays.
- We can access data anytime and anywhere.
- It is secure way of storage of data.
- Graphs models real life problems
- It allows processing of data on software system

Disadvantages:-

- It is applicable only for advanced users.
- Slow access in case of some data types

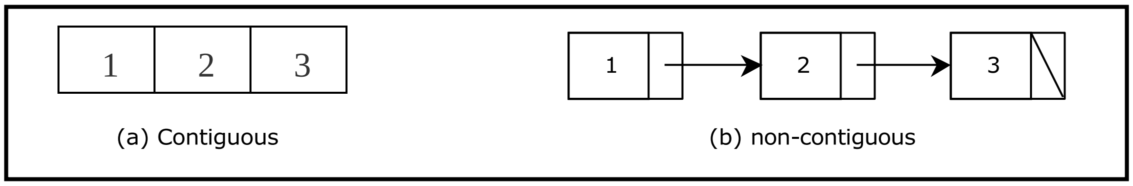
Organization of data

The collections of data you work with in a program have some kind of structure or organization. They can be broken down into two fundamental types:

- Contiguous
- Non-Contiguous.

In contiguous structures, data are kept together in memory (either RAM or in a file). An array is an example of a contiguous structure.

In non-contiguous structure data are scattered in memory, but we linked to each other in some way. A linked list is an example of a non-contiguous data structure.



Contiguous and Non-contiguous structures

Abstract Data Type (ADT)

An abstract data type is a theoretical construct that consists of data as well as the operations to be performed on the data while hiding implementation. Some Examples are Stack, Queue, List etc.

Algorithm

An algorithm is a finite sequence of instructions, each of which has a clear meaning and can be performed with a finite amount of effort in a finite length of time. An algorithm terminates after executing a finite number of instructions. Every algorithm must satisfy the following criteria:

Input: there are zero or more quantities, which are externally supplied.

Output: at least one quantity is produced.

Definiteness: each instruction must be clear and unambiguous.

Finiteness: if we trace out the instructions of an algorithm, then for all cases the algorithm will terminate after a finite number of steps.

Effectiveness: every instruction must be sufficiently basic that it can in principle be carried out by a person using only pencil and paper. It is not enough that each operation be definite, but it must also be feasible.

Algorithm design issues:

Choosing an efficient algorithm or data structure is just one part of the design process. Next, will look at some design issues that are broader in scope. There are three basic design goals that we should use in a program:

- Try to save time (Time complexity).
- Try to save space (Space complexity).
- Try to have face.

A program that runs faster is a better program, so saving time is an obvious goal. Likewise, a program that saves space over a competing program is considered desirable. We want to “save face” by preventing the program from locking up or generating amounts of garbled data.

Performance of a program:

The performance of a program is the amount of computer memory and time needed to run a program. We use two approaches to determine the performance of a program. One is analytical, and the other experimental. In performance analysis we use analytical methods, while in performance measurement we conduct experiments.

Time Complexity:

The time needed by an algorithm expressed as a function of the size of a problem is called the time complexity of the algorithm. The time complexity of a program is the amount of computer time it needs to run to completion.

Space Complexity:

The space complexity of a program is the amount of memory it needs to run to completion. The space need by a program has the following components:

Instruction space: Instruction space is the space needed to store the compiled version of the program instructions.

Data space: Data space is the space needed to store all constant and variable values. Data space has two components:

- Space needed by constants and simple variables in program.
- Space needed by dynamically allocated objects such as arrays and class instances.

Environment stack space: The environment stack is used to save information needed to resume execution of partially completed functions.

Instruction Space: The amount of instructions space that is needed depends on factors such as:

- The compiler used to complete the program into machine code.
- The compiler options in effect at the time of compilation
- The target computer.

Classification of Algorithms

If ‘n’ is the number of data items to be processed or degree of polynomial or the size of the file to be sorted or searched or the number of nodes in a graph etc.

1 Next instructions of most programs are executed once or at most only a few times. If all the instructions of a program have this property, we say that its running time is a constant.

log n When the running time of a program is logarithmic, the program gets slightly slower as n grows. This running time commonly occurs in programs that solve a big problem by transforming it into a smaller problem, cutting the size by some constant fraction., When n is a million, log n is a doubled whenever n doubles, log n increases by a constant, but log n does not double until n increases to n2.

n	When the running time of a program is linear, it is generally the case that a small amount of processing is done on each input element. This is the optimal situation for an algorithm that must process n inputs.
$n. \log n$	This running time arises for algorithms but solve a problem by breaking it up into smaller sub-problems, solving them independently, and then combining the solutions. When n doubles, the running time more than doubles.
n^2	When the running time of an algorithm is quadratic, it is practical for use only on relatively small problems. Quadratic running times typically arise in algorithms that process all pairs of data items (perhaps in a double nested loop) whenever n doubles, the running time increases fourfold.
n^3	Similarly, an algorithm that process triples of data items (perhaps in a triple–nested loop) has a cubic running time and is practical for use only on small problems. Whenever n doubles, the running time increases eightfold.
2^n	Few algorithms with exponential running time are likely to be appropriate for practical use, such algorithms arise naturally as "brute–force" solutions to problems.

Complexity of Algorithms

The complexity of an algorithm M is the function f(n) which gives the running time and/or storage space requirement of the algorithm in terms of the size 'n' of the input data. Mostly, the storage space required by an algorithm is simply a multiple of the data size 'n'. Complexity shall refer to the running time of the algorithm.

The function f(n), gives the running time of an algorithm, depends not only on the size 'n' of the input data but also on the particular data. The complexity function f(n) for certain cases are:

1.	Best Case	:	The minimum possible value of f(n) is called the best case.
2.	Average Case	:	The expected value of f(n).
3.	Worst Case	:	The maximum value of f(n) for any key possible input.

Rate of Growth

Big–Oh (O), Big–Omega (□), Big–Theta (□) and Little–Oh

1. T(n) = O(f(n)), (pronounced order of or big oh), says that the growth rate of T(n) is less than or equal (<) that of f(n)

2. T(n) = □(g(n)) (pronounced omega), says that the growth rate of T(n) is greater than or equal to (>) that of g(n)

3. T(n) = □(h(n)) (pronounced theta), says that the growth rate of T(n) equals (=) the growth rate of h(n) [if T(n) = O(h(n)) and T(n) = □(h(n)]

4. T(n) = o(p(n)) (pronounced little oh), says that the growth rate of T(n) is less than the growth rate of p(n) [if T(n) = O(p(n)) and T(n) □□(p(n))].

Analyzing Algorithms

Suppose 'M' is an algorithm, and suppose 'n' is the size of the input data. Clearly the complexity f(n) of M increases as n increases. It is usually the rate of increase of f(n) we want to examine. This is usually done by comparing f(n) with some standard functions. The most common computing times are:

O(1), O(log2n), O(n), O(n. log2n), O(n2), O(n3), O(2n), n! and nn

Numerical Comparison of Different Algorithms

The execution time for six of the typical functions is given below:

S. No.	log n	n	n. log n	n2	n3	2n
1	0	1	1	1	1	2
2	1	2	2	4	8	4
3	2	4	8	16	64	16
4	3	8	24	64	512	256
5	4	16	64	256	4096	65536

Exercises

1. Define algorithm.
2. Define efficiency of an algorithm?
3. State the various methods to estimate the efficiency of an algorithm.
4. Define time complexity of an algorithm?
5. Define worst case, average case, best case of an algorithm.
6. Mention the various spaces utilized by a program.
7. Define space complexity of an algorithm.

Multiple Choice Questions

1. is a step-by-step recipe for solving an instance of problem. [A]
   A. Algorithm B. Complexity
   C. Pseudocode D. Analysis

2. is used to describe the algorithm, in less formal language. [C]
   A. Cannot be defined B. Natural Language
   C. Pseudocode D. None

3. is used to define the worst-case running time of an algorithm. [A]
   A. Big-Oh notation B. Cannot be defined
   C. Complexity D. Analysis

4. of an algorithm is the amount of time (or the number of steps) needed by a program to complete its task. [D]
   A. Space Complexity B. Dynamic Programming
   C. Divide and Conquer D. Time Complexity

5. of a program is the amount of memory used at once by the algorithm until it completes its execution. [C]
   A. Divide and Conquer B. Time Complexity
   C. Space Complexity D. Dynamic Programming

*****