Classification of Data Structure: A Review

Anjana Singh, Assistant Professor in Computer Science,
AIJHM College, Rohtak, Anjanatalan81@gmail.com

Abstract

Data has always been and is becoming a resource that needs to be judiciously used and shared for the benefit of the organizations and institutions. More than ever, there is a massive data sharing these days with the increasing technological updates and social networking sites. The processed data is called as information and the task of finding them from the existing repository are called as information retrieval. Information retrieval is an area of research actively involving multiple domains of computer science. Information retrieval exploits many of the fundamental concepts of computer science and is used as a tool in various advanced research areas of computer science. For an example, the field of text mining uses information retrieval as a first step for before applying other mining operations.

Keywords: Data Structures, Information Retrieval, Multiple Domains, Organizations etc.

Introduction

Data may contain a single element or sometimes it may be a set of elements. Whether it is a single element or multiple elements but it must be organized in a particular way in the computer memory system. In this article, we are going to discuss some very important terms like what is data structure & classification of data structure – Primitive Data Structure, Non-Primitive Data Structure etc.

The data structure is basically a technique of organizing and storing of different types of data items in computer memory. It is considered as not only the storing of data elements but also the maintaining of the logical relationship existing between individual data elements.

The Data structure can also be defined as a mathematical or logical model, which relates to a particular organization of different data elements.

Specification of data structure:

- Organization of data
- Accessing methods
- Degree of associatively
- Processing alternatives for information

Data structures are considered as the main building blocks of a computer program. So, at the time of selection of data structure, we should follow these two things so that our selection is efficient enough to solve our problem.

1. The data structure must be powerful enough to handle the different relationship existing between the data.
2. The structure of data also to be simple, so that we can efficiently process data when required.

Basically, the term data structure and algorithm both are somehow related to each other. Set of a well-defined algorithm and data structure makes a computer program efficient.

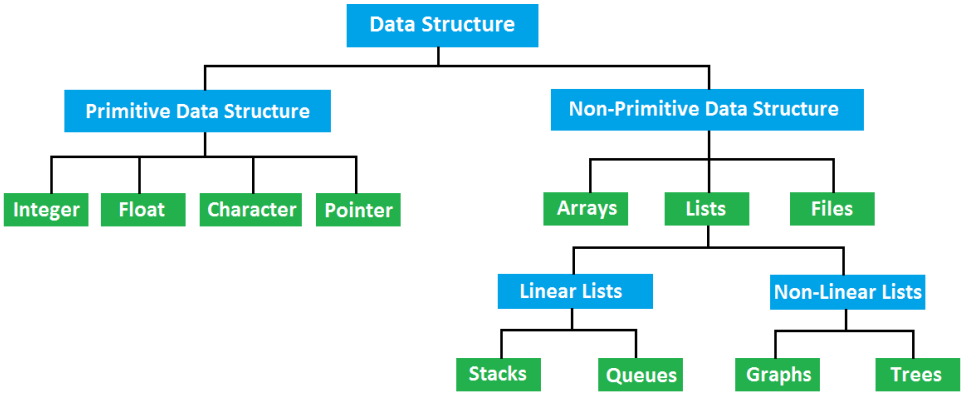
Algorithm + Data Structure = Program

A program can be made using the different data structure to produce the same output. But the efficiency of the program will depend on the choice of the best data structure to create that particular program. We have a classification of data structures to choose to make this happen.

Classification of data structure:

The classification of data structure mainly consists of:

1. Primitive data structure
2. Non-primitive data structure



Primitive data structure:

The primitive data structures are known as basic data structures. These data structures are directly operated upon by the machine instructions. Normally, primitive data structures have different representation on different computers.

Example of primitive data structure:

- Integer
- Float
- Character
- Pointer

Integer:

The integers are signed or unsigned whole numbers with the specified range such as 5, 39, -1917, 0 etc. They have no fractional parts. Integers can be positive or negative but whether or not they can have negative values, it depends upon the integer types.

Float:

Float refers floating point or real number. It can hold a real number or a number having a fractional part like 3.112 or 588.001 etc. The decimal point signals that it is a floating point number, not an integer. The number 15 is an integer but 15.0 is a floating point number.

Character:

It can store any member of the basic character set. If a character from this set is stored in a character variable, its value is equivalent to the integer code of that character basically known as ASCII code. It can hold one letter/symbol like a, B, d etc. Characters can also be of different types.

Pointer:

A pointer is but a variable-like name points or represents a storage location in memory (RAM). RAM contains many cells to store values. Each cell in memory is 1 byte and has a unique address to identify it. The memory address is always an unsigned integer.

Non-Primitive data structure:

The non-primitive data structures are highly developed complex data structures. Basically, these are developed from the primitive data structure. The non-primitive data structure is responsible for organizing the group of homogeneous and heterogeneous data elements.

Example of Non-primitive data structure:

- Arrays
- Lists
- Files

Arrays:

Arrays are the set of homogeneous data elements stored in RAM. So, they can hold only one type of data. The data may be all integers, all floating numbers or all characters. Values in an array are identified using array name with subscripts. Single sub-scripted variables are known as a one-dimensional array or linear array; two sub-scripted variables are referred as a two-dimensional array.

Lists:

A list is a collection of a variable number of data items. Lists fall in the non-primitive type of data structure in the classification of data structure. Every element on a list contains at least two fields, one is used to store data and the other one is used for storing the address of next element.

Files:

Files contain data or information, stored permanently in the secondary storage device such as Hard Disk and Floppy Disk. It is useful when we have to store and process a large amount of data. A file stored in a storage device is always identified using a file name like HELLO.DAT or TEXTNAME.TXT and so on. A file name normally contains a primary and a secondary name which is separated by a dot (.).

Stack:

Like arrays, a stack is also defined as an ordered collection of elements. A stack is a non-primitive linear data structure having a special feature that we can delete and insert elements from only one end, referred as TOP of the stack. The stack is also known as Last In First Out (LIFO) type of data structure for this behaviour.

When we perform insertion or deletion operation on a stack, its base remains unchanged but the top of the stack changes. Insertion in a stack is called Push and deletion of elements from the stack is known as Pop.

We can implement a stack using 2 ways:

- Static implementation (using arrays)
- Dynamic implementation (using pointers)

Queues:

Queues are also non-primitive linear data structure. But unlike stacks, queues are the First in First out (FIFO) type of data structures. We can insert an element in a queue from the REAR end but we have to remove an element from the only FRONT end.

We can also implement queues using 2 ways:

- Using arrays
- Using pointers

Trees:

Trees fall into the category of non-primitive non-linear data structures in the classification of data structure. They contain a finite set of data items referred as nodes. We can represent a hierarchical relationship between the data elements using trees.

A Tree has the following characteristics:

- The top item in a hierarchy of a tree is referred as the root of the tree.
- The remaining data elements are partitioned into a number of mutually exclusive subsets and they itself a tree and are known as the sub tree.
- Unlike natural trees, trees in the data structure always grow in length towards the bottom.

Graph:

Graph falls in the non-primitive non-linear type of data structure in the classification of data structure. Graphs are capable of representing different types of physical structures. Apart from computer science, they are used broadly in the fields of Geography, Chemistry & Engineering Sciences.

Conclusion:

The term data structure is used to describe the way data is stored, and the term algorithm is used to describe the way data is processed. Data structures and algorithms are interrelated. Choosing a data structure affects the kind of algorithm you might use, and choosing an algorithm affects the data structures we use. 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. No matter what the input values may be, an algorithm terminates after executing a finite number of instructions.

Bibliography:

[1] S. Ceri et al., Web Information Retrieval, Data-Centric Systems and Applications, DOI 10.1007/978-3-642-39314-3_2, © Springer Verlag Berlin Heidelberg 2013

[2] Fei Song, W Bruce Croft. A general language model for information retrieval. Proceedings of the eighth international conference on Information and knowledge management (ACM) 1999/11/1.pp316-321

[3] Falley. P “Categories of Data Structures”, Journal of Computing Sciences in Colleges - Papers of the Fourteenth Annual CCSC Midwestern Conference and Papers of the Sixteenth Annual CCSC Rocky Mountain Conference. Volume 23 Issue 1, October 2007. PP. 147-153, 2007-10-01

[4] B. Zhou and Y. Yao Evaluating information retrieval system performance based on user preference JIIS, 34:227–248, 2010

[5] Rudolf Bayer and Karl Unterauer, "Prefix B Trees" ACM Transactions on Database Systems, Vol. 2, No. 1, March 1977, Pages 11-26.

[6] Morin, Patrick. "Data Structures for Strings", chapter 7, March 2012.

[7] Pogue, C. & Willett, P. (1987). Use of Text Signatures for Document Retrieval in a Highly Parallel Environment. Parallel Computing, 4, 259-268.

[8] Nicholajs. B Elkin, W.B Rucec Roft,. Retrieval Techniques, Annual Review of Information Science and Technology, Volume 22. 1987. Martha E. Williams, Editor Published for the American Society for Information Science (ASIS) bv Elsevier Science Publishers.

[9] Tao Qin, Tie-Yan Liu, Xu-Dong Zhang, Zheng Chen, and WeiYing Ma. A study of relevance propagation for web search. In SIGIR '05: Proceedings of the 28th annual international ACM SIGIR conference on Research and development in information retrieval, pages 408–415, New York, NY, USA, 2005. ACMPress.