Introduction to Basic Data Structures and Algorithms

- Introduction to Basic Data Structures and Algorithms
- Classification of Data Structures
  - Arrays
  - Linked List
  - Stack
  - Queue
  - Tree
  - Heap
  - Dictionary
  - Hash Table
  - Graph
- References

Introduction to Basic Data Structures and Algorithms

Before introducing data structures we should understand that computers do store, retrieve, and process a large amount of data. If the data is stored in well organized way on storage media and in computer's memory then it can be accessed quickly for processing that further reduces the latency and the user is provided fast response.

Data structure introduction refers to a scheme for organizing data, or in other words a data structure is an arrangement of data in computer's memory in such a way that it could make the data quickly available to the processor for required calculations. A data structure should be seen as a logical concept that must address two

fundamental concerns. First, how the data will be stored, and second, what operations will be performed on it? As data structure is a scheme for data organization so the functional definition of a data structure should be independent of its implementation. The functional definition of a data structure is known as ADT (Abstract Data Type) which is independent of implementation. The implementation part is left on developers who decide which technology better suits to their project needs.

For example, a stack ADT is a structure which supports operations such as push and pop. A stack can be implemented in a number of ways, for example using an array or using a linked list.

Along with data structures introduction, in real life, problem solving is done with help of data structures and algorithms. An algorithm is a step by step process to solve a problem. In programming, algorithms are implemented in form of methods or functions or routines. To get a problem solved we not only want algorithm but also an efficient algorithm. One criteria of efficiency is time taken by the algorithm, another could be the memory it takes at run time.

Sometimes, we can have more than one algorithm for the same problem to process a data structure, and we have to choose the best one among available algorithms. This is done by algorithm analysis. The best algorithm is the one which has a fine balance between time taken and memory consumption. But, as we know the best exists rarely, and we generally give more priority to the time taken by the algorithm rather than the memory it consumes. Also, as memory is getting cheaper and computers have more memory today than previous time, therefore, run time analysis becomes more significant than memory. The analysis of algorithms is an entirely separate topic and we will discuss that separately.

Classification Data Structures

Data structures can be broadly classified in two categories - *linear structures* and *hierarchical structures*. Arrays, linked lists, stacks, and queues are linear structures, while trees, graphs, heaps etc. are hierarchical structures.

Every data structure has its own strengths, and weaknesses. Also, every data structure specially suits to specific problem types depending upon the operations performed and the data organization. For example, an array is suitable for read operations. Following is a quick introduction to important data structures.

Arrays

Arrays are statically implemented data structures by some programming languages

like C and C++; hence the size of this data structure must be known at compile time and cannot be altered at run time. But modern programming languages, for example, Java implements arrays as objects and give the programmer a way to alter the size of them at run time. Arrays are the most common data structure used to store data.

Arrays are unarguably easier data structures to use and access. But inserting an item to an array and deleting it from the array are situation dependent. If you want to insert an item at a particular position which is already occupied by some element then you have to shift all items one position right from the position new element has to be inserted then insert the new item. The time taken by insert operation is depend on how big the array is, and at which position the item is being inserted. The same is true about deletion of an item.

If the array is unsorted then search operation is also proved costly and takes O(N) time in worst case, where N is size of the array. But if the array is sorted then search performance is improved magically and takes O(logN) time in worst case.

Linked List

Linked list data structure provides better memory management than arrays. Because linked list is allocated memory at run time, so, there is no waste of memory. Performance wise linked list is slower than array because there is no direct access to linked list elements.

Linked list is proved to be a useful data structure when the number of elements to be stored is not known ahead of time.

There are many flavors of linked list you will see: linear, circular, doubly, and doubly circular.

Stack

Stack is a last-in-first-out strategy data structure; this means that the element stored in last will be removed first. Stack has specific but very useful applications; some of them are as follows:

- Solving Recursion - recursive calls are placed onto a stack, and removed from there once they are processed.
- Evaluating post-fix expressions
- Solving Towers of Hanoi

- Backtracking
- Depth-first search
- Converting a decimal number into a binary number

Queue

Queue is a first-in-first-out data structure. The element that is added to the queue data structure first, will be removed from the queue first. Dequeue, priority queue, and circular queue are the variants of queue data structure. Queue has the following application uses:

- Access to shared resources (e.g., printer)
- Multiprogramming
- Message queue

Trees

Tree is a hierarchical data structure. The very top element of a tree is called the root of the tree. Except the root element every element in a tree has a parent element, and zero or more children elements. All elements in the left sub-tree come before the root in sorting order, and all those in the right sub-tree come after the root.

Tree is the most useful data structure when you have hierarchical information to store. For example, directory structure of a file system; there are many variants of tree you will come across. Some of them are Red-black tree, threaded binary tree, AVL tree, etc.

Heap

Heap is a binary tree that stores a collection of keys by satisfying heap property. Max heap and min heap are two flavors of heap data structure. The heap property for max heap is: each node should be greater than or equal to each of its children. While, for min heap it is: each node should be smaller than or equal to each of its children. Heap data structure is usually used to implement priority queues.

Dictionary

Dictionary is a data structure that maintains a set of items indexed on basis of keys. Dictionary stores data in form of key-element pairs.

Hash Table

Hash Table is again a data structure that stores data in form of key-element pairs. A key is a non-null value which is mapped to an element. And, the element is accessed on the basis of the key associated with it. Hash table is a useful data structure for implementing dictionary.

Graph

Graph is a networked data structure that connects a collection of nodes called vertices, by connections, called edges. An edge can be seen as a path or communication link between two nodes. These edges can be either directed or undirected. If a path is directed then you can move in one direction only, while in an undirected path the movement is possible in both directions.

Last Word

In this tutorial we saw a brief introduction of various important data structures. We also talked of what data structures are, and why data structures so much important for information systems. Data structures along with algorithms is a core subject of computer science. Hope you have enjoyed reading this tutorial. Please do write us if you have any suggestion/comment or come across any error on this page. Thanks for reading!

References

1. Algorithm Design [Foundations, Analysis, and Internet Examples]
2. Algorithms, 4th Edition
3. Handbook of Data Structures and Applications

About the Author

Krishan Kumar is the main author for cs-fundamentals.com. He is a software professional (post graduated from BITS-Pilani) and loves writing technical articles on programming and data structures.

Today's Tech News

'Safe' search engine blocks common words

Posted on Tuesday March 01, 2016

A search engine aimed at children, which blocks many common search terms like menstruation and balls, has gone viral.

First look at virtual reality Minecraft

Posted on Tuesday March 01, 2016

Microsoft has shown off how the immersive world of Minecraft will look like in virtual reality.

Software export rules 'to be rewritten'

Posted on Tuesday March 01, 2016

An arms-control deal that restricts the export of hacking tools is set to be renegotiated by American authorities, a US congressman says.

Courtesy BBC News