**Data Structures and Algorithms**

martes, 28 de marzo de 2023

6:23 p. m.

Source: <https://www.programiz.com/dsa>

Source: [Data Structures Illustrated](https://www.youtube.com/playlist?list=PLkZYeFmDuaN2-KUIv-mvbjfKszIGJ4FaY)

DSA (Data Structure and Algorithms)

Data structures and Data type are slightly different. Data structures are collections of data types arranged in a specific order.

There are two categories of Data Structures: Linear and Non-linear.

Linear

Collections where data is stored one after the other.

Popular ones are:

* Arrays: Elements arranged in a continuous memory, all element are the same type.
* Stack: Elements stored in LIFO
* Queue: Elements store in FIFO
* Linked List: Elements are connected through a series of nodes. And, each node contains the data items and address to the next node

Non-Linear

Collections where data is stored in no particular sequence but they are arranged in a hierarchical manner where one element will be connected to one or more elements.

The two main categories of non-linear data structures are graphs and tree based data structures:

Graphs DS

Within this category, each node is called a 'vertex' and each one is connected to other vertices through edges.

Most popular Graph Based DS are

* Spanning Tree and Minimum Spanning Tree
* Strongly Connected Components
* Adjacency Matrix
* Adjacency List

Trees DS

On the other hand, a Tree is a collection of vertices and edges. However, in this type of DS there can only be one edge between two vertices.

Most popular Tree Based DS are

* Binary Tree
* Binary Search Tree
* AVL Tree
* B-Tree
* B+ Tree
* Read-black Tree

*Comparative*

|  |  |
| --- | --- |
| **Linear Data Structures** | **Non Linear Data Structures** |
| * + The data items are arranged in sequential order, one after the other. | * + The data items are arranged in non-sequential order (hierarchical manner). |
| * + All the items are present on the single layer. | * + The data items are present at different layers. |
| * + It can be traversed on a single run. That is, if we start from the first element, we can traverse all the elements sequentially in a single pass. | * + It requires multiple runs. That is, if we start from the first element it might not be possible to traverse all the elements in a single pass. |
| * + The memory utilization is not efficient. | * + Different structures utilize memory in different efficient ways depending on the need. |
| * + The time complexity increase with the data size. | * + Time complexity remains the same. |
| * + Example: Arrays, Stack, Queue | * + Example: Tree, Graph, Map |

On Algorithms

Based on the notion that algorithm are simple a set of order determined instructions to solve problems, the difference between one to another is how efficiently they solve certain problem.

The two variables considered to asses how efficient is an algorithm are time and memory.

And when a algorithm must require more time depending on how large is the size of the input is called a Linearly scalable algorithm, but on the other hand if the problem is solved no matter the size of the input it will then known as a constant-time algorithm.

Asymptotic Analysis

The study of change in performance of the algorithm with the change in the order of the input size is defined as asymptotic analysis.

This video explains pretty well the asymptotic analysis. - [Asymptotic Bounding 101: Big O, Big Omega, & Theta (Deeply Understanding Asymptotic Analysis)](https://www.youtube.com/watch?v=0oDAlMwTrLo)

Asymptotic 
Bounds 

**Asymptotic Notation**s

This are mathematical notations used to describe the running time of an algorithm when the input tends towards a particular value or a limiting value.

For instance, the Bubble sort algorithm works its fastest when the array to be sorted is already sorted, so the time taken by the algorithm is known as 'Linear', which is the best case,

But in the other hand, when the input is in the opposite order, the algorithm works its slowest and takes maximum time and is know as 'Quadratic', this will be the worst case.

And between this two limit cases is when the input is neither sorted or reversed, then it will take average time to be solved. And this durations are denoted using asymptotic notations.

The main use of this notations is to fairly compare complexity and performance of different algorithms and choose.

There are mainly 3 asymptotic notations:

* Big-O: worst case performance.
* Omega notation: best case performance.
* Theta notation: average case performance.

Master Theorem

There is a theorem that helps to understand the complexity of divide-and-conquer algorithms according to the relation between some constants of its recurrence relations.

This tutorial illustrates the case - [What is the Master Theorem?](https://www.youtube.com/watch?v=2H0GKdrIowU)

Master Theorem 
If T (n) = + O(nd) 
I.d z O). them 
if d > 

Arrays and Linked Lists 
Heaps, Stacks and Queues 
"ash Tables 
Binary Search Trees 
Recursion 
Sorting Algorithms 

Data Structures top by usage among industries

