Data Structure

**Data structures**are essential components that help organize and store data efficiently in computer memory.

**Why are Data Structures Important?**

Data structures are essential for the following reasons:

* **Efficient Data Management:** They enable efficient storage and retrieval of data, reducing processing time and improving performance.
* **Data Organization:**They organize data in a logical manner, making it easier to understand and access.
* **Data Abstraction:**They hide the implementation details of data storage, allowing programmers to focus on the logical aspects of data manipulation.
* **Reusability:**Common data structures can be reused in multiple applications, saving time and effort in development.
* **Algorithm Optimization:** The choice of the appropriate data structure can significantly impact the efficiency of algorithms that operate on the data.

**Classification of Data Structures**

Data structures can be classified into two main categories:

* **Linear Data Structures:**These structures store data in a sequential order this allowing for easy **insertion**and **deletion**operations. Examples include arrays, linked lists, and queues.
* **Non-Linear Data Structures:**These structures store data in a hierarchical or interconnected manner this allowing for more complex relationships between data elements. Examples include trees, graphs, and hash tables.

## Types of Data Structures

Basically, data structures are divided into two categories:

### ****Linear Data Structures:****

* **Array:**A collection of elements of the same type stored in contiguous memory locations.
* **Linked List:** A collection of elements linked together by pointers, allowing for dynamic insertion and deletion.
* **Queue:**A First-In-First-Out (FIFO) structure where elements are added at the end and removed from the beginning.
* **Stack:** A Last-In-First-Out (LIFO) structure where elements are added and removed from the top.

### Non-Linear Data Structures:

* **Tree:**A hierarchical structure where each node can have multiple child nodes.
* **Graph: A** collection of nodes connected by edges, representing relationships between data elements.
* **Hash Table:** A data structure that uses a hash function to map keys to values, allowing for fast lookup and insertion.

**Advantages of Static Memory allocation**

* The memory is allocated during compile time.
* It is easy to use.
* It uses a stack Data Structure.
* The execution time is efficiently controlled.

**Disadvantages of Static Memory allocation**

* This allocation method leads to memory wastage.
* Memory cannot be changed while executing a program.
* Exact memory requirements must be known.
* If memory is not required, it cannot be freed.

**Advantages of Dynamic Memory allocation**

* This allocation method has no memory wastage.
* The memory allocation is done at run time.
* Memory size can be changed based on the requirements of the dynamic memory allocation.
* If memory is not required, it can be freed.

**Disadvantages of Dynamic Memory allocation**

* It requires more execution time due to execution during runtime.
* The compiler does not help with allocation and deallocation. Programmer needs to both allocate and deallocate the memory.

`

**Applications of Data Structures**

Data structures are widely used in various applications, including:

* **Database Management Systems:** To store and manage large amounts of structured data.
* **Operating Systems:**To manage memory, processes, and files.
* **Compiler Design:**To represent source code and intermediate code.
* **Artificial Intelligence:**To represent knowledge and perform reasoning.
* **Graphics and Multimedia:** To store and process images, videos, and audio data.
* **Databases**: Data structures like B-trees and hash tables are used to implement indexing and storage mechanisms in databases for efficient data retrieval and manipulation.
* **Operating Systems**: Data structures such as queues, stacks, and trees are used in operating systems for process scheduling, memory management, file systems, and resource allocation.
* **Compilers**: Abstract syntax trees (ASTs) and symbol tables are essential data structures used in compilers for parsing, semantic analysis, optimization, and code generation.
* **Networking**: Graphs and trees are used to model and optimize network topologies, routing algorithms, and communication protocols.
* **Web Development**: Data structures like arrays, linked lists, and hash maps are used extensively in web development for managing session data, caching, routing, and database interactions.
* **Artificial Intelligence and Machine Learning**: Graphs and trees are used to represent knowledge, decision trees, and search algorithms in AI and machine learning applications.
* **Graphics and Game Development**: Data structures such as trees and graphs are used for scene graphs, collision detection, spatial indexing, and pathfinding algorithms in graphics rendering and game development.
* **Cryptography**: Data structures like hash tables and trees are used in cryptographic algorithms for secure storage, authentication, and encryption.
* **Bioinformatics**: Data structures are used in bioinformatics for storing and analyzing DNA sequences, protein structures, and biological networks.
* **Finance and Trading**: Data structures like priority queues, hash maps, and trees are used in financial applications for portfolio management, algorithmic trading, and risk analysis.
* **Geographic Information Systems (GIS)**: Spatial data structures like quad trees and R-trees are used to efficiently store and retrieve geographic data such as maps, satellite imagery, and location-based services.
* **Text Editors and Word Processors**: Data structures such as buffers and trees are used in text editors and word processors for efficient text manipulation, searching, and formatting.

## Memory allocation definition

Memory allocation is the process of reserving virtual or physical computer space for a specific purpose (e.g., for computer programs and services to run). Memory allocation is part of the management of computer memory resources, known as memory management. Through memory allocation, computer programs and services are assigned a specific memory portion, depending on how much memory they need.

**How memory allocation works**

* Memory allocation is a **computer hardware operation** managed through the operating system.
* Through the process of memory allocation, a **portion of the computer memory is set aside** for running programs and processes. How much memory is allocated depends on the program’s requirements.
* When the program or application has finished its operation, the memory is **released** and assigned to another program.

**Types of memory allocation**

* **Static memory allocation (compile-time).** Programs and services are allocated memory at compile time. Compile time refers to the period when the programming code is converted to machine code for the computer to understand.
* **Dynamic memory allocation (run-time).** Programs and services are allocated memory at run time or execution time. Dynamic memory allocation allows computers to use memory space more efficiently.

|  |  |  |
| --- | --- | --- |
| S.No | Static Memory Allocation | Dynamic Memory Allocation |
| 1 | In the static memory allocation, variables get allocated permanently, till the program executes or function call finishes. | In the Dynamic memory allocation, variables get allocated only if your program unit gets active. |
| 2 | Static Memory Allocation is done before program execution. | Dynamic Memory Allocation is done during program execution. |
| 3 | It uses [stack](https://www.geeksforgeeks.org/stack-data-structure/) for managing the static allocation of memory | It uses [heap](https://www.geeksforgeeks.org/heap-data-structure/) for managing the dynamic allocation of memory |
| 4 | It is less efficient | It is more efficient |
| 5 | In Static Memory Allocation, there is no memory re-usability | In Dynamic Memory Allocation, there is memory re-usability and memory can be freed when not required |
| 6 | In static memory allocation, once the memory is allocated, the memory size can not change. | In dynamic memory allocation, when memory is allocated the memory size can be changed. |
| 7 | In this memory allocation scheme, we cannot reuse the unused memory. | This allows reusing the memory. The user can allocate more memory when required. Also, the user can release the memory when the user needs it. |
| 8 | In this memory allocation scheme, execution is faster than dynamic memory allocation. | In this memory allocation scheme, execution is slower than static memory allocation. |
| 9 | In this memory is allocated at compile time. | In this memory is allocated at run time. |
| 10 | In this allocated memory remains from start to end of the program. | In this allocated memory can be released at any time during the program. |
| 11 | **Example:** This static memory allocation is generally used for [array](https://www.geeksforgeeks.org/introduction-to-arrays/). | **Example:** This dynamic memory allocation is generally used for [linked list](https://www.geeksforgeeks.org/data-structures/linked-list/). |

**Stack Allocation:** The allocation happens on contiguous blocks of memory. We call it a stack memory allocation because the allocation happens in the function call stack. The size of memory to be allocated is known to the compiler and whenever a function is called, its variables get memory allocated on the stack. And whenever the function call is over, the memory for the variables is de-allocated. This all happens using some predefined routines in the compiler. A programmer does not have to worry about memory allocation and de-allocation of stack variables. This kind of memory allocation is also known as Temporary memory allocation because as soon as the method finishes its execution all the data belonging to that method flushes out from the stack automatically. This means any value stored in the stack memory scheme is accessible as long as the method hasn’t completed its execution and is currently in a running state.

**Heap Allocation:**The memory is allocated during the execution of instructions written by programmers. Note that the name heap has nothing to do with the [heap data structure](https://www.geeksforgeeks.org/heap-data-structure/). It is called a heap because it is a pile of memory space available to programmers to allocate and de-allocate. Every time when we made an object it always creates in Heap-space and the referencing information to these objects is always stored in Stack-memory. Heap memory allocation isn’t as safe as Stack memory allocation because the data stored in this space is accessible or visible to all threads. If a programmer does not handle this memory well, a [memory leak](https://www.geeksforgeeks.org/what-is-memory-leak-how-can-we-avoid/) can happen in the program.

**The Heap-memory allocation is further divided into three categories:-** These three categories help us to prioritize the data(Objects) to be stored in the Heap-memory or in the [Garbage collection](https://www.geeksforgeeks.org/garbage-collection-java/).

* **Young Generation –** It’s the portion of the memory where all the new data(objects) are made to allocate the space and whenever this memory is completely filled then the rest of the data is stored in Garbage collection.
* **Old or Tenured Generation –** This is the part of Heap-memory that contains the older data objects that are not in frequent use or not in use at all are placed.
* **Permanent Generation –**This is the portion of Heap-memory that contains the JVM’s metadata for the runtime classes and application methods.

A memory leak occurs when programmers create a memory in a heap and forget to delete it.

* The consequence of the memory leak is that it reduces the performance of the computer by reducing the amount of available memory. Eventually, in the worst case, too much of the available memory may become allocated, all or part of the system or device stops working correctly, the application fails, or the system slows down vastly.
* Memory leaks are particularly serious issues for programs like daemons and servers which by definition never terminate.

Time complexity is a type of computational complexity that describes the time required to execute an algorithm

Constant time is when the algorithm does not depend on the size of the input. Linear time is when the algorithm is proportional to the size of the input.

An algorithm is a sequence of steps executed by a computer that takes an input and transforms it into a target output.

**What is the need for algorithms?**

1. Algorithms are necessary for solving complex problems efficiently and effectively.
2. They help to automate processes and make them more reliable, faster, and easier to perform.
3. Algorithms also enable computers to perform tasks that would be difficult or impossible for humans to do manually.
4. They are used in various fields such as mathematics, computer science, engineering, finance, and many others to optimize processes, analyze data, make predictions, and provide solutions to problems.

Asymptotic Notations are mathematical tools that allow you to analyze an algorithm’s running time by identifying its behavior as its input size grows.

.This is also referred to as an algorithm’s growth rate

## ****Big-O Notation (O-notation):****

*Big-O notation represents the upper bound of the running time of an algorithm. Therefore, it gives the worst-case complexity of an algorithm.*

## *.It is the most widely used notation for Asymptotic analysis. .It specifies the upper bound of a function. .The maximum time required by an algorithm or the worst-case time complexity.*

## Omega Notation (Ω-****Notation):****

*.*

*It returns the highest possible output value(big-O) for a given input.  
.Big-Oh(Worst Case) It is defined as the condition that allows an algorithm to complete statement execution in the longest amount of time possible.*

*Omega notation represents the lower bound of the running time of an algorithm. Thus, it provides the best case complexity of an algorithm.*

## Theta Notation (Θ-Notation):

*Theta notation encloses the function from above and below. Since it represents the upper and the lower bound of the running time of an algorithm, it is used for analyzing the****average-case****complexity of an algorithm.*

*.Theta (Average Case) You add the running times for each possible input combination and take the average in the average case.*

Asymptotic analysis is a technique used in computer science to analyze the efficiency of algorithms in terms of their time and space complexity. It provides a high-level understanding of how the performance of an algorithm scales with the size of the input.

There are two main types of asymptotic analysis:

1. **Time Complexity:** Time complexity describes the amount of time an algorithm takes to complete as a function of the size of its input. It provides an upper bound on the running time of an algorithm, usually expressed using Big O notation (e.g., O(n), O(n^2), O(log n)).
2. **Space Complexity:** Space complexity describes the amount of memory space an algorithm uses as a function of the size of its input. Similar to time complexity, it provides an upper bound on the amount of memory required by an algorithm, also expressed using Big O notation.

Pending

Remove nth node from end

Doubly linkedlist

Leetcode