**VERİ YAPILARI TERMİNOLOJİ**

|  |  |
| --- | --- |
|  |  |
| **Data Structures:** | A **data structure** is an arrangement of data in a computer’s memory (or sometimes on a disk) |
|  |  |
| **Array:** | The **array** is the most commonly used data storage structure; it’s built into most programming languages. |
|  |  |
| **Stack:** | A **stack** allows access to only one data item: the last item inserted. |
|  |  |
| **Queue:** | In computer science a **queue** is a data structure that is somewhat like a stack, except that in a queue the first item inserted is the first to be removed (First-In-First-Out, FIFO), while in a stack, as we’ve seen, the last item inserted is the first to be removed (LIFO). |
|  |  |
| **Queue🡪Enqueue:** | The **enqueue** method adds element into the tail of the queue. |
|  |  |
| **Queue🡪Dequeue:** | The **dequeue** method removes an element from the head of the queue and returns the removed element. |
|  |  |
| **Queue🡪Front:** | **The Front** is used to reference the first or the oldest element of the queue container. |
|  |  |
| **Queue🡪Rear:** | **The Rear** is used to reference the last or the newest element of the queue container. |
|  |  |
| **Queue🡪** **Priority Queue:** | A **priority queue** is a special type of queue in which each element is associated with a priority value. |
|  |  |
| **List🡪head:** | The first node of a linked list usually is called the **head** of the list. |
|  |  |
| **List🡪Tail:** | The last node of a linked list usually is called the **tail** of the list. |
|  |  |
| **List🡪Doubly link list:** | **Doubly Linked List** is a variation of Linked list in which navigation is possible in both ways, either forward and backward easily as compared to Single Linked List. |
| **List🡪Circular link lists:** | **Circular linked list** is a linked list where all nodes are connected to form a circle. |
|  |  |
| **Binary tree🡪Root:** | The node at the top of the tree is called the **root.** |
|  |  |
| **Binary tree🡪Parent:** | Any node (except the root) has exactly one edge running upward to another node. The node above it is called the **parent** of the node. |
|  |  |
| **Binary tree🡪Child:** | Any node may have one or more lines running downward to other nodes. These nodes below a given node are called its **child.** |
|  |  |
| **Binary tree🡪Leaf:** | Node with no children is called **leaf,** or external node. |
|  |  |
| **Binary tree🡪inorder successor node:** | In Binary Search Tree,  **inorder successor node**  can also be defined as the node with the smallest key greater than the key of the current node. |
|  |  |
| **Binary tree🡪height:** | The **height** of a binary tree is the height of the root node in the whole binary tree. |
|  |  |
| **Binary tree🡪depth:** | The **depth** of a node in a binary tree is the total number of edges from the root node to the target node. Similarly, the **depth** of a binary tree is the total number of edges from the root node to the most distant leaf node. |
|  |  |
| **Binary tree🡪ancestor:** | An **ancestor** is a node that is present in the upper layer of a given node. |
|  |  |
| **Stack 🡪Push:** | The **push**(Object element) method adds the specified element to the stack |
|  |  |
| **Stack 🡪Pop:** | The **pop** method removes the top element from the stack and returns it. |
|  |  |
| **Stack 🡪Peek:** | The **peek**() method retrieves the element at the top of the stack without removing it. |
|  |  |
| **Graph🡪vertex:** | "**Vertex**" is a synonym for a node of a graph, i.e., one of the points on which the graph is defined and which may be connected by graph edges. |
|  |  |
| **Graph🡪edge:** | An **edge** (or link) of a network (or graph) is one of the connections between the nodes (or vertices) of the network. |