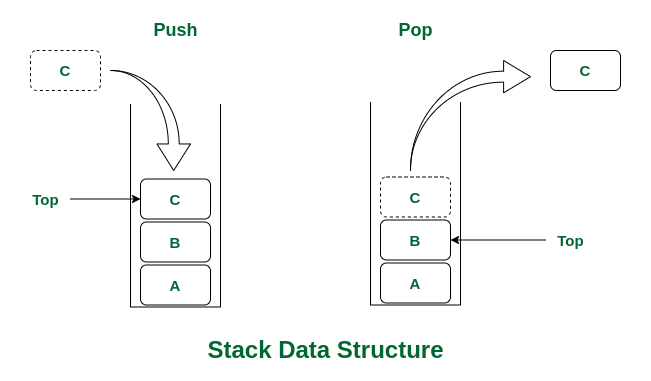
STACKS AND QUEUES

STACK

Stack is a linear data structure that follows a particular order in which the operations are performed. The order may be LIFO(Last In First Out) or FILO(First In Last Out). LIFO implies that the element that is inserted last, comes out first and FILO implies that the element that is inserted first, comes out last.



There are many real-life examples of a stack. Consider an example of plates stacked over one another in the canteen. The plate which is at the top is the first one to be removed, i.e. the plate which has been placed at the bottommost position remains in the stack for the longest period of time. So, it can be simply seen to follow LIFO(Last In First Out)/FILO(First In Last Out) order.

**Basic Operations on Stack**

In order to make manipulations in a stack, there are certain operations provided to us.

* **push()** to insert an element into the stack
* **pop()**to remove an element from the stack
* **top()** Returns the top element of the stack.
* **isEmpty()**returns true if stack is empty else false.
* **size()** returns the size of stack.

**Complexity Analysis:**

* **Time Complexity**

| **Operations** | **Complexity** |
| --- | --- |
| push() | O(1) |
| pop() | O(1) |
| isEmpty() | O(1) |
| size() | O(1) |

**Types of Stacks:**

* **Fixed Size Stack**: As the name suggests, a fixed size stack has a fixed size and cannot grow or shrink dynamically. If the stack is full and an attempt is made to add an element to it, an overflow error occurs. If the stack is empty and an attempt is made to remove an element from it, an underflow error occurs.
* **Dynamic Size Stack**: A dynamic size stack can grow or shrink dynamically. When the stack is full, it automatically increases its size to accommodate the new element, and when the stack is empty, it decreases its size. This type of stack is implemented using a linked list, as it allows for easy resizing of the stack.

In addition to these two main types, there are several other variations of Stacks, including:

1. **Infix to Postfix Stack**: This type of stack is used to convert infix expressions to postfix expressions.
2. **Expression Evaluation Stack**: This type of stack is used to evaluate postfix expressions.
3. **Recursion Stack**: This type of stack is used to keep track of function calls in a computer program and to return control to the correct function when a function returns.
4. **Memory Management Stack**: This type of stack is used to store the values of the program counter and the values of the registers in a computer program, allowing the program to return to the previous state when a function returns.
5. **Balanced Parenthesis Stack**: This type of stack is used to check the balance of parentheses in an expression.
6. **Undo-Redo Stack**: This type of stack is used in computer programs to allow users to undo and redo actions.

**Applications of the stack:**

* [Infix to Postfix](https://www.geeksforgeeks.org/stack-set-2-infix-to-postfix/) /Prefix conversion
* Redo-undo features at many places like editors, photoshop.
* Forward and backward features in web browsers
* Used in many algorithms like [Tower of Hanoi,](https://www.geeksforgeeks.org/recursive-functions/)[tree traversals](https://www.geeksforgeeks.org/618/), [stock span problems](https://www.geeksforgeeks.org/the-stock-span-problem/), and [histogram problems](https://www.geeksforgeeks.org/largest-rectangular-area-in-a-histogram-set-1/).
* Backtracking is one of the algorithm designing techniques. Some examples of backtracking are the Knight-Tour problem, N-Queen problem, find your way through a maze, and game-like chess or checkers in all these problems we dive into someway if that way is not efficient we come back to the previous state and go into some another path. To get back from a current state we need to store the previous state for that purpose we need a stack.
* In Graph Algorithms like [Topological Sorting](https://www.geeksforgeeks.org/topological-sorting/) and [Strongly Connected Components](https://www.geeksforgeeks.org/strongly-connected-components/)
* In Memory management, any modern computer uses a stack as the primary management for a running purpose. Each program that is running in a computer system has its own memory allocations
* String reversal is also another application of stack. Here one by one each character gets inserted into the stack. So the first character of the string is on the bottom of the stack and the last element of a string is on the top of the stack. After Performing the pop operations on the stack we get a string in reverse order.
* Stack also helps in implementing function call in computers. The last called function is always completed first.
* Stacks are also used to implement the undo/redo operation in text editor.

QUEUE

*A****Queue****is defined as a linear data structure that is open at both ends and the operations are performed in First In First Out (FIFO) order.*

We define a queue to be a list in which all additions to the list are made at one end, and all deletions from the list are made at the other end.  The element which is first pushed into the order, the operation is first performed on that.

**FIFO Principle of Queue:**

* A Queue is like a line waiting to purchase tickets, where the first person in line is the first person served. (i.e. First come first serve).
* Position of the entry in a queue ready to be served, that is, the first entry that will be removed from the queue, is called the **front** of the queue(sometimes, **head** of the queue), similarly, the position of the last entry in the queue, that is, the one most recently added, is called the **rear** (or the **tail**) of the queue. See the below figure.

## [Queue Data Structure](https://media.geeksforgeeks.org/wp-content/cdn-uploads/20221213113312/Queue-Data-Structures.png)

## [FIFO Principle of Queue](https://www.geeksforgeeks.org/fifo-first-in-first-out-approach-in-programming/):

* A Queue is like a line waiting to purchase tickets, where the first person in line is the first person served. (i.e. First come first serve).
* Position of the entry in a queue ready to be served, that is, the first entry that will be removed from the queue, is called the **front** of the queue(sometimes, **head** of the queue), similarly, the position of the last entry in the queue, that is, the one most recently added, is called the **rear** (or the**tail**) of the queue. See the below figure.

**Characteristics of Queue:**

* Queue can handle multiple data.
* We can access both ends.
* They are fast and flexible.
* **Queue:** the name of the array storing queue elements.
* **Front**: the index where the first element is stored in the array representing the queue.
* **Rear:** the index where the last element is stored in an array representing the queue.

## [Types of Queue:](https://www.geeksforgeeks.org/different-types-of-queues-and-its-applications/)

There are different types of queues:

1. **Input Restricted Queue:** This is a simple queue. In this type of queue, the input can be taken from only one end but deletion can be done from any of the ends.
2. **Output Restricted Queue:** This is also a simple queue. In this type of queue, the input can be taken from both ends but deletion can be done from only one end.
3. [**Circular Queue**](https://www.geeksforgeeks.org/introduction-and-array-implementation-of-circular-queue/)**:** This is a special type of queue where the last position is connected back to the first position. Here also the operations are performed in FIFO order. To know more refer [this](https://www.geeksforgeeks.org/introduction-and-array-implementation-of-circular-queue/).
4. [**Double-Ended Queue (Dequeue)**](https://www.geeksforgeeks.org/deque-set-1-introduction-applications/)**:** In a double-ended queue the insertion and deletion operations, both can be performed from both ends. To know more refer [this](https://www.geeksforgeeks.org/deque-set-1-introduction-applications/).
5. [**Priority Queue**](https://www.geeksforgeeks.org/priority-queue-set-1-introduction/)**:** A priority queue is a special queue where the elements are accessed based on the priority assigned to them. To know more refer [this](https://www.geeksforgeeks.org/priority-queue-set-1-introduction/).

**Basic Operations for Queue in Data Structure:**

Some of the basic operations for Queue in Data Structure are:

1. **Enqueue() –** Adds (or stores) an element to the end of the queue..
2. **Dequeue() –** Removal of elements from the queue.
3. **Peek() or front()-** Acquires the data element available at the front node of the queue without deleting it.
4. **rear() –** This operation returns the element at the rear end without removing it.
5. **isFull() –** Validates if the queue is full.
6. **isNull() –** Checks if the queue is empty.

[**Applications of Queue:**](https://www.geeksforgeeks.org/applications-of-queue-data-structure/)

Application of queue is common. In a computer system, there may be queues of tasks waiting for the printer, for access to disk storage, or even in a time-sharing system, for use of the CPU. Within a single program, there may be multiple requests to be kept in a queue, or one task may create other tasks, which must be done in turn by keeping them in a queue.

* It has a single resource and multiple consumers.
* It synchronizes between slow and fast devices.
* In a network, a queue is used in devices such as a router/switch and mail queue.
* Variations: dequeue, priority queue and double-ended priority queue.

## FAQs (Frequently asked questions) on Queue:

### 1. What data structure can be used to implement a priority queue?

Priority queues can be implemented using a variety of data structures, including linked lists, arrays, binary search trees, and heaps. Priority queues are best implemented using the heap data structure.

### 2. Queues are used for what purpose?

In addition to making your data persistent, queues reduce errors that occur when different parts of your system are down.

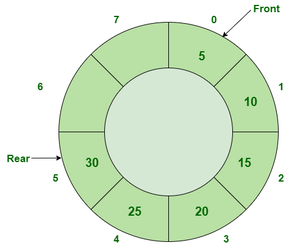
### 3. In data structures, what is a double-ended queue?

In a double-ended queue, elements can be inserted and removed at both ends.

### 4. What is better, a stack or a queue?

If you want things to come out in the order you put them in, use a queue. Stacks are useful when you want to reorder things after putting them in.

[**Circular Queue**](https://www.geeksforgeeks.org/circular-queue-set-1-introduction-array-implementation/) is a linear data structure that follows FIFO (first in first out) principle which means the item that is inserted first in the queue will be taken out first.  It is also known as circular/ring buffer because the last position of the queue is circled back and connected with the first element thereby, forming a circular structure.



**Applications of Circular Queue:**

* In a page replacement algorithm, a circular list of pages is maintained and when a page needs to be replaced, the page in the front of the queue will be chosen.
* Computer systems supply a holding area for maintaining communication between two processes or two programs. This memory area is also known as a ring buffer.
* CPU Scheduling: In the Round-Robin scheduling algorithm, a circular queue is utilized to maintain processes that are in a ready state.
* Inter-process communication: Circular queue can be used for communication between different processes.
* Resource allocation: In operating systems, circular queue is used for managing resources.

**Real-time Applications of Circular Queue:**

* Months in a year: Jan – Feb – March – and so on upto Dec- Jan – . . .
* Eating: Breakfast – lunch – snacks – dinner – breakfast – . . .
* Traffic Light is also a real-time application of circular queue.

**Advantages of Circular Queue:**

* It provides a quick way to store FIFO data with a maximum size.
* Efficient utilization of the memory.
* Doesn’t use dynamic memory.
* Simple implementation.
* All operations occur in O(1) constant time.

**Disadvantages of Circular Queue:**

* In a circular queue, the number of elements you can store is only as much as the queue length, you have to know the maximum size beforehand.
* Some operations like deletion or insertion can be complex in circular queue.
* The implementation of some algorithms like priority queue can be difficult in circular queue.
* Circular queue has a fixed size, and when it is full, there is a risk of overflow if not managed properly.

DEQUE

**Applications of Deque:** Since Deque supports both stack and queue operations, it can be used as both. The Deque data structure supports clockwise and anticlockwise rotations in O(1) time which can be useful in certain applications. Also, the problems where elements need to be removed and or added to both ends can be efficiently solved using Deque. For example see the [Maximum of all subarrays of size k problem.](https://www.geeksforgeeks.org/maximum-of-all-subarrays-of-size-k/), [0-1 BFS,](https://www.geeksforgeeks.org/0-1-bfs-shortest-path-binary-graph/) and [Find the first circular tour that visits all petrol pumps](https://www.geeksforgeeks.org/find-a-tour-that-visits-all-stations/). See the [wiki page](http://en.wikipedia.org/wiki/Double-ended_queue#Applications)for another example of the A-Steal job scheduling algorithm where Deque is used as deletions operation is required at both ends.

**Some Practical Applications of Deque**:

* Applied as both stack and queue, as it supports both operations.
* Storing a web browser’s history.
* Storing a software application’s list of undo operations.
* Job scheduling algorithm

**Monotonic Deque :**

* It is deque which stores elements in strictly increasing order or in strictly decreasing order
* To maintain monotonicity, we need to delete elements
  + For example – Consider monotonic(decreasing) deque **dq** =  {5, 4, 2, 1}
  + Insert 3 into dq
    - So we need to delete elements till dq.back() < 3   to insert 3 into dq  (*2,1 are the deleted elements)*
    - Resulting dq = {5, 4, 3}
* Applications of monotonic deque :
  + It can be used to get **next maximum** in a subarray ([sliding-window-maximum-of-all-subarrays-of-size-k](https://www.geeksforgeeks.org/sliding-window-maximum-maximum-of-all-subarrays-of-size-k/)) by using monotonically decreasing deque
  + Like this it can be used to get **previous maximum** also in a subarray
  + It is frequentlyused in**sliding window problems (hard)**

**Other Applications:**

Deques have several other applications, some of which include:

* **Palindrome checking:** Deques can be used to check if a word or phrase is a palindrome. By inserting each character of the word or phrase into a deque, it is possible to check if the word or phrase is a palindrome by comparing the first and last characters, the second and second-to-last characters, and so on.
* **Graph traversal:** Deques can be used to implement Breadth-First Search (BFS) on a graph. BFS uses a queue to keep track of the vertices to be visited next, and a deque can be used as an alternative to a queue in this case.
* **Task scheduler:** Deques can be used to implement a task scheduler that keeps track of tasks to be executed. Tasks can be added to the back of the deque, and the scheduler can remove tasks from the front of the deque and execute them.
* **Multi-level undo/redo functionality:** Deques can be used to implement undo and redo functionality in applications. Each time a user performs an action, the current state of the application is pushed onto the deque. When the user undoes an action, the front of the deque is popped, and the previous state is restored. When the user redoes an action, the next state is popped from the deque.
* In computer science, deque can be used in many algorithms like *LRU Cache*, *Round Robin Scheduling*, *Expression Evaluation*.