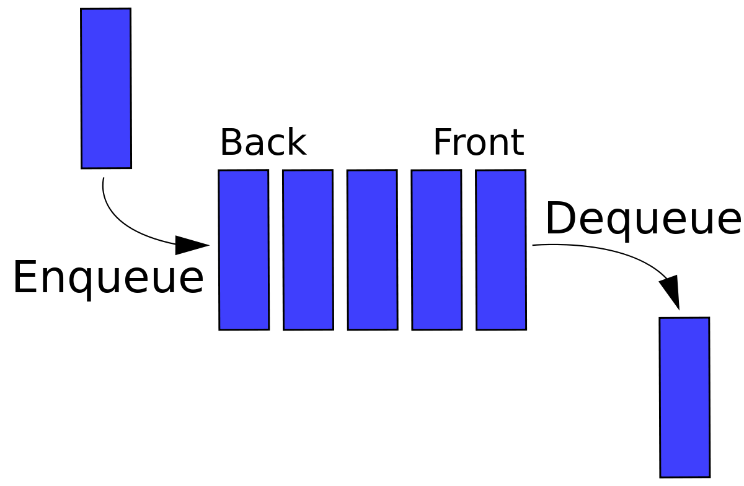
**Queue**

Queue is a sequential access data structure which follows the FIFO (First in First Out) order for adding or removing elements.

Element-3

Element-2

Element-1



Common methods used in Queue

1. Enqueue

2. Dequeue

3. Peek

4. Contains

**Enqueue**

Enqueue is used to add the element at the end/tail of the queue and increase the size of the queue by 1.

demo\_queue. enqueue(“hi”)

demo\_queue. enqueue(“hello”)

demo\_queue. enqueue(“world”)

world

hello

hi

demo\_queue. enqueue (“send email”)

Send email

**Dequeue**

Dequeue is used to remove the element from the queue. This will remove the element from head of the queue (this means element which is added first). This will decrease the size of the queue by 1.

Dequeue will not take any argument.

demo\_queue. dequeue ()

demo\_queue. dequeue ()

demo\_queue. dequeue ()

Send email

world

hello

hi

**Peek**

Peek is used to know the 1st element in the queue without removing it.

Peek won’t take any arguments.

Send mail

world

hello

hi

demo\_queue. peek () ===🡺 hi

demo\_queue. dequeue ()

demo\_queue. peek () ===🡺 hello

**Contains**

Contains method will take an object and search for the object in the queue and returns True if found otherwise returns False.

Send mail

world

hello

hi

demo\_queue. contains(“hi”) ===🡺 True

demo\_queue. contains(“avenger”) ===🡺 False

**isEmpty**

isEmpty method is used to check whether the queue is empty or not. It returns True if queue is empty otherwise returns False.

Send mail

world

hello

hi

demo\_queue. IsEmpty () ===🡺 False

**isFull**

isFull is used to check whether the queue is Full or not. It returns True if queue is full otherwise returns False.

Send mail

world

hello

hi

demo\_queue. IsFull () ===🡺 True

**Time complexity equations of Queue**

**Accessing**

The time complexity for accessing an element in a queue is O(n).

Let’s say we have a queue like below

Send mail

world

hello

hi

if we want to access the element at tail/last (send mail), first we need to dequeue the elements (3 element) before it (send mail), then we have to dequeue the element we want to access. So, time complexity is O(n).

**Searching**

The time complexity for searching an element in a queue is O(n).

Let’s say we have a queue like below

Send mail

world

hello

hi

We want to search whether the element “send mail” is there or not. So, we have to iterate through all elements in the queue. we found the element at end/tail of the queue. hence the time complexity for searching is O(n).

**Inserting**

The time complexity for inserting an element in a queue is O(1).

Let’s say we have a queue like below

Send mail

world

hello

hi

If we want to add one more element to the queue the new element will always be added at tail/end of the queue. it requires only one operation to insert the new element. So, the time complexity for inserting is O(1).

**Deleting**

The time complexity for deleting an element in a queue is O(1).

Let’s say we have a queue

Send mail

world

hello

hi

When we delete an element from the queue it’s always the first/head element to be removed. It requires only one operation to remove. So the time complexity is O(1).

**Report of Queue**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Accessing | Searching | Inserting | Deleting |
| Time complexity | O(n) | O(n) | O(1) | O(1) |

**Note:**

Queues are widely used in Job scheduling to queue the tasks.

Queues are used in networking to queue the requests.