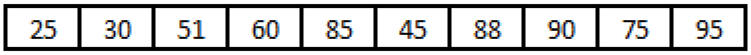
Disadvantages of queues

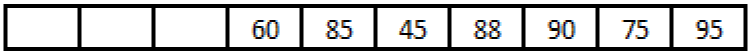
A major disadvantage of a classical queue is that a new element can only be inserted when *all* of the elements are deleted from the queue.

As an example, consider the queue:



25	30	51	60	85	45	88	90	75	95

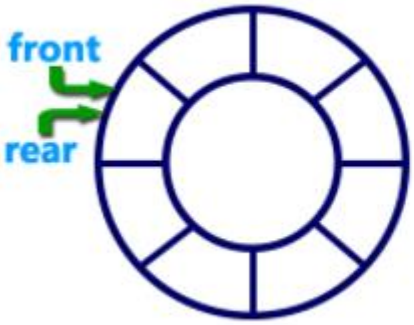
Now, if the first three members are de-queued from the front (left hand side) of the queue, we get:



			60	85	45	88	90	75	95

Where the queue remains full but we can not insert a new element because, the back of the queue (right hand side) remains as it was before. this is the major limitation of a classical queue, i.e. even if there is space available at the front of the queue we can not use it.

So to overcome the problem above, we can use a *circular queue*. With reference to Circular Queue - Data Structures, this can be defined as a "linear data structure in which the operations are performed based on FIFO (First In First Out) principle and the last position is connected back to the first position to make a circle." We can represent this circular queue as:



If a queue is considered circular, when a de-queue operation occurs, re-pointing the head of the queue to the next element is a simple assignment. This also avoids extensive re-buffering when all the elements would otherwise *move one to the left.*

Difference between STACK AND QUEUE

STACK	QUEUE
1. stack uses LIFO (last in first out) method to access and add data elements	1. Queue uses FIFO (First in first out) method to access and add data elements.
2. Stack has only one end open for pushing and popping the data elements	2. Queue has both ends open for enqueuing and dequeuing the data elements.
3. Number of pointers used is one	3. Number of pointers used is two
4. Operations performed are Push and Pop	4. Operations performed are Enqueue and dequeue
5. In stack, top = -1 indicates the underflow condition	5. In Queue, Front == -1 || Front == Rear + 1 Indicates the underflow condition
6. In stack, Top == Max – 1 indicates the overflow condition	6. In queue, Rear == Max – 1 indicates the overflow condition
7. It does not have variants.	7. It has variants like circular queue, priority queue, doubly ended queue.
8. Implementation of stack is simple	8. Implementation of queue is Comparatively complex

Difference between STACK AND ARRAYS

STACKS	ARRAYS
1. Stack is a linear data structure that is represented by a collection of items arranged in the form of a physical stack or a pile	1. An array, on the other hand, is a random access data structure used to store large number of data values to reduce the complexity of the program.
2. In stack, objects are inserted and deleted from one end only i.e. is called as top.	2. In an array, the objects are stored linearly, one after another for efficient memory management.
3. A stack is an abstract data type that can store meaning it can contain	3. An array will only store homogenous data meaning it refers to the collection

different data types (heterogeneous data)	of similar data types.
4. It is a limited-access data structure in which the objects can be added or removed in a particular order	4. Arrays have a list of ordered elements that can be accessed at any time.
5. stack uses LIFO (last in first out) method to access and add data elements	5. An array is a collection of objects which you can access at any time meaning objects can be inserted and removed randomly irrespective of their order.
6. Two operations can be performed on stack push and pop	6. Many operations can be performed on an array such as Traversing, Insertion, Deletion, Searching, Sorting, and Merging.

Difference between STACK AND QUEUE

ARRAY	**LINKED LIST**
1. Arrays are index based data structure where each element associated with an index.	1. Linked list relies on references where each node consists of the data and the references to the previous and next element.
2. An array is a set of similar data objects stored in sequential memory locations under a common heading or a variable name.	2. Linked list is a data structure which contains a sequence of the elements where each element is linked to its next element. It has two fields : data and link
3. Array has a fixed size and required to be declared prior	3. Linked List is not restricted to size and expand and contract during execution.
4. It is a consistent set of a fixed number of data items.	4. It is an ordered set comprising a variable number of data items.
5. Size should be Specified during declaration.	5. No need to specifysize, grow and shrink during execution.
6. Element location is allocated during compile time.	6. Element position is assigned during run time.
7. Order of the elements Stored consecutively	7. Order of the elements Stored randomly
8. In array, we can access the elements Direct or randomly i.e., Specify the array index or subscript.	8. In linked list, we can access the elements Sequentially i.e., Traverse starting from the first node in the list

	by the pointer.
9. Insertion and deletion of element is Slow relatively as shifting is required.	9. Insertion and deletion of element is Easier, fast and efficient.
10. Binary search and linear search is used	10. linear search is used
11.less Memory is required	11. more Memory is required
12. Memory Utilization is Ineffective	12. Memory Utilization is Efficient