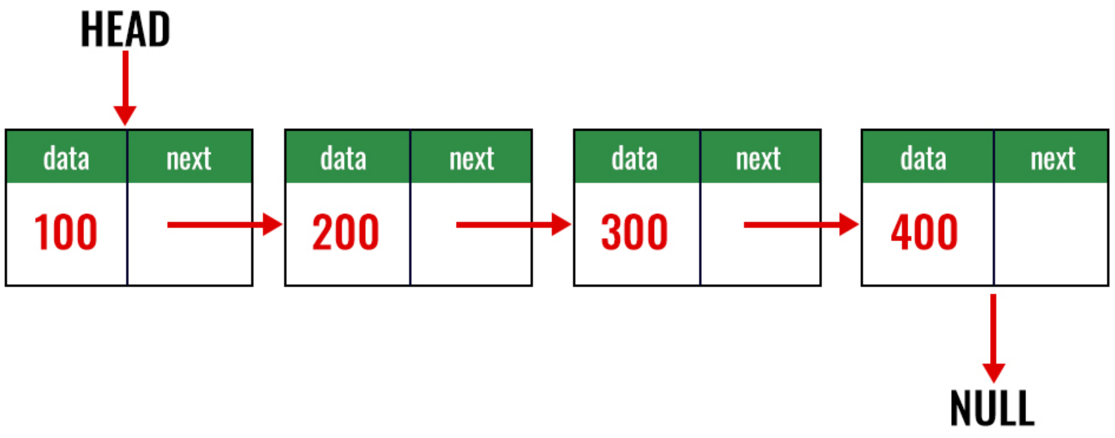
Collections

List

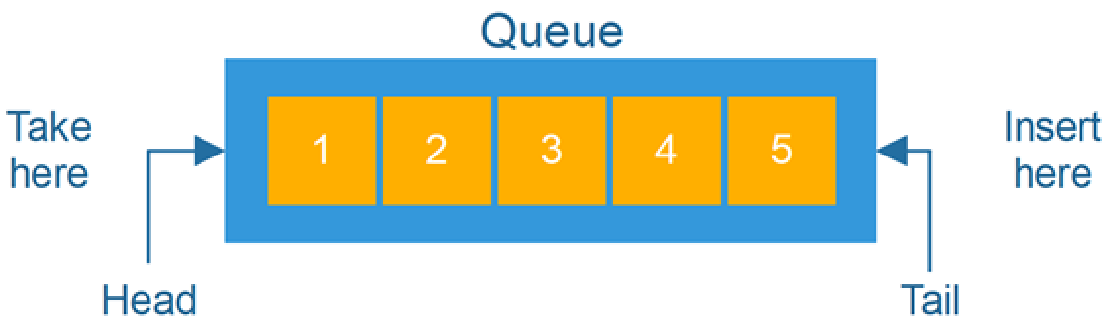
A List is an ordered Collection, an interface that extends Collection interface. It provides control over the position where you can insert an element while still being able to access elements by their index or searching directly the elements in the list.

- **ArrayList**
  - Java class that you can use to store a lists of Objects.
  - The difference between an Array and an ArrayList is that an Array has a constant size, while an ArrayList has a variable size.
  - The main goal of an ArrayList is to replicate the functionality of an Array while adding more features that solves Arrays problems.
- **LinkedList**
  - Dynamic data structure whose increases as you add the elements and decreases as you remove the elements from the list.
  - The elements in the LinkedList are stored in individual containers pointing to only the next and previous container.
  - The list points to the first container

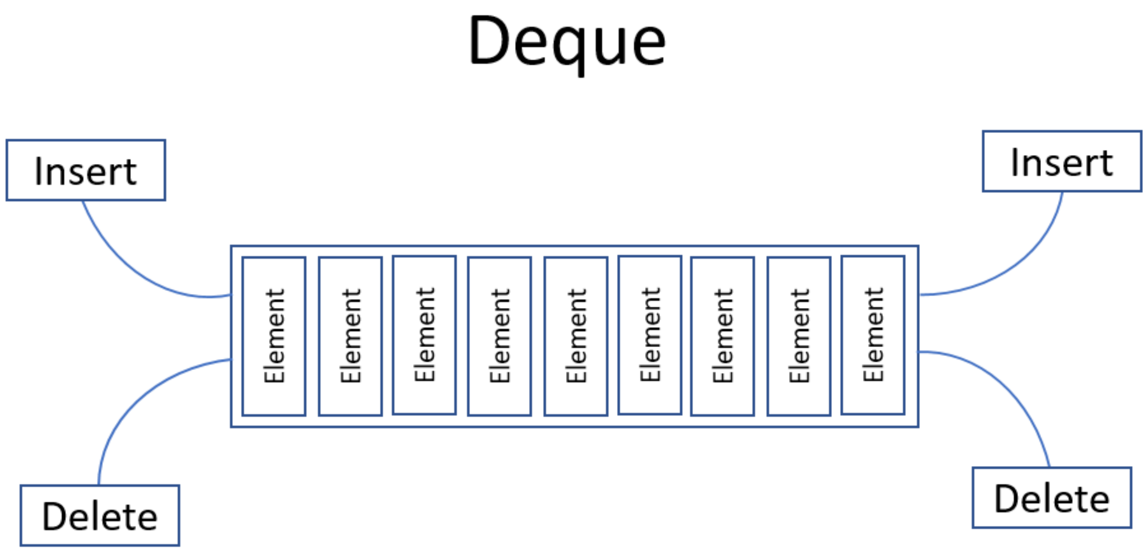


Queue

A Queue is an object that represents a data structure designed to have the element inserted at the end of the Queue, and the element removed from the beginning of the Queue



- **Deque**
  - A linear Collection that supports element insertion and removal at both ends.
  - Deque means "Double ended Queue and is usually pronounced "deck"



- **PriorityQueue**
    - A PriorityQueue stores a Collection prioritized by natural ordering or by a custom Comparator
    - The elements are prioritized with the least value element at the head of the Queue and Queue methods.

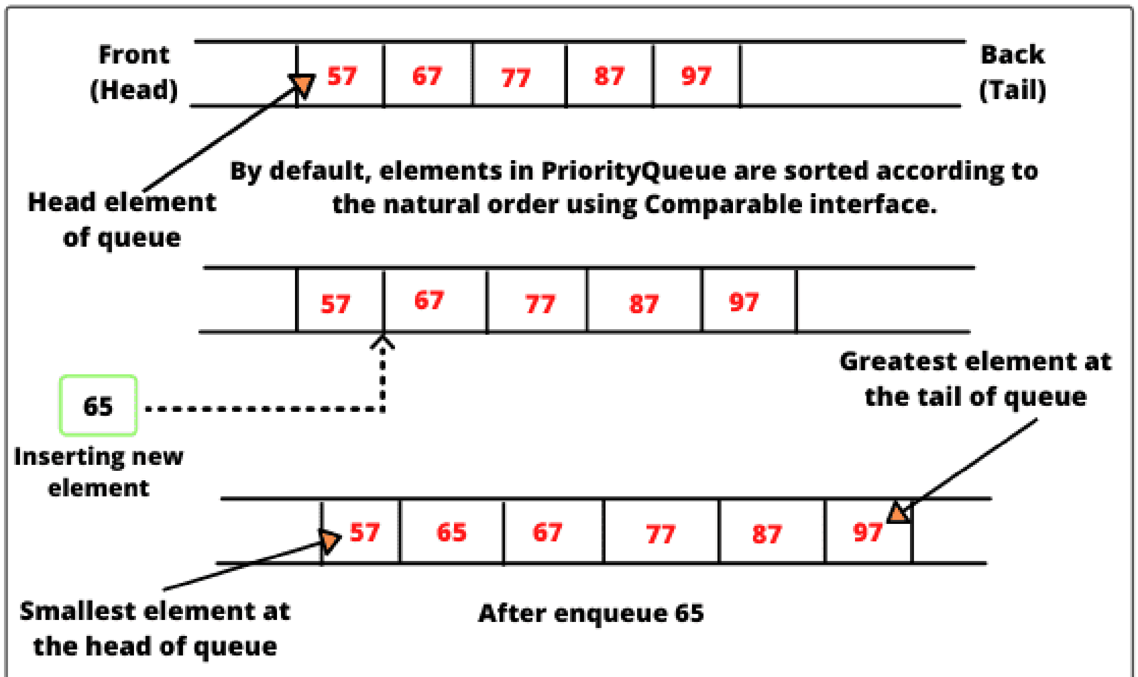

Front (Head)
57
67
77
87
97
Back (Tail)
Head element of queue
By default, elements in PriorityQueue are sorted according to the natural order using Comparable interface.
57
67
77
87
97
65
Inserting new element
Greatest element at the tail of queue
57
65
67
77
87
97
Smallest element at the head of queue
After enqueue 65

Fig: Operation of adding element in priority queue

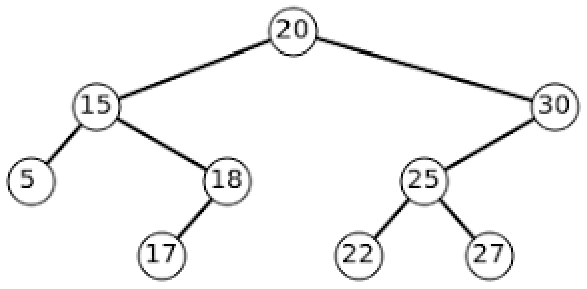
- **ArrayDeque**
    - An ArrayDeque is a special kind of growable array that allows us to add or remove an element from both sides
    - An ArrayDeque can be used as a Stack (Last-In-First-Out) or a Queue (First-In-Last-Out)

Set

A Set is an unordered Collection of Objects in which duplicate values cannot be stored.

- **SortedSet**
    - A sorted set is a set that uses natural values or custom Comparators that provides a total ordering in its elements.
- **NavigableSet**
    - Subtype of a SortedSet, behaving like a SortedSet, but with an additional set of navigation methods available in addition to the sorting mechanism of the SortedSet

- **TreeSet**
  - A TreeSet provides an implementation of the Set that uses a tree for storage.
  - Objects are stored in a sorted and ascending order.
  - Access and retrieval times are quite fast and, which makes them an excellent choice when storing large amounts of sorted information that has to be found quickly.



Map (Not a Collection)

A Map is a Java interface that stores the data in key-value pairs not allowing duplicate keys

- **SortedMap**
  - A Map that maintains its entries in ascending order, sorted according to the key's natural ordering or a custom Comparator provided at the CREATION of the SortedMap.
- **HashMap**
  - Data structure that uses the Map interface and a hash table to provide efficient access and manipulation of data based unique keys

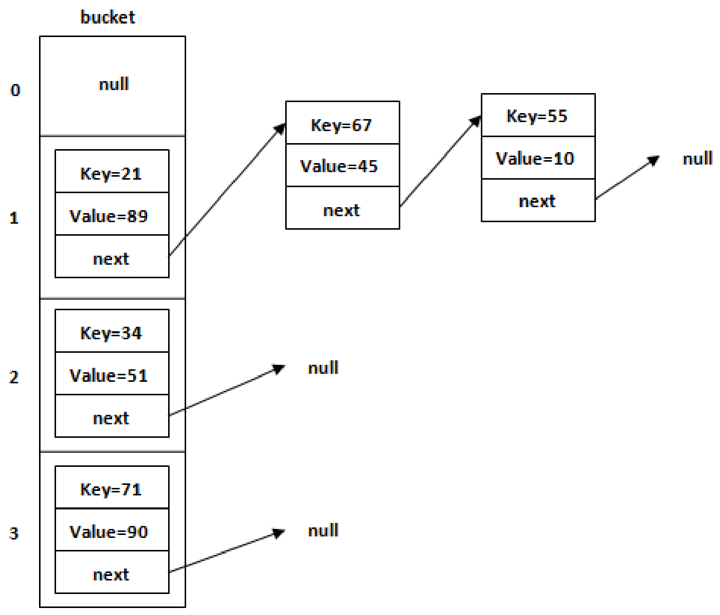
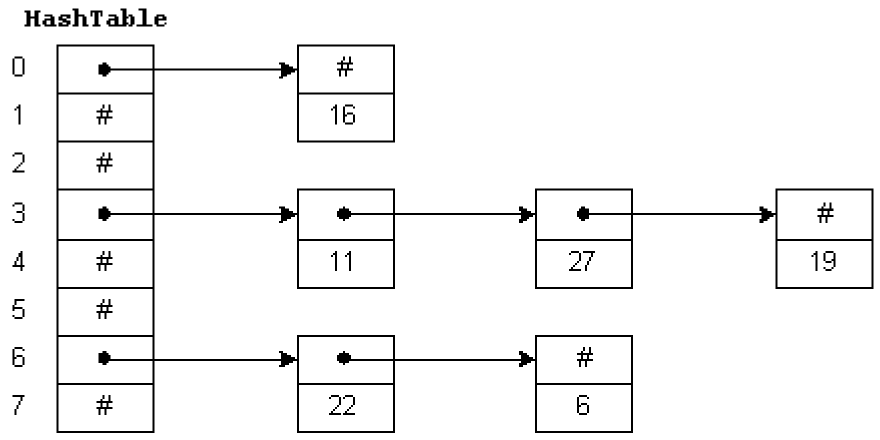


Figure: Allocation of nodes in Bucket

- **HashTable**
  - Unordered collection of key-value pairs, with a unique key for each value.
  - Data is stored in an array of list format, with a distinct index value for each data value.
  - The hash table efficiently combines retrieval, edit, and delete operations.


HashTable
0
1
2
3
4
5
6
7
#
16
11
27
19
22
6