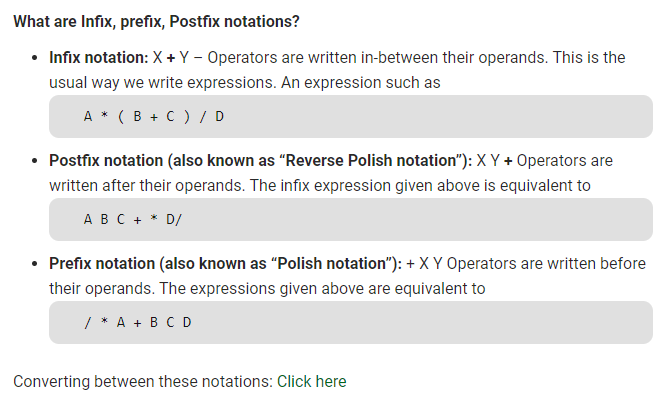
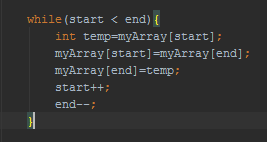
# 



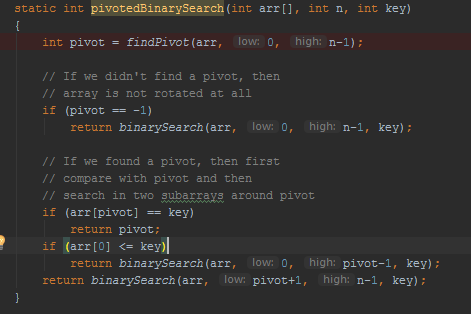
# Program for array rotation

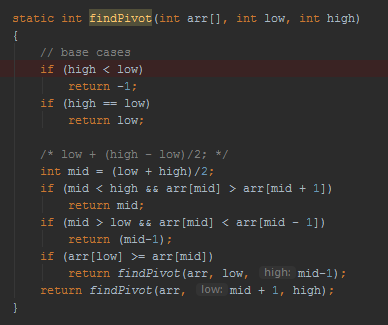
* **Using temp array** 
  + **Time complexity :** O(n)
  + **Auxiliary Space :** O(d)
* **Rotate one by one**
  + **Time complexity :** O(n \* d)
  + **Auxiliary Space :** O(1)

# Program to reverse array without temp array with complexity o(n)/ [Program to cyclically rotate an array by one](https://www.geeksforgeeks.org/c-program-cyclically-rotate-array-one/)

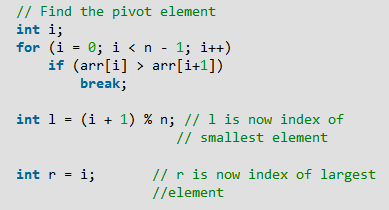


# Search an element in a sorted and rotated array





Below is the easy Approach to find Pivot:



1. [Given a sorted and rotated array, find if there is a pair with a given sum](https://www.geeksforgeeks.org/given-a-sorted-and-rotated-array-find-if-there-is-a-pair-with-a-given-sum/).

The idea is to first

1. find the largest element in array which is the pivot point also and the element just after largest is the smallest element.
2. Once we have indexes largest and smallest elements, we use similar meet in middle algorithm to find if there is a pair.
3. The only thing new here is indexes are incremented and decremented in rotational manner using modular arithmetic.

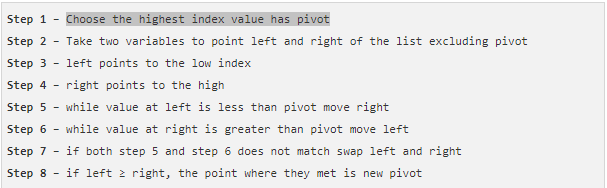
# 

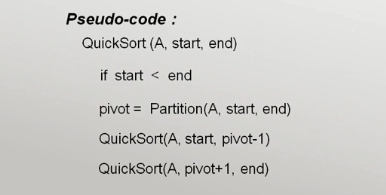
# 

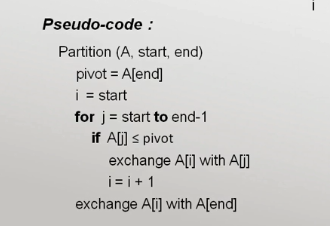
# Quick Sort Algorithm:

1. Average complexity: O (nlogn)
2. Worst Case: O(n2)

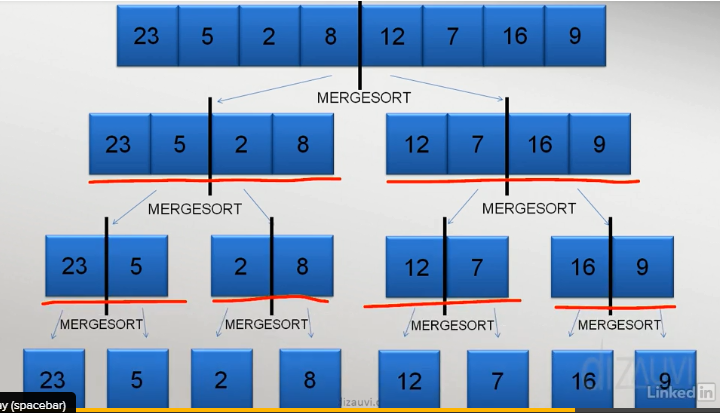
Algorithm:

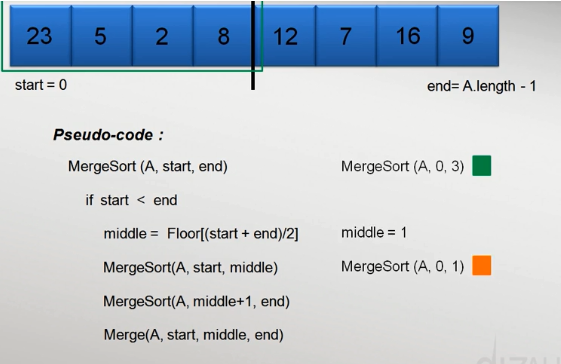


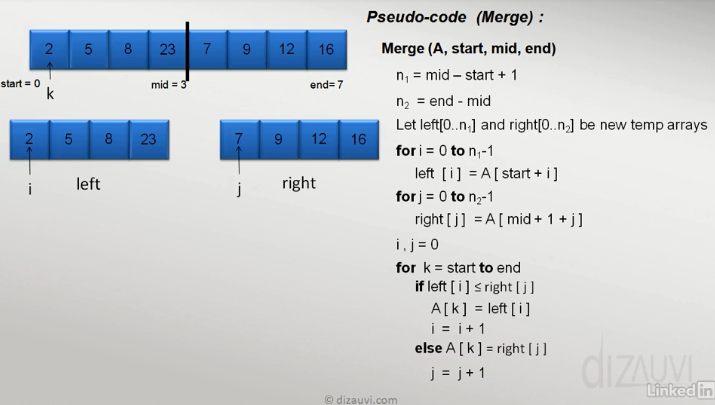




# Merge Sort Algorithm:





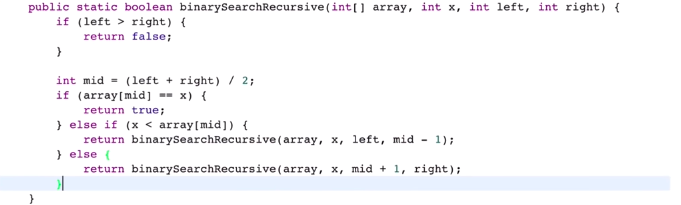


## **Array Sorting Algorithms**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Algorithm | Time Complexity | | | Space Complexity |
|  | Best | Average | Worst | Worst |
| [Quicksort](http://en.wikipedia.org/wiki/Quicksort) | Ω(n log(n)) | Θ(n log(n)) | O(n^2) | O(log(n)) |
| [Mergesort](http://en.wikipedia.org/wiki/Merge_sort) | Ω(n log(n)) | Θ(n log(n)) | O(n log(n)) | O(n) |
| [Bubble Sort](http://en.wikipedia.org/wiki/Bubble_sort) | Ω(n) | Θ(n^2) | O(n^2) | O(1) |
| [Insertion Sort](http://en.wikipedia.org/wiki/Insertion_sort) | Ω(n) | Θ(n^2) | O(n^2) | O(1) |
| [Selection Sort](http://en.wikipedia.org/wiki/Selection_sort) | Ω(n^2) | Θ(n^2) | O(n^2) | O(1) |

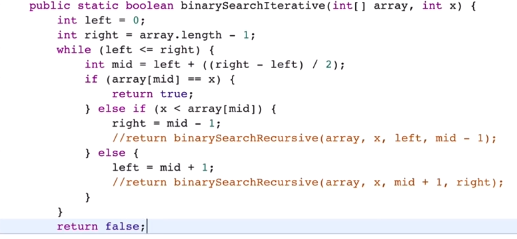
# Binary Search Algorithm:

**Recursive:**

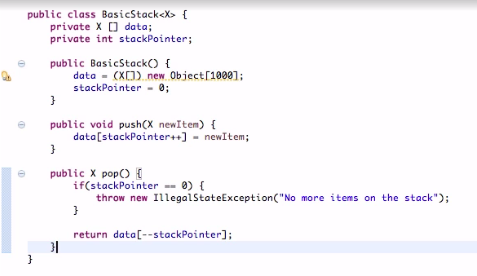


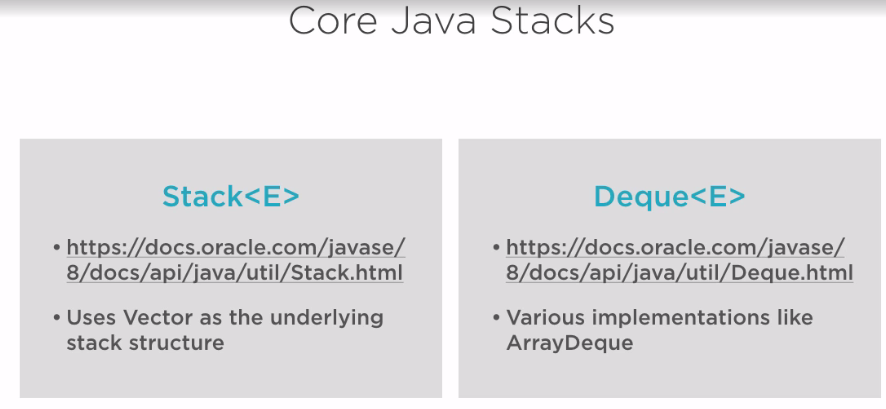


**Iterative:**

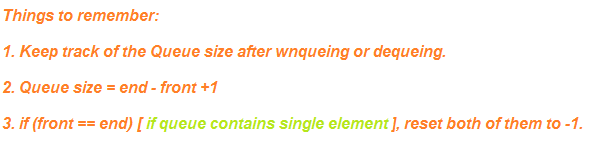


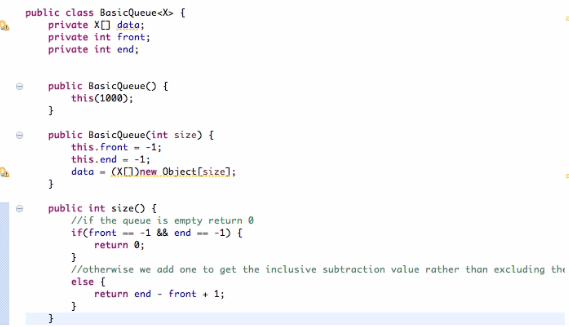
**Stack:**

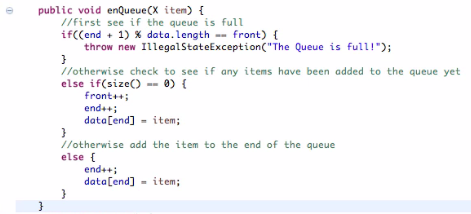


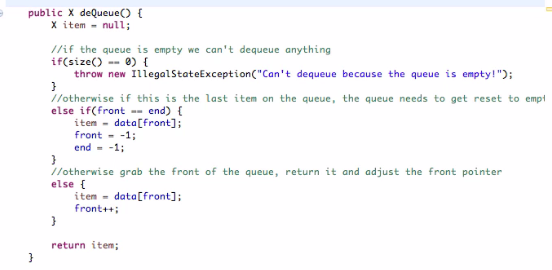


**Queue:**

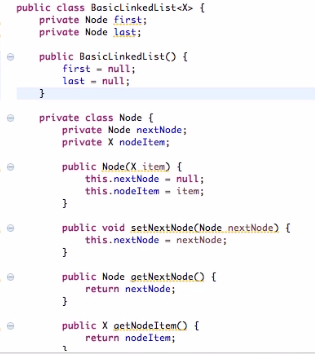
****



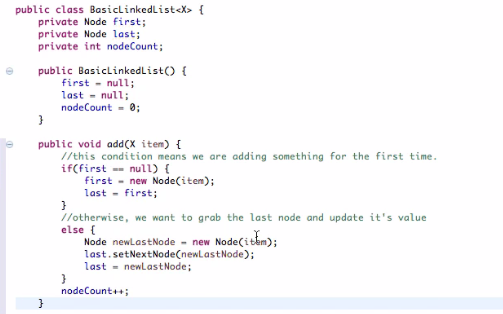




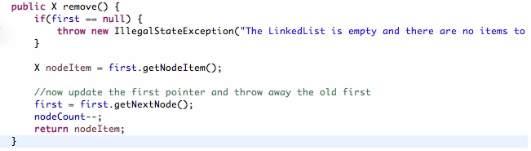
**LinkedList:**



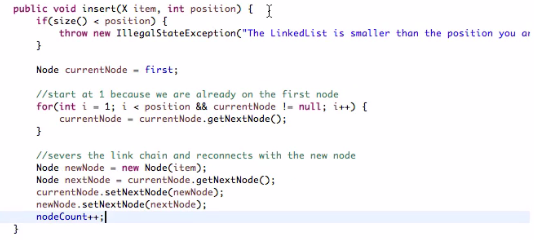
**Add Element To LinkedList:**



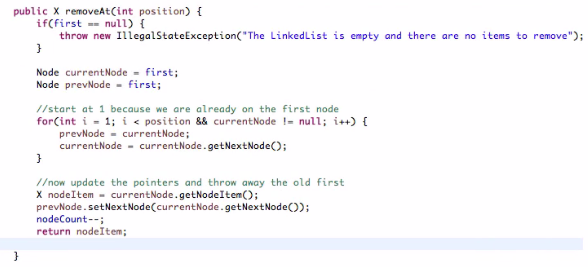
**Remove Item from LinkedList**



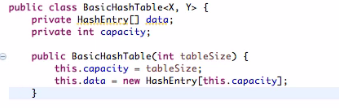
**Insert Item into LinkedList**

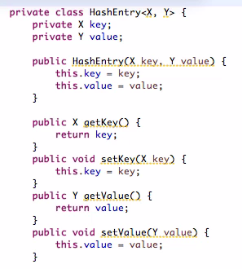


**Remove Item from LinkedList**

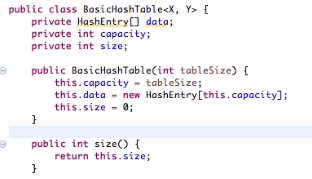


**Hash Data Structure:**

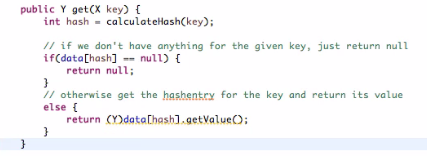


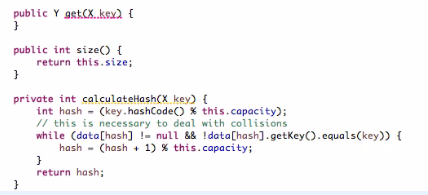


**Create Size() of Hash Table:**

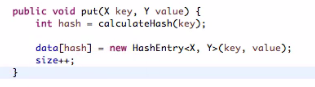


**Get value by Key from HashTable:**





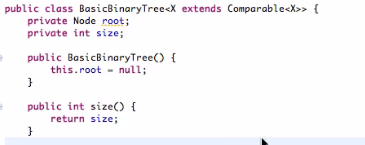
**Put Value into HashTable:**



**Binary Tree:**

 **Other getter & setter**

**Size() of Binary Tree:**



**Add Note into Binary Tree**

