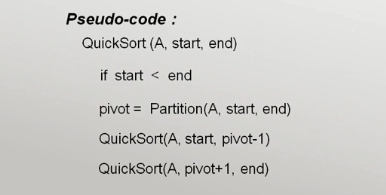
# 

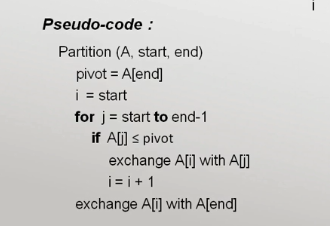
# 

# Quick Sort Algorithm:

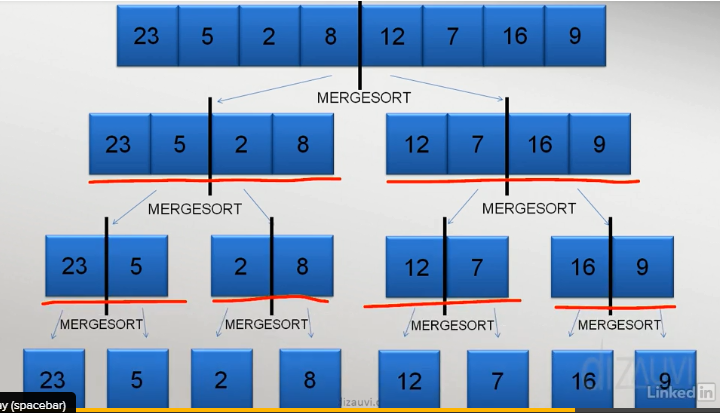
1. Average complexity: O (nlogn)
2. Worst Case: O(n2)

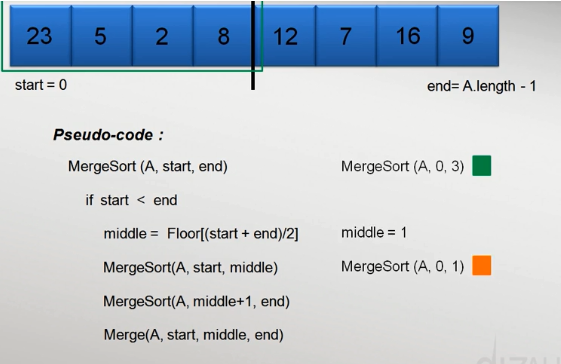
Algorithm:

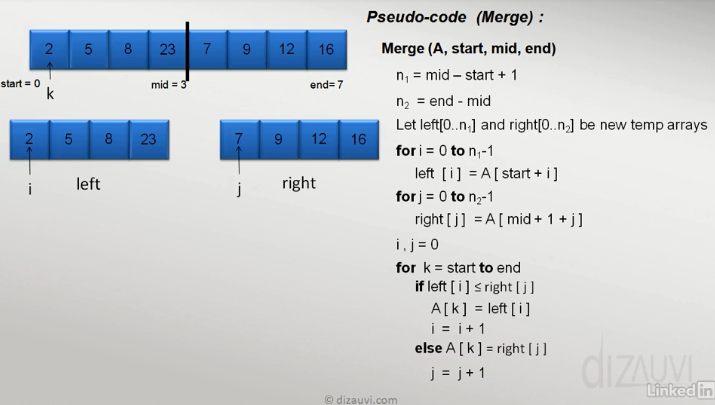




# Merge Sort Algorithm:





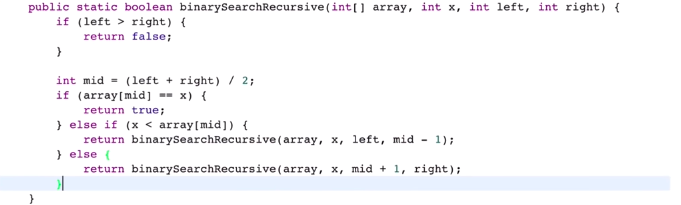


## **Array Sorting Algorithms**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 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) |

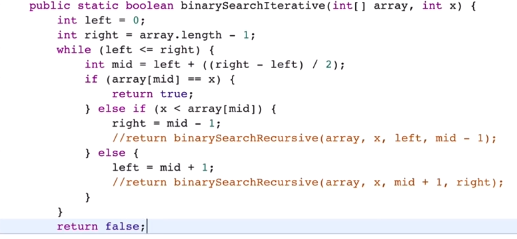
# Binary Search Algorithm:

**Recursive:**

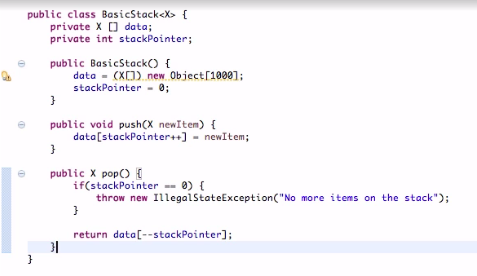


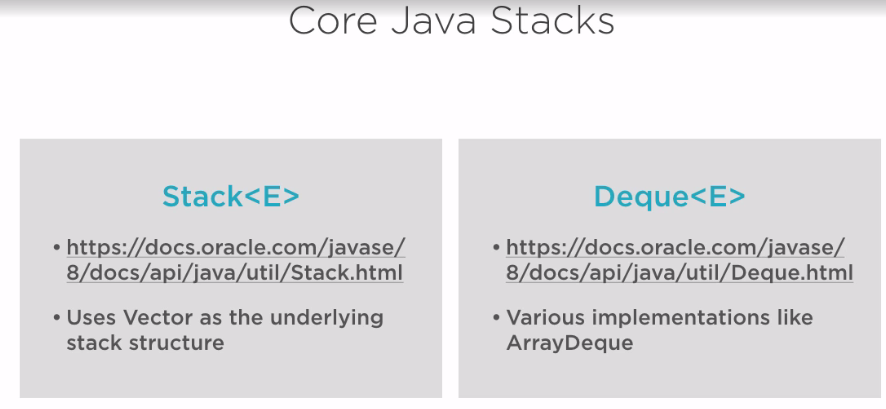


**Iterative:**

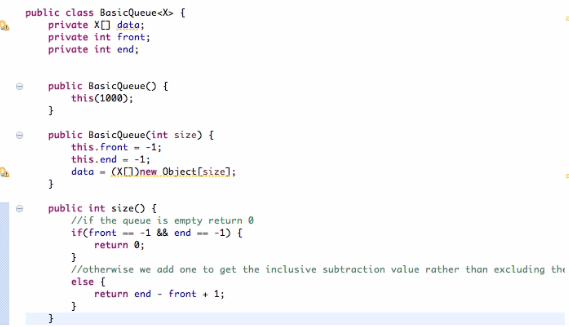


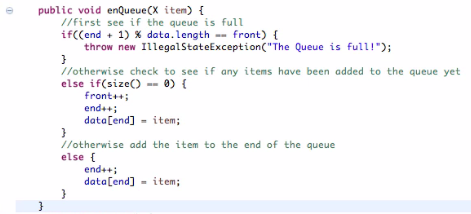
**Stack:**

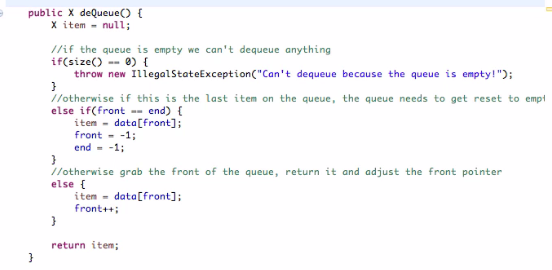




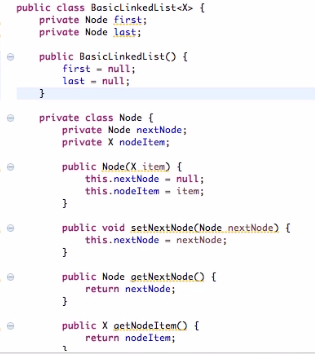
**Queue:**



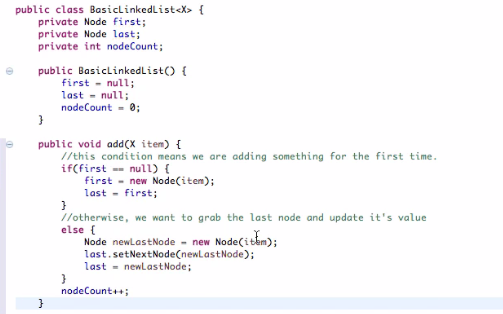




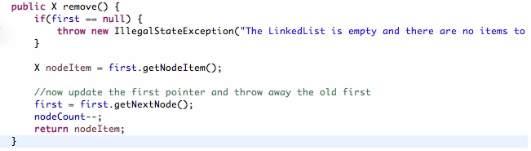
**LinkedList:**



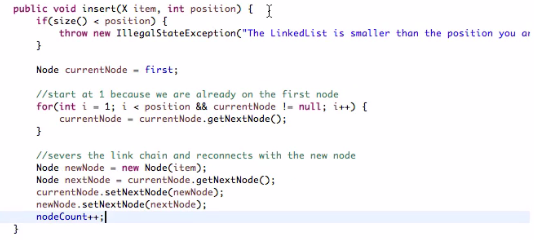
**Add Element To LinkedList:**



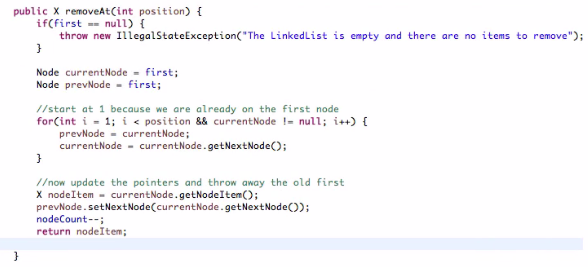
**Remove Item from LinkedList**



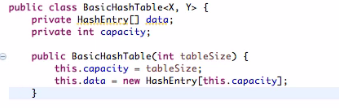
**Insert Item into LinkedList**

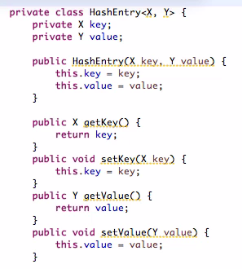


**Remove Item from LinkedList**

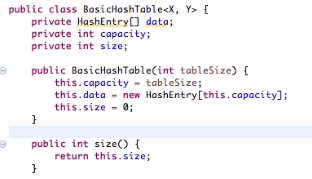


**Hash Data Structure:**

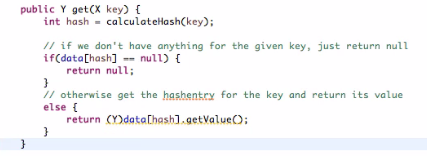


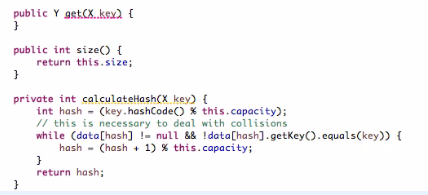


**Create Size() of Hash Table:**

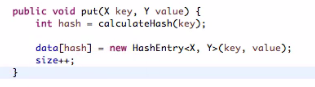


**Get value by Key from HashTable:**





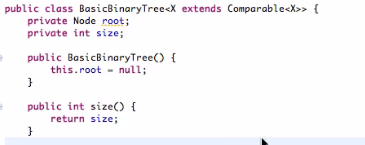
**Put Value into HashTable:**



**Binary Tree:**

 **Other getter & setter**

**Size() of Binary Tree:**



**Add Note into Binary Tree**

