Big-Oh For Common Operations and Algorithms

Searching
- Linear Search is **O**(n)
- Binary Search in **O**(log n)

Sorting
- Selection Sort is **O**(n^2)
- Insertion Sort is **O**(n^2)
- MergeSort is **O**(n log n)
- QuickSort is **O**(n log n)

ArrayLists
- Insert at front is **O**(n)
- Remove from front is **O**(n)
- Insert at end is **O**(1)
- Remove from end is **O**(1)
- Linear traversal is **O**(n)
- Random access is **O**(1)

LinkedList
- Insert at front is **O**(1)
- Remove from front is **O**(1)
- Insert at end is **O**(1)
- Remove from end is **O**(1)
- Linear traversal is **O**(n)
- Random access is **O**(n)

Stacks
- *push* and *pop* are **O**(1)

Queues
- *enqueue* and *dequeue* are **O**(1)

HashSet, HashMap
- *add*, *remove*, and *contains* are usually **O**(1), but the worst case is **O**(n)

TreeSet, TreeMap
- *add*, *remove*, and *contains* are **O**(log n) (because they are implemented as balanced binary trees)

Heaps
- Insertion is **O**(log n)
- Deletion is **O**(log n)

HeapSort
- is **O**(n log n)

Fastest Running Time				Slower Running Times		
1	(log n)	n	(n log n)	n^2	n^3	a^n

Priority Queues

Implemented with an <u>unsorted array or ArrayList</u>

- Insertion is **O**(1) (Insertions at the end of the array)
- *removeMin* is **O**(n) (Find it (**O**(n)) and delete it.)

Implemented with a <u>sorted array or ArrayList</u>

- Insertion is **O**(n) (Find insertion spot and shift elements over)
- *removeMin* is **O**(1) (Min element is in the last position.)

Implemented with an <u>unsorted LinkedList</u>

- Insertion is **O**(1) (Insertions at the front of the list)
- *removeMin* is **O**(n) (Find it (**O**(n)) and delete it.)

Implemented with a <u>sorted LinkedList</u>

- Insertion is **O**(n) (Find insertion spot)
- *removeMin* is **O**(1)

Implemented with a <u>TreeSet</u>

- Insertion is **O**(log n)
- *removeMin* is **O**(log n)

Implemented with a <u>Heap</u>

- Insertion is **O**(log n)
- *removeMin* is **O**(log n)