Note on Bubble Sort , Stack And Arraylist in Java

Name: Akshay Murali

Roll No.: 07

Section: MCA A **BATCH**

BUBBLE SORT

Bubble sort is a simple sorting algorithm that compares adjacent elements of an array and swaps them if the element on the right is smaller than the one on the left.

It is an in-place sorting algorithm i.e. no extra space is needed for this sort, the array itself is modified.

- The first step is to create a method, bubbleSort, that takes the array as the input to be sorted, sort_arr, and the length of the array, len,.
- The second step is to create an outer for loop which will iterate over each element of the array.
- The third step is to create an inner for loop . This for loop starts from the first element of the array till the second last index, (len - i - 1).

  Each time one index less than the last is traversed as at the end of each iteration, the largest element for that iteration reaches the end.
- Within the nested for loop is the if condition. This checks that if the element on the left is greater than that on the right. If so, it swaps the two elements.

Time Complexity

- Since there are two nested loops within the algorithm, the time complexity will be *O(n^2)* where *n* is equivalent to the length of the array to be sorted.

Stack Data Structure

Stack is a linear data structure which follows a particular order in which the operations are performed. The order may be LIFO(Last In First Out) or FILO(First In Last Out).

Mainly the following three basic operations are performed in the stack:

- **Push:** Adds an item in the stack. If the stack is full, then it is said to be an Overflow condition.
- **Pop:** Removes an item from the stack. The items are popped in the reversed order in which they are pushed. If the stack is empty, then it is said to be an Underflow condition.
- **Peek or Top:** Returns top element of stack.
- **isEmpty:** Returns true if stack is empty, else false.

How to understand a stack practically?

There are many real-life examples of a stack. Consider the simple example of plates stacked over one another in a canteen. The plate which is at the top is the first one to be removed, i.e. the plate which has been placed at the bottommost position remains in the stack for the longest period of time. So, it can be simply seen to follow LIFO/FILO order.

Time Complexities of operations on stack:

push(), pop(), isEmpty() and peek() all take O(1) time. We do not run any loop in any of these operations.

ArrayList

ArrayList is a part of collection framework and is present in java.util package. It provides us with dynamic arrays in Java. Though, it may be slower than standard arrays but can be helpful in programs where lots of manipulation in the array is needed. This class is found in java.util package.

Performing Various Operations on ArrayList

Adding Elements: In order to add an element to an ArrayList, we can use the add() method. This method is overloaded to perform multiple operations based on different parameters. They are:

- **add(Object):** This method is used to add an element at the end of the ArrayList.
- **add(int index, Object):** This method is used to add an element at a specific index in the ArrayList.

Changing Elements: After adding the elements, if we wish to change the element, it can be done using the set() method. Since an ArrayList is indexed, the element which we wish to change is referenced by the index of the element. Therefore, this method takes an index and the updated element which needs to be inserted at that index.

Removing Elements: In order to remove an element from an ArrayList, we can use the remove() method. This method is overloaded to perform multiple operations based on different parameters. They are:

- **remove(Object):** This method is used to simply remove an object from the ArrayList. If there are multiple such objects, then the first occurrence of the object is removed.
- **remove(int index):** Since an ArrayList is indexed, this method takes an integer value which simply removes the element present at that specific index in the ArrayList. After removing the element, all the elements are moved to the left to fill the space and the indices of the objects are updated.

Iterating the ArrayList: There are multiple ways to iterate through the ArrayList. The most famous ways are by using the basic for loop in combination with a get() method to get the element at a specific index and the advanced for loop.

Important Features:

- ArrayList inherits AbstractList class and implements List interface.
- ArrayList is initialized by the size. However, the size is increased automatically if the collection grows or shrinks if the objects are removed from the collection.
- Java ArrayList allows us to randomly access the list.

- ArrayList can not be used for primitive types, like int, char, etc. We need a wrapper class for such cases.