Learning Consolidation

Implement Sorting on Arrays

In this sprint, you have learned to:

- Explain sorting
- Explore the time taken for execution of a sorting algorithm
- Explore quick sort

Sorting an Array

- Sorting an array means to arrange the elements in the array in ascending or descending order.
- Sorting can be used in multiple situations when dealing with data in the array.
  - If student roll numbers are stored in an array, they need to be in sorted order whenever a teacher takes roll call.
  - To determine the top student of the year, the array that holds student scores must be in the order of highest performers.

Sorting Algorithms

- Sorting can be performed on array elements by using multiple sorting algorithms.
- Some sorting algorithms are as follows:
  - Bubble sort
  - Selection sort
  - Insertion sort
  - Merge sort
  - Quick sort
  - Heap sort
- The simplest sorting algorithms are bubble sort and insertion sort.

Time Taken for a Sort Algorithm

- The time taken for a sort algorithm on an array can be determined by using the `System.currentTimeMillis()` method.
- The `System.currentTimeMillis()` method can be called before and after calling the method for the sorting of the array.
- The time delay can be obtained by subtracting both the values as shown below.

```
long start = System.currentTimeMillis();
int[] sortedArray = bubbleSort.bubbleSort(gradesOfStudents);
long end = System.currentTimeMillis();
System.out.println("Time taken for bubbleSort to run : "+(end-start)+" milliseconds");
```

Quick Sort

- Quick sort is one of the most efficient sorting algorithms.
- It is based on the divide-and-conquer approach.
- It successively divides the problems into smaller parts until the problems become so small that they can be directly solved.

Implementing Quick Sort

- To implement quick sort
  - Select an element from the list as the pivot.
  - Partition the list into two parts, so that
    - All the elements towards the left end of the list are smaller than the pivot.
    - All the elements towards the right end of the list are greater than the pivot.
  - Store the pivot at the correct position between the two parts of the list.
    - Repeat this process for each of the two sub lists created after partitioning.
    - This process continues until one element is left in each sub list.

Representing Quick Sort