# PRACTICE SET – 6

Meena Rhoshini C

Nov 18, 24

1. **Bubble Sort**

```
import java.util.Arrays;

public class BubbleSort {

    public static void bubbleSort(int[] array) {
        int n = array.length;
        for (int i = 0; i < n - 1; i++) {
            for (int j = 0; j < n - i - 1; j++) {
                if (array[j] > array[j + 1]) {
                    int temp = array[j];
                    array[j] = array[j + 1];
                    array[j + 1] = temp;
                }
            }
        }
    }

    public static void main(String[] args) {
        int[] arr = {4, 1, 3, 9, 7};
        bubbleSort(arr);
        System.out.println(Arrays.toString(arr));

        int[] arr1 = {10, 9, 8, 7, 6, 5, 4, 3, 2, 1};
        bubbleSort(arr1);
        System.out.println(Arrays.toString(arr1));

        int[] arr2 = {1, 2, 3, 4, 5};
        bubbleSort(arr2);
        System.out.println(Arrays.toString(arr2));
    }
}
```

OUTPUT:

```
C:\Users\Rhoshini\Desktop\dsa>javac  BubbleSort.java

C:\Users\Rhoshini\Desktop\dsa>java  BubbleSort
[1, 3, 4, 7, 9]
[1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
[1, 2, 3, 4, 5]
```

Time Complexity: O(n^2)
Space Complexity: O(1)

**2. Quick Sort**

```
import java.util.Arrays;

public class QuickSort {

   public static void quickSort(int[] arr, int low, int high) {
      if (low < high) {
         int pivotIndex = partition(arr, low, high);
         quickSort(arr, low, pivotIndex - 1);
         quickSort(arr, pivotIndex + 1, high);
      }
   }

   public static int partition(int[] arr, int low, int high) {
      int pivot = arr[high];
      int i = low - 1;
      for (int j = low; j < high; j++) {
         if (arr[j] <= pivot) {
            i++;
            int temp = arr[i];
            arr[i] = arr[j];
            arr[j] = temp;
         }
      }
      int temp = arr[i + 1];
      arr[i + 1] = arr[high];
      arr[high] = temp;
      return i + 1;
   }

   public static void main(String[] args) {
      int[] arr1 = {4, 1, 3, 9, 7};
      quickSort(arr1, 0, arr1.length - 1);
      System.out.println(Arrays.toString(arr1));

      int[] arr2 = {2, 1, 6, 10, 4, 1, 3, 9, 7};
      quickSort(arr2, 0, arr2.length - 1);
      System.out.println(Arrays.toString(arr2));

      int[] arr3 = {5, 5, 5, 5};
      quickSort(arr3, 0, arr3.length - 1);
      System.out.println(Arrays.toString(arr3));
   }
}
```

OUTPUT:

```
C:\Users\Rhoshini\Desktop\dsa>javac QuickSort.java

C:\Users\Rhoshini\Desktop\dsa>java QuickSort
[1, 3, 4, 7, 9]
[1, 1, 2, 3, 4, 6, 7, 9, 10]
[5, 5, 5, 5]
```

Time Complexity: O(n^2)
Space Complexity: O(logn)

---

3. **Non Repeating Character**

```
import java.util.LinkedHashMap;
import java.util.Map;

public class NonRepeatingCharacter {

    public static char firstNonRepeatingChar(String s) {
        Map<Character, Integer> charCount = new LinkedHashMap<>();
        for (char c : s.toCharArray()) {
            charCount.put(c, charCount.getOrDefault(c, 0) + 1);
        }
        for (Map.Entry<Character, Integer> entry : charCount.entrySet()) {
            if (entry.getValue() == 1) {
                return entry.getKey();
            }
        }
        return '$';
    }

    public static void main(String[] args) {
        String str1 = "geeksforgeeks";
        System.out.println(firstNonRepeatingChar(str1));

        String str2 = "racecar";
        System.out.println(firstNonRepeatingChar(str2));

        String str3 = "aabbccc";
        System.out.println(firstNonRepeatingChar(str3));
    }
}
```

OUTPUT:

```
C:\Users\Rhoshini\Desktop\dsa>javac NonRepeatingCharacter.java

C:\Users\Rhoshini\Desktop\dsa>java NonRepeatingCharacter
f
e
$
```

Time Complexity: O(n)
Space Complexity: O(n)

---

4. **Edit Distance**

```
public class EditDistance {

    public static int minEditDistance(String s1, String s2) {
        int m = s1.length();
        int n = s2.length();
        int[][] dp = new int[m + 1][n + 1];

        for (int i = 0; i <= m; i++) {
            for (int j = 0; j <= n; j++) {
                if (i == 0) {
                    dp[i][j] = j;
                } else if (j == 0) {
                    dp[i][j] = i;
                } else if (s1.charAt(i - 1) == s2.charAt(j - 1)) {
                    dp[i][j] = dp[i - 1][j - 1];
                } else {
                    dp[i][j] = 1 + Math.min(dp[i - 1][j - 1], Math.min(dp[i - 1][j], dp[i][j - 1]));
                }
            }
        }
        return dp[m][n];
    }

    public static void main(String[] args) {
        String s1 = "geek";
        String s2 = "gesek";
        System.out.println(minEditDistance(s1, s2));

        String s3 = "gfg";
        String s4 = "gfg";
        System.out.println(minEditDistance(s3, s4));

        String s5 = "abc";
        String s6 = "def";
        System.out.println(minEditDistance(s5, s6));
    }
}
```

OUTPUT:

```
C:\Users\Rhoshini\Desktop\dsa>javac EditDistance.java

C:\Users\Rhoshini\Desktop\dsa>java EditDistance
1
0
3
```

Time complexity: O(m*n)
Space complexity: O(m*n)

**5. k largest elements**

```
import java.util.*;

public class KLargestElements {

   public static int[] findKLargest(int[] arr, int k) {
      PriorityQueue<Integer> minHeap = new PriorityQueue<>();
      for (int num : arr) {
         minHeap.add(num);
         if (minHeap.size() > k) {
            minHeap.poll();
         }
      }
      int[] result = new int[k];
      for (int i = k - 1; i >= 0; i--) {
         result[i] = minHeap.poll();
      }
      return result;
   }

   public static void main(String[] args) {
      int[] arr1 = {12, 5, 787, 1, 23};
      int k1 = 2;
      System.out.println(Arrays.toString(findKLargest(arr1, k1)));

      int[] arr2 = {1, 23, 12, 9, 30, 2, 50};
      int k2 = 3;
      System.out.println(Arrays.toString(findKLargest(arr2, k2)));

      int[] arr3 = {12, 23};
      int k3 = 1;
      System.out.println(Arrays.toString(findKLargest(arr3, k3)));
   }
}
```

OUTPUT:

```
C:\Users\Rhoshini\Desktop\dsa>javac KLargestElements.java

C:\Users\Rhoshini\Desktop\dsa>java KLargestElements
[787, 23]
[50, 30, 23]
[23]
```

Time complexity: O(nlogk)
Space Complexity : O(k)

6. **Form the Largest Number**

```
import java.util.*;

public class LargestNumber {

   public static String formLargestNumber(int[] arr) {
      String[] strArr = new String[arr.length];
      for (int i = 0; i < arr.length; i++) {
         strArr[i] = String.valueOf(arr[i]);
      }
      Arrays.sort(strArr, (a, b) -> (b + a).compareTo(a + b));
      if (strArr[0].equals("0")) {
         return "0";
      }
      StringBuilder result = new StringBuilder();
      for (String num : strArr) {
         result.append(num);
      }
      return result.toString();
   }

   public static void main(String[] args) {
      int[] arr1 = {3, 30, 34, 5, 9};
      System.out.println(formLargestNumber(arr1));

      int[] arr2 = {54, 546, 548, 60};
      System.out.println(formLargestNumber(arr2));

      int[] arr3 = {3, 4, 6, 5, 9};
      System.out.println(formLargestNumber(arr3));
   }
}
```

OUTPUT:

```
C:\Users\Rhoshini\Desktop\dsa>javac LargestNumber.java

C:\Users\Rhoshini\Desktop\dsa>java LargestNumber
9534330
6054854654
96543
```

Time Complexity: O(nlogn)
Space Complexity: O(n)