DSA PRACTICE PROBLEMS – SET 6

Madhulekha R – AIML - [18/11/2024]

1. **Question:** Bubble sort

   Code:

```
import java.io.*;

class Main5{
    public static void main(String args[]){
        int arr[] = {64, 34, 25, 12, 22, 11, 90};
        int n = arr.length;
        bubbleSort(arr, n);
        System.out.println("Sorted array: ");
        printArray(arr, n);
    }

    static void bubbleSort(int arr[], int n){
        int i, j, temp;
        boolean swapped;
        for (i = 0; i < n - 1; i++) {
            swapped = false;
            for (j = 0; j < n - i - 1; j++) {
                if (arr[j] > arr[j + 1]) {
                    temp = arr[j];
                    arr[j] = arr[j + 1];
                    arr[j + 1] = temp;
                    swapped = true;
                }
            }
            if (swapped == false)
                break;
        }
    }

    static void printArray(int arr[], int size){
        int i;
        for (i = 0; i < size; i++){
            System.out.print(arr[i] + " ");
        }
        System.out.println();
```

```
    }
}
```

Output:

```
C:\Users\HP\Documents>javac Main5.java

C:\Users\HP\Documents>java Main5
Sorted array:
11 12 22 25 34 64 90
```

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

2. **Question:** Quick sort

Code:

```
import java.util.Arrays;

class Main5{
    public static void main(String[] args) {
        int[] arr = {10, 7, 8, 9, 1, 5};
        int n = arr.length;
        quickSort(arr, 0, n - 1);
        for (int val : arr) {
            System.out.print(val + " ");
        }
    }

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

```
    static void swap(int[] arr, int i, int j) {
        int temp = arr[i];
        arr[i] = arr[j];
        arr[j] = temp;
    }

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

Output:

```
C:\Users\HP\Documents>javac Main5.java

C:\Users\HP\Documents>java Main5
1 5 7 8 9 10
```

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

3. **Question:** Non-Repeating character

Code:

```
class Main5{
    public static void main(String[] args) {
        String s = "racecar";
        System.out.println(nonRepeatingChar(s));
    }

    static char nonRepeatingChar(String s) {
        int n = s.length();
        for (int i = 0; i < n; ++i) {
            boolean found = false;
            for (int j = 0; j < n; ++j) {
                if (i != j && s.charAt(i) == s.charAt(j)) {
                    found = true;
                    break;
                }
```

```
            }
            if (found == false)
                return s.charAt(i);
        }
        return '$';
    }
}
```

Output:

```
C:\Users\HP\Documents>javac Main5.java

C:\Users\HP\Documents>java Main5
e
```

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

4. **Question:** Edit Distance

Code:

```
public class Main5{
    public static void main(String[] args) {
        String s1 = "GEEXSFRGEEKKS";
        String s2 = "GEEKSFORGEEKS";
        System.out.println(func1(s1, s2));
    }

    private static int func(String s1, String s2, int m, int n, int[][] memo){
        if (m == 0) return n;
        if (n == 0) return m;
        if (memo[m][n] != -1) return memo[m][n];
        if (s1.charAt(m - 1) == s2.charAt(n - 1)) {
            memo[m][n] = func(s1, s2, m - 1, n - 1, memo);
        } else {
            int insert = func(s1, s2, m, n - 1, memo);
            int remove = func(s1, s2, m - 1, n, memo);
            int replace = func(s1, s2, m - 1, n - 1, memo);
            memo[m][n] = 1 + Math.min(insert, Math.min(remove, replace));
        }
        return memo[m][n];
    }
```

```
    public static int func1(String s1, String s2) {
        int m = s1.length(), n = s2.length();
        int[][] memo = new int[m + 1][n + 1];
        for (int i = 0; i <= m; i++) {
            for (int j = 0; j <= n; j++) {
                memo[i][j] = -1;
            }
        }
        return func(s1, s2, m, n, memo);
    }
}
```

Output:

```
C:\Users\HP\Documents>javac Main5.java

C:\Users\HP\Documents>java Main5
3
```

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

5. **Question:** k largest elements

Code:

```
import java.util.*;

class Main5{
    static int partition(int[] arr, int left, int right){
        int pivot = arr[right];
        int i = left;
        for (int j = left; j < right; j++){
            if (arr[j] >= pivot) {
                int temp = arr[i];
                arr[i] = arr[j];
                arr[j] = temp;
                i++;
            }
        }

        int temp = arr[i];
        arr[i] = arr[right];
```

```
        arr[right] = temp;
        return i;
    }

    static void quickSelect(int[] arr, int left, int right, int k) {
        if (left <= right) {
            int pivotIdx = partition(arr, left, right);
            int leftCnt = pivotIdx - left + 1;
            if (leftCnt == k)
                return;
            if (leftCnt > k)
                quickSelect(arr, left, pivotIdx - 1, k);
            else
                quickSelect(arr, pivotIdx + 1, right, k - leftCnt);
        }
    }

    static ArrayList<Integer> kLargest(int[] arr, int k) {
        quickSelect(arr, 0, arr.length - 1, k);
        ArrayList<Integer> res = new ArrayList<>();
          for(int i = 0; i < k; i++)
              res.add(arr[i]);
          Collections.sort(res, Collections.reverseOrder());
        return res;
    }

    public static void main(String[] args) {
        int[] arr = {1, 23, 12, 9, 30, 2, 50};
        int k = 3;
        ArrayList<Integer> res = kLargest(arr, k);
        for (int ele : res)
            System.out.print(ele + " ");
    }
}
```

Output:

```
C:\Users\HP\Documents>javac Main5.java

C:\Users\HP\Documents>java Main5
50 30 23
```

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

6. **Question:** Form the largest numbers

Code:

```
import java.util.*;

class Main5{
    public static void main(String[] args) {
        int[] arr = { 3, 30, 34, 5, 9 };
        System.out.println(findLargest(arr));
    }

    static boolean myCompare(String s1, String s2) {
        return (s1 + s2).compareTo(s2 + s1) > 0;
    }

    static String findLargest(int[] arr){
        ArrayList<String> numbers = new ArrayList<>();
        for (int ele : arr) {
            numbers.add(Integer.toString(ele));
        }

        Collections.sort(numbers, (s1, s2) -> myCompare(s1, s2) ? -1 : 1);
        if (numbers.get(0).equals("0")) {
            return "0";
        }
        StringBuilder res = new StringBuilder();
        for (String num : numbers) {
            res.append(num);
        }

        return res.toString();
    }
}
```

Output:

```
C:\Users\HP\Documents>javac Main5.java

C:\Users\HP\Documents>java Main5
9534330
```

Time Complexity: O(n log n)
Space Complexity: O(1)