C-DAC Mumbai

Date 26/09/2024

Subject: Algorithm and Data Structure
Assignment 1

1. Printing Patterns
Program:

```
import java.util.Scanner;

public class StarPattern {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        System.out.print("Enter number of rows: ");
        int n = sc.nextInt();

        for (int i = 1; i <= n; i++) {
            for (int j = 1; j <= i; j++) {
                System.out.print("*");
            }
            System.out.println();
        }
        sc.close();
    }
}
```

Flowchart:

1. Start
2. Input n
3. Outer loop from 1 to n
4. Inner loop from 1 to current value of outer loop
5. Print "*"
6. Move to next line
7. End

Explanation: This program prints a right-angled triangle pattern. The outer loop runs for each row, while the inner loop prints stars corresponding to the row number.
Output:
markdown

```
Enter number of rows: 3
*
**
***
```

Time Complexity: O(n^2)

Space Complexity: O(1)

2. Remove Array Duplicates
Program:

```
import java.util.*;

public class RemoveDuplicates {
    public static int removeDuplicates(int[] arr) {
        if (arr.length == 0) return 0;
        int j = 0;
        for (int i = 1; i < arr.length; i++) {
            if (arr[i] != arr[j]) {
                j++;
                arr[j] = arr[i];
            }
        }
        return j + 1;
    }

    public static void main(String[] args) {
        int[] arr = {1, 1, 2};
        int length = removeDuplicates(arr);
        System.out.println("New length: " + length);
    }
}
```

Flowchart:

1. Start
2. Input array
3. Initialize j = 0
4. Loop from 1 to length of array
5. If current element is not equal to previous, update j
6. Return j + 1 as the new length
7. End

Explanation: This program removes duplicates from a sorted array in-place and returns the new length of the array.

Output:
perl

```
New length: 2
```

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

3. Remove White Spaces from String
Program:

```
public class RemoveWhiteSpace {
   public static String removeSpaces(String str) {
      return str.replaceAll("\\s", "");
   }

   public static void main(String[] args) {
      String input = "  Java   Programming  ";
      String output = removeSpaces(input);
      System.out.println(output);
   }
}
```
Flowchart:

1. Start
2. Input string
3. Replace all spaces
4. Output result
5. End

Explanation: This program removes all white spaces from the input string using the replaceAll method.

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

4. Reverse a String
Program:

```
public class ReverseString {
   public static String reverse(String str) {
      StringBuilder sb = new StringBuilder(str);
      return sb.reverse().toString();
   }

   public static void main(String[] args) {
      String input = "hello";
      String output = reverse(input);
      System.out.println(output);
   }
}
```
Flowchart:

1. Start

2. Input string
3. Reverse string using StringBuilder
4. Output result
5. End

Explanation: This program reverses the input string using the StringBuilder class’s reverse method.
Output:

```
olleh
```

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

5. Reverse Array in Place
Program:

```java
import java.util.Arrays;

public class ReverseArray {
  public static void reverseArray(int[] arr) {
    int left = 0, right = arr.length - 1;
    while (left < right) {
      int temp = arr[left];
      arr[left] = arr[right];
      arr[right] = temp;
      left++;
      right--;
    }
  }

  public static void main(String[] args) {
    int[] arr = {1, 2, 3, 4};
    reverseArray(arr);
    System.out.println(Arrays.toString(arr));
  }
}
```

Flowchart:

1. Start
2. Input array
3. Swap elements from start and end
4. Repeat until array is reversed
5. Output result
6. End

Explanation: This program reverses the input array in place by swapping the elements from both ends.
Output:

```
csharp
[4, 3, 2, 1]
```
Time Complexity: O(n)
Space Complexity: O(1)

6. Reverse Words in a String
Program:

```
public class ReverseWords {
    public static String reverseWords(String str) {
        String[] words = str.split(" ");
        StringBuilder reversed = new StringBuilder();
        for (int i = words.length - 1; i >= 0; i--) {
            reversed.append(words[i]).append(" ");
        }
        return reversed.toString().trim();
    }

    public static void main(String[] args) {
        String input = "Hello World";
        String output = reverseWords(input);
        System.out.println(output);
    }
}
```
Flowchart:

1. Start
2. Input string
3. Split string into words
4. Reverse words and join them
5. Output result
6. End

Explanation: This program reverses the words in the input string while keeping the characters of each word intact.
Output:
```
World Hello
```
Time Complexity: O(n)
Space Complexity: O(n)

7. Reverse a Number
Program:

```
public class ReverseNumber {
    public static int reverse(int num) {
        int reversed = 0;
        boolean isNegative = num < 0;
        num = Math.abs(num);

        while (num > 0) {
            int digit = num % 10;
            reversed = reversed * 10 + digit;
            num /= 10;
        }

        return isNegative ? -reversed : reversed;
    }

    public static void main(String[] args) {
        int number = 12345;
        System.out.println("Reversed number: " + reverse(number));

        number = -9876;
        System.out.println("Reversed number: " + reverse(number));
    }
}
```

Flowchart:

1. Start
2. Input number
3. Check if the number is negative
4. Loop through the digits
    - Extract the last digit
    - Append it to the reversed number
5. Multiply reversed by -1 if the original number was negative
6. Output reversed number
7. End

Explanation: This program reverses the digits of an integer. It handles both positive and negative numbers by first checking the sign and using Math.abs() to work with positive digits.

Output:

```
yaml
Reversed number: 54321
Reversed number: -6789
```

Time Complexity: O(log10(n))
Space Complexity: O(1)

8. Array Manipulation
Program:

```
public class ArrayManipulation {
    public static long arrayManipulation(int n, int[][] queries) {
        long[] arr = new long[n + 1];

        for (int[] query : queries) {
            int start = query[0] - 1;
            int end = query[1];
            int value = query[2];

            arr[start] += value;
            if (end < n) arr[end] -= value;
        }

        long max = 0, current = 0;
        for (long x : arr) {
            current += x;
            if (current > max) max = current;
        }

        return max;
    }

    public static void main(String[] args) {
        int[][] queries = {{1, 2, 100}, {2, 5, 100}, {3, 4, 100}};
        System.out.println(arrayManipulation(5, queries)); // Output: 200

        queries = new int[][] {{1, 3, 50}, {2, 4, 70}};
        System.out.println(arrayManipulation(4, queries)); // Output: 120
    }
}
```

Flowchart:

1. Start
2. Input n and queries
3. Initialize an array of length n+1
4. For each query, update the start and end indices in the array
5. Traverse the array to calculate cumulative values and find the maximum value
6. Output the maximum value
7. End

Explanation: This program performs efficient range updates on an array using a "difference array" technique. Instead of updating all elements within a range for each query, the start and end of the range are adjusted, and cumulative sums are used to compute the final values.

Output:

```
200
120
```

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

9. String Palindrome
Program:
java
Copy code

```
public class PalindromeChecker {
    public static boolean isPalindrome(String str) {
        int left = 0, right = str.length() - 1;

        while (left < right) {
            if (str.charAt(left) != str.charAt(right)) {
                return false;
            }
            left++;
            right--;
        }

        return true;
    }

    public static void main(String[] args) {
        String input = "madam";
        System.out.println(isPalindrome(input)); // Output: true

        input = "hello";
        System.out.println(isPalindrome(input)); // Output: false
    }
}
```

Flowchart:

1. Start
2. Input string
3. Initialize two pointers (start and end)
4. Compare characters from both ends
5. If a mismatch is found, return false
6. Return true if the entire string is checked
7. End

Explanation: This program checks if a string is a palindrome by comparing characters from both ends. If all corresponding characters match, the string is a palindrome.

Output:

```
arduino
true
false
```

Time Complexity: O(n)

Space Complexity: O(1)

10. Array Left Rotation

Program:

```
import java.util.Arrays;

public class ArrayLeftRotation {
    public static void rotateLeft(int[] arr, int d) {
        d = d % arr.length;
        reverse(arr, 0, d - 1);
        reverse(arr, d, arr.length - 1);
        reverse(arr, 0, arr.length - 1);
    }

    public static void reverse(int[] arr, int start, int end) {
        while (start < end) {
            int temp = arr[start];
            arr[start] = arr[end];
            arr[end] = temp;
            start++;
            end--;
        }
    }

    public static void main(String[] args) {
        int[] arr = {1, 2, 3, 4, 5};
        rotateLeft(arr, 2);
        System.out.println(Arrays.toString(arr)); // Output: [3, 4, 5, 1, 2]

        arr = new int[]{10, 20, 30, 40};
        rotateLeft(arr, 1);
        System.out.println(Arrays.toString(arr)); // Output: [20, 30, 40, 10]
    }
}
```

Flowchart:

1. Start
2. Input array and rotation count d
3. Rotate the array using the reverse technique in three steps
4. Output rotated array
5. End

Explanation: This program rotates an array to the left by d positions using the reversal method. It first reverses the first d elements, then the remaining elements, and finally the whole array.

Output:

csharp

```
[3, 4, 5, 1, 2]
[20, 30, 40, 10]
```

Time Complexity: O(n)

Space Complexity: O(1)