Solution 23-11-23

662. Maximum Width of Binary Tree

```
import java.util.LinkedList;
import java.util.Queue;
import java.util.Scanner;

class TreeNode {
    int val;
    TreeNode left, right;

    public TreeNode(int item) {
        val = item;
        left = right = null;
    }
}

public class MaximumWidthOfBinaryTree {

    // Function to find the maximum width of a binary tree
    public static int maxWidth(TreeNode root) {
        if (root == null) {
            return 0;
        }

        int maxWidth = 0;
        Queue<TreeNode> queue = new LinkedList<>();
        queue.add(root);

        while (!queue.isEmpty()) {
            int count = queue.size();
```

```
            maxWidth = Math.max(maxWidth, count);

            while (count-- > 0) {
                TreeNode front = queue.poll();

                if (front.left != null) {
                    queue.add(front.left);
                }

                if (front.right != null) {
                    queue.add(front.right);
                }
            }
        }

        return maxWidth;
    }

    // Function to build a binary tree from user input
    public static TreeNode buildTree() {
        Scanner scanner = new Scanner(System.in);
        System.out.println("Enter the value for the root node: ");
        int rootValue = scanner.nextInt();
        TreeNode root = new TreeNode(rootValue);

        Queue<TreeNode> queue = new LinkedList<>();
        queue.add(root);

        while (!queue.isEmpty()) {
            TreeNode current = queue.poll();
```

```
        System.out.println("Enter the value for the left child of " + current.val + " (Enter -1 if no left child): ");
        int leftValue = scanner.nextInt();
        if (leftValue != -1) {
            current.left = new TreeNode(leftValue);
            queue.add(current.left);
        }

        System.out.println("Enter the value for the right child of " + current.val + " (Enter -1 if no right child): ");
        int rightValue = scanner.nextInt();
        if (rightValue != -1) {
            current.right = new TreeNode(rightValue);
            queue.add(current.right);
        }
      }

      return root;
   }

   public static void main(String[] args) {
      TreeNode root = buildTree();
      int result = maxWidth(root);
      System.out.println("Maximum width of the binary tree: " + result);
   }
}
```

Otuput:-

```
PS C:\Users\Ajeet\Desktop\java> java MaximumWidthOfBinaryTree
Enter the value for the root node:
1
```

```
Enter the value for the left child of 1 (Enter -1 if no left child):
2
Enter the value for the right child of 1 (Enter -1 if no right child):
3
Enter the value for the left child of 2 (Enter -1 if no left child):
-1
Enter the value for the right child of 2 (Enter -1 if no right child):
-1
Enter the value for the left child of 3 (Enter -1 if no left child):
4
Enter the value for the right child of 3 (Enter -1 if no right child):
5
Enter the value for the left child of 4 (Enter -1 if no left child):
-1
Enter the value for the right child of 4 (Enter -1 if no right child):
-1
Enter the value for the left child of 5 (Enter -1 if no left child):
-1
Enter the value for the right child of 5 (Enter -1 if no right child):
-1
Maximum width of the binary tree: 2
```

2570. Merge Two 2D Arrays by Summing Values

```
import java.util.Scanner;

public class MergeTwo2DArrays {
  public static int[][] mergeArrays(int[][] arr1, int[][] arr2) {
    int rows = Math.max(arr1.length, arr2.length);
    int cols = Math.max(arr1[0].length, arr2[0].length);
```

```
  int[][] result = new int[rows][cols];

  for (int i = 0; i < rows; i++) {
    for (int j = 0; j < cols; j++) {
      int value1 = (i < arr1.length && j < arr1[i].length) ? arr1[i][j] : 0;
      int value2 = (i < arr2.length && j < arr2[i].length) ? arr2[i][j] : 0;
      result[i][j] = value1 + value2;
    }
  }

  return result;
}

public static void printArray(int[][] arr) {
  for (int[] row : arr) {
    for (int value : row) {
      System.out.print(value + " ");
    }
    System.out.println();
  }
}

public static void main(String[] args) {
  Scanner scanner = new Scanner(System.in);

  // Input for the first array
  System.out.println("Enter the dimensions of the first array (rows and columns):");
  int rows1 = scanner.nextInt();
  int cols1 = scanner.nextInt();

  System.out.println("Enter the elements of the first array:");
```

```
    int[][] arr1 = new int[rows1][cols1];
    for (int i = 0; i < rows1; i++) {
      for (int j = 0; j < cols1; j++) {
        arr1[i][j] = scanner.nextInt();
      }
    }

    // Input for the second array
    System.out.println("Enter the dimensions of the second array (rows and columns):");
    int rows2 = scanner.nextInt();
    int cols2 = scanner.nextInt();

    System.out.println("Enter the elements of the second array:");
    int[][] arr2 = new int[rows2][cols2];
    for (int i = 0; i < rows2; i++) {
      for (int j = 0; j < cols2; j++) {
        arr2[i][j] = scanner.nextInt();
      }
    }

    // Merge arrays and print the result
    int[][] result = mergeArrays(arr1, arr2);
    System.out.println("Merged Array:");
    printArray(result);
  }
}
```

Output

```
PS C:\Users\Ajeet\Desktop\java> java MergeTwo2DArrays
Enter the dimensions of the first array (rows and columns):
3 3
```

Enter the elements of the first array:

1 2 3

4 5 6

7 8 9

Enter the dimensions of the second array (rows and columns):

3 3

Enter the elements of the second array:

1 2 3

4 5 6

7 8 9

Merged Array:

2 4 6

8 10 12

14 16 18

67. Add Binary

```
import java.util.Scanner;


public class AddBinary {


  public static String addBinary(String a, String b) {
    StringBuilder result = new StringBuilder();


    int carry = 0;
    int i = a.length() - 1;
    int j = b.length() - 1;


    while (i >= 0 || j >= 0 || carry > 0) {
      int digitA = (i >= 0) ? Character.getNumericValue(a.charAt(i--)) : 0;
      int digitB = (j >= 0) ? Character.getNumericValue(b.charAt(j--)) : 0;
```

```
    int sum = digitA + digitB + carry;
    result.insert(0, sum % 2);
    carry = sum / 2;
   }

   return result.toString();
  }

  public static void main(String[] args) {
   Scanner scanner = new Scanner(System.in);

   System.out.println("Enter the first binary number:");
   String binary1 = scanner.nextLine();

   System.out.println("Enter the second binary number:");
   String binary2 = scanner.nextLine();

   // Add binary numbers and print the result
   String sum = addBinary(binary1, binary2);
   System.out.println("Sum of the binary numbers: " + sum);
  }
}
```

Output

```
PS C:\Users\Ajeet\Desktop\java> javac AddBinary.java
PS C:\Users\Ajeet\Desktop\java> java AddBinary
Enter the first binary number:
1101
Enter the second binary number:
1010
Sum of the binary numbers: 10111
```

202. Happy Number

```
import java.util.HashSet;
import java.util.Scanner;
import java.util.Set;

public class HappyNumber {

  public static boolean isHappy(int n) {
    Set<Integer> seen = new HashSet<>();

    while (n != 1 && !seen.contains(n)) {
      seen.add(n);
      n = getNext(n);
    }

    return n == 1;
  }

  private static int getNext(int n) {
    int sum = 0;
    while (n > 0) {
      int digit = n % 10;
      sum += digit * digit;
      n /= 10;
    }
    return sum;
  }

  public static void main(String[] args) {
    Scanner scanner = new Scanner(System.in);
```

```
    System.out.println("Enter a number to check if it is a happy number:");
    int number = scanner.nextInt();

    // Check if the number is a happy number and print the result
    boolean result = isHappy(number);
    if (result) {
      System.out.println(number + " is a happy number.");
    } else {
      System.out.println(number + " is not a happy number.");
    }
  }
}
```

Output

```
PS C:\Users\Ajeet\Desktop\java> javac HappyNumber.java
PS C:\Users\Ajeet\Desktop\java> java HappyNumber
Enter a number to check if it is a happy number:
19
19 is a happy number.
```

412. Fizz Buzz

```
import java.util.Scanner;

public class FizzBuzz {

  public static void fizzBuzz(int n) {
    for (int i = 1; i <= n; i++) {
      if (i % 3 == 0 && i % 5 == 0) {
        System.out.println("FizzBuzz");
      } else if (i % 3 == 0) {
        System.out.println("Fizz");
```

```
    } else if (i % 5 == 0) {
      System.out.println("Buzz");
    } else {
      System.out.println(i);
    }
   }
  }

  public static void main(String[] args) {
   Scanner scanner = new Scanner(System.in);

   System.out.println("Enter the value of n:");
   int n = scanner.nextInt();

   // Output Fizz Buzz sequence
   fizzBuzz(n);
  }
}
```

Output

```
PS C:\Users\Ajeet\Desktop\java> java FizzBuzz
Enter the value of n:
15
1
2
Fizz
4
Buzz
Fizz
7
8
```

Fizz

Buzz

11

Fizz

13

14

FizzBuzz

1780. Check if Number is a Sum of Powers of Three

```
import java.util.Scanner;

public class SumOfPowersOfThree {

 public static boolean checkPowersOfThree(int n) {
  while (n > 0) {
   if (n % 3 == 2) {
    return false;
   }
   n /= 3;
  }
  return true;
 }

 public static void main(String[] args) {
  Scanner scanner = new Scanner(System.in);

  System.out.println("Enter a number to check if it is a sum of powers of three:");
  int number = scanner.nextInt();

  // Check if the number can be represented as the sum of powers of three
```

```
    boolean result = checkPowersOfThree(number);

    // Print the result
    System.out.println("Is the number a sum of powers of three? " + result);
  }
}
```

Output

```
PS C:\Users\Ajeet\Desktop\java> javac SumOfPowersOfThree.java
PS C:\Users\Ajeet\Desktop\java> java SumOfPowersOfThree
Enter a number to check if it is a sum of powers of three:
21
Is the number a sum of powers of three? False
```

273. Integer to English Words

```
import java.util.Scanner;

public class IntegerToEnglishWords {

    private static final String[] LESS_THAN_20 = {
            "", "One", "Two", "Three", "Four", "Five", "Six", "Seven", "Eight", "Nine", "Ten",
            "Eleven", "Twelve", "Thirteen", "Fourteen", "Fifteen", "Sixteen", "Seventeen", "Eighteen",
"Nineteen"
    };

    private static final String[] TENS = {
            "", "", "Twenty", "Thirty", "Forty", "Fifty", "Sixty", "Seventy", "Eighty", "Ninety"
    };

    private static final String[] THOUSANDS = {"", "Thousand", "Million", "Billion"};

    public static String numberToWords(int num) {
```

```
    if (num == 0) {
        return "Zero";
    }

    int i = 0;
    String words = "";

    while (num > 0) {
        if (num % 1000 != 0) {
            words = helper(num % 1000) + THOUSANDS[i] + " " + words;
        }
        num /= 1000;
        i++;
    }

    return words.trim();
}

private static String helper(int num) {
    if (num == 0) {
        return "";
    } else if (num < 20) {
        return LESS_THAN_20[num] + " ";
    } else if (num < 100) {
        return TENS[num / 10] + " " + helper(num % 10);
    } else {
        return LESS_THAN_20[num / 100] + " Hundred " + helper(num % 100);
    }
}

public static void main(String[] args) {
```

```
        Scanner scanner = new Scanner(System.in);

        System.out.println("Enter a non-negative integer:");
        int num = scanner.nextInt();

        // Convert integer to English words and print the result
        String result = numberToWords(num);
        System.out.println("English representation: " + result);
    }
}
```

Output

```
PS C:\Users\Ajeet\Desktop\java> javac IntegerToEnglishWords.java
PS C:\Users\Ajeet\Desktop\java> java IntegerToEnglishWords
Enter a non-negative integer:
1234567
English representation: One Million Two Hundred Thirty Four Thousand Five Hundred Sixty Seven
```

160. Intersection of Two Linked Lists

```
import java.util.Scanner;

class ListNode {
  int val;
  ListNode next;

  public ListNode(int val) {
    this.val = val;
    this.next = null;
  }
}

public class IntersectionOfTwoLinkedLists {
```

```
public static ListNode getIntersectionNode(ListNode headA, ListNode headB) {
  if (headA == null || headB == null) {
    return null;
  }

  ListNode a = headA;
  ListNode b = headB;

  while (a != b) {
    a = (a == null) ? headB : a.next;
    b = (b == null) ? headA : b.next;
  }

  return a;
}

public static ListNode createLinkedList(int[] values) {
  ListNode dummy = new ListNode(-1);
  ListNode current = dummy;

  for (int value : values) {
    current.next = new ListNode(value);
    current = current.next;
  }

  return dummy.next;
}

public static void printLinkedList(ListNode head) {
  ListNode current = head;
```

```
    while (current != null) {
      System.out.print(current.val + " ");
      current = current.next;
    }
    System.out.println();
  }

  public static void main(String[] args) {
    Scanner scanner = new Scanner(System.in);

    System.out.println("Enter values for the first linked list (space-separated):");
    int[] valuesA = parseInput(scanner.nextLine());

    System.out.println("Enter values for the second linked list (space-separated):");
    int[] valuesB = parseInput(scanner.nextLine());

    System.out.println("Enter the index (0-based) where the two linked lists intersect (enter -1 if no
intersection):");
    int intersectionIndex = scanner.nextInt();

    // Create linked lists
    ListNode headA = createLinkedList(valuesA);
    ListNode headB = createLinkedList(valuesB);

    // Intersect the linked lists at the specified index
    if (intersectionIndex >= 0) {
      ListNode intersectionNode = getNodeAtIndex(headA, intersectionIndex);
      getLastNode(headB).next = intersectionNode;
    }

    // Print linked lists
```

```
    System.out.println("Linked List A:");
    printLinkedList(headA);

    System.out.println("Linked List B:");
    printLinkedList(headB);

    // Find and print the intersection node
    ListNode intersectionNode = getIntersectionNode(headA, headB);
    System.out
        .println("Intersection Node Value: " + ((intersectionNode != null) ? intersectionNode.val : "No intersection"));
  }

  private static int[] parseInput(String input) {
    String[] tokens = input.split("\\s+");
    int[] values = new int[tokens.length];
    for (int i = 0; i < tokens.length; i++) {
      values[i] = Integer.parseInt(tokens[i]);
    }
    return values;
  }

  private static ListNode getNodeAtIndex(ListNode head, int index) {
    ListNode current = head;
    for (int i = 0; i < index; i++) {
      current = current.next;
    }
    return current;
  }

  private static ListNode getLastNode(ListNode head) {
```

```
    ListNode current = head;
    while (current.next != null) {
      current = current.next;
    }
    return current;
  }
}
```

Output:-

```
PS C:\Users\Ajeet\Desktop\java> javac IntersectionOfTwoLinkedLists.java
PS C:\Users\Ajeet\Desktop\java> java IntersectionOfTwoLinkedLists
Enter values for the first linked list (space-separated):
1 2 3 4 5
Enter values for the second linked list (space-separated):
6 7 8
Enter the index (0-based) where the two linked lists intersect (enter -1 if no intersection):
2
Linked List A:
1 2 3 4 5
Linked List B:
6 7 8 3 4 5
Intersection Node Value: 3
```

172. Factorial Trailing Zeroes

```
import java.util.Scanner;

public class FactorialTrailingZeroes {

  public static int trailingZeroes(int n) {
```

```
    int count = 0;

    // Keep dividing by 5 and accumulating the count
    while (n >= 5) {
      n /= 5;
      count += n;
    }

    return count;
  }

  public static void main(String[] args) {
    Scanner scanner = new Scanner(System.in);

    System.out.println("Enter a non-negative integer:");
    int number = scanner.nextInt();

    // Calculate trailing zeroes and print the result
    int result = trailingZeroes(number);
    System.out.println("Number of trailing zeroes in the factorial: " + result);
  }
}
```

output

```
PS C:\Users\Ajeet\Desktop\java> javac FactorialTrailingZeroes.java
PS C:\Users\Ajeet\Desktop\java> java FactorialTrailingZeroes
Enter a non-negative integer:
25
Number of trailing zeroes in the factorial: 6
```