Leet Code Problem (03-10-2024)

Q1.

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


class TreeNode {
    int val;
    TreeNode left;
    TreeNode right;
    TreeNode() {}
    TreeNode(int val) { this.val = val; }
    TreeNode(int val, TreeNode left, TreeNode right) {
        this.val = val;
        this.left = left;
        this.right = right;
    }
}


public class Solution {
    class NodePosition {
        TreeNode node;
        int position;


        NodePosition(TreeNode node, int position) {
            this.node = node;
            this.position = position;
        }
    }
```

```
public int widthOfBinaryTree(TreeNode root) {
    if (root == null) return 0;

    int maxWidth = 0;
    Queue<NodePosition> queue = new LinkedList<>();
    queue.offer(new NodePosition(root, 0));

    while (!queue.isEmpty()) {
        int size = queue.size();
        int minPos = queue.peek().position;
        int first = 0, last = 0;

        for (int i = 0; i < size; i++) {
            NodePosition current = queue.poll();
            int currPos = current.position - minPos;

            if (i == 0) first = currPos;
            if (i == size - 1) last = currPos;

            if (current.node.left != null) {
                queue.offer(new NodePosition(current.node.left, 2 * currPos));
            }
            if (current.node.right != null) {
                queue.offer(new NodePosition(current.node.right, 2 * currPos + 1));
            }
        }

        maxWidth = Math.max(maxWidth, last - first + 1);
    }
```

```
        return maxWidth;
    }
}
```

Q2.

```
import java.util.ArrayList;
import java.util.List;

public class Solution {
    public int[][] mergeArrays(int[][] nums1, int[][] nums2) {
        List<int[]> resultList = new ArrayList<>();
        int i = 0, j = 0;

        while (i < nums1.length && j < nums2.length) {
            int id1 = nums1[i][0];
            int id2 = nums2[j][0];

            if (id1 == id2) {
                resultList.add(new int[] {id1, nums1[i][1] + nums2[j][1]});
                i++;
                j++;
            } else if (id1 < id2) {
                resultList.add(new int[] {id1, nums1[i][1]});
                i++;
            } else {
                resultList.add(new int[] {id2, nums2[j][1]});
                j++;
            }
```

```
        }

        while (i < nums1.length) {
            resultList.add(new int[] {nums1[i][0], nums1[i][1]});
            i++;
        }

        while (j < nums2.length) {
            resultList.add(new int[] {nums2[j][0], nums2[j][1]});
            j++;
        }

        int[][] resultArray = new int[resultList.size()][2];
        for (int k = 0; k < resultList.size(); k++) {
            resultArray[k] = resultList.get(k);
        }

        return resultArray;
    }

    public static void main(String[] args) {
        Solution solution = new Solution();

    }
}
```

Q3.

```
public class Solution {
    public String addBinary(String a, String b) {
```

```
    StringBuilder result = new StringBuilder();
    int i = a.length() - 1;
    int j = b.length() - 1;
    int carry = 0;

    while (i >= 0 || j >= 0 || carry > 0) {
        int sum = carry;

        if (i >= 0) {
            sum += a.charAt(i) - '0';
            i--;
        }

        if (j >= 0) {
            sum += b.charAt(j) - '0';
            j--;
        }

        result.append(sum % 2);
        carry = sum / 2;
    }

    return result.reverse().toString();
}

public static void main(String[] args) {
    Solution solution = new Solution();
```

```
    }
}
```

Q4.

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

public class Solution {
    public boolean isHappy(int n) {
        Set<Integer> seenNumbers = new HashSet<>();

        while (n != 1 && !seenNumbers.contains(n)) {
            seenNumbers.add(n);
            n = getSumOfSquares(n);
        }

        return n == 1;
    }

    private int getSumOfSquares(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) {
        Solution solution = new Solution();

         }
}
```

Q5.

```
import java.util.ArrayList;
import java.util.List;

public class Solution {
    public List<String> fizzBuzz(int n) {
        List<String> result = new ArrayList<>();

        for (int i = 1; i <= n; i++) {
            if (i % 3 == 0 && i % 5 == 0) {
                result.add("FizzBuzz");
            } else if (i % 3 == 0) {
                result.add("Fizz");
            } else if (i % 5 == 0) {
                result.add("Buzz");
            } else {
                result.add(Integer.toString(i));
            }
        }

        return result;
    }
```

```
    public static void main(String[] args) {
        Solution solution = new Solution();
    }
}
```

Q6.

```
public class Solution {
    public boolean checkPowersOfThree(int n) {
        while (n > 0) {
            if (n % 3 == 2) {

                return false;
            }
            n /= 3;
        }
        return true;
    }

    public static void main(String[] args) {
        Solution solution = new Solution();

    }
}
```