PALINDROME LINKED LIST

CODE

```
import java.util.*;
class ListNode {
    int val;
    ListNode next;
    ListNode(int x) { val = x; }
}
public class PalindromeLL {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.print("Enter the number of nodes in the list: ");
        int n = scanner.nextInt();
        if (n == 0) {
            System.out.println("The list is empty, hence it is considered a palindrome.");
            return;
        }
        System.out.println("Enter the elements of the list: ");
        ListNode head = new ListNode(scanner.nextInt());
        ListNode current = head;
        for (int i = 1; i < n; i++) {
            int val = scanner.nextInt();
            current.next = new ListNode(val);
            current = current.next;
        }
        if (isPalindrome(head)) {
```

```
            System.out.println("The linked list is a palindrome.");
        } else {
            System.out.println("The linked list is not a palindrome.");
        }
    }
    public static boolean isPalindrome(ListNode head) {
        Stack<Integer> stack = new Stack<>();
        ListNode current = head;
        while (current != null) {
            stack.push(current.val);
            current = current.next;
        }
        current = head;
        while (current != null) {
            if (current.val != stack.pop()) {
                return false;
            }
            current = current.next;
        }
        return true;
    }
}
```

OUTPUT

G:\java programs\hw>java PalindromeLL

Enter the number of nodes in the list: 5

Enter the elements of the list:

6 8 4 8 7

The linked list is not a palindrome.

TIME COMPLEXITY:

O(n)

BALANCED TREE CHECK

CODE

```
import java.util.*;
class TreeNode {
    int val;
    TreeNode left;
    TreeNode right;
    TreeNode(int x) {
        val = x;
        left = null;
        right = null;
    }
}
public class BalancedBinaryTree {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.println("Enter the root value (or -1 to indicate no node): ");
        int rootVal = scanner.nextInt();
        TreeNode root = buildTree(scanner, rootVal);
        boolean balanced = isBalanced(root);
        System.out.println("Is the tree balanced? " + balanced);
    }
    public static boolean isBalanced(TreeNode root) {
        return checkHeight(root) != -1;
    }
    private static int checkHeight(TreeNode root) {
```

```
        if (root == null) {
            return 0;
        }
        int leftHeight = checkHeight(root.left);
        if (leftHeight == -1) return -1;
        int rightHeight = checkHeight(root.right);
        if (rightHeight == -1) return -1;
        if (Math.abs(leftHeight - rightHeight) > 1) {
            return -1;
        }
        return Math.max(leftHeight, rightHeight) + 1;
    }
    private static TreeNode buildTree(Scanner scanner, int val) {
        if (val == -1) {
            return null;
        }
        TreeNode node = new TreeNode(val);
        System.out.println("Enter left child of " + val + " (or -1 if no child): ");
        int leftVal = scanner.nextInt();
        node.left = buildTree(scanner, leftVal);
        System.out.println("Enter right child of " + val + " (or -1 if no child): ");
        int rightVal = scanner.nextInt();
        node.right = buildTree(scanner, rightVal);
        return node;
    }
}
```

OUTPUT

G:\java programs\hw>java BalancedBinaryTree

Enter the root value (or -1 to indicate no node): 7

Enter left child of 7 (or -1 if no child): 3

Enter left child of 3 (or -1 if no child): 4

Enter left child of 4 (or -1 if no child): -1

Enter right child of 4 (or -1 if no child): -1

Enter right child of 3 (or -1 if no child): 8

Enter left child of 8 (or -1 if no child): -1

Enter right child of 8 (or -1 if no child): -1

Enter right child of 7 (or -1 if no child): 9

Enter left child of 9 (or -1 if no child): 8

Enter left child of 8 (or -1 if no child): 7

Enter left child of 7 (or -1 if no child): -1

Enter right child of 7 (or -1 if no child): -1

Enter right child of 8 (or -1 if no child): 5

Enter left child of 5 (or -1 if no child): -1

Enter right child of 5 (or -1 if no child): -1

Enter right child of 9 (or -1 if no child): 8

Enter left child of 8 (or -1 if no child): -1

Enter right child of 8 (or -1 if no child): -1

Is the tree balanced? true

TIME COMPLEXITY

O(n)

1-0KNAPSACK PROBLEM

CODE

```
import java.util.*;
public class Knapsack {
    public static int knapsack(int capacity, int[] val, int[] wt) {
        int n = val.length;
        int[][] dp = new int[n + 1][capacity + 1];
        for (int i = 1; i <= n; i++) {
            for (int w = 0; w <= capacity; w++) {
                if (wt[i - 1] <= w) {
                    dp[i][w] = Math.max(dp[i - 1][w], dp[i - 1][w - wt[i - 1]] + val[i - 1]);
                } else {
                    dp[i][w] = dp[i - 1][w];
                }
            }
        }
        return dp[n][capacity];
    }
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.print("Enter the capacity of the knapsack: ");
        int capacity = scanner.nextInt();
        System.out.print("Enter the number of items: ");
        int n = scanner.nextInt();
        int[] val = new int[n];
```

```
        int[] wt = new int[n];
        System.out.println("Enter the values of the items: ");
        for (int i = 0; i < n; i++) {
            val[i] = scanner.nextInt();
        }
        System.out.println("Enter the weights of the items: ");
        for (int i = 0; i < n; i++) {
            wt[i] = scanner.nextInt();
        }
        int result = knapsack(capacity, val, wt);
        System.out.println("The maximum value that can be obtained is: " + result);
        scanner.close();
    }
}
```

OUTPUT

```
G:\java programs\hw>java Knapsack
Enter the capacity of the knapsack: 5
Enter the number of items: 4
Enter the values of the items:
1 5 6 7
Enter the weights of the items:
8 4 2 6
The maximum value that can be obtained is: 6
```

TIME COMPLEXITY

O(n)

FLOOR IN SORTED ARRAY

CODE

```
import java.util.Scanner;
public class FloorOfK {
    public static int findFloor(int[] arr, int k) {
        int left = 0;
        int right = arr.length - 1;
        int result = -1;
        while (left <= right) {
            int mid = left + (right - left) / 2;
            if (arr[mid] <= k) {
                result = mid;
                left = mid + 1;
            } else {
                right = mid - 1;
            }
        }
        return result;
    }
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.print("Enter the size of the array: ");
        int n = scanner.nextInt();
        int[] arr = new int[n];
        System.out.println("Enter the elements of the array in sorted order: ");
```

```
        for (int i = 0; i < n; i++) {
            arr[i] = scanner.nextInt();
        }
        System.out.print("Enter the value of k: ");
        int k = scanner.nextInt();

        int index = findFloor(arr, k);
        System.out.println("The index of the floor of " + k + " is: " + index);
        scanner.close();
    }
}
```

OUTPUT

```
G:\java programs\hw>java FloorOfK
Enter the size of the array: 5
Enter the elements of the array in sorted order:
3 6 4 2 7
Enter the value of k: 4
The index of the floor of 4 is: 3
```

TIME COMPLEXITY

O(log n)

CHECK EQUAL ARRAY

CODE

```
import java.util.*;
public class ArrayEquality {
    public static boolean areArraysEqual(int[] arr1, int[] arr2) {
        if (arr1.length != arr2.length) {
            return false;
        }
        HashMap<Integer, Integer> map1 = new HashMap<>();
        HashMap<Integer, Integer> map2 = new HashMap<>();
        for (int num : arr1) {
            map1.put(num, map1.getOrDefault(num, 0) + 1);
        }
        for (int num : arr2) {
            map2.put(num, map2.getOrDefault(num, 0) + 1);
        }
        return map1.equals(map2);
    }
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.print("Enter the size of the arrays: ");
        int n = scanner.nextInt();
        int[] arr1 = new int[n];
        int[] arr2 = new int[n];
        System.out.println("Enter the elements of the first array: ");
```

```
        for (int i = 0; i < n; i++) {
            arr1[i] = scanner.nextInt();
        }
        System.out.println("Enter the elements of the second array: ");
        for (int i = 0; i < n; i++) {
            arr2[i] = scanner.nextInt();
        }
        boolean result = areArraysEqual(arr1, arr2);
        System.out.println("Are the arrays equal? " + result);
        scanner.close();
    }
}
```

OUTPUT

```
G:\java programs\hw>java ArrayEquality
Enter the size of the arrays: 5
Enter the elements of the first array:
1 5 9 0 3
Enter the elements of the second array:
 3 5 0 1 9
Are the arrays equal? true
```

TIME COMPLEXITY

O(n)

TRIPLET SUM IN ARRAY

CODE

```
import java.util.*;
public class TripletSum {
    public static boolean hasTripletSum(int[] arr, int x) {
        Arrays.sort(arr);
        int n = arr.length;
        for (int i = 0; i < n - 2; i++) {
            int left = i + 1;
            int right = n - 1;
            while (left < right) {
                int sum = arr[i] + arr[left] + arr[right];
                if (sum == x) {
                    return true;
                } else if (sum < x) {
                    left++;
                } else {
                    right--;
                }
            }
        }
        return false;
    }
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
```

```
        System.out.print("Enter the size of the array: ");
        int n = scanner.nextInt();
        int[] arr = new int[n];
        System.out.println("Enter the elements of the array: ");
        for (int i = 0; i < n; i++) {
            arr[i] = scanner.nextInt();
        }
        System.out.print("Enter the target sum (x): ");
        int x = scanner.nextInt();
        boolean result = hasTripletSum(arr, x);
        System.out.println("Is there a triplet with sum " + x + "? " + (result ? 1 : 0));
        scanner.close();
    }
}
```

OUTPUT

```
G:\java programs\hw>java TripletSum
Enter the size of the array: 6
Enter the elements of the array:
40 10 20 3 6 7
Enter the target sum (x): 24
Is there a triplet with sum 24? 0
```

TIME COMPLEXITY

$O(n^2)$