11-11-2024

CODING PRACTICE PROBLEMS

1. **0-1 knapsack problem**

```
import java.util.*;
import java.util.Scanner;

class Problem1 {

    public static int knapsack(int capacity, int[] val, int[] wt, int n) {
        int[][] dp = new int[n + 1][capacity + 1];

        for (int i = 0; i <= n; i++) {
            for (int w = 0; w <= capacity; w++) {
                if (i == 0 || w == 0) {
                    dp[i][w] = 0;
                } else if (wt[i - 1] <= w) {
                    dp[i][w] = Math.max(val[i - 1] + dp[i - 1][w - wt[i - 1]], dp[i - 1][w]);
                } else {
                    dp[i][w] = dp[i - 1][w];
                }
            }
        }

        return dp[n][capacity];
    }

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

        System.out.print("Enter the capacity of the knapsack: ");
        int capacity = sc.nextInt();
```

```
        System.out.print("Enter the number of items: ");
        int n = sc.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] = sc.nextInt();
        }

        System.out.println("Enter the weights of the items: ");
        for (int i = 0; i < n; i++) {
            wt[i] = sc.nextInt();
        }

        int result = knapsack(capacity, val, wt, n);
        System.out.println("Maximum value in knapsack = " + result);

        sc.close();
    }
}
```

```
C:\Users\shali\OneDrive\Desktop\9-11-2024-java\day-1>java Problem1
Enter the capacity of the knapsack: 4
Enter the number of items: 3
Enter the values of the items:
1 2 3
Enter the weights of the items:
4 5 1
Maximum value in knapsack = 3
```

2. Floor in sorted array

```
import java.util.*;
import java.util.Scanner;
```

```
class Problem2 {

    public static int findFloor(int[] arr, int k) {
        int low = 0, high = arr.length - 1;
        int result = -1;

        while (low <= high) {
            int mid = low + (high - low) / 2;

            if (arr[mid] == k) {
                return mid;
            } else if (arr[mid] < k) {
                result = mid;
                low = mid + 1;
            } else {
                high = mid - 1;
            }
        }

        return result;
    }

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

        System.out.print("Enter the size of the array: ");
        int n = sc.nextInt();

        int[] arr = new int[n];
        System.out.println("Enter the elements of the sorted array: ");
        for (int i = 0; i < n; i++) {
            arr[i] = sc.nextInt();
```

```
        }

        System.out.print("Enter the value of k: ");
        int k = sc.nextInt();

        int index = findFloor(arr, k);
        if (index == -1) {
            System.out.println("No element less than or equal to " + k + " found.");
        } else {
            System.out.println("The index of the largest element less than or equal to " + k + " is: " +
index);
        }

        sc.close();
    }
}
```

```
C:\Users\shali\OneDrive\Desktop\9-11-2024-java\day-1>javac problem2.java

C:\Users\shali\OneDrive\Desktop\9-11-2024-java\day-1>java Problem2
Enter the size of the array: 7
Enter the elements of the sorted array:
1 2 8 10 11 12 19
Enter the value of k: 5
The index of the largest element less than or equal to 5 is: 1
```

3. Check equal arrays

```
import java.util.*;
import java.util.HashMap;
import java.util.Scanner;

class Problem3 {

    public static boolean areArraysEqual(int[] arr1, int[] arr2) {
        if (arr1.length != arr2.length) {
            return false;
        }
```

```
    HashMap<Integer, Integer> map = new HashMap<>();

    for (int num : arr1) {
        map.put(num, map.getOrDefault(num, 0) + 1);
    }

    for (int num : arr2) {
        if (!map.containsKey(num)) {
            return false;
        }
        int count = map.get(num);
        if (count == 1) {
            map.remove(num);
        } else {
            map.put(num, count - 1);
        }
    }

    return map.isEmpty();
}

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

    System.out.print("Enter the size of the first array: ");
    int n1 = sc.nextInt();

    int[] arr1 = new int[n1];
    System.out.println("Enter the elements of the first array: ");
    for (int i = 0; i < n1; i++) {
        arr1[i] = sc.nextInt();
```

```
        }

        System.out.print("Enter the size of the second array: ");
        int n2 = sc.nextInt();

        int[] arr2 = new int[n2];
        System.out.println("Enter the elements of the second array: ");
        for (int i = 0; i < n2; i++) {
            arr2[i] = sc.nextInt();
        }

        boolean result = areArraysEqual(arr1, arr2);
        if (result) {
            System.out.println("The arrays are equal.");
        } else {
            System.out.println("The arrays are not equal.");
        }

        sc.close();
    }
}
```

```
C:\Users\shali\OneDrive\Desktop\9-11-2024-java\day-1>javac problem3.java

C:\Users\shali\OneDrive\Desktop\9-11-2024-java\day-1>java Problem3
Enter the size of the first array: 5
Enter the elements of the first array:
1 2 5 4 0
Enter the size of the second array: 5
Enter the elements of the second array:
2 4 5 0 1
The arrays are equal.
```

4. Palindrome linked list

```
import java.util.*;
import java.util.Scanner;

class Problem4 {
```

```
static class ListNode {
    int val;
    ListNode next;
    ListNode(int x) {
        val = x;
        next = null;
    }
}

public static boolean isPalindrome(ListNode head) {
    if (head == null || head.next == null) {
        return true;
    }
    ListNode slow = head;
    ListNode fast = head;

    while (fast != null && fast.next != null) {
        slow = slow.next;
        fast = fast.next.next;
    }
    ListNode prev = null;
    ListNode curr = slow;
    while (curr != null) {
        ListNode next = curr.next;
        curr.next = prev;
        prev = curr;
        curr = next;
    }
    ListNode firstHalf = head;
    ListNode secondHalf = prev;
    while (secondHalf != null) {
```

```
        if (firstHalf.val != secondHalf.val) {
            return false;
        }
        firstHalf = firstHalf.next;
        secondHalf = secondHalf.next;
    }

    return true;
}

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

    System.out.print("Enter the number of nodes: ");
    int n = sc.nextInt();

    System.out.println("Enter the values of the nodes: ");
    ListNode head = new ListNode(sc.nextInt());
    ListNode current = head;
    for (int i = 1; i < n; i++) {
        int value = sc.nextInt();
        current.next = new ListNode(value);
        current = current.next;
    }

    boolean result = isPalindrome(head);
    if (result) {
        System.out.println("The linked list is a palindrome.");
    } else {
        System.out.println("The linked list is not a palindrome.");
    }
```

```
        sc.close();
    }
}
```

```
C:\Users\shali\OneDrive\Desktop\9-11-2024-java\day-1>javac problem4.java

C:\Users\shali\OneDrive\Desktop\9-11-2024-java\day-1>java Problem4
Enter the number of nodes: 6
Enter the values of the nodes:
1 2 1 1 2 1
The linked list is a palindrome.

C:\Users\shali\OneDrive\Desktop\9-11-2024-java\day-1>java Problem4
Enter the number of nodes: 4
Enter the values of the nodes:
1 2 3 4
The linked list is not a palindrome.
```

5. Balanced tree check

```
Import java.util.*;
import java.util.Scanner;

class Problem5 {

    static class TreeNode {
        int val;
        TreeNode left;
        TreeNode right;
        TreeNode(int x) {
            val = x;
            left = null;
            right = null;
        }
    }

    public static boolean isBalanced(TreeNode root) {
        return height(root) != -1;
    }
```

```
private static int height(TreeNode node) {
    if (node == null) {
        return 0;
    }

    int leftHeight = height(node.left);
    int rightHeight = height(node.right);

    // If left or right subtree is unbalanced, return -1
    if (leftHeight == -1 || rightHeight == -1) {
        return -1;
    }

    // If the current node is unbalanced, return -1
    if (Math.abs(leftHeight - rightHeight) > 1) {
        return -1;
    }

    // Return the height of the current node
    return Math.max(leftHeight, rightHeight) + 1;
}

public static TreeNode buildTree(Scanner sc) {
    System.out.print("Enter the value of the node (-1 for no node): ");
    int val = sc.nextInt();
    if (val == -1) {
        return null;
    }

    TreeNode node = new TreeNode(val);
    System.out.println("Enter left child of " + val + ":");
    node.left = buildTree(sc);
```

```
        System.out.println("Enter right child of " + val + ":");
        node.right = buildTree(sc);

        return node;
    }

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

        System.out.println("Enter the binary tree structure:");
        TreeNode root = buildTree(sc);

        if (isBalanced(root)) {
            System.out.println("1");  // The tree is balanced
        } else {
            System.out.println("0");  // The tree is not balanced
        }

        sc.close();
    }
}
```

```
C:\Users\shali\OneDrive\Desktop\9-11-2024-java\day-1>javac problem5.java

C:\Users\shali\OneDrive\Desktop\9-11-2024-java\day-1>java Problem5
Enter the binary tree structure:
Enter the value of the node (-1 for no node): 1
Enter left child of 1:
Enter the value of the node (-1 for no node): 2
Enter left child of 2:
Enter the value of the node (-1 for no node): -1
Enter right child of 2:
Enter the value of the node (-1 for no node): 3
Enter left child of 3:
Enter the value of the node (-1 for no node): -1
Enter right child of 3:
Enter the value of the node (-1 for no node): -1
Enter right child of 1:
Enter the value of the node (-1 for no node): -1
0
```

6. Triplet sum in array

```
import java.util.*;
import java.util.Scanner;

class Problem6 {

    public static boolean findTriplet(int[] arr, int n, int x) {
        for (int i = 0; i < n - 2; i++) {
            for (int j = i + 1; j < n - 1; j++) {
                for (int k = j + 1; k < n; k++) {
                    if (arr[i] + arr[j] + arr[k] == x) {
                        return true;
                    }
                }
            }
        }
        return false;
    }

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

        System.out.print("Enter the size of the array: ");
        int n = sc.nextInt();

        int[] arr = new int[n];
        System.out.println("Enter the elements of the array: ");
        for (int i = 0; i < n; i++) {
            arr[i] = sc.nextInt();
        }

        System.out.print("Enter the value of x: ");
```

```
        int x = sc.nextInt();

        if (findTriplet(arr, n, x)) {
            System.out.println("1");
        } else {
            System.out.println("0");
        }

        sc.close();
    }
}
```

```
C:\Users\shali\OneDrive\Desktop\9-11-2024-java\day-1>javac problem6.java

C:\Users\shali\OneDrive\Desktop\9-11-2024-java\day-1>java Problem6
Enter the size of the array: 6
Enter the elements of the array:
1 2 4 3 6 7
Enter the value of x: 13
1
```