Problem Solving 11/11/2024

1. Knapsack
   Time Complexity: O(n * W)
   Space Complexity: O(n * W)

```java
package program11thnov;

import java.util.Scanner;

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

        System.out.print("Enter the number of items: ");
        int itemCount = scanner.nextInt();

        int[] values = new int[itemCount];
        int[] weights = new int[itemCount];

        System.out.println("Enter the values for each item:");
        for (int i = 0; i < itemCount; i++) {
            values[i] = scanner.nextInt();
        }

        System.out.println("Enter the weights for each item:");
        for (int i = 0; i < itemCount; i++) {
            weights[i] = scanner.nextInt();
        }

        System.out.print("Enter the maximum weight capacity of the knapsack: ");
        int capacity = scanner.nextInt();

        int maxProfit = knapSack(capacity, weights, values, itemCount);
        System.out.println(maxProfit);

        scanner.close();
    }
    static int knapSack(int capacity, int weights[], int values[], int itemCount) {
        int[][] dp = new int[itemCount + 1][capacity + 1];

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

        return dp[itemCount][capacity];
    }


}

```

```
<terminated> Knapsack [Java Application] C:\Program Files\Java\bin\javaw.exe  (11-Nov-2024, 2:45:50 pm – 2:47:26 pm) [pid: 21816]
Enter the number of items: 3
Enter the values for each item:
1 2 3
Enter the weights for each item:
4 5 1
Enter the maximum weight capacity of the knapsack: 4
3
```

2. Floor in Sorted array

Time Complexity: O(log n)

Space Complexity: O(1)

```java
package program11thnov;

import java.util.*;

public class FloorSortedArray {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.print("Enter the number of elements: ");
        int n = scanner.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] = scanner.nextInt();
        }
        System.out.print("Enter the target number: ");
        int target = scanner.nextInt();
        int index = floorSearch(arr, 0, n - 1, target);
        if (index == -1) {
            System.out.println("Floor of " + target + " doesn't exist in the array.");
        } else {
            System.out.println("Floor of " + target + " is " + arr[index]);
        }
        scanner.close();
    }
    static int floorSearch(int arr[], int low, int high, int target) {
        if (low > high) return -1;
        if (target >= arr[high]) return high;
        int mid = (low + high) / 2;
        if (arr[mid] == target) return mid;
        if (mid > 0 && arr[mid - 1] <= target && target < arr[mid]) {
            return mid - 1;
        }
        if (target < arr[mid]) return floorSearch(arr, low, mid - 1, target);
        return floorSearch(arr, mid + 1, high, target);
    }
}
```

```
Enter the number of elements: 7
Enter the elements of the sorted array:
1 2 8 10 10 12 19
Enter the target number: 5
Floor of 5 is 2
```

3. Check Equal Arrays

Time Complexity: O(n)

Space Complexity: O(n)

```java
package program11thnov;

import java.util.*;

public class CheckEqualArray {
    public static boolean areEqual(int[] arr1, int[] arr2) {
        int N = arr1.length;
        int M = arr2.length;
        if (N != M) return false;
        Map<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) || map.get(num) == 0) return false;
            map.put(num, map.get(num) - 1);
        }
        return true;
    }

    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.println("Enter the size of the first array:");
        int n1 = scanner.nextInt();
        int[] arr1 = new int[n1];
        System.out.println("Enter elements of the first array:");
        for (int i = 0; i < n1; i++) {
            arr1[i] = scanner.nextInt();
        }
        int n2=n1;
        int[] arr2 = new int[n2];
        System.out.println("Enter elements of the second array:");
        for (int i = 0; i < n2; i++) {
            arr2[i] = scanner.nextInt();
        }
        if (areEqual(arr1, arr2))
            System.out.println("Yes, the arrays are equal.");
        else
            System.out.println("No, the arrays are not equal.");

        scanner.close();
```

```
<terminated> CheckEqualArray [Java Application] C:\Program Files\Java\bin\j
Enter the size of the first array:
5
Enter elements of the first array:
1 2 5 4 0
Enter elements of the second array:
2 4 5 0 1
Yes, the arrays are equal.
```

4. Palindrome Linked List

   Time Complexity: O(n)

   Space Complexity: O(n)

```java
package program11thnov;
import java.util.*;
class ListNode {
    int val;
    ListNode next;
    ListNode(int val) {
        this.val = val;
    }
}
public class PalindromeLinkedList {
    public static boolean isPalindrome(ListNode head) {
        Stack<Integer> stack = new Stack<>();
        ListNode curr = head;
        while (curr != null) {
            stack.push(curr.val);
            curr = curr.next;
        }
        curr = head;
        while (curr != null && curr.val == stack.pop()) {
            curr = curr.next;
        }
        return curr == null;
    }
    public static ListNode createLinkedList(int[] values) {
        ListNode head = new ListNode(values[0]);
        ListNode curr = head;
        for (int i = 1; i < values.length; i++) {
            curr.next = new ListNode(values[i]);
            curr = curr.next;
        }
        return head;
    }
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.println("Enter the number of elements:");
        int n = scanner.nextInt();
        int[] values = new int[n];
        System.out.println("Enter the elements of the linked list:");
        for (int i = 0; i < n; i++) {
            values[i] = scanner.nextInt();
        }
        ListNode head = createLinkedList(values);
        if (isPalindrome(head)) {
            System.out.println("The linked list is a palindrome.");
        } else {
            System.out.println("The linked list is not a palindrome.");
        }
        scanner.close();
    }
}
```

```
<terminated> PalindromeLinkedList [Java Application] C:\Program Files\Java\b
Enter the number of elements:
5
Enter the elements of the linked list:
1 2 3 2 1
The linked list is a palindrome.
```

5. Balanced Tree Check

Time Complexity: O(n)

Space Complexity: O(h)

```java
package program11thnov;
import java.util.*;
class Node {
    int key;
    Node left;
    Node right;
    Node(int k) {
        key = k;
        left = right = null;
    }
}
public class BalancedTree {
    public static int isBalanced(Node root) {
        if (root == null)
            return 0;
        int lh = isBalanced(root.left);
        if (lh == -1)
            return -1;
        int rh = isBalanced(root.right);
        if (rh == -1)
            return -1;
        if (Math.abs(lh - rh) > 1)
            return -1;
        else
            return Math.max(lh, rh) + 1;
    }
```

```java
    public static Node buildTree(Scanner scanner) {
        System.out.println("Enter node value (-1 for no node):");
        int value = scanner.nextInt();
        if (value == -1) return null;
        Node root = new Node(value);
        System.out.println("Enter left child of " + value);
        root.left = buildTree(scanner);
        System.out.println("Enter right child of " + value);
        root.right = buildTree(scanner);
        return root;
    }

    public static void main(String args[]) {
        Scanner scanner = new Scanner(System.in);
        System.out.println("Build your binary tree:");
        Node root = buildTree(scanner);
        if (isBalanced(root) > 0)
            System.out.print("Balanced");
        else
            System.out.print("Not Balanced");

        scanner.close();
    }
}
```

```
<terminated> BalancedTree [Java Application] C:\Program Files\Ja
Build your binary tree:
Enter node value (-1 for no node):
5
Enter left child of 5
Enter node value (-1 for no node):
4
Enter left child of 4
Enter node value (-1 for no node):
3
Enter left child of 3
Enter node value (-1 for no node):
-1
Enter right child of 3
Enter node value (-1 for no node):
2
Enter left child of 2
Enter node value (-1 for no node):
1
Enter left child of 1
Enter node value (-1 for no node):
-1
Enter right child of 1
Enter node value (-1 for no node):
-1
Enter right child of 2
Enter node value (-1 for no node):
-1
Enter right child of 4
Enter node value (-1 for no node):
-1
Enter right child of 5
Enter node value (-1 for no node):
-1
Not Balanced
```

6. Triplet Sum in Array
   Time Complexity: O(n^2)
   Space Complexity: O(n)

```java
package program11thnov;
import java.util.*;
public class TripletSumArray {
    static boolean find3Numbers(int[] arr, int sum) {
        int n = arr.length;
        for (int i = 0; i < n - 2; i++) {
            Set<Integer> s = new HashSet<>();
            int currSum = sum - arr[i];
            for (int j = i + 1; j < n; j++) {
                int requiredValue = currSum - arr[j];
                if (s.contains(requiredValue)) {
                    System.out.println("Triplet is " + arr[i] + ", " + arr[j] + ", " + requiredValue);
                    return true;
                }
                s.add(arr[j]);
            }
        }
        System.out.println("No triplet found");
        return false;
    }

    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.print("Enter the number of elements in 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("Target Sum ");
        int sum = scanner.nextInt();
        find3Numbers(arr, sum);
        scanner.close();
    }
}
```

```
<terminated> TripletSumArray [Java Application] C:\Program Files\Java\bin\j
Enter the number of elements in the array: 8
Enter the elements of the array:
2 4 5 1 8 6 3 9
Target Sum 13
Triplet is 2, 6, 5
```