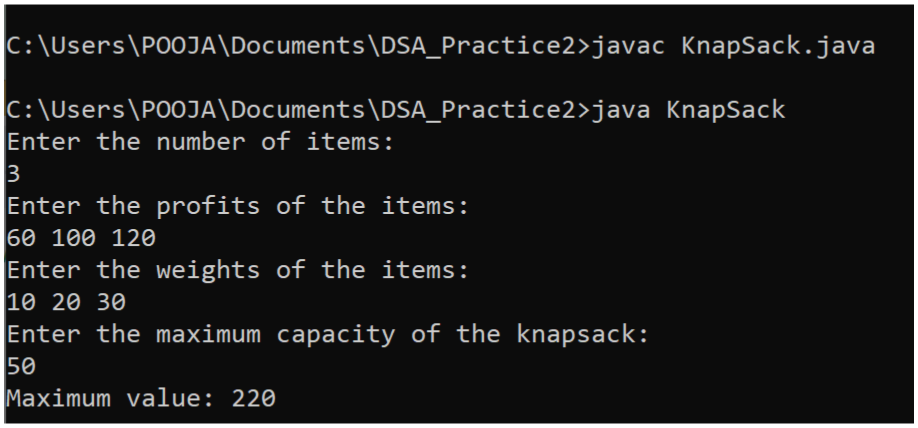
11/11/2024 DSA PRACTICE 2

1.0-1 knapsack problem

```
import java.util.Scanner;


class KnapSack {


    static int knapSack(int W, int wt[], int val[], int n) {

        if (n == 0 || W == 0)

            return 0;


        if (wt[n - 1] > W)

            return knapSack(W, wt, val, n - 1);


        return Math.max(knapSack(W, wt, val, n - 1), val[n - 1] + knapSack(W - wt[n - 1], wt, val, n -
1));

    }


    public static void main(String args[]) {

        Scanner sc = new Scanner(System.in);


        System.out.println("Enter the number of items:");

        int n = sc.nextInt();


        int[] profit = new int[n];

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

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

        System.out.println("Enter the maximum capacity of the knapsack:");
        int W = sc.nextInt();

        System.out.println("Maximum value: " + knapSack(W, weight, profit, n));
        sc.close();
    }
}
```

Time Complexity : O(2^n)

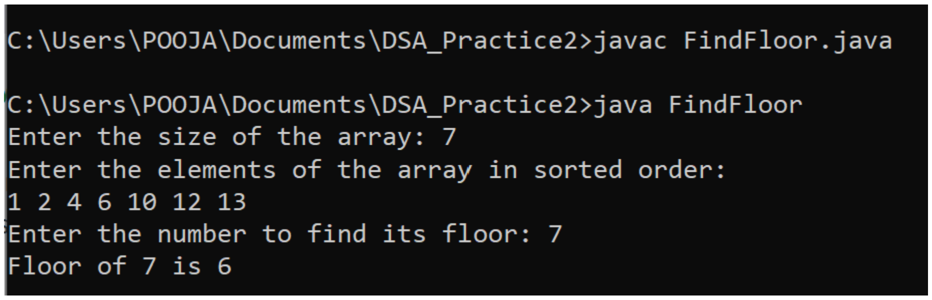


```
C:\Users\POOJA\Documents\DSA_Practice2>javac KnapSack.java

C:\Users\POOJA\Documents\DSA_Practice2>java KnapSack
Enter the number of items:
3
Enter the profits of the items:
60 100 120
Enter the weights of the items:
10 20 30
Enter the maximum capacity of the knapsack:
50
Maximum value: 220
```

2.Floor in sorted array

```
import java.util.Scanner;

class FindFloor{

    static int floorSearch(int arr[], int n, int x) {
        if (x >= arr[n - 1])
            return n - 1;
        if (x < arr[0])
            return -1;
        for (int i = 1; i < n; i++)
            if (arr[i] > x)
                return (i - 1);
        return -1;
    }

    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 in sorted order:");
        for (int i = 0; i < n; i++) {
            arr[i] = sc.nextInt();
        }
        System.out.print("Enter the number to find its floor: ");
        int x = sc.nextInt();

        int index = floorSearch(arr, n, x);
        if (index == -1)
            System.out.println("Floor of " + x + " doesn't exist in array.");
        else
            System.out.println("Floor of " + x + " is " + arr[index]);

        sc.close();
    }
}
```

Time Complexity : O(n)

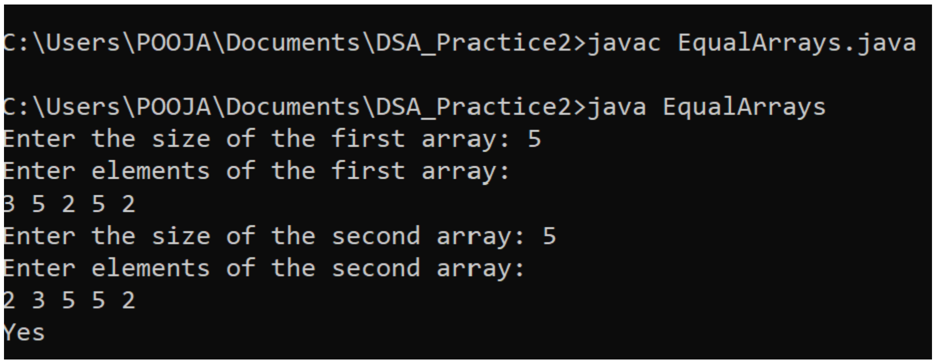


```
C:\Users\POOJA\Documents\DSA_Practice2>javac FindFloor.java

C:\Users\POOJA\Documents\DSA_Practice2>java FindFloor
Enter the size of the array: 7
Enter the elements of the array in sorted order:
1 2 4 6 10 12 13
Enter the number to find its floor: 7
Floor of 7 is 6
```

3.Check equal arrays

```
import java.util.Arrays;
import java.util.Scanner;

class EqualArrays{
    public static boolean areEqual(int arr1[], int arr2[]) {
        int N = arr1.length;
        int M = arr2.length;
        if (N != M)
            return false;

        Arrays.sort(arr1);
        Arrays.sort(arr2);

        for (int i = 0; i < N; i++)
            if (arr1[i] != arr2[i])
                return false;

        return true;
    }

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

        System.out.print("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();
        }

        System.out.print("Enter the size of the second array: ");
        int n2 = scanner.nextInt();
        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");
        else
            System.out.println("No");

        scanner.close();
    }
}
```

Time Complexity : O(NlogN+MlogM)



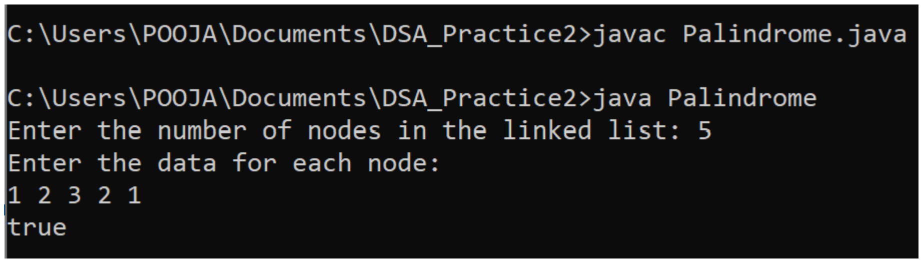
```
C:\Users\POOJA\Documents\DSA_Practice2>javac EqualArrays.java

C:\Users\POOJA\Documents\DSA_Practice2>java EqualArrays
Enter the size of the first array: 5
Enter elements of the first array:
3 5 2 5 2
Enter the size of the second array: 5
Enter elements of the second array:
2 3 5 5 2
Yes
```

4.Palindrome linked list

```
import java.util.Scanner;

class Node {
    int data;
    Node next;
    Node(int d) {
        data = d;
        next = null;
    }
}

class Palindrome{
    static Node reverseList(Node head) {
        Node prev = null;
        Node curr = head;
        Node next;
        while (curr != null) {
            next = curr.next;
            curr.next = prev;
            prev = curr;
            curr = next;
        }
        return prev;
    }

    static boolean isIdentical(Node n1, Node n2) {
        while (n1 != null && n2 != null) {
            if (n1.data != n2.data)
                return false;
            n1 = n1.next;
            n2 = n2.next;
        }
        return true;
```

```
}

static boolean isPalindrome(Node head) {
        if (head == null || head.next == null)
                return true;

        Node slow = head, fast = head;
        while (fast.next != null && fast.next.next != null) {
                slow = slow.next;
                fast = fast.next.next;
        }

        Node head2 = reverseList(slow.next);
        slow.next = null;
        boolean ret = isIdentical(head, head2);
        head2 = reverseList(head2);
        slow.next = head2;

        return ret;
}

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

        System.out.print("Enter the number of nodes in the linked list: ");
        int n = scanner.nextInt();

        System.out.println("Enter the data for each node:");
        Node head = null, tail = null;

        for (int i = 0; i < n; i++) {

            int data = scanner.nextInt();

            Node newNode = new Node(data);

            if (head == null) {

                head = newNode;

                tail = newNode;

            } else {

                tail.next = newNode;

                tail = newNode;

            }

        }


        boolean result = isPalindrome(head);

        System.out.println(result ? "true" : "false");


        scanner.close();

    }

}
```

Time Complexity : O(n)



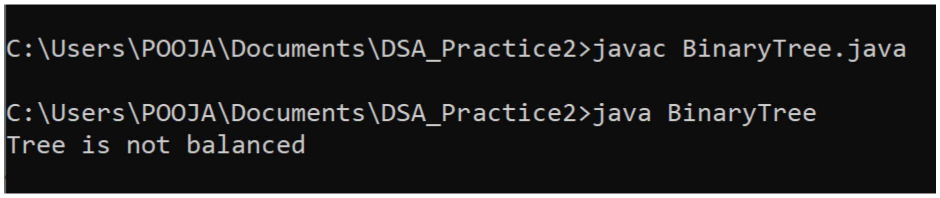
```
C:\Users\POOJA\Documents\DSA_Practice2>javac Palindrome.java

C:\Users\POOJA\Documents\DSA_Practice2>java Palindrome
Enter the number of nodes in the linked list: 5
Enter the data for each node:
1 2 3 2 1
true
```

5.Balanced tree check

```
class Node {
        int data;
        Node left, right;
        Node(int d) {
                data = d;
                left = right = null;
        }
}

class BinaryTree {
        Node root;

        boolean isBalanced(Node node) {
                int lh;
                int rh;

                if (node == null)
                        return true;

                lh = height(node.left);
                rh = height(node.right);

                if (Math.abs(lh - rh) <= 1 && isBalanced(node.left) && isBalanced(node.right))
                        return true;

                return false;
        }

        int height(Node node) {
```

```
        if (node == null)
            return 0;
        return 1 + Math.max(height(node.left), height(node.right));
    }

    public static void main(String args[]) {
        BinaryTree tree = new BinaryTree();
        tree.root = new Node(1);
        tree.root.left = new Node(2);
        tree.root.right = new Node(3);
        tree.root.left.left = new Node(4);
        tree.root.left.right = new Node(5);
        tree.root.left.left.left = new Node(8);

        if (tree.isBalanced(tree.root))
            System.out.println("Tree is balanced");
        else
            System.out.println("Tree is not balanced");
    }
}
```

Time Complexity : O(n^2)



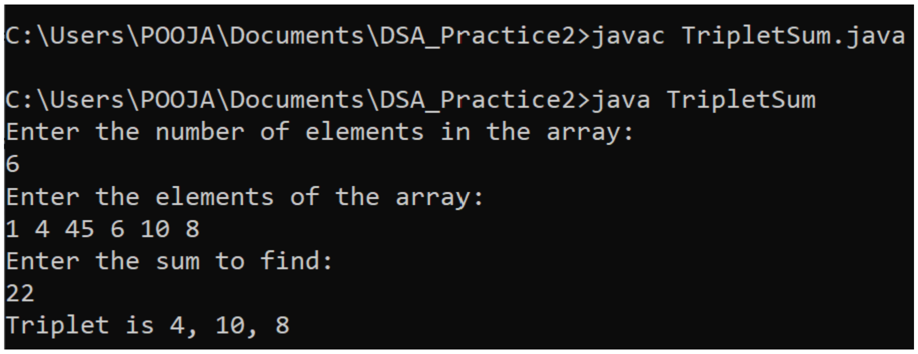
```
C:\Users\POOJA\Documents\DSA_Practice2>javac BinaryTree.java

C:\Users\POOJA\Documents\DSA_Practice2>java BinaryTree
Tree is not balanced
```

6.Triplet sum in array

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

```
public class TripletSum {

    static boolean find3Numbers(int[] arr, int sum) {

        int n = arr.length;


        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] == sum) {

                        System.out.println("Triplet is " + arr[i] + ", " + arr[j] + ", " + arr[k]);

                        return true;

                    }

                }

            }

        }

        return false;

    }


    public static void main(String[] args) {

        Scanner scanner = new Scanner(System.in);


        System.out.println("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.println("Enter the sum to find:");
        int sum = scanner.nextInt();

        if (!find3Numbers(arr, sum)) {
            System.out.println("No triplet found with the given sum.");
        }

        scanner.close();
    }
}
```

Time Complexity : O(n^3)



```
C:\Users\POOJA\Documents\DSA_Practice2>javac TripletSum.java

C:\Users\POOJA\Documents\DSA_Practice2>java TripletSum
Enter the number of elements in the array:
6
Enter the elements of the array:
1 4 45 6 10 8
Enter the sum to find:
22
Triplet is 4, 10, 8
```