

1.Prime factors program

import java.util.Scanner;

public class PrimeFactors {

public static void main(String args[]){

int n umber;

Scanner sc = new Scanner(System.in);

System.out.println("Enter a number ::");

number = sc.nextInt();

for(int i = 2; i< number; i++) {

while(number%i == 0) {

System.out.println(i+" ");

number = number/i;

}

}

if(number >2) {

System.out.println(number);

}

}

}

2.Armstrong program

import java.util.Scanner;

public class Armstrong {

public static void main(String[] args) {

int number, originalNumber, remainder, result = 0;

Scanner sc = new Scanner(System.in);

System.out.println("Enter a number ::");

number = sc.nextInt();

originalNumber = number;

while (originalNumber != 0)

{

remainder = originalNumber % 10;

result += Math.pow(remainder, 3);

originalNumber /= 10;

}

if(result == number)

System.out.println(number + " is an Armstrong number.");

else

System.out.println(number + " is not an Armstrong number.");

}

}

3.Factorical program

import java.util.Scanner;

public class Factorial {

public static void main(String[] args) {

int num;

Scanner sc = new Scanner(System.in);

System.out.println("Enter a number ::");

num = sc.nextInt();

long factorial = 1;

for(int i = 1; i <= num; ++i)

{

// factorial = factorial \* i;

factorial \*= i;

}

System.out.printf("Factorial of %d = %d", num, factorial);

}

}

Day 2:

Bubble sort

import java.io.\*;

class Bubblesorting {

// An optimized version of Bubble Sort

static void bubbleSort(int arr[], int n){

int i, j, temp;

boolean swapped;

for (i = 0; i < n - 1; i++) {

swapped = false;

for (j = 0; j < n - i - 1; j++) {

if (arr[j] > arr[j + 1]) {

// Swap arr[j] and arr[j+1]

temp = arr[j];

arr[j] = arr[j + 1];

arr[j + 1] = temp;

swapped = true;

}

}

// If no two elements were

// swapped by inner loop, then break

if (swapped == false)

break;

}

}

// Function to print an array

static void printArray(int arr[], int size){

int i;

for (i = 0; i < size; i++)

System.out.print(arr[i] + " ");

System.out.println();

}

// Driver program

public static void main(String args[]){

int arr[] = { 64, 34, 25, 12, 22, 11, 90 };

int n = arr.length;

bubbleSort(arr, n);

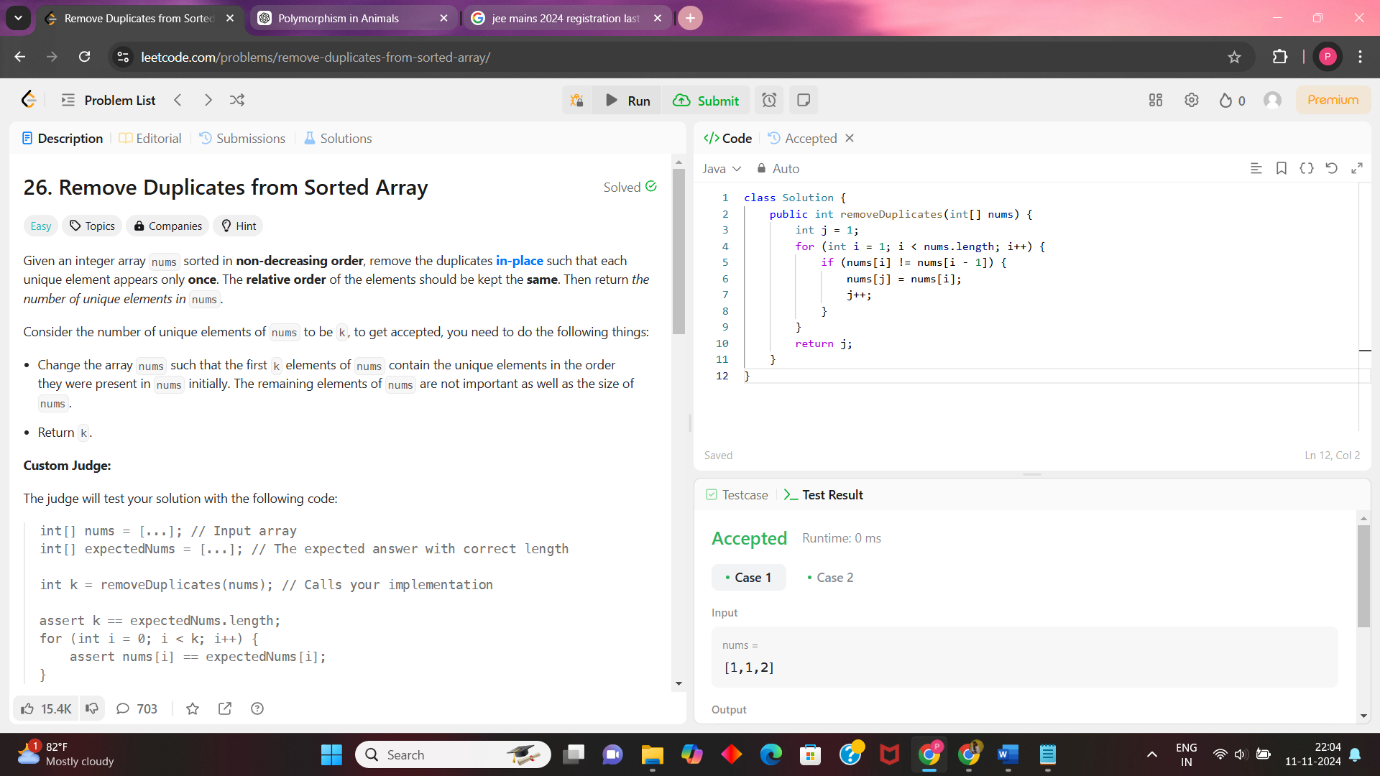
System.out.println("Sorted array: ");

printArray(arr, n);

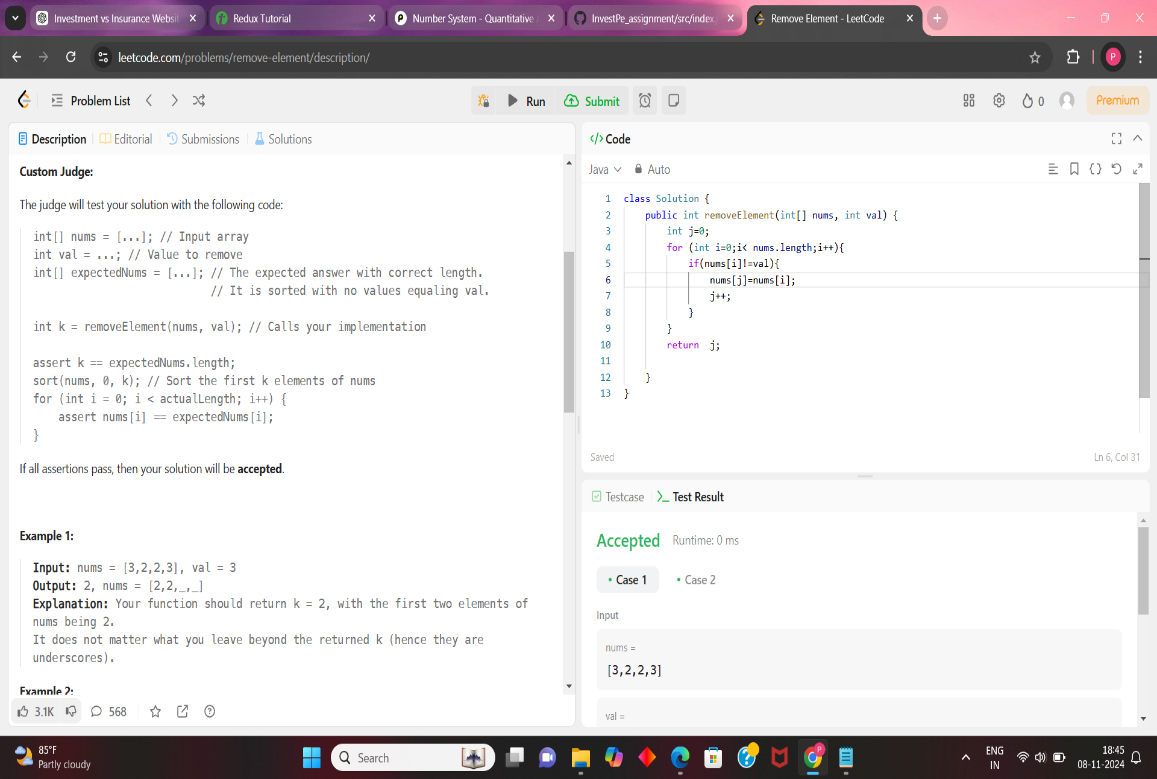
}

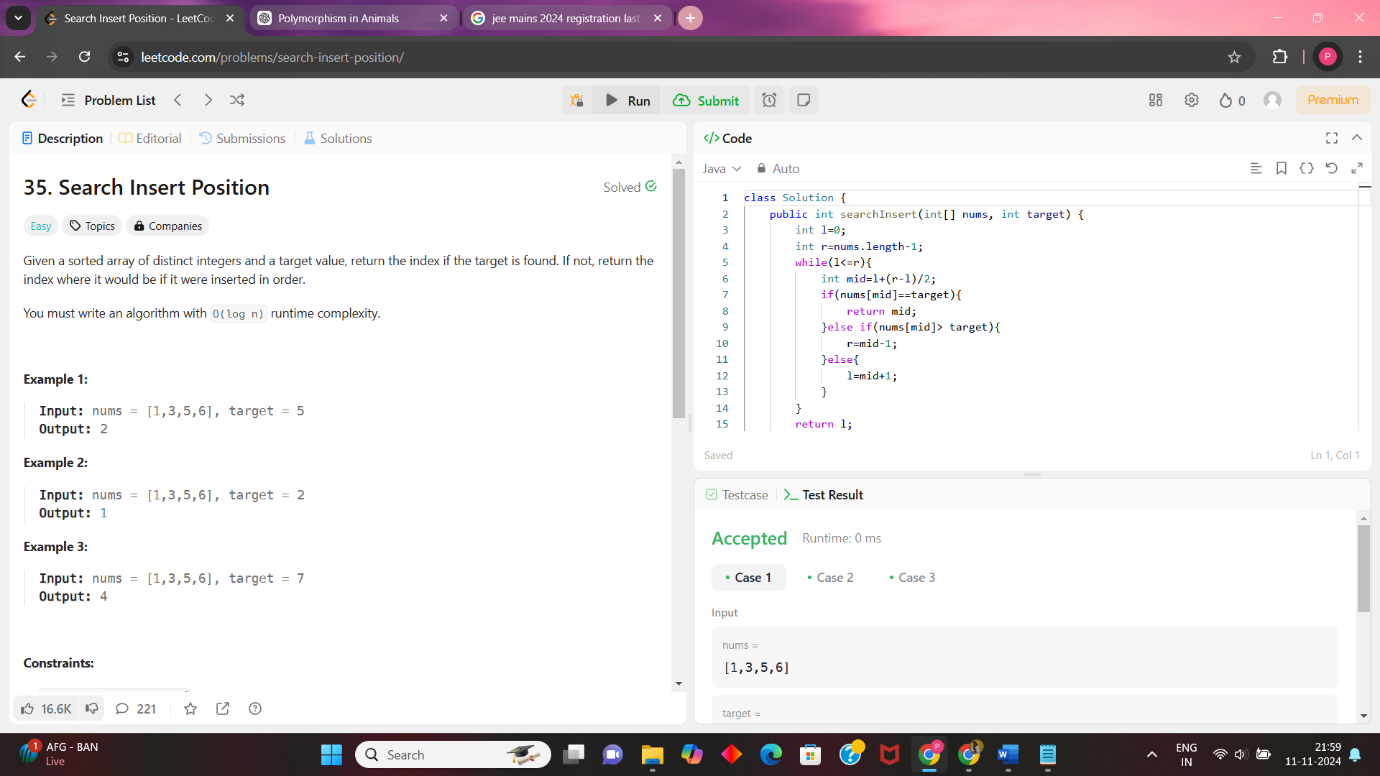
}

DAY 3



DAY4:



DAY 5:

Insertion sort using binary search:

public class InsertionSortBinarySearch {

// Function to perform binary search and find the index to insert element

private static int binarySearch(int[] arr, int item, int low, int high) {

while (low <= high) {

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

if (arr[mid] == item) {

return mid + 1; // If item is equal, insert after the current mid

} else if (arr[mid] < item) {

low = mid + 1;

} else {

high = mid - 1;

}

}

return low; // Position where the item should be inserted

}

// Function to perform insertion sort using binary search

public static void insertionSort(int[] arr) {

for (int i = 1; i < arr.length; i++) {

int current = arr[i];

int j = i - 1;

// Find the position to insert the element

int pos = binarySearch(arr, current, 0, j);

// Shift elements to the right to make room for current element

while (j >= pos) {

arr[j + 1] = arr[j];

j--;

}

// Insert current element at its correct position

arr[j + 1] = current;

}

}

// Test the insertion sort with binary search

public static void main(String[] args) {

int[] arr = {37, 23, 0, 17, 12, 72, 31, 46, 100, 88, 54};

System.out.println("Original array:");

for (int num : arr) {

System.out.print(num + " ");

}

insertionSort(arr);

System.out.println("\nSorted array:");

for (int num : arr) {

System.out.print(num + " ");

}

}

}

Day 6:

Median of the two unsorted arrays(without using concatenation)

public class MedianOfTwoArraysSimple {

public static double findMedian(int[] nums1, int[] nums2) {

int m = nums1.length;

int n = nums2.length;

int totalLength = m + n;

int mid1 = -1, mid2 = -1;

int i = 0, j = 0;

for (int count = 0; count <= totalLength / 2; count++) {

mid1 = mid2; //

if (i < m && (j >= n || nums1[i] < nums2[j])) {

mid2 = nums1[i++];

} else {

mid2 = nums2[j++];

}

}

if (totalLength % 2 == 0) {

return (mid1 + mid2) / 2.0;

} else {

return mid2;

}

}

public static void main(String[] args) {

int[] nums1 = {1, 3, 8};

int[] nums2 = {7, 9, 10, 11};

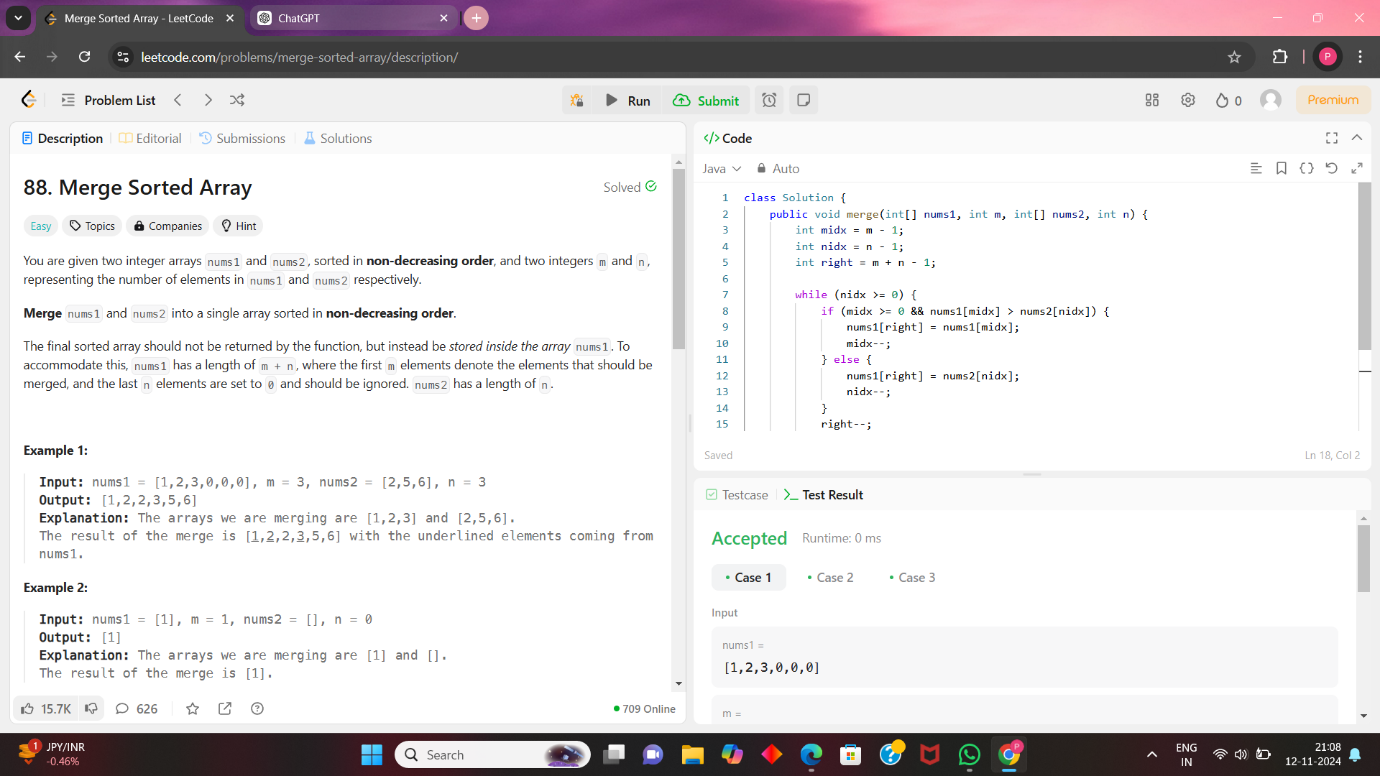
System.out.println("Median is: " + findMedian(nums1, nums2));

}

}

Insertion sort in binary search techinque

Leecode 88



Leetcode 217

