Άσκηση 1: Δυαδική Αναζήτηση (Binary Search)

Περιγραφή

Γράψτε έναν αλγόριθμο που χρησιμοποιεί τη δυαδική αναζήτηση για να βρει τη θέση ενός στοιχείου σε έναν ταξινομημένο πίνακα. Αν το στοιχείο δεν υπάρχει στον πίνακα, η συνάρτηση πρέπει να επιστρέφει -1.

Λύση

```
using System;

class Program
{
    static void Main()
    {
        int[] array = { 1, 2, 3, 4, 5, 6, 7, 8, 9, 10 };
        int key = 7;
        int result = BinarySearch(array, 0, array.Length - 1, key);

        if (result != -1)
        {
            Console.WriteLine($"Element {key} found at index: {result}");
        }
        else
        {
            Console.WriteLine($"Element {key} not found in the array.");
        }
    }

    static int BinarySearch(int[] array, int left, int right, int key)
    {
        if (left > right)
        {
            return -1;
        }

        int middle = (left + right) / 2;

        if (array[middle] == key)
        {
            return middle;
        }
        else if (array[middle] > key)
        {
            return BinarySearch(array, left, middle - 1, key);
```

```
        }
        else
        {
            return BinarySearch(array, middle + 1, right, key);
        }
    }
}
```

Άσκηση 2: Ταξινόμηση Συγχώνευσης (Merge Sort)

Περιγραφή

Γράψτε έναν αλγόριθμο που χρησιμοποιεί την ταξινόμηση συγχώνευσης για να ταξινομήσει έναν πίνακα ακεραίων.

Λύση

```
using System;

class Program
{
    static void Main()
    {
        int[] array = { 38, 27, 43, 3, 9, 82, 10 };
        MergeSort(array);

        Console.WriteLine("Sorted array: " + string.Join(", ", array));
    }

    static void MergeSort(int[] array)
    {
        if (array.Length <= 1)
            return;

        int mid = array.Length / 2;
        int[] left = new int[mid];
        int[] right = new int[array.Length - mid];

        Array.Copy(array, 0, left, 0, mid);
        Array.Copy(array, mid, right, 0, array.Length - mid);

        MergeSort(left);
        MergeSort(right);
        Merge(array, left, right);
    }
```

```
    static void Merge(int[] array, int[] left, int[] right)
    {
        int i = 0, j = 0, k = 0;

        while (i < left.Length && j < right.Length)
        {
            if (left[i] <= right[j])
            {
                array[k++] = left[i++];
            }
            else
            {
                array[k++] = right[j++];
            }
        }

        while (i < left.Length)
        {
            array[k++] = left[i++];
        }

        while (j < right.Length)
        {
            array[k++] = right[j++];
        }
    }
}
```

Άσκηση 3: Μέγιστο Υποπίνακα (Maximum Subarray Problem)

Περιγραφή

Γράψτε έναν αλγόριθμο που χρησιμοποιεί την τεχνική διαίρεσης και κυρίαρχίας για να βρει το μέγιστο άθροισμα συνεχόμενων στοιχείων σε έναν πίνακα (Maximum Subarray Problem).

Λύση

```
using System;

class Program
```

```
{
    static void Main()
    {
        int[] array = { -2, 1, -3, 4, -1, 2, 1, -5, 4 };
        int maxSum = FindMaximumSubarray(array, 0, array.Length - 1);
        Console.WriteLine($"Maximum subarray sum is: {maxSum}");
    }

    static int FindMaximumSubarray(int[] array, int left, int right)
    {
        if (left == right)
        {
            return array[left];
        }

        int middle = (left + right) / 2;

        int leftMax = FindMaximumSubarray(array, left, middle);
        int rightMax = FindMaximumSubarray(array, middle + 1, right);
        int crossMax = FindMaxCrossingSubarray(array, left, middle, right);

        return Math.Max(Math.Max(leftMax, rightMax), crossMax);
    }

    static int FindMaxCrossingSubarray(int[] array, int left, int middle, int right)
    {
        int leftSum = int.MinValue;
        int sum = 0;
        for (int i = middle; i >= left; i--)
        {
            sum += array[i];
            if (sum > leftSum)
            {
                leftSum = sum;
            }
        }

        int rightSum = int.MinValue;
        sum = 0;
        for (int i = middle + 1; i <= right; i++)
        {
            sum += array[i];
            if (sum > rightSum)
            {
                rightSum = sum;
            }
```

```
        }

        return leftSum + rightSum;
    }
}
```