```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Diagnostics;
using System.Text;
using System.Threading.Tasks;

namespace L2
{
    class Program
    {
        static int[] x = { 1, 2, 3, 4, 5, 6, 7 };
        static int[] y = { 1, 2, 3, 5, 4, 6, 8, 6 };
        static int[] dov = { 1, 2, 3, 5, 4, 6, 8, 6 };
        static int[] A = new int[0];
        static int[] B = new int[0];
        static int min;
        static void Main(string[] args)
        {
            int[] numOfData = { 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20 ,40, 60, 80,
100};
            Benchmark(numOfData);
            Random randNum = new Random();


        }

        private static void Benchmark(int[] dataCount)
        {
            Console.WriteLine("Dinaminio programavimo laikai: ");
            for (int i = 0; i < dataCount.Length; i++)
            {
                x = randomArray(dataCount[i] * 10000000);
                y = randomArray(dataCount[i] * 10000000);
                var sw = new Stopwatch();
                sw.Start();
                FD2(x.Length, y.Length);
                sw.Stop();
                Console.WriteLine("{0} - {1}", dataCount[i], sw.Elapsed); ;
            }
            Console.WriteLine("Rekursinio programavimo laikai: ");
            for (int i = 0; i < dataCount.Length - 3; i++)
            {
                x = randomArray(dataCount[i]);
                y = randomArray(dataCount[i]);
                var sw = new Stopwatch();
                sw.Start();
                FD(x.Length, y.Length);
                sw.Stop();
                Console.WriteLine("{0} - {1}", dataCount[i], sw.Elapsed); ;
            }
            Console.WriteLine("Paralelinio programavimo laikai: ");
            for (int i = 0; i < dataCount.Length; i++)
            {
                x = randomArray(dataCount[i]);
                y = randomArray(dataCount[i]);
                var sw = new Stopwatch();
                sw.Start();
            FP(x.Length, y.Length);
            sw.Stop();
            Console.WriteLine("{0} - {1}", dataCount[i], sw.Elapsed); ;
        }
        Console.WriteLine("Dinaminio dovanų paskirstymo laikai: ");
        for (int i = 0; i < dataCount.Length; i++)
        {
            dov = randomArray(dataCount[i] * 2000);
            var sw = new Stopwatch();
            sw.Start();
            Dov2();
            sw.Stop();
            Console.WriteLine("{0} - {1}", dataCount[i], sw.Elapsed); ;
        }
        Console.WriteLine("Rekursiško dovanų paskirstymo laikai: ");
        for (int i = 0; i < dataCount.Length; i++)
        {
            dov = randomArray(dataCount[i]);
            var sw = new Stopwatch();
            sw.Start();
            reorder(dov, 0, dov.Length - 1);
            sw.Stop();
            Console.WriteLine("{0} - {1}", dataCount[i], sw.Elapsed); ;
        }
    }
    private static int FD(int m, int n)
    {
        if (n == 0) return m;
        if (m == 0 && n > 0) return n;

        int temp;
        int min = 1 + FD(m - 1, n);
        if ((temp = 1 + FD(m, n - 1)) < min)
            min = temp;
        if ((temp = D(m, n) + FD(m - 1, n - 1)) < min)
            min = temp;
        return min;
    }

    private static int FD2(int m, int n)
    {
        int tm = m;
        int tn = n;
        int cm = 0;
        int rez1 = 1 * m + n;
        int rez2 = 1 * n + m;
        int rez3 = 0;
        int min = rez1;

        while (n > 0 && m > 0)
        {
            tn--;
            tm--;
            if (tn == 0)
            {
                rez3 = m + cm;
                break;
            }
        if (tm == 0)
        {
            rez3 = n + cm;
            break;
        }
        if (x[tm] == y[tn])
            cm++;
    }
    if (rez2 < rez1)
        min = rez2;
    if (rez3 < min)
        min = rez3;
    return min;
}

private static int FP(int m, int n)
{
    if (n == 0) return m;
    if (m == 0 && n > 0) return n;
    int countCPU = 3;
    Task[] tasks = new Task[countCPU];
    var task1 = Task.Factory.StartNew(() => 1 + FD(m - 1, n));
    var task2 = Task.Factory.StartNew(() => 1 + FD(m, n - 1));
    var task3 = Task.Factory.StartNew(() => D(m, n) + FD(m - 1, n - 1));
    Task.WaitAll(task1, task2, task3);
    int min = task1.Result;
    if (task2.Result < min)
        min = task2.Result;
    if (task3.Result < min)
        min = task3.Result;
    return min;
}

private static int D(int i, int j)
{
    if (i >= x.Length || j >= y.Length) return 0;
    if (x[i] == y[j]) return 1;
    return 0;
}

public static void reorder(int[] list, int k, int m)
{
    if (list.Length == 0)
        return;
    if (k == m)
    {
        split(list, 1);
    }
    else
    {
        for (int i = k; i <= m; i++)
        {
            swap(ref list[k], ref list[i]);
            reorder(list, k + 1, m);
            swap(ref list[k], ref list[i]);
        }
    }
}
```

```
public static void swap(ref int a, ref int b)
{
    int temp = a;
    a = b;
    b = temp;
}

public static void split(int[] list, int s)
{
    if (s == list.Length)
        return;
    int Sa = 0;
    int Sb = 0;
    int[] Aa = new int[s];
    for (int i = 0; i < s; i++)
    {
        Sa += list[i];
        Aa[i] = list[i];
    }
    int[] Ab = new int[list.Length - s];
    int c = 0;
    for (int i = s; i < list.Length; i++)
    {
        Sb += list[i];
        Ab[c++] = list[i];
    }
    if (Math.Abs(Sa - Sb) < min)
    {
        min = Math.Abs(Sa - Sb);
        A = new int[Aa.Length];
        for (int i = 0; i < Aa.Length; i++)
            A[i] = Aa[i];
        B = new int[Ab.Length];
        for (int i = 0; i < Ab.Length; i++)
            B[i] = Ab[i];
    }
    if (min == 0)
        return;
    split(list, ++s);
}
private static int getMax(int[] list)
{
    int max = 0;
    foreach (var item in list)
        if (item > max && item > 0)
            max = item;
    return max;
}
private static int getMin(int[] list)
{
    int min  = getMax(list);
    int ind = 0;
    for (int i = 0; i < list.Length; i++)
        if (list[i] < min && list[i] > 0)
        {
            min = list[i];
            ind = i;
        }
    list[ind] *= -1;
    return min;
}
private static void Dov2()
{
    //Console.WriteLine("Dinaminis");
    int sumA = 0;
    int sumB = 0;
    int[] dov2 = dov;
    int dovC = dov2.Length;
    bool broken = false;
    for (int i = 0; i < dov2.Length; i++)
    {
        if (sumA == sumB)
        {
            sumA += getMin(dov2);
            dovC--;
            if (dovC == 0)
            {
                broken = true;
                break;
            }
        }
        if (broken) break;
        while (sumA < sumB)
        {
            sumA += getMin(dov2);
            dovC--;
            if (dovC == 0)
            {
                broken = true;
                break;
            }
        }
        if (broken) break;
        while (sumB < sumA)
        {
            sumB += getMin(dov2);
            dovC--;
            if (dovC == 0)
            {
                broken = true;
                break;
            }
        }
        if (broken) break;
    }
    //Console.WriteLine("A " + sumA);
    //Console.WriteLine("B " + sumB);
}

private static void printArray(string mesg, int[] array)
{
    Console.WriteLine(mesg);
    foreach (var it in array)
        Console.Write(it + " ");
    Console.WriteLine();
}
private static int[] randomArray(int length)
{
    Random randNum = new Random();
    int[] ar = new int[length];
    for (int i = 0; i < ar.Length; i++)
        ar[i] = randNum.Next(1, 20);
    return ar;
}
}
}
```