Задача 1:

long Compute(int[] arr)

{

long count = 0;

for (int i=0; i<arr.Length; i++)

{

int start = 0, end = arr.Length-1;

while (start < end)

if (arr[start] < arr[end])

{ start++; count++; }

else

end--;

}

return count;

}

При arr.length = n.

Време на работа (Worst case) - O(n\*n).

Обосновка - външният for цикъл винаги ще се изпълнява n на брой пъти. Вътрешният while цикъл ще се изпълни винаги n - 1 на брой пъти, защото началната му стойност винаги е нула, крайната n - 1, и промяната или на start или end винаги е с едно. Като крайна приблизителна бройка на екзекуции имаме n \* (n - 1 + 2) (+2 заради вътршният if/else.)

Задача 2:

long CalcCount(int[,] matrix)

{

long count = 0;

for (int row=0; row<matrix.GetLength(0); row++)

if (matrix[row, 0] % 2 == 0)

for (int col=0; col<matrix.GetLength(1); col++)

if (matrix[row,col] > 0)

count++;

return count;

}

При matrix.GetLength(0) = n и matrix.GetLength(1) = m;

Време на работа (Worst case) - O(n \* m);

Обосновка - въшният цикъл винаги ще се изпълни n на брой пъти. Вътрешният цикъл е зависим от услвоният конструктор но в най-лошият случай които е и ограничаващият кайс ще се изпълнява на всеки цикъл. Самият вътрешен цикъл обхожда всикчи стойности на m което ни дава крайна броика в най лошият случай - n \* m;

Задача 3:

long CalcSum(int[,] matrix, int row)

{

long sum = 0;

for (int col = 0; col < matrix.GetLength(0); col++)

sum += matrix[row, col];

if (row + 1 < matrix.GetLength(1))

sum += CalcSum(matrix, row + 1);

return sum;

}

Console.WriteLine(CalcSum(matrix, 0));

При matrix.GetLength(0) = n и matrix.GetLength(1) = m;

Забележка - има малка грешка в кода което води до това че col реално обхожда по n което е броят на редовете и row обхожда по m което е броят на колоните. От тук насетне ще третирам кода все едно е написан правилно а именно:

long CalcSum(int[,] matrix, int row)

{

long sum = 0;

for (int col = 0; col < matrix.GetLength(1); col++)

sum += matrix[row, col];

if (row + 1 < matrix.GetLength(0))

sum += CalcSum(matrix, row + 1);

return sum;

}

Console.WriteLine(CalcSum(matrix, 0));

Време на работа (Worst case) - O(n \* m)

Обосновка - for цикълът винаги ще се изпълнява m на брой пъти. If условието ще се изпълни рекурсивно n на брой пъти, което води до обща стойност на сложност - n \* (m + 2) което е n \* m + 2 \* n което е n \* m за достатъчно големи n и m. + 2 идва от създаването на стойност и проверката на if-a при всяко изпълнение на CalcSum.