1. **What is the expected running time of the following C# code? Explain why. Assume the array's size is n.**

**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;**

**}**

|  |
| --- |
| *Explanation:* |
|  | *Running time O(n^2), number of elementary steps ~(n\*(n-1)).* |
|  | *Outer loop "for" will make n cycles. Inner loop "while" will make n-1 cycles because start will allways lower than end.* |

1. **What is the expected running time of the following C# code? Explain why.**

**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;**

**}**

**Assume the input matrix has size of n \* m.**

|  |
| --- |
|  |
|  | *Running time average and worst O(n\*m), best O(n).*  *Number of elementary steps is ~(n + x\*m) where x is the number of cells in the first column of the matrix holding value divisible by 2.* |
|  | *Outer loop "for" will make n cycles. Inside we have a conditional branch with m operations that run every time when the value of the current first column is divisible by 2.* |
|  | *Best case is when none of them is divisible by 2.* |

1. **\* What is the expected running time of the following C# code? Explain why.**

**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));**

**Assume the input matrix has size of n \* m.**

|  |
| --- |
|  |
|  | *Runing time O(n\*m).*  *Number of elementary steps is ~(m + (m\*(n-row)))* |
|  | *The algorithm passes recursively through each of the rows of the matrix with (n - row) steps.* |
|  | *For each row we nake m steps to calculate the sum.* |