Lab Growth of Functions

Create a project.

Include the following in the cs file:

using System;

using System.Diagnostics;

using System.Numerics;

Also add the reference “System.Numerics” in the Solution Explorer by right clicking “References” and searching for “System.Numerics” in the .NET tab.

A function that takes input n, and runs in n2 steps:

private static void nsquaredgrowthversion1(int n)

{

int sum = 0;

for (int i = 0; i < n; i++)

for (int j = 0; j < n; j++)

sum = sum + 1;

}

Part 1

1. Add a stop watch to track the time the nsquaredgrowthversion1 function. You will call the stopwatch after getting the number from the user and stop the stopwatch immediately after the function call.

Stopwatch sw = Stopwatch.StartNew(); //creates sw and starts stopwatch

//code to time goes here, you can call nsquaredgrowthversion1 here

sw.Stop(); // stops stopwatch

Console.WriteLine("Time used: {0} secs", sw.Elapsed.TotalMilliseconds / 1000);

1. 1. At the command prompt ask the user for a number for n.
   2. Read in a string and covert to a integer
   3. Time how long it takes to run the nsquaredgrowthversion1
   4. The nsquaredgrowthversion1, n= \_\_1000\_\_, time elaped= \_\_0.3721 sec\_\_\_
2. Change the Build Configuration to “Release” mode.
3. Find the largest n, so that the time elapsed is less than 60 seconds. What is n ?

N = 450,000 58.38 sec

1. Change “sum = sum + 1;” to “sum+=1;” Is the code any faster?

Not really.

1. Change “sum = sum + 1;” to “sum++;” Is the code any faster?

All pretty much the same.

1. Change “sum = sum + 1;” to “sum=sum+31” Is the code any slower?
2. All pretty much the same.
3. Explain the difference (internally) between “sum = sum + 1;” , “sum+=1;” and “sum++;”

Sum = sum +1 is finding sum in memory preforming an add 1 to it and storing it back in the sum position in memory.

Sum+=1 is finding the sum in memory and adding one to it.

Sum++ is incrementing sum.

1. Part 2

Write functions for each of the following using “sum++.” Create unique code for everything except 2n and n!. For those two just make a function call for the code used for n. Submit your code to the dropbox. Complete the chart by finding the largest input n such that the code runs in the time limit given. Record that value of n. For lg n use BigInteger instead of ints with 1 second time limit only-it just has to be close to 1 second for this.

|  |  |  |
| --- | --- | --- |
| function | 1 second | 1 minute(60 s) |
| lg n (use BigInteger) | 10000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000 | ~~xxxxxxxxxxx~~ |
| n | 1,200,000,000 | 100,000,000,000 |
| n lg n (use Binary Search) | 4,500,000 | 150,000,000 |
| n2 | 58,000 | 450,000 |
| n3 | 1500 | 5900 |
| n4 | 227 | 660 |
| 2n | 30 | 36 |
| n! | 12 | 14 |

Part 3 Complete the chart

For each function category and time value given below, find the largest input size that can be computed on a machine that runs 10,000,000 instructions per second , in the allotted time – for example, how much input can be processed by an n2 algorithm in 1 minute?

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| function | 1 second | 1 minute  60s\* | 2 hours  10800s | 2 days  259,200s | 2 months  23,328,000s | 2 years  94,608,000s | 30,000,000,000s  (which is about 1000 years) |
| n | 10,000,000 | 600,000,000 | 108,000,000,000 | 2,592,000,000,000 | 233,280,000,000,000 | 946,080,000,000,000 | 300,000,000,000,000,000 |
| n2 | 3,162 | 24,494 | 328,633 | 1,609,968 | 15,273,506 | 30,758,413 | 547,722,557 |
| n3 | 215 | 843 | 4,762 | 13,736 | 61,559 | 98,169 | 669,432 |
| 2n | 24 | 25 | 36 | 41 | 47 | 49 | 58 |
| n! | 10 | 12 | 14 | 15 | 16 | 17 | 19 |