Joseph Lewis

9/10/2018

PA2 Report

Student ID 11567186

**Problem Statement:**

The goal of this programming is to compare running time for four different functions used to find “maxsubsum” of an array of integers. The “maxsubsum” is a max sum of integers added consecutively from a beginning point to an ending point of an array.

**Experimental Setup:**

I first had to implement the coded functions provided in class that will be test for running time. Then I created an algorithm that prompted the user which file they would like to test using ‘ifstream’. I used the given text files to conduct my experiment. I created an algorithm that would open the specific type file and add integers into a vector array. After that, I called the functions “maxSubSum1”, “maxSubSum2”, “maxSubSum3”, and “maxSubSum4” found the max sub sum by entering the vector array in the parameters. To record running time, I used to Chrono library to record the time of each function in microseconds. The CPU I used for this experiment is a “Intel(R)Core(TM)i7-4710HQ”, which is a 64-bit operating system. The clock speed used for testing was 2.50 GHz. The RAM use for this experiment had a memory value of 8 gigabytes with a speed of 1600 MHz. The text files used for testing had different input sizes: 8, 6, 32, 128, 256, 512, 1024, 2048, 4096, 8192. For each of the sizes, there were ten different text files used for testing. I tested every text file for each of the given input size. Then I computed the average running times for each ‘maxSubSum’ function by the input sizes. After that, I had four functions to compare running times and eleven data points needed to plot the results. All the testing in this experiment was conducted in the EECS SSH server, which involved the Unix environment.

**Experimental Results:**

Overall, the results did confirm my theoretical expectations because there was evidence that the order from greatest to least value in running time were: maxSubSum1, maxSubSum2, maxSubSum3, and maxSubSum4. The results proved that the running time for “maxSubSum1” was significantly larger than the other functions. This was caused by “maxSubSum1” runtime complexity is theta(n3). This runtime was caused by a for loop being inside two for loops. The function “maxSubSum2” had the second longest running time, which was caused by the runtime complexity equaling theta(n2). The determined runtime time complexity occurred because there was a for loop inside another for loop. The function with the third longest running time was “maxSubSum3”, which had a runtime complexity of O(n\*Log(n)). This runtime complexity was caused by a recursive call from the use of the function “maxSumRec” which caused the tested array to be split in pieces many times when tested. This caused O(Log(n)) to occur for the runtime complexity. Also, there was a for loop used in the function, which caused the result of time complexity to equal O(N\*Log(N)). The fastest runtime in this experiment was “maxSubSum4”, which had a runtime complexity of theta(n). There was only one for loop in the function “maxSubSum4”, which caused this function to have the smallest runtime complexity compared to the other tested functions. After the 8192-bit file test the average runtime results were: maxSubSum1 = 318362587 microseconds, maxSubSum2 = 213219 microseconds, maxSubSum3 = 926 microseconds, and maxSubSum4 = 67 microseconds. I created three line, graphs that were used to plot the data after the testing. I had to create three graphs because I needed to make the runtime growth comparable for each of the functions. The graphs showed a significant difference of running time growth.

The first graph shows the function maxSubSum3’s growth is significantly larger than the other functions.

The second graph compares runtime growth between functions maxSubSum2, maxSubSum3, and maxSubSum4. Overall the runtime growth of maxSubSum2 was significantly greater.

The third graph compared runtime growth between functions maxSubSum3 and maxSubSum4.