CSCI 421 Design and Analysis of Algorithms Spring 2019

Assignment 1

**Running Time Comparison of 3-Sum Algorithms**

We have implemented two faster algorithms (~N2) for the 3-Sum problem in Lecture 1 Activity 1. Now we would like to compare them with the brute-force algorithm (~N3) and the binary-search based algorithm (~N2logN, shown in Lecture 1 slide 48) to see their running time in practice.

You need to find the number of triples of N random int values that sum to 0, where the values are uniformly distributed between –M and M, where M is not small. For instance, N can range from 250, 500, 1000, 2000, 4000, 8000, 16000 and you need to find the running time of all the four algorithms aforementioned. Generate a table and a standard plot as shown in Lecture 1 slide 16 and 17, respectively.

Note 1: Regarding the binary-search based algorithm, you need to implement the binary search method by yourself. To obtain a sorted array, you can use the java.util.Arrays library. Regarding the hash-table based algorithm, do not include the table creation part when you measure the running time.

Note 2: A slide showing how to measure running time in Java is attached. You can replace “System.currentTimeMillis” with “System.nanoTime” if you find the running time is too small to be displayed.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Array Size | Brute Force | 3 Sum Hash | Three Sum Imporved | Binary insertion |
| 250 | 15 | 14 | 3 | 1 |
| 500 | 23 | 22 | 6 | 3 |
| 1000 | 133 | 114 | 10 | 6 |
| 2000 | 657 | 759 | 17 | 8 |
| 4000 | 23938 | 24059 | 52 | 21 |
| 8000 | 214069 | 172817 | 232 | 35 |
| 16000 | 195755 | 160020 | 214 | 31 |

import java.util.Random;

//Driver class

public class Assign1{

public static void main(String[] args) {

Random r = new Random();

int[] integers = new int[16000];

for (int i = 0; i < integers.length; i++) {

integers[i] = r.nextInt();

}

long startTime, endTime, elapsed;

System.out.println("Array Size of random numbers: " + integers.length);

ThreeSumBrute tsb = new ThreeSumBrute();

ThreeSumHash tsh = new ThreeSumHash();

ThreeSumImp tsi = new ThreeSumImp();

BinSearch bs = new BinSearch();

startTime = System.currentTimeMillis();

tsb.count(integers);

endTime = System.currentTimeMillis();

elapsed = endTime - startTime;

System.out.println("Time using Brute Force Method: " + elapsed);

startTime = System.currentTimeMillis();

tsh.count(integers);

endTime = System.currentTimeMillis();

elapsed = endTime - startTime;

System.out.println("Time using Hash method: " + elapsed);

startTime = System.currentTimeMillis();

tsi.count(integers);

endTime = System.currentTimeMillis();

elapsed = endTime - startTime;

System.out.println("Time using Three sum Improved: " + elapsed);

startTime = System.currentTimeMillis();

bs.binaryInsSort(integers);

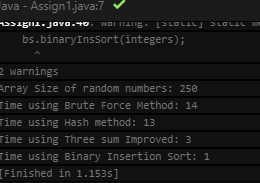
endTime = System.currentTimeMillis();

elapsed = endTime - startTime;

System.out.println("Time using Binary Insertion Sort: " + elapsed);

}

}



\*\*just to show you its running