**Program Structures and Algorithms**

**Assignment-2**

**Summer-2022**

**Dimpleben Kanjibhai Patel – 002965372**

**Problem:** 3-SUM

**Task**: Solve 3-SUM using the Quadrithmic, Quadratic and quadraticWithCalipers approaches.

Submit:

* Evidence of your unit tests running
* A spreadsheet showing your timing observations using the doubling method for at least five values of N- for each of the algorithms.
* Your brief explanation of why the quadratic methods works.

**Output:**

1. Evidence of all unit test passing:

Calendar

Description automatically generated with low confidence

1. Timing Observation:

Observation Table for Raw time per run:

|  |  |  |  |
| --- | --- | --- | --- |
| **N** | **Qaudratic (n2)(msec)** | **Quadrithmic (n2log(n))(msec)** | **Cubic(n3)(msec)** |
| 250 | 24 | 12 | 25 |
| 500 | 34 | 33 | 69 |
| 1000 | 36 | 53 | 276 |
| 2000 | 87 | 139 | 1898 |
| 4000 | 318 | 548 | 14458 |
| 8000 | 1098 | 1924 | 122062 |
| 16000 | 3563 | 7287 | 2566687 |

Graph for Qaudratic Vs Qaudrithmic Vs Cubic

A picture containing line chart

Description automatically generated

As from the above graph, we can observe for larger value of n, time is drastically increases. We can not see clear difference for Quadratic and Quadrithic solutions, so we will plot separate graph for that.

Graph for Quadratic Vs Quadrithmic

Chart, line chart

Description automatically generated

Observation Table for Normalized time per run:

|  |  |  |  |
| --- | --- | --- | --- |
| **N** | **Qaudratic (n2)** | **Quadrithmic (n2log(n))** | **Cubic(n3)** |
| 250 | 384 | 24.10 | 1.60 |
| 500 | 136 | 14.72 | 0.55 |
| 1000 | 36 | 5.32 | 0.28 |
| 2000 | 21.75 | 3.17 | 0.24 |
| 4000 | 19.88 | 2.86 | 0.23 |
| 8000 | 17.16 | 2.32 | 0.24 |
| 16000 | 13.92 | 2.04 | 0.63 |

A picture containing graphical user interface

Description automatically generated

1. Explanation:

Cubic solution is a brute force approach which takes O(n3) time complexity as there are three nested loops for traversing the array.

For Qaudrithmic solution, first we sort the array, then we have two nest for loop for the elements and third element we can search using binary search algorithm. Therefore it will take O(n2log(n)) time complexity.

For Quadratic solution, First we will sort the array which will take nlog(n) time complexity. After sorting an array, we will fix first element of the triplet. Then we will use two pointer method to find whether there is any pair whose sum is equal to (0 – first element). The two-pointer method will take linear time i.e. n and fixing each element will take n time so total time complexity will be O(n2).