Program Structures and Algorithms

Spring 2023(SEC 03)

NAME: Vipul Rajderkar

NUID: 002700991

**Task:**

Solve 3-SUM using the Quadrithmic, Quadratic, and (bonus point) quadraticWithCalipers approaches, as shown in skeleton code in the repository.

**Relationship Conclusion:**

After solving 3-Sum problem using three different approaches, it can be conferred that Quadratic Approach (quadratic as well as quadratic with calipers) performs better than rest 2 viz. Quadrithimic and cubic.

Time complexities of all approaches are as below:

**Quadratic: O(n²)**

**Quadrithimic: O(n² log n)**

**Cubic: O(n3)**

**Evidence to support that conclusion:**

Approach 1: 3-Sum Quadratic

Step 1: Select a number as a middle number (say x) from an array (complexity: O(n))

Step 2: Use 2 pointers pointing number less than and greater than middle number (x)

Step 3: While both pointers don’t reach middle number’s index -> check the sum and increment corresponding pointers depending on whether the sum is less than or equal to 0.

Code Snippet:

Text

Description automatically generated

Approach 2: 3-Sum Quadratic with Calipers

The computation is less heavy than approach 1 because of the invariant that remaining 2 numbers are going to be more than first number.

Step 1: Select a number as a first number (say x) from a sorted array (complexity: O(n))

Step 2: Use 2 pointers pointing numbers just greater than x and the largest number.

Step 3: While both pointers don’t reach middle number’s index -> check the sum and increment/decrement corresponding pointers depending on whether the sum is less than or equal to 0.

Code Snippet:

Text

Description automatically generated

Approach 3: 3-Sum Quadrithimic

Step 1: Select a pair of number x and y (Complexity: O(n²))

Step 2: Find the third element(z = -(x+y)) using BinarySearch (Complexity: O(log n))

Code Snippet:

Text

Description automatically generated

**Benchmarking Code Snippet and Output:**

Graphical user interface, text

Description automatically generated

Graphical user interface, text

Description automatically generated

Graphical user interface, text

Description automatically generated

**Graphical Representation:**

Timing Table:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **N** | **Quadratic(ms)** | **Quadratic with Calipers(ms)** | **Quadrithimic(ms)** | **Cubic(ms)** |
| 250 | 1.84 | 1.09 | 1.08 | 5.97 |
| 500 | 3.14 | 1.14 | 2.88 | 40.68 |
| 1000 | 6.8 | 4.25 | 14.25 | 317.9 |
| 2000 | 28.2 | 19 | 71.4 | 2518.7 |
| 4000 | 149.2 | 91.8 | 332.6 | 20007.6 |
| 8000 | 686.67 | 527.67 | 1466.33 | 161619 |
| 16000 | 2676.5 | 2263.5 | 6888 | 1285873 |

**3-Sum Quadratic:**

**3-Sum Quadrithimic:**

**Cubic:**

**Unit Test Screenshots:**

**Text

Description automatically generated**

**Explanation of why Quadratics work:**

Quadratic approach performs definitely better than the rest as it is most efficient approach with a time complexity of O(N2). Since we use a two-pointer approach, the computation time is reduced as depending on the difference between expected and actual sum only required pointers can be moved.