Program Structures and Algorithms

Summer I 2023(SEC – 1)

Assignment -2 (3 Sum)

NAME: Amrita Dubey

NUID: 002760658

**Task:**

* **3-SUM using the Quadrithmic, Quadratic, and quadraticWithCalipers approaches**
* **Record timing observations--using the doubling method for at least five values of N--for each of the algorithms.**
* **Explanation of algorithm**

**Explanation of why the quadratic method(s) work:**

The algorithm find triples of numbers in a sorted array such that the sum of the three numbers is equal to 0.

This algorithm utilizes a two-pointer approach, where one pointer starts from the left of the middle index (l = j - 1) and the other pointer starts from the right of the middle index (r = j + 1). These pointers will move towards each other.

Inside the while loop, the code calculates the sum of the numbers at positions l (left), j (input), and r(right) using the expression a[l] + a[j] + a[r].

If the sum is equal to 0, it means the current combination of numbers forms a valid triplet. And increase the left and right pointer by 1.

If the sum is less than 0, Which means larger number is needed to make it 0 so move “right” pointer towards higher values.

If the sum is greater than 0, which means smaller number is needed to make sum as 0 so move “left” pointer towards lesser values.

By repeatedly adjusting the pointers and checking different combinations, it explores all possible triples in the array. And resulting a list of tiples is returned to getTriples() method which will sort the list of triples and remove duplicates by using distinct() method.

Overall, the code efficiently finds triples that sum up to zero by leveraging the sorted nature of the array and employing a two-pointer approach.

This approach works because the array is sorted, the algorithm relies on this sorted array when it is looking for s triplet that adds to 0. From the given mid-point (j in this case) all the numbers to its left are smaller and the numbers to the right are larger. When attempting to balance the sum:

* When its greater than 0, the algorithm looks for a smaller (A negative number) number to make the resultant sum 0.
* When its small than 0, the algorithm looks for a larger (A positive number) number to make the resultant sum 0.

**Note: In attached spreadsheet of timing observation, we can observe that timing for quadratic is approximately half of the Quadrithimic which makes this algorithm more efficient.**

On the other hand, **quadraticWithCalipers** utilizes pointers in a distinct manner.

The method utilizes the pointers to traverse the array inwardly, considering different combinations of numbers. The loop continues if the “to” pointer is less than the “from” pointer (to < from).

Within each iteration of the loop, a new triple is created using the values at indices i, to, and from (Triple triple = new Triple(a[i], a[to], a[from])). The function parameter is then applied to this triple to calculate a sum.

If sum == 0, the triple is added to the triples list and both the “to” and “from” towards the centre of the array.

If sum < 0, it indicates that the current triple's sum is smaller than 0 so increase the “to” pointer to move towards centre.

If sum > 0, it means that the current triple's sum larger than 0 so decrease the “from” pointer to move towards centre.

By iteratively adjusting the pointers and evaluating different combinations of numbers, the method explores all possible triples.

The quadraticWithCalipers method effectively utilizes the inward movement of the pointers to find triples offering an alternative approach to solve the problem.

**Spreadsheet showing timing observations:**

****

**Screenshot of graph for quadratic timing observations:**

**A screen shot of a graph

Description automatically generated with low confidence**

**Screenshots of Unit Tests and Benchmarking:**

**A screenshot of a computer

Description automatically generated with medium confidenceA screenshot of a computer program

Description automatically generated with medium confidenceA screenshot of a computer program

Description automatically generated with medium confidenceA screenshot of a computer program

Description automatically generated with medium confidence**

A screenshot of a computer

Description automatically generated with medium confidenceA screenshot of a computer program

Description automatically generated with medium confidence