**Program Structures and Algorithms**

**Spring 2023(SEC –1)**

**Assignment-2**

**Name:** Naga Venkata Nishanth Sayana

**NU ID:** 002930970

**a) Test cases:**

**A picture containing scatter chart

Description automatically generated**

**b) Benchmark Observations**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **N** |  | **Quadratic** |  | **Quadrithimic** |  | **Cubic** |  |
|  |  | **(millisecond)** |  | **(millisecond)** |  | **(millisecond)** |  |
| **250** | **Raw Time** | 1.06 |  | .94 |  | 5.71 |  |
|  | **Normalized** | 16.96 |  | 1.89 |  | .37 |  |
| **500** | **Raw Time** | 2.38 |  | 3.48 |  | 42.82 |  |
|  | **Normalized** | 9.52 |  | 1.55 |  | .34 |  |
| **1000** | **Raw Time** | 4.40 |  | 15.50 |  | 333.20 |  |
|  | **Normalized** | 4.40 |  | 1.56 |  | .33 |  |
| **2000** | **Raw Time** | 19.00 |  | 77.20 |  | 2656.00 |  |
|  | **Normalized** | 4.75 |  | 1.76 |  | .33 |  |
| **4000** | **Raw Time** | 106.60 |  | 373.40 |  | 21438.60 |  |
|  | **Normalized** | 6.66 |  | 1.95 |  | .33 |  |
| **8000** | **Raw Time** | 601.67 |  | 1738.67 |  |  |  |
|  | **Normalized** | 9.40 |  | 2.10 |  |  |  |
| **16000** | **Raw Time** | 3103.50 |  | 7385.00 |  |  |  |
|  | **Normalized** | 12.12 |  | 2.07 |  |  |  |

**Chart, line chart

Description automatically generatedTime vs N graph plot**

As we can see from the chart we plotted based on the benchmark values, we can see a clear distinction between time taken by different solution. As expected, quadratic solution gives us the best time complexity.

**c) Why quadratic methods work?**

Our goal here is establish the difficulty level of the problem and develop optimal algorithms.

Instead of considering the triplet as 3 single entities and performing the solution in an exhaustive way (), we can consider that the triplet is made up of a pair and single element. In that case we have at least pairs to solve a 3 Sum problem, this way we are able to reduce the time complexity of the solution from to , hence quadratic is the best possible solution to solve the problem.

Based on the time consumed by the quadratic solution in the time spreadsheet illustrated above, we are able to prove that the quadratic solution is the most optimal solution.

**Quadratic solution:**

In this solution we use a combination of if loop and the two pointers, we use 3 variables (i, j, k) as the indexes for the 3 elements required for the 3 sum. The middle index(j) is fixed and passed as parameter to the getTriplets method, then we start the first index at zero(i=0) and the last index at the end(k=array.length-1), then run the while loop and log the triplets if the 3 sum is zero.

In this the outer loop for j runs n times and the while loop inside the getTriplets method run for n times leading to O(n^2) complexity.

**Quadratic Calipers solution:**

In this solution, the first index(i) would be fixed and we will run it in a loop of n, and then for each and every run, we run the while loop in the getTriplets method for values j=i+1 and k=array.length-1, as the name suggests, we will move the j and k in a motion similar to calipers which will be around n times, as soon as we found a 3 sum triplet we will add it to the result. The outer loop runs for n timed and the inner loop where j and k move run for n times, leading to O(n^2) complexity.