Write an efficient algorithm to solve a cubic equation of the form $x^3 - n = 0$, for some input value of n, by using a repetitive algorithm. Draw a graph of accuracy/error in the answer vs. total number of repetitions. Do the necessary experimentation and analysis with your algorithm.

Parag Parihar
Roll No:- IIT2016095
iit2016095@iiita.ac.in

Rakshit Sai
Roll No:- IIT2016126
iit2016126@iiita.ac.in

Adarsh Agrawal
Roll No:- IIT2016516
iit2016516@iiita.ac.in

Nilotpal Pramanik
Roll No:- IRM2016501
irm2016501@iiita.ac.in

I. Introduction and Literature Survey

The given task for us was to design and analyze a problem. The problem given was to generate an efficient algorithm to solve a cubic equation of the form $x^3 - n = 0$, for some input value n, by using a repetitive algorithm. Then we have to draw the graph for accuracy/error in the answer vs. total number of repetitions. The actual deal with the problem is that there wont be any direct equation given, we will have to scan the number n in the equation that is the constant in the equation and then calculate the cube root of the equation with the repetitive algorithm which must be as efficient as it could be. The difficult part here is the repetitive algorithm. Then we have to plot the graph between the error/accuracy of the root vs. the number of iterations so as to know how close is the answer for each iteration. Basically the first iteration gives us a high error answer than the last one because that is how we try to reduce the error in the answer so it is very obvious.

II. Algorithm Design

According to the problem we have to generate an efficient algorithm to solve a cubic equation of the form $x^3 - n = 0$, for some input value n, by using a repetitive algorithm.

As an input we are going to take an input "n".And we have initialize two variable left and right with -1000000 and 1000000 respectively to define the range of the root.

Then through while loop we will check weather the difference between the left and right is less than or equal to 0.0000000000001 much or not.If true then we will print just the mean of those values as the answer and break.

And then to divide the difference of range in 10 parts we are storing the value in d.And through for loop we will traverse till 10 to assign the value $(left + d * i)^3 - n$ in ans1 and $(right + d * (i + 1))^3 - n$ in ans2.Then further check weather the ans1 is 0.0 or not if true then we will print $left + d * (i)$ as the answer and stop through break.And similarly follow same logic for ans2 and print $left + d * (i+1)$ as the answer of the given problem.

And for $ans1 < 0$ and for $ans2 > 0$ we have to swap the left and right ranges follow the logic to solve and through break end the loop.

Algorithm 1

INPUT: ***n***
OUTPUT: Accurate value of cube root
Initialization : *left* $\leftarrow -1000000$ *right* $\leftarrow 1000000$
while TRUE **do**
  if $right - left \leq 0.0000000000001$ **then**
    print answer $(\frac{(right+left)}{2})$
    break
  end if
  $d \leftarrow (right - left)/10$
  for $i = 0$ *to* 10 **do**
    $ans1 \leftarrow (left + d * i)^3 - n$
    $ans2 \leftarrow (right + d * (i + 1))^3 - n$
    if $ans1 = 0.0$ **then**
      $flag \leftarrow 1$
      $ans \leftarrow left + d * (i)$
      print **ans**
      break
    end if
    if $ans2 = 0.0$ **then**
      $flag \leftarrow 1$
      $ans \leftarrow left + d * (i + 1)$
      print **ans**

```
        break
      end if
      if ans1 < 0 and ans2 > 0 then
        tmp ← left
        left ← left + d * i.
        right ← tmp + d * (i + 1)
        break
      end if
    end for
    if flag = 1 then
      break
    end if
  end while
```

III. Analysis and Discussion

A. Best Case

: When we will give an input of $x^3 + 10^{18} = 0$, the root will be -10^6 , and since on dividing by 10, the first segment we have will be from $[-10^6 to(-10^6 + 2 * 10^5)]$ which itself includes the root, hence it will break there only, so both while and for loop will run only 1 time. So in the best case the no of instructions = 42.

B. Worst Case

: The worst case will be when we will have $x^3 - n = 0$, where $n >> 0$, there will be a n for which the while loop and for loop will run the most until we reach the precision of 10-12 , in such a case the while loop will run 20 times and for loop 10 times to find value upto 12 decimal places. So in worst case the no of instructions = 5986.

C. Average Case

: Since our complexity not only depends on input, but on several other factors like amount of precisions, in how many parts we are dividing the algorithm, so cannot find a relation between given n and total no of instructions.Still, Total no of instructions in average case is between 42 and 5986.

D. Graph for error vs. no. of iterations

:

IV. Experimental Analysis

We revised the commands and basic functions of **GNUPlot** to plot the time complexity analysis graphs and other relevant analysis related to our algorithm.
While making this report we also learnt the basics of making reports using LATEX in IEEE format using **IEEEtran class**.

For making the graph between **error vs no. of iterations**, we have taken any random number. And for every iterations, we have tried to find value of **CUBIC ROOT**. After getting cubic root for every iteration we have made, we have generated corresponding **error**. From the formula

$$\%error = \left| \frac{exact\_value - aproximate\_value}{exact\_value} \right| * 100$$

exact_value $= given\ number(of\ which\ we\ have\ to\ find\ cubic\ root)$
approximate_value $= cube\ of\ output\ value$

Suppose the number of which we have to find cubic root is 100.Table for the error after every iteration is:-

iteration	error
2	5.000000
3	3.823000
4	0.544625
5	0.038053
6	0.005741
7	0.000075
8	0.000011
9	0.000004
10	0.000000
11	0.000000
12	0.000000
13	0.000000

As we can see here for 2nd and 3rd iteration there is relatively comparable error but after 3rd iteration error decreases exponentially, as we can see from graph.

V. Conclusion

Through different experimental studies we have tried to analyze this algorithm perfectly and properly using the Divide into ten repetitive algorithm. By finding the root of the equation upto a given decimal number we found out that the time complexity, i.e., total no of instructions in best case is equal to 42 and in the worst case it is equal to 5986 as depicted in the graphs.Through the experimental analysis, we concluded that average case is between the best case and worst case and it has no particular relation with the given input n. By the analysis we concluded that on increasing the number of repetitions of the the repetitive algorithm, the root of the cubic equation becomes more and more accurate as we decrease the size of segment in the method, and hence as depicted in the error vs repetitions graph, the error in the root becomes less.Therefore we get a exponentially decreasing graph and the accuracy graph is exponentially increasing.