Design and Analysis of Algorithm for finding roots of a quadratic equation using Secant Method

Nairit Banerjee
IIT2016505

Kamal Nayan Chaubey
IIT2016510

Ashutosh Vishwakarma
IIT2016017

Mandeep Chakma
IIT2016012

February 20, 2018

Abstract

This document is a final report of our study on finding the roots of a quadratic equation using Secant Method.

Keywords: *Interval,function,error,Accuracy, Number of iterations*

1 Introduction And Literature Survey

In mathematics and computer science, the process of finding the roots of a equation has number of methods. One such method is Secant Method. A method that takes interval,error as an input and outputs the approximate roots of that equation.
Our equation is of the form f(x) = x^2−n, where n will be taken as input.For example we have an equation f(x) = x^2 - 4,by simple method we know that root of the above equation is 2,-2 as f(2) = f(-2) = 0.
But secant method is as follows :-
Assume an interval **[a,b]** where we know that one of the root of equation lies in the given interval i.e $c \in [a, b]$ such that **f(c) = 0** and f(a)*f(b) < 0. c is calculated as follows:

$$c = \frac{(a * f(b) - b * f(a))}{(f(b) - f(a))} ....(1)$$

The above formula is calculated by using the basic definition of slope of a curve at a point.

$$f'(c) = \frac{(f(b) - f(a))}{b - a}$$

, where

$$f'(c) = \frac{f(b)}{b - c}$$

, on solving for the value of c we get the eq(1).

2 Algorithm Design

The first task in Algorithm design to be done is to produce the interval where our root will lie. The algorithm description for the problem is as follows:

(1) procedure NextPermutation

1. Find largest index i such that arr[i-1]<arr[i]. (If no such i exists, then this is the last permutation).
2. Find largest index j such that $j \geq i$ and arr[j]>arr[i-1]
3. Swap arr[j] and arr[i-1]
4.Reverse the suffix starting at arr[i]

(2) procedure PossibleWord

1. We go through the loop from 1 to $2^7 - 1$ and then for each number that comes in between, we have their binary representation consisting of only 0's and 1's.

2. Now for each number we go through their binary representation and check the position of bitset(that is the position where the binary digit is equal to 1) in the binary representation of the respective number.

3. Now the positions at which bit is set represents that, at present we are using those position characters from the randomly generated character string, for forming our different words using nextPermutation procedure.

4. As the words are generated, they are checked whether they are previously generated or not using the function check.If previously generated then we avoid it,Else we search it using search function.

2.1 Pseudo Code

Algorithm 1 Function Value

```
1: procedure F(x)
2:     a ← x^2 − n
3:     return a
4: end procedure
5: procedure MAIN
6:     if n == 0 then
7:         Roots are both = 0
8:         return 0
9:     end if
10:    if n < 0 then
11:        Roots are imaginory
12:        return 0
13:    end if
14:    while F(i) ≤ 0 do
15:        i ← i + 1.0
16:    end while
17:    b ← i − 1
18:    c ← i
19:    e ← 10^−6
20:    while abs(c − b) ≥ e do
21:        a ← b
22:        b ← c
23:        c ← b−((b−a)/(F(b)−F(a)))∗F(b)
24:        if F(c) == 0 then
25:            The root of equation is c
26:            return 0
27:        end if
28:    end while
29:    The root of equation is c
30:    return 0
31: end procedure
```

3 Analysis and Discussions

3.1 Time Complexity

The time complexity for the above stated algorithm can be computed by adding units of time taken by each statement multiplied by the number of times the statement is executed. Hence,

$T = \Sigma(cost(i) * no.of.times(i))$,for all statements

The running time will be dependent on the coefficient n in the equation

$$x^2 - n = 0$$

If n is zero or negative, it results in best case and returns output in constant time. Whereas if n is any positive number, the algorithm will take time to estimate the range between which a root may lie. In other words time taken will be dependent on how far the control has to travel along the x-axis to find a change in nature(sign) of the function.

$$T_{best} \propto c_1$$
$$T_{average} \propto \sqrt{n} + 10c_2$$
$$T_{worst} \propto \sqrt{n} + 10c_3$$

Hence the order of growth for best case, average case and worst case are

$$\Omega(1), \Theta(\sqrt{n}), O(\sqrt{n}) \text{ respectively.}$$

The average and worst case differ in constant units of time. This is because if n is a perfect square, one of the ends of the estimated range will be a root. Whereas if n is not a perfect square , we now need to actually iterate through the secant method to reach to the root.

For the dataset,

Table 1: Data for plotting running time graph

n	T_{best}	T_{avg}	T_{worst}
0	4	-	-
-100	5	-	-
64	-	83	-
625	-	163	-
69	-	-	385
1001	-	-	249

We will get the following time complexity graph for the above stated and a few more values of n

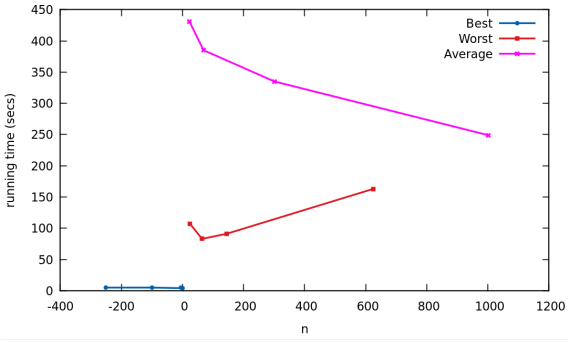

Figure 1: Running time graph.

Now we will plot a graph between the number of iterations for which we run the secant function and the error percentage in finding the root of the equation

$$x^2 - 67 = 0$$

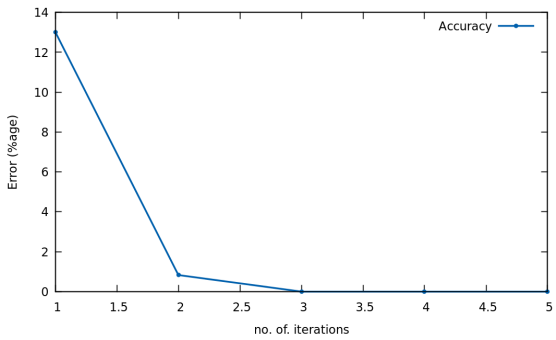

Figure 2: Percentage Error Graph

3.2 Space Complexity

The algorithm does not consume any extra space in the memory other than the variables used. Hence space complexity is linear.

$$Space \propto 1$$

4 Experimental Study

We may assume the following examples to study the various running times possible,

Best Case: The best case will depend on the value of n :
(i)when **n=0**, in this both roots will be **0**.
(ii)when **n<0**, in this case **imaginary roots**.

Average Case: When some of the letters are repeated then we need to generate all the possible combination and permutation of those letters to generate different words and hence find out the validity of word in dictionary.For example a, a, a, b, l, l, o.

Worst Case:In ts case no any letter is repeated and hence it takes the highest computation time to compute all the combinations and permutations of the letters and then search the word in dictionary to check whether the word is present in dictionary or not.For example suppose we take only 3 letters as a,b,c then all the posssible combination and permutation will be as follows:
ab,ba,ac,ca,bc,cb,abc,acb,bac,bca,cab,cba
hi Thus we can claim that the running time depends on the variation in the types of alphabets randomly generated. More distinct the aplhabets generated more will be the computation time to form every possible permutation.

5 Conclusion

In this paper we proposed an algorithm to find the roots of a quaratic equation of the form : x^2 - n using **Secant method**.

Hence practical implications are: Communication networks, cryptography and network security. Permutations are frequently used in communication networks and parallel and distributed systems.

References

[1] Introduction To Algorithms (Second Edition) by Thomas H.Cormen, Charles E.Leiserson , R.L.Rivest and Clifford Stein.

[2] How to Solve it by Computer by R.G.Dromey.

[3] The C Programming Language by Brian Kernighan and Dennis Ritchie.

[4] https://www.geeksforgeeks.org/

[5] https://www.stackoverflow.com/

[6] http://mathworld.wolfram.com/SecantMethod.html