Concordia University

Problem 3

SOEN 6011 - Software Engineering Process

ETERNITY: FUNCTION (σ)

Pragya Tomar

Student ID : 40197757

Repository Address: https://github.com/pragya231/SOEN6011

Contents

1 Algorithms

1.1 Description

For calculating the standard deviation of a group of numbers, there are two algorithms. For Algorithm 1, the standard deviation is usually calculated in two passes. In first pass, we will find the mean and then in second step, we will calculate the standard deviation of a group of numbers from the calculated mean. But we can do the same thing in one pass. So, Algorithm 2 do the same thing in one pass. It just rewrite the formula in a different way to calculate the mean and standard deviation in a single pass. These algorithms are using various functions such as power function and square root function.

1.2 Advantages

- Multipass Algorithm
  - It is most efficient algorithm to use.
  - This algorithm is fast and can take collection of input data in the form of a file.
- Singlepass Algorithm:-
  - It is easy to understand.
  - The algorithm is taking up less time as there is only one iteration.

1.3 Disadvantages

- Multipass Algorithm
  - It will be difficult to develop in any other language.
  - Memory stack will get full as it is using recursion and not iteration.
- Singlepass Algorithm:-
  - It gives inaccurate result when the array contains large numbers.
  - If there are more input data then this algorithm takes more time.

1.4 Mind Map for Psuedo-Code format

By checking through many refernces and implementing various algorithms, I used the multi-pass algorithm here in order to calculate standard deviation. This psuedo-code is sapce-efficient as well as time-efficient and have good memory requirements. There are no errors and by using proper debugger, all the errors and exceptions are handled properly.

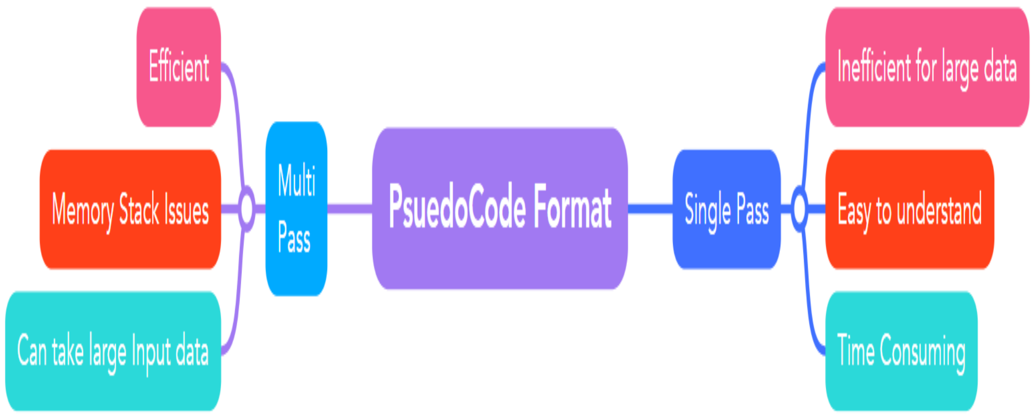

Figure 1: Mind Map for psuedo code format

1.5 Psuedo-Code

There are two algorithms for which the psuedo code is provided below:-

Algorithm 1 Multi Pass Algorithm for calculating Standard Deviation

```
function StandardDeviation(numArray[])
    $Sum \leftarrow 0.0$
    $Mean \leftarrow 0.0$
    $SD1 \leftarrow 0.0$
    $iLength \leftarrow numArray.count()$
    for $i = 0$ to $iLength$ do
        Sum = Sum + numArray[i]
    $Mean = Sum/iLength$
    for $i = 0$ to $iLength$ do
        diff = numArray[i]-Mean
        SD1 =SD1 + (diff*diff)
    return SquareRoot(SD1 /iLength)

function SquareRoot(input)
    $error \leftarrow 0.00001$
    $errorPrecision \leftarrow 1$
    $dup \leftarrow input$
    $iLength \leftarrow numArray.count()$
    while $errorPrecision > error$ do
        $input = (input + dup/input)/2$
        $errorPrecision = input - dup/input$
    return input
```

Algorithm 2 Single Pass Algorithm for calculating Standard Deviation

```
function StandardDeviation(numArray[])
    $Sum \leftarrow 0.0$
    $Mean \leftarrow 0.0$
    $SD1 \leftarrow 0.0$
    $iLength \leftarrow numArray.count()$
    for $i = 0$ to $iLength$ do
        Sum = Sum + numArray[i]
        SqSum = SqSum + (numArray[i]*numArray[i])
    $Mean = Sum/iLength$
    $Variance = SqSum/n - Mean * Mean$
    return SquareRoot(SD1 /iLength)

function SquareRoot(input)
    $error \leftarrow 0.00001$
    $errorPrecision \leftarrow 1$
    $dup \leftarrow input$
    $iLength \leftarrow numArray.count()$
    while $errorPrecision > error$ do
        $input = (input + dup/input)/2$
        $errorPrecision = input - dup/input$
    return input
```

Bibliography

[1] Peter Kankowski
`https://www.strchr.com/standard_deviation_in_one_pass?allcomments=1`

[2] Standard Deviations and Standard Errors,
`https://www.ncbi.nlm.nih.gov/pmc/articles/PMC1255808/`