SOEN 6011 : SOFTWARE ENGINEERING PROCESSES

SUMMER 2022

ETERNITY

PROBLEM - 1

Function Description

Author

Neona Sheetal Pinto

https://www.overleaf.com/project/62e9f079b389ca1bdb5b238c

Contents

List of Algorithms

1 PROBLEM 3 - F6: $B(x, y)$

1.1 Algorithm Description and Pseudo-Code

SOEN 6011 - Summer 2022

1.2 Mind-map for pseudo-code format decision

The figure 1 shows the mind-map showcasing features supported by various algorithm/pseudo-code packages in Latex. The mind-map shows how most of the features are supported by almost all the packages. The *Customization* feature highlighted in red block is a powerful feature which provides flexibility in adding new customised features which is supported by **algpseudocode** + **algorithmicx**, Hence the reason in choosing the *Customization* format for representing the algorithms in this document.[7]

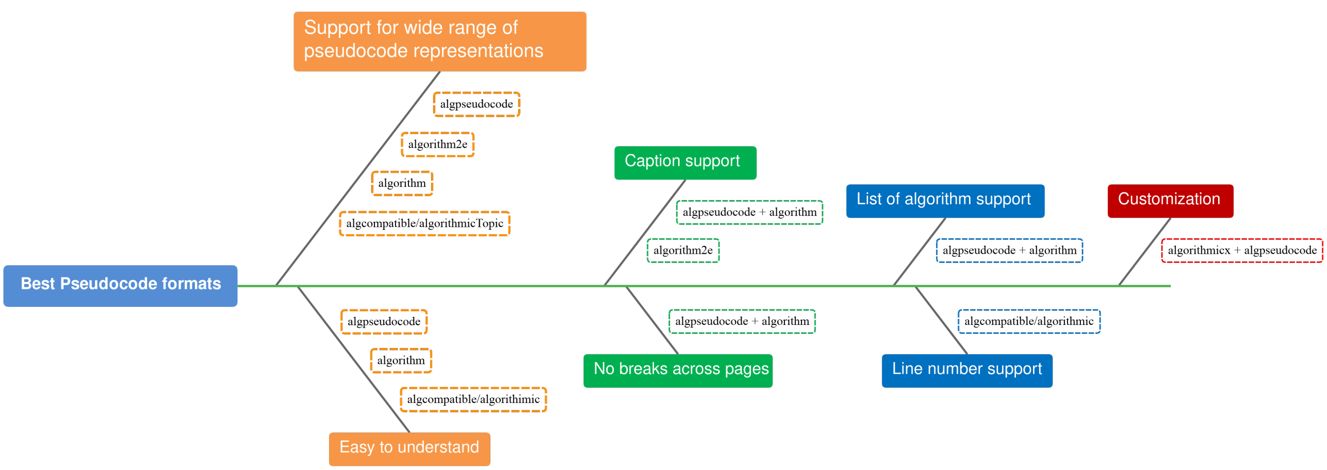

Figure 1: Mind map for pseudocode format decision

1.3 Algorithms for Beta function:

The following are the ways to find Beta value of the given variables (x,y):

- **Algorithm 1: Trapezoidal Rule** The trapezoidal rule chooses points in [a,b] starting with a and ending with b. Trapezoidal Rule is a rule that evaluates the area under the curves by dividing the total area into smaller trapezoids rather than using rectangles. This integration works by approximating the region under the graph of a function as a trapezoid, and it calculates the area. This rule takes the average of the left and the right sum [5]

- Let f(x) be a continuous function on the interval [a, b]. Now divide the intervals [a, b] into n equal sub intervals with each of width,

  $\Delta\ x = (b - a)/n, ta = x0 < x1 < x2 < x3 < \ldots.. < xn = b$

  Then the Trapezoidal Rule formula for area approximating the definite integral $\int_a^b f(x)\, dx$

  $\int_a^b f(x)dx \approx T\_n = \frac{\triangle x}{2}[f(x_0) + 2f(x_1) + 2f(x_2) + \ldots .2f(x_{n-1}) + f(x_n)]$
  Where, $xi = a + i\Delta x$

  If n $\rightarrow\infty$, R.H.S of the expression approaches the definite integral

  $$\int_a^b f(x)\, dx$$

- The name trapezoidal is because when the area under the curve is evaluated, then the total area is divided into small trapezoids instead of rectangles. Then we find the area of these small trapezoids in a definite interval.

- **Algorithm 2: Simpson's Rule** In numerical analysis, Simpson's rule is a method for numerical approximation of definite integrals. Specifically, it is the following approximation:

  $$\int_a^b f(x)dx \approx \frac{(b-a)}{6}\left(f(a) + 4f\frac{(a+b)}{2} + f(b)\right)$$

  In Simpson's 1/3 Rule, we use parabolas to approximate each part of the curve.We divide the area into n equal segments of width $\mathbf{\Delta x}$. Simpson's rule can be derived by approximating the integrand f (x) (in blue) by the quadratic interpolant P(x).

- In order to integrate any function f(x) in the interval (a, b), follow the steps given below: 1.Select a value for n, which is the number of parts the interval is divided into. 2.Calculate the width, h = (b-a)/n 3.Calculate the values of x0 to xn as x0 = a, x1 = x0 + h, ...xn-1 = xn-2 + h, xn = b. Consider y = f(x). Now find the values of y(y0 to yn) for the corresponding x(x0 to xn) values. 4.Substitute all the above found values in the Simpson's Rule Formula to calculate the integral value. Approximate value of the integral can be given by Simpson's Rule:

  $$\int_a^b f(x)dx \approx \frac{h}{3}\left(f_0 + f_n + 4 * \sum_{i=1,3,5}^{n-1} f_i + 2 * \sum_{i=2,4,6}^{n-2} f_i\right)$$

  [4]

1.4 Technical Reasons for selecting Trapezoidal Integration for calculating the definite integral of the Beta function:

Advantages:

- It requires less steps than the rectangular method to get the same accuracy so it is somewhat faster on a computer.
- The trapezoidal method is a little more complicated but is still relatively easy to understand.
- The trapezoidal rule is mostly used in the numerical analysis process as they provide more accurate results.
- Trapezoid rule is its rather easy conceptualization and derivation. [6]

Disadvantages:

- The Trapezoidal Rule does not give accurate value as Simpson's Rule when the underlying function is smooth
- It follows that if the integrand is concave up (and thus has a positive second derivative), then the error is negative and the trapezoidal rule overestimates the true value.

Therefore, the reasons above show the use of trapezoidal rule for calculating the definite integral in the Beta function.

1.5 Pseudo Code for Trapezoidal rule for the definite integral to calculate Beta function

1.5.1 Description

To find the area from a to b , we can divide the area into n trapezoids, and the width of each trapezoid is w, so we can say that (b - a) = nw. When the number of trapezoids increases, the result of area calculation will be more accurate. The function passed inside the integral is what produces the Beta function output.[3]

1.5.2 Algorithm : Trapezoidal rule of Integral to calculate Beta function - algorithm1

Algorithm 1 Trapezoidal rule for definite integral

Require: $x > 0$ AND $y > 0, x, y \in R$

function INTEGRAL(a, b, x, y) ▷ algorithm for $\int_0^1 t^{x-1}(1-t)^{y-1}\,dt$
    $area \leftarrow 0$
    $modifier \leftarrow 1$
    $a \leftarrow 0$
    $b \leftarrow 1$
    for $i \leftarrow a + increment, i < b, i \leftarrow i + increment$ **do** ▷ increment is equal to 1E-4
        $dist \leftarrow i - a$
        $f1 \leftarrow$ EXPONENTIAL$(a + dist - increment, x, y)$
        $f2 \leftarrow$ EXPONENTIAL$(a + dist - increment, x, y)$
        $area = area + increment/2 * f1 + f2$
    end for
return $area * modifier$
end function

1.6 Pseudo Code for Simpson's rule for the definite integral

1.6.1 Description

Simpson's Rule is based on the fact that given three points, we can find the equation of a quadratic through those points. The function passed inside the integral is what produces the Beta function output.

1.6.2 Algorithm : Simpson's - algorithm2

Algorithm 2 Simpson's rule for definite integral

Require: $x > 0$ AND $y > 0, x, y \in R$

function INTEGRAL(a, b, x, y) ▷ algorithm for $\int_0^1 t^{x-1}(1-t)^{y-1}\,dt$
    $number \leftarrow 10000$ ▷ precision parameter
    $w \leftarrow (b-a)/(number-1)$ ▷ Step size
    $sum \leftarrow 1.0/3.0 * \text{EXPONENTIAL}(x, y) + \text{EXPONENTIAL}(x, y)$
    for $i \leftarrow 1; i < number - 1; i \leftarrow i + 2)$ **do**
        $x \leftarrow a + w * i$
        $sum \leftarrow sum + 4.0/3.0 * \text{EXPONENTIAL}(x, y)$
    end for
    for $i \leftarrow 2; i < number - 1; i \leftarrow i + 2)$ **do**
        $x \leftarrow a + w * i$
        $sum \leftarrow sum + 2.0/3.0 * \text{EXPONENTIAL}(x, y)$
    end forreturn $sum * w$;
end function

1.7 Pseudo Code for Taylor's Series for calculating Exponential value

1.7.1 Description

Taylor series is a representation of a function as an infinite sum of terms that are calculated from the values of the function's derivatives at a single point. This is a sub function used in Beta function for calculating the definite integral. There are 2 utility functions used by the main function exponent to calculate the x^y

1.7.2 Algorithm : Exponent - algorithm3

1.8 Annexure:

- **Trello Board :** *https://trello.com/eternity119*
- **Code Version Control :** *https://github.com/neonapinto/Scientific_calculator*
- **Overleaf :** *https://www.overleaf.com/project/62e9f079b389ca1bdb5b238c*

Algorithm 3 Exponentiation by Taylor Series

```
Require: x ≠ 0 AND y > 0
function Exponent(base, power)    ▷ algorithm for x^y considering all conditions and using sub functions
    result ← 1
    roots ← 1
    base_value ← x
    power ← y                     ▷ Power is integer simply multiply and return result
    if power_value%1 ≡ 0 then
        for counter <= power_value do
            result ← result * x
        end for                   ▷ Power is a decimal number
    else
        if power_value >= 1 then
            exponent_value[] ← Exponential(base_value, power_value)
            roots = roots * exponential_value[0]
            power_value ← exponential_value[1]
        end if
    end if
    if power_value > 0 AND power_value < 1 then
        precision ← 1
        findroot ← findClosestRoot(base_value, den, 0, precision)
        while base_value < Exponential(findroot, den) AND precision > 0.000001 do
            findroot ← findroot − precision
            precision ← precision * 0.1
            findroot ← findClosestRoot(base_value, den, findroot, precision)
        end while
        value ← Exponential(findroot, power_value * den)[0]
        roots ← roots * value
        result ← root
    end if
return result
end function
function findClosestRoot(base, power, root, precision)    ▷ algorithm for finding the root with closest precision
    closestRoot ← closestRoot + precision;
    temp ← Exponential(closestRoot, power)
    while temp[0] < base do
        closestRoot ← closestRoot + precision;
        temp ← Exponential(closestRoot, power)
    end while
return closestRoot
end function
function exponential(x, y)    ▷ algorithm for x^y
    exponent_value ← 1
    while power > 0 do
        exponent_value ← exponent_value * base
        power ← power − 1
        if power < 1 then
            break;
        end if
    end while
return exponent, power
```

References

[1] Concordia Logo : The logo of Concordia University with transparent background
https://www.pngwing.com/en/free-png-djpzn

[2] Latex Guidelines : How to write robust and effective latex documents
https://medium.com/@pbeliveau/latex-coding-standards-f82743b7866b

[3] Trapezoidal Rule for definite integral
https://www.tutorialspoint.com/Trapezoidal-Rule-for-definite-integral

[4] Simpson Rule for definite integral
https://www.javatpoint.com/simpson-method

[5] Trapezoidal Rule for definite integral
https://byjus.com/maths/trapezoidal-rule/

[6] 8 Difference Between Trapezoidal Rule And Simpson's Rule In Surveying
https://vivadifferences.com/difference-between-trapezoidal-rule-and-simpsons-rule-in-surveyin

[7] Pseudo-code in Latex: Different packages to represent algorithm or pseudo-code in latex.
https://www.overleaf.com/learn/latex/Algorithms