SOEN 6011 : SOFTWARE ENGINEERING PROCESSES

SUMMER 2021

SUPER CALCULATOR

PROBLEM - 3

Pseudo-code and Algorithms

Authors

Rokeya Begum Keya

Kyle Taylor Lange

Sijie Min

Manimaran Palani

https://www.overleaf.com/project/610304de4e6b8d24f7c781b6

Contents

Decision on Pseudo-Code Format

Our Team conducted a brainstorming session and referred several resources [2] [3] to decide on Pseudo code algorithm/pattern. As a conclusion, Everyone had agreed to make a pseudo code of their respective algoriths in the Algorithmicx (algpseudocode) [1] format available in Overleaf Latex.

Example of Algorithmicx (algpseudocode) Pseudo code Pattern

Algorithm 1 Algorithmicx (algpseudocode) Pseudo code Pattern

Require: $n \geq 0$
Ensure: $y = x^n$
  $y \leftarrow 1$
  $X \leftarrow x$
  $N \leftarrow n$
  while $N \neq 0$ **do**
    if N is even **then**
      $X \leftarrow X \times X$
      $N \leftarrow \frac{N}{2}$ ▷ This is a comment
    else if N is odd **then**
      $y \leftarrow y \times X$
      $N \leftarrow N - 1$
    end if
  end while

Algorithm Description and Pseudo-Code

PROBLEM 3 - F2: $tan(x)$

SOEN 6011 - Summer 2021 **Rokeya Begum Keya**
Software Engineering Processes **40183615**
Repository address : https://github.com/Dakatsu/SOEN6011Calculator

Technical Reasons for selecting Maclaurin Series:

- There are many reasons for selecting Maclaurin Series for calculating the value of $tan(x)$ function. Below are some advantages for which I selected Maclaurin Series:

Advantages:

- Maclaurin series provides more approximate values for the tangent function.
- The formula to calculate the value of $sin(x)$ and $cos(x)$ function to get the value of tangent function is easy to understand.

$$tan(x) = \frac{sin(x)}{cos(x)}$$

Disadvantages:

- There is an another form of Maclaurin series to calculate the tangent function. For example: derivation of $tan(x)$ function. In this formula there are no use of $sin(x)$ and $cos(x)$ function. However, using this formula we can not get the approximate value of $tan(x)$ function.

Therefore, I select the Maclaurin series of $sin(x)$ and $cos(x)$ to calculate the tangent function.

Algorithm 1 - Maclaurin Series:

- In this project to calculate $tan(x)$ function, I select the Maclaurin Series. Maclaurin series is just a special case of taylor series where region near $x = 0$.
- The $tan(x)$ function's approximation is derived by the Maclaurin Series's explicit forms of $sin(x)$ and $cos(x)$.

$$sin(x) = x - x^3/3! + x^5/5! - x^7/7! + ... \tag{1}$$

$$cos(x) = 1 - x^2/2! + x^4/4! - x^6/6! + ... \tag{2}$$

- As, $tan(x)$ is an odd function, odd derivatives when x=0 of Maclaurin series are used to calculate $tan(x)$ function.
- The output are in integer and provide an approximate value for tangent function.

Pseudo Code for Maclaurin Series

Algorithm 2 Maclaurin Series

Require: $retVal = 1$ AND $tmpResult = 1$ AND $i = 1$
  function EXPLICIT FORM(cos(x))
    for $i \leftarrow i + 2$ **do**
      $value = (-1) * x * x/(i * (i + 1))$ $\triangleright$ $Seriesforcos(x)$
      $tmpResult = tmpResult * value$
      if check($value$)$<=$ EPS **then**
      end if
      $retVal = retVal + tmpResult$
    end for
  return $retVal$ $\triangleright$ $getvalueofcos(x)$
  end function
Require: $retVal = x$ AND $tmpResult = x$ AND $i = 0$
  function EXPLICIT FORM(sin(x))
    for $i \leftarrow i + 1$ **do**
      $value = ((-1) * x * x/((2 * i + 2) * (2 * i + 3)))$ $\triangleright$ $Seriesforsin(x)$
      $tmpResult = tmpResult * value$
      if check($value$)$<=$ EPS **then**
      end if
      $retVal = retVal + tmpResult$
    end for
  return $retVal$ $\triangleright$ $getvalueofsin(x)$
  end function
Require: $x = Rad(x)$ AND $SinVal = retVal$ AND $CosVal = reVal$
  function CALCULATE(tan(x))
    if $SinVal < EPSvalMini$ **then**
  return 0
    end if
    if $CosVal < EPSvalMini$ **then**
  return $undefined$
    end if
  return $SinVal/CosVal$ $\triangleright$ $calculationfortan(x)$
  end function
  $resulttan(x)$

PROBLEM 3 - F3: Hyperbolic Sine, $sinh(x)$

SOEN 6011 - Summer 2021
Software Engineering Processes
https://www.overleaf.com/project/610304de4e6b8d24f7c781b6
https://github.com/Dakatsu/SOEN6011Calculator

Kyle Taylor Lange
27627696
Repository address :

PROBLEM 3 - F5

SOEN 6011 - Summer 2021 **Sijie Min**
Software Engineering Processes **401*****
Repository address : https://github.com/Dakatsu/SOEN6011Calculator

Team please add your content here
pseudo code
Algorithm f5(a,b,x)
Double type a and b,integer type x
if x=0 then
return a
c=x
if c¡0 then
c=-c
r=b
y=1
while c¿0 do
if c
c=c/2
r=r*r
else
c=c-1
y=y*r
if x¡0 then
return a/y
else
return a*y

PROBLEM 3 - F7 : x^y

SOEN 6011 - Summer 2021 **Manimaran Palani**
Software Engineering Processes **40167543**
Repository address : https://github.com/Dakatsu/SOEN6011Calculator

Algorithm : Montgomery's Ladder Technique[4]

- Montgomerym's ladder technique addresses defence against side-channel attacks for exponentiation computation.

  The algorithm prevents the recovery of the exponent involved in the computation which could possibly benefit an attacker

- The algorithm performs a fixed sequence of operations (up to log n): a multiplication and squaring takes place for each bit in the exponent, regardless of the bit's specific value.

Advantages	**Disadvantages**
It addresses the concern of MIM(Middle Man attack) observing the sequence of squaring and multiplications can (partially) recover the exponent involved in the computation.	Cache timing attacks are not yet protected and memory access latency might still be observable to an attacker

Algorithm : Taylor series

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.

$$x^y = e^{y \ln x} \tag{3}$$

3 evaluation of x^y. Here, e is a mathematical constant approximately equal to 2.71828

$$e^x = 1 + x/1! + x^2/2! + x^3/3! + ... \tag{4}$$

4 express e^x using Taylor Series

$$e^x = 1 + (x/1)(1 + (x/2)(1 + (x/3)(........))) \tag{5}$$

5 The series 4 can be re-written as above

$$log(1 + x) = x - x^2/2 + x^3/3 - ... \tag{6}$$

6 express ln x using Taylor Series

Advantages	Disadvantages
Very useful for derivations	Successive terms get very complex and hard to derive
Can be used to get theoretical error bounds	Truncation error tends to grow rapidly away from expansion point
Power series can be inverted to yield the inverse function	Almost always not as efficient as curve fitting or direct approximation

Algorithm 3 Montgomery's ladder Exponential Function

Require: $x_1 = x$; $x_2 = x^2$

for $i = k - 2$ to 0 *do* **do**
    if $n_{\mathrm{i}} = 0$ **then**
        $x_2 = x_1 * x_2$ ▷ $x_1 = {x_1}^2$
    else:
        $x_1 = x_1 * x_2$ ▷ $x2 = {x_2}^2$
        return x_1
    end if
end for

Algorithm 4 Exponentiation by Taylor Series

Require: $x \neq 0$ AND $y > 0$

function LOGARITHM(n) ▷ $algorithmforlog(n)$
    $sum \leftarrow 0$
    while $n > 1$ **do**
        $n \leftarrow n/e$ ▷ e is a constant approximately equal to 2.71828
        $y \leftarrow y + 1$
    end while
return y
end function
function EXPONENTIAL(x) ▷ $algorithmfore^x$
    $sum \leftarrow 1$
    $n \leftarrow 10$
    for $i \leftarrow n - 1, 1$ **do**
        $sum \leftarrow 1 + x * sum/i$
    end for
return sum
end function
$logx$ ←LOGARITHM(x)
$result$ ←EXPONENTIAL(y*logx)

Bibliography

[1] Algorithmicx (algpseudocode)
`https://www.overleaf.com/latex/examples/pseudocode-example/pbssqzhvktkj`

[2] Pseudo-Code Standard
`http://users.csc.calpoly.edu/j̃dalbey/SWE/pdl_std.html`

[3] Szasz Janos, The algorithmicx package
`http://tug.ctan.org/macros/latex/contrib/algorithmicx/algorithmicx.pdf`

[4] Montgomery, Peter L. (1987). ”Speeding the Pollard and Elliptic Curve Methods of Factorization” (PDF)
`https://www.ams.org/journals/mcom/1987-48-177/S0025-5718-1987-0866113-7/S0025-5718-1987-0866113-7.pdf`