Q1) The following set of statement sorts a data frame in R:

D <- data.frame(x=c(1,2,3,1), y=c(7,19,2,2))

indexes <- order(D$x)

D[indexes,]

D[rev(order(D$y)),]

Q2) The following function computes the log factorial of a given number using iteration.

ComputeFactorialbyIteration <- function(n)

{

result = 0

for (i in seq(1,n, by = 1))

result = result + log(i)

return (result)

}

Q3)The following function computes the log factorial of a given number using recursion.

ComputeFactorialbyRecursion <- function(n)

{

result = 0

if(n == 1)

return (log(n))

else

return(log(n) + ComputeFactorialbyRecursion(n-1))

}

Q4)The following function computes the sum of log factorials using different techniques such as recursion, iteration or lfactorial() depending upon the value passed in the input parameter iFunctionType.

ComputeSumOfLogFactorial <- function(n, iFunctionType)

{

result = 0

for(i in seq(1,n, by = 1))

{

if(iFunctionType == 1)

result = result + ComputeFactorialbyRecursion(i)

else if(iFunctionType == 2)

result = result + ComputeFactorialbyIteration(i)

else if(iFunctionType == 3)

result = result + lfactorial(i)

}

return (result)

}

Q5) The following function plots the execution time for all three sum of factorial functions (recursion, iteration and lFactorial()) versus the value of N by increasing it from 1 to 500000.

PlotLatency<-function(n, bComplex)

{

iter\_vect <- vector(mode="numeric", length=n)

rcsn\_vect <- vector(mode="numeric", length=n)

lfac\_vect <- vector(mode="numeric", length=n)

for(i in seq(1,n, by = 1))

{

if(bComplex)

{

rcsn\_vect[i] = (system.time(ComputeSumOfLogFactorial(i, iFunctionType = 1))[3])

iter\_vect[i] = (system.time(ComputeSumOfLogFactorial(i, iFunctionType = 2))[3])

lfac\_vect[i] = (system.time(ComputeSumOfLogFactorial(i, iFunctionType = 3))[3])

}

else

{

rcsn\_vect[i] = (system.time(ComputeFactorialbyRecursion(i))[3])

iter\_vect[i] = (system.time(ComputeFactorialbyIteration(i))[3])

lfac\_vect[i] = (system.time(lfactorial(i))[3])

}

}

df <- data.frame(N = 1:n,

recursion = rcsn\_vect,

iteration = iter\_vect,

lfactorial = lfac\_vect)

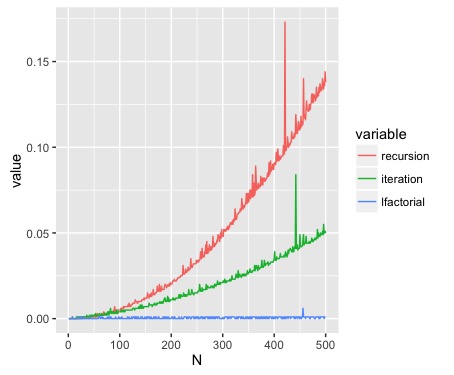
df <- melt(df , id.vars = 'N', variable.name = 'series')

g\_1<-ggplot(df, aes(N,value)) + geom\_line(aes(colour=variable))

print(g\_1)

}

PlotLatency(5000, 1)



As can be seen above, recursion and iteration are extremely slow compared to lFactorial() function. The time it takes to perform recursion and iteration increases quadratically as N increases.

Initially,for n >=5000, R was not able to complete its job and gave the following error:

> PlotLatency(5000, 1)

Error: evaluation nested too deeply: infinite recursion / options(expressions=)?

Error during wrapup: evaluation nested too deeply: infinite recursion / options(expressions=)?

I later increased options(expressions=5000000) and re-ran my code. It took many hours for the simulation to complete but when it did finish, it was apparent that lFactorial() behaved much faster than iteration and recursion. Recursion was the worst.