DataVisualAnalytics_HW1

Weifeng She

August 25, 2016

2. Implement a function that computes the log of the factorial value of an integer using a for loop. Note that implementing it using log(A)+log(B)+ $\cdots$ avoids overflow while implementing it as log(A $\cdot$ B $\cdot$ $\cdots$) creates an overflow early on.

```
library(ggplot2)
library(reshape)
logfact1 <- function(n){
   res = 0
   for(i in 1:n){
           res <- res + log(i)
   }
   return(res)
}
```

3. Implement a function that computes the log of the factorial value of an integer using recursion.

```
logfact2 <- function(n){
   if(n == 1) {
           return(log(1))
   }else{
           return(logfact2(n-1) + log(n))
   }
}
```

4. Using your two implementations of log-factorial in (2) and (3) above, compute the sum of the log-factorials of the integers 1, 2,. . . ,N for various N values.

```
# Define the function to calculate the sum using log-factorial
sumLogFact <- function(N, func){
        res = 0
        for(i in 1:N){
                res = res + func(i)
        }
        res
}
NList1 <- seq(from = 0, to = 5000, by = 200)
NList1[[1]] <- 1
# calculate the sum using log-factorial in (2)
logfact1Value = c()
for(N in NList1){
        result <- sumLogFact(N, logfact1)
        logfact1Value <- c(logfact1Value, result)
        print(result)
}
```

```
## [1] 0
## [1] 77012.76
## [1] 361684.6
## [1] 885240.3
## [1] 1664362
## [1] 2710708
## [1] 4033270
## [1] 5639375
## [1] 7535210
## [1] 9726128
## [1] 12216849
## [1] 15011592
## [1] 18114174
## [1] 21528079
## [1] 25256511
## [1] 29302434
## [1] 33668611
## [1] 38357623
## [1] 43371897
## [1] 48713719
## [1] 54385253
## [1] 60388551
## [1] 66725564
## [1] 73398153
## [1] 80408098
## [1] 87757100
```

```
# calculate the sum using log-factorial in (3)
options(expressions=500000)
NList2 = seq(from = 0, to = 2200, by = 200)
NList2[[1]] <- 1
logfact2Value = c()
for(N in NList2){
        result <- sumLogFact(N, logfact2)
        logfact2Value <- c(logfact2Value, result)
        print(result)
}
```

```
## [1] 0
## [1] 77012.76
## [1] 361684.6
## [1] 885240.3
## [1] 1664362
## [1] 2710708
## [1] 4033270
## [1] 5639375
## [1] 7535210
## [1] 9726128
## [1] 12216849
## [1] 15011592
```

5. Compare the execution times of your two implementations for (4) with an implementation based on the offcial R function lfactorial(n). You may use the function system.time() to measure execution

time. What are the growth rates of the three implementations as N increases? Use the command options(expressions=500000) to increase the number of nested recursions allowed. Compare the timing of the recursion implementation as much as possible, and continue beyond that for the other two implementations.

```
logfact1Time <- c()
logfact2Time <- c()
logfact3Time <- c()
for(N in NList1){
        time1 <- system.time(sumLogFact(N, logfact1))[3]
        time3 <- system.time(sumLogFact(N, lfactorial))[3]
        logfact1Time <- c(logfact1Time, time1)
        logfact3Time <- c(logfact3Time, time3)
}

for(N in NList2){
        time2 <- system.time(sumLogFact(N, logfact2))[3]
        logfact2Time <- c(logfact2Time, time2)
}
sup <- replicate(n=length(logfact1Time) -length(logfact2Time), NaN )
logfact2Time <- c(logfact2Time, sup)
timeVector <- cbind(N = NList1, forloop = logfact1Time, recursion = logfact2Time, lfactorial = logfact3
row.names(timeVector) <- NULL
timeVector <- as.data.frame(timeVector)
head(timeVector)
```

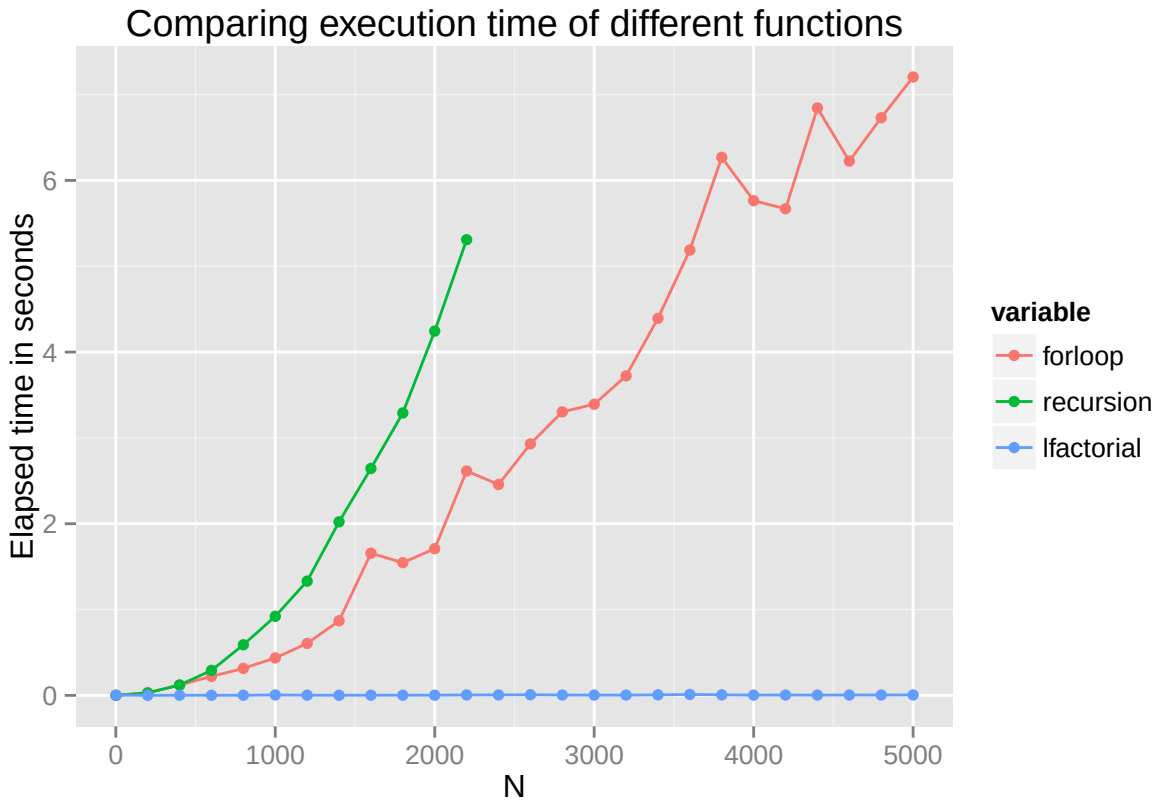
```
##      N forloop recursion lfactorial
## 1    1   0.000     0.000      0.003
## 2  200   0.031     0.028      0.000
## 3  400   0.121     0.120      0.001
## 4  600   0.221     0.291      0.001
## 5  800   0.314     0.589      0.001
## 6 1000   0.436     0.921      0.005
```

```
mtimeVector <- melt(timeVector, id = c("N"))

ggplot(mtimeVector, aes(x=N, y=value, color = variable)) +
  geom_line() +
  geom_point() +
  ggtitle("Comparing execution time of different functions") +
  ylab("Elapsed time in seconds")
```

```
## Warning in loop_apply(n, do.ply): Removed 14 rows containing missing values
## (geom_path).
```

```
## Warning in loop_apply(n, do.ply): Removed 14 rows containing missing values
## (geom_point).
```



From the graph we can see the growth rate of execution time for recursion implementation is expotiential, almost linear for lfactorial, and implementation for for loop is between other two.