lab05

Derek Olson
11/13/2019

Question 1:

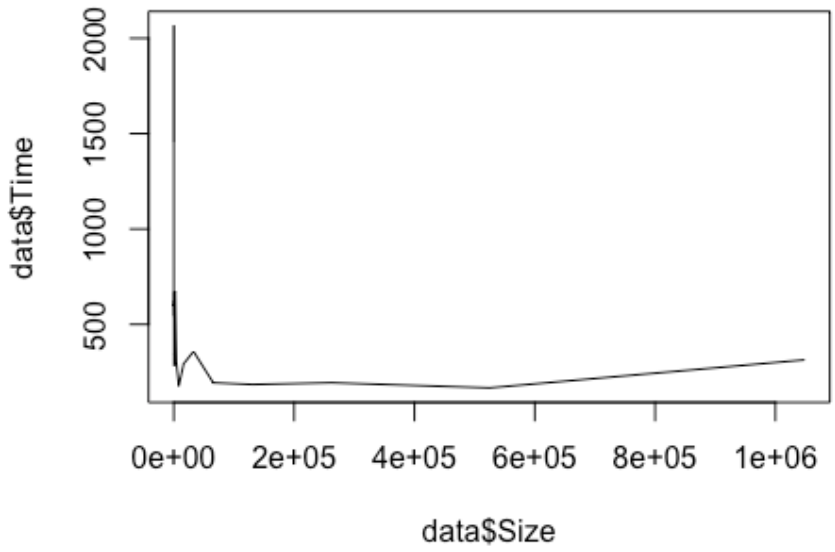
From an implementation perspective it would have been more efficient to back the sorted set with a Java list. This is due to the fact that Java Lists are part of the standard library and have methods already defined.This makes Java lists more flexible to work with.

However, arrays more efficienlty perform get and set operations. In many instances this performace gain may only be in the realm of nanoseconds.

Question 2:

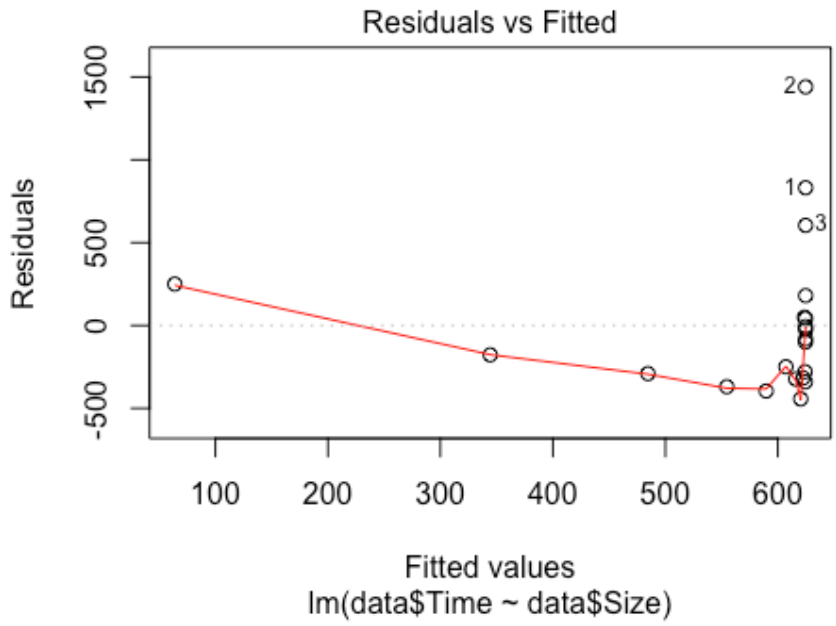
I expect the contains function to have a Big-O behavior of logN as the sorted set allows for a binary search to determine if the element is contained or not. The binary search uses the sorted nature of the set to systematically drop (not search) large chunks of the set.

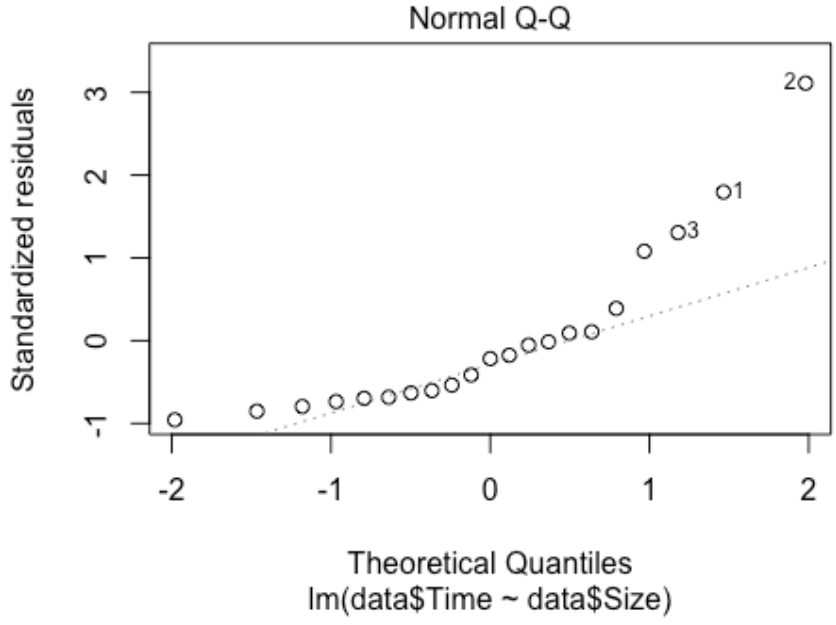
```
#Question 3:
library(readr)
data <- read_csv("timesContains.csv", col_names = FALSE)
## Parsed with column specification:
## cols(
##   X1 = col_double(),
##   X2 = col_double()
## )
colnames(data) = c("Size", "Time")
plot(data$Size, data$Time, type = "l")
```

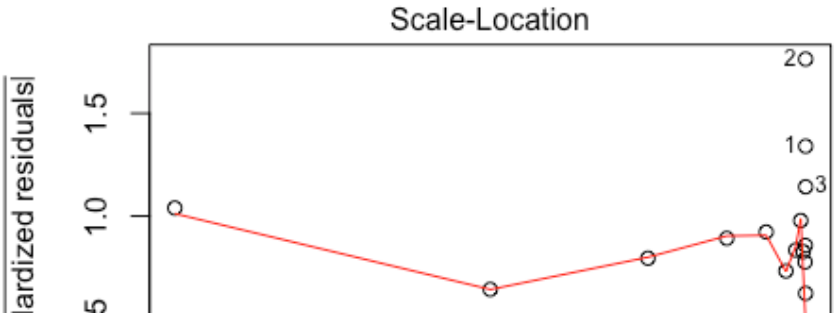


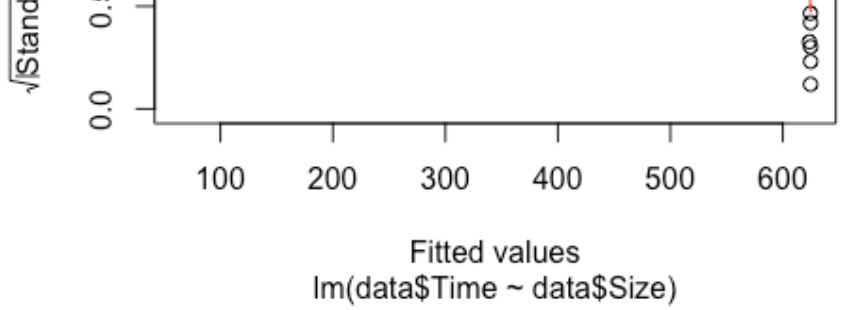

```
model = lm(data$Time ~ data$Size)
summary(model)
##
## Call:
## lm(formula = data$Time ~ data$Size)
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -443.3 -316.5 -100.6   49.4 1440.3
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)  6.247e+02  1.123e+02   5.563  2.3e-05 ***
```

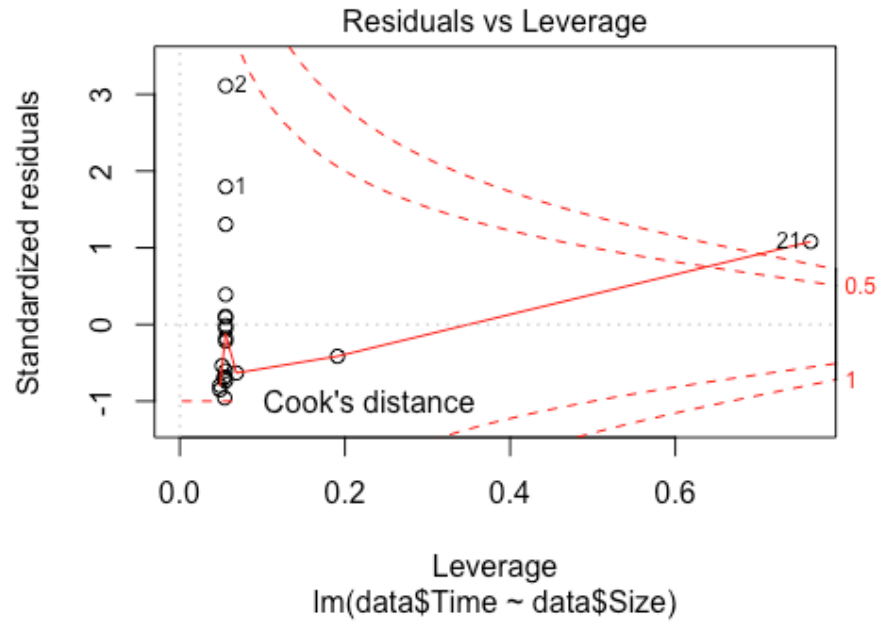
```
## data$Size   -5.348e-04  4.250e-04  -1.258    0.224
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 476.5 on 19 degrees of freedom
## Multiple R-squared:  0.07692,    Adjusted R-squared:  0.02833
## F-statistic: 1.583 on 1 and 19 DF,  p-value: 0.2235
plot(model)
```



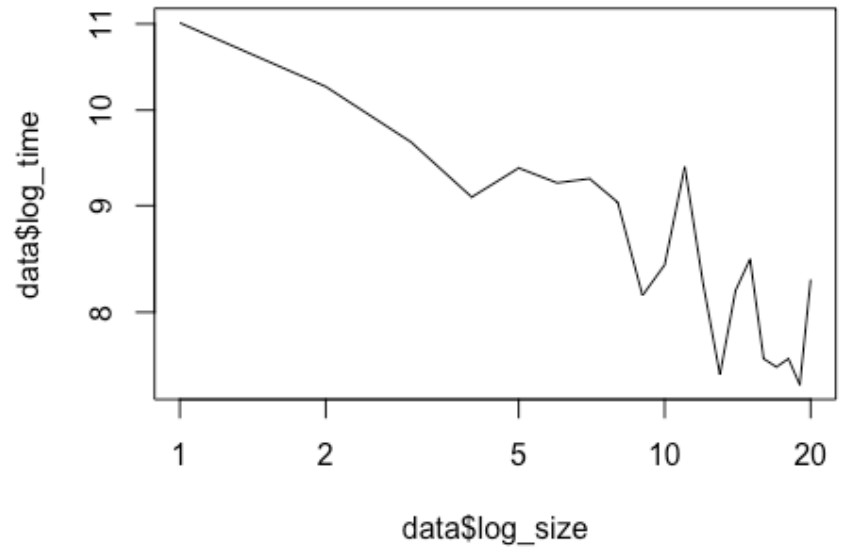









```
data['log_size'] = log2(data$Size)
data['log_time'] = log2(data$Time)
plot(data$log_size, data$log_time, type = "l", log="xy")
## Warning in xy.coords(x, y, xlabel, ylabel, log): 1 x value <= 0 omitted
## from logarithmic plot
```



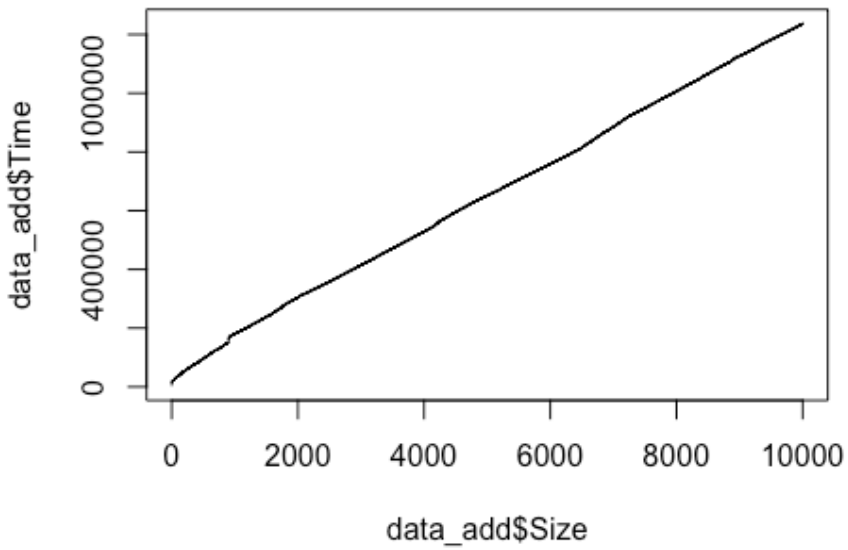
The plots do not match my expectations. The run times trend lower as the size of the set becomes larger. I expected the run times to become larger at a very slow rate as the set became larger.

Question 4:

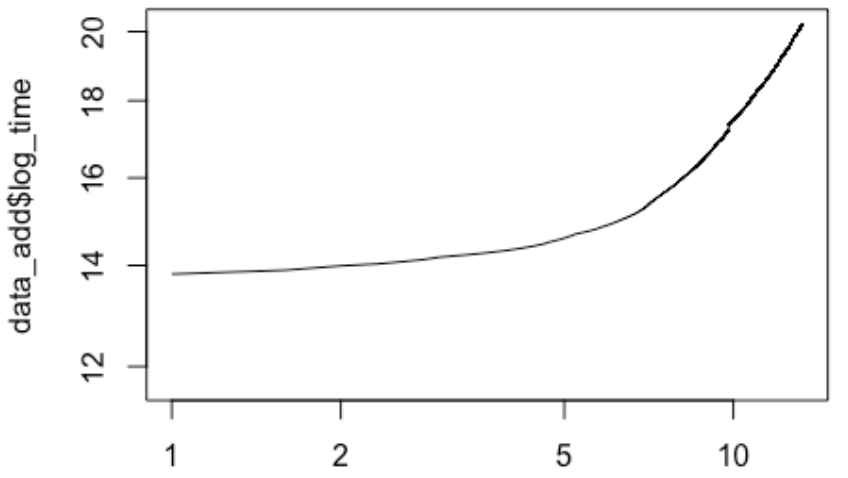
I am really not understanding this question. I think it can be interpreted 2 different ways. The first is how long it takes to determine the location of the insertion given a set of size N, which is just doing a binary search on size N. This is exaclty the same question as above, at least how I am interpreting it. This would result in a big-O complexity of logN because a binary seacrh has a big-O complexity of logN.

How I am going to analyze question 4 to differentiate it from question 3 is by timing the full add element method into a sorted set given a set of size N. This should result in a logN time for the binary search and a N time for the actual insertion. This is N+logN which is a big-O complexity of N.

```
data_add <- read_csv("timesAdd.csv", col_names = FALSE)
## Parsed with column specification:
## cols(
##   X1 = col_double(),
##   X2 = col_double()
## )
colnames(data_add) = c("Size", "Time")
plot(data_add$Size, data_add$Time, type = "l")
```




```
data_add['log_size'] = log2(data_add$Size)
data_add['log_time'] = log2(data_add$Time)
plot(data_add$log_size, data_add$log_time, type = "l", log="xy")
## Warning in xy.coords(x, y, xlabel, ylabel, log): 1 x value <= 0 omitted
## from logarithmic plot
```



data_add$log_size