insert.R

Preston

2020-05-23

```
# Preston Dunton
# CS320 Honors Option
# May 20, 2020
# pdunton@rams.colostate.edu

# insert() Should be O(logn)

insert_binomial = read.csv("./insert_binomial.csv")
attach(insert_binomial)
```

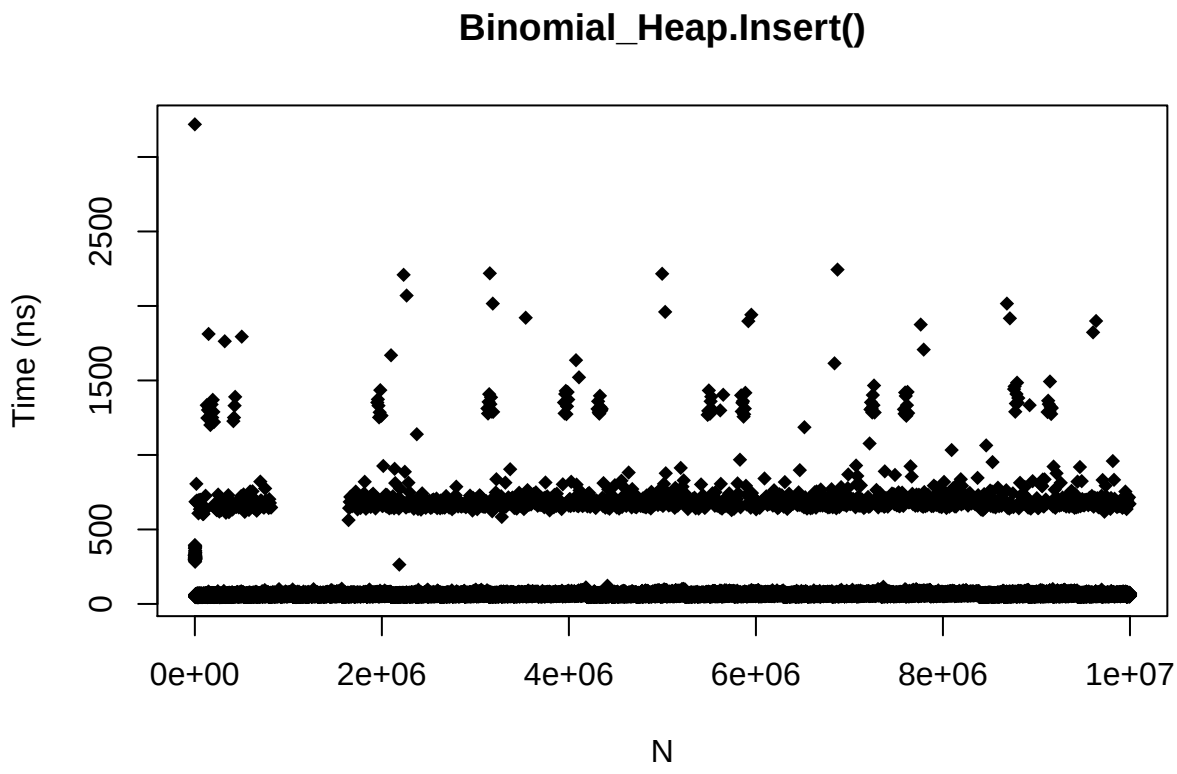
```
## The following object is masked from package:base:
##
##     T
```

```
summary(T)
```

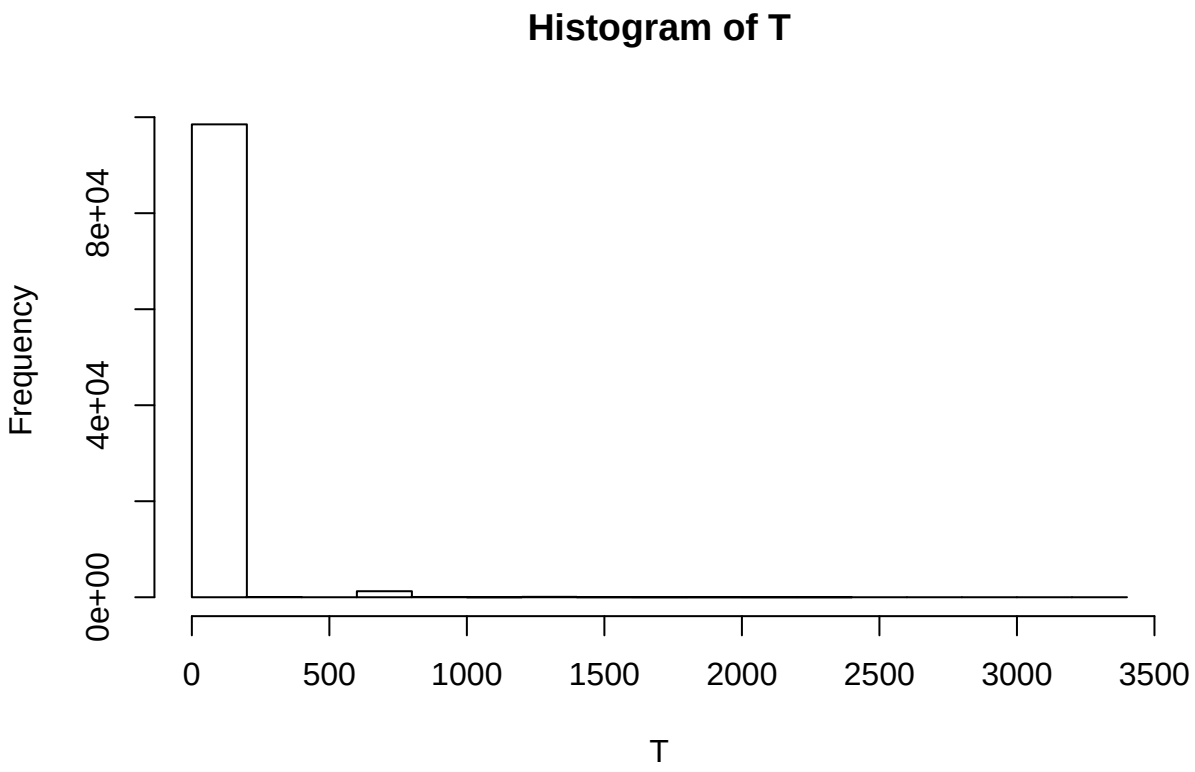
```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    45.0    59.0    65.0    74.7    70.0  3219.0
```

```
# min 45
# q1 59
# median 65
# mean 74.4
# q3 70.0
# max 3219

plot(N,T,pch=18,xlab="N",ylab="Time (ns)",main="Binomial_Heap.Insert()")
```




```
hist(T,breaks=20)
```




```
# Let's see if we can remove some outliers

quantile(T,seq(0,1,0.1))
```

```
##   0%  10%  20%  30%  40%  50%  60%  70%  80%  90% 100%
##   45   55   58   61   63   65   67   68   71   74 3219
```

```
quantile(T,seq(0.9,1,0.01))
```

```
##     90%     91%     92%     93%     94%     95%     96%     97%     98%     99%
##   74.00   74.00   75.00   76.00   77.00   78.00   79.00   82.00   86.00  666.01
##    100%
## 3219.00
```

```
quantile(T,seq(0.98,0.99,0.001))
```

```
##    98%  98.1%  98.2%  98.3%  98.4%  98.5%  98.6%  98.7%  98.8%  98.9%    99%
##  86.00  87.00  88.00  90.00  91.00 100.00 638.00 650.00 657.00 662.00 666.01
```

```
# Let's separate the top 1.5% and analyze

# Top 1.5%
sum(T>100) # There are 1498 outliers
```

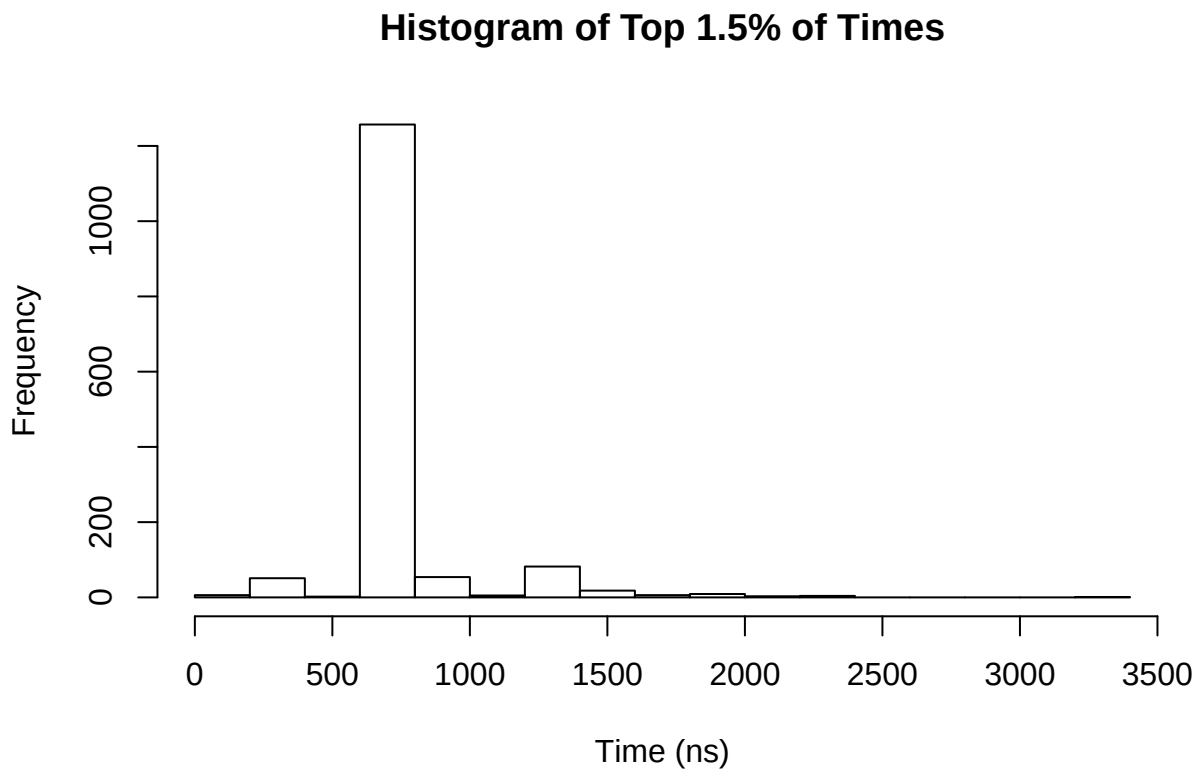
```
## [1] 1498
```

```
summary(T[which(T>100)])
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
```

```
##   101.0   661.0   679.0   738.2   712.0  3219.0
```

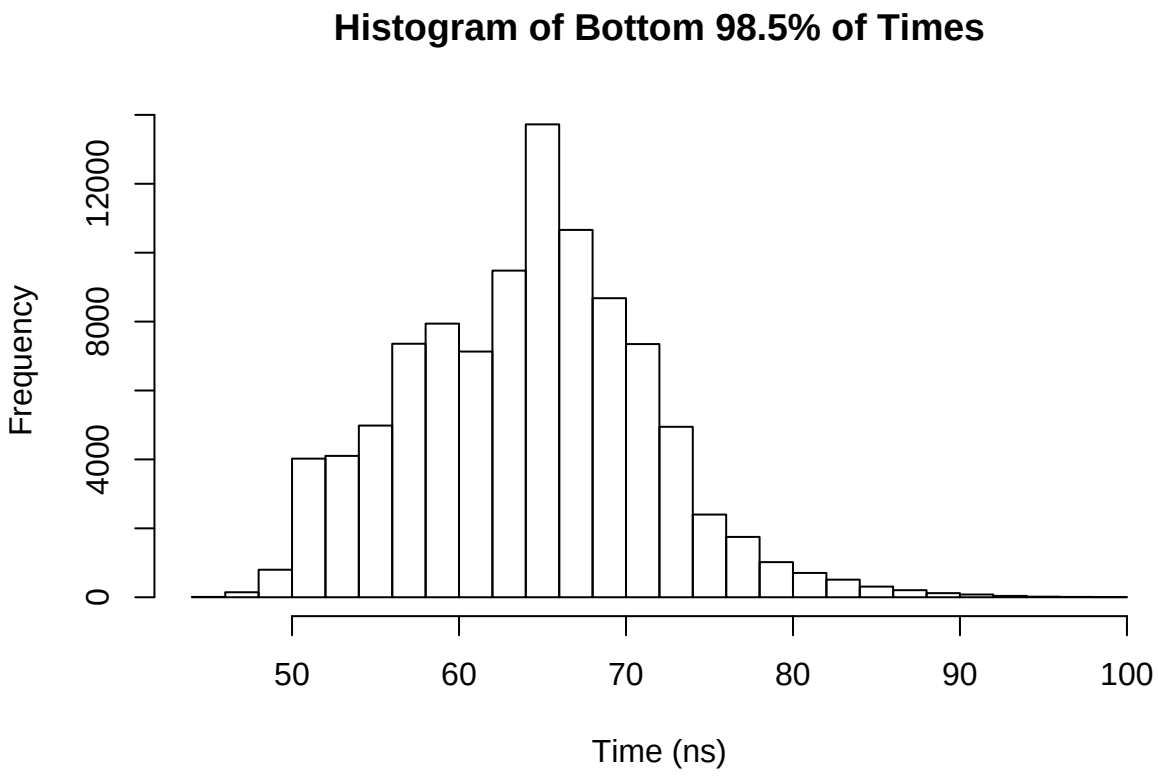
```
# min 101
# q1 661
# median 679
# mean 738.2
# q3 712
# max 3219
hist(T[which(T>100)],main="Histogram of Top 1.5% of Times",xlab="Time (ns)",breaks=20)
```




```
# Bottom 98.5%
summary(T[which(T<=100)])
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   45.00   59.00   65.00   64.61   69.00  100.00
```

```
# min 45
# q1 59
# median 65
# mean 64.61
# q3 69
# max 100
hist(T[which(T<=100)],main="Histogram of Bottom 98.5% of Times",xlab="Time (ns)",breaks=20)
```




```
# 98.5% of insertions take less than 100 ns
# There also appears to be no large corelation between N and insertion time
# The implementation must be correct for an O(logn) time

detach(insert_binomial)
```