# clear.R

Preston

2020-05-23

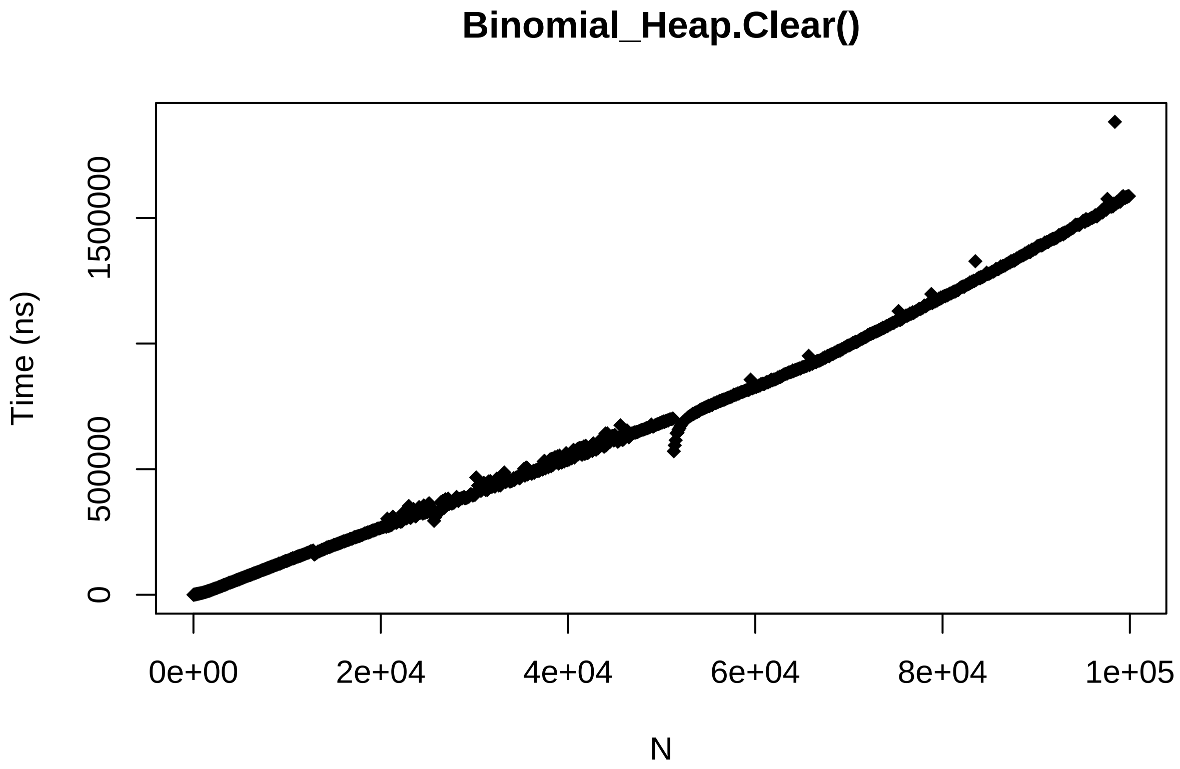
```r
# Preston Dunton
# CS320 Honors Option
# May 23, 2020
# pdunton@rams.colostate.edu

# clear() should be O(n) since we must delete every element in the heap

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

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

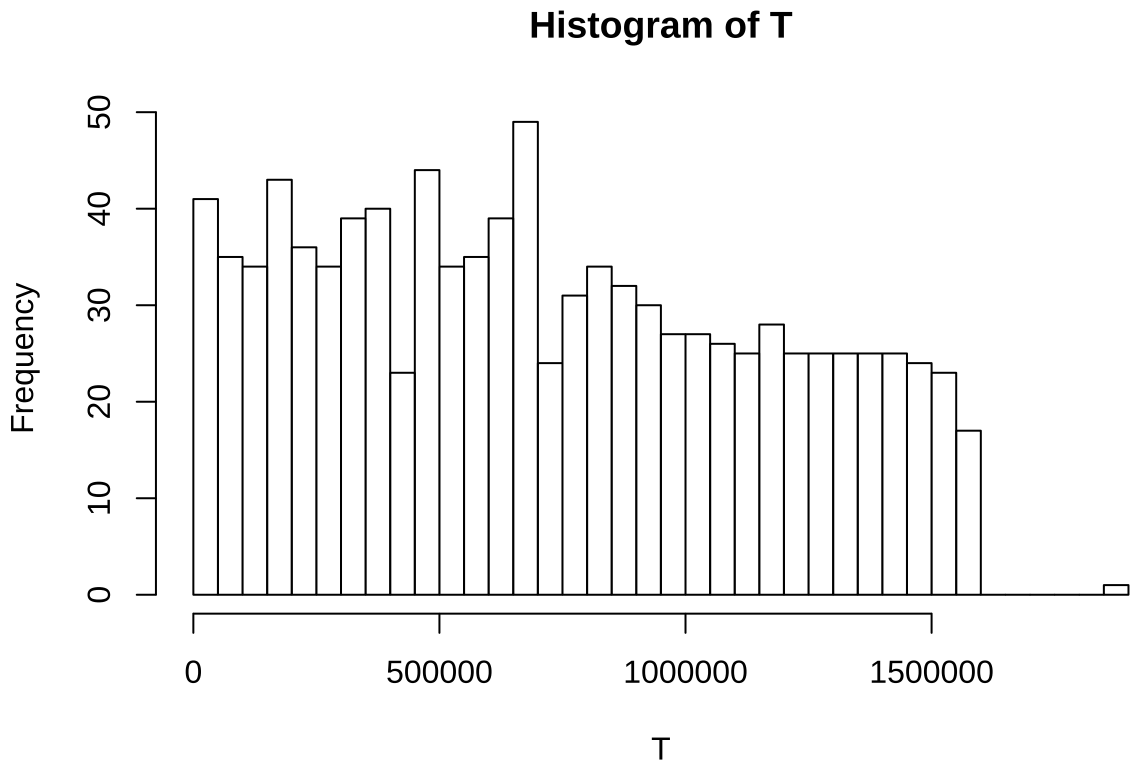
```r
summary(T)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
##     140  332821  674865  719946 1086782 1882524
```

```r
# min 140
# q1 332821
# median 674865
# mean 719946
# q3 1086782
# max 1882524

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



```
hist(T,breaks=30)
```



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

sum(T>1600000) # There's only one point that seems to lie above the rest.
```

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

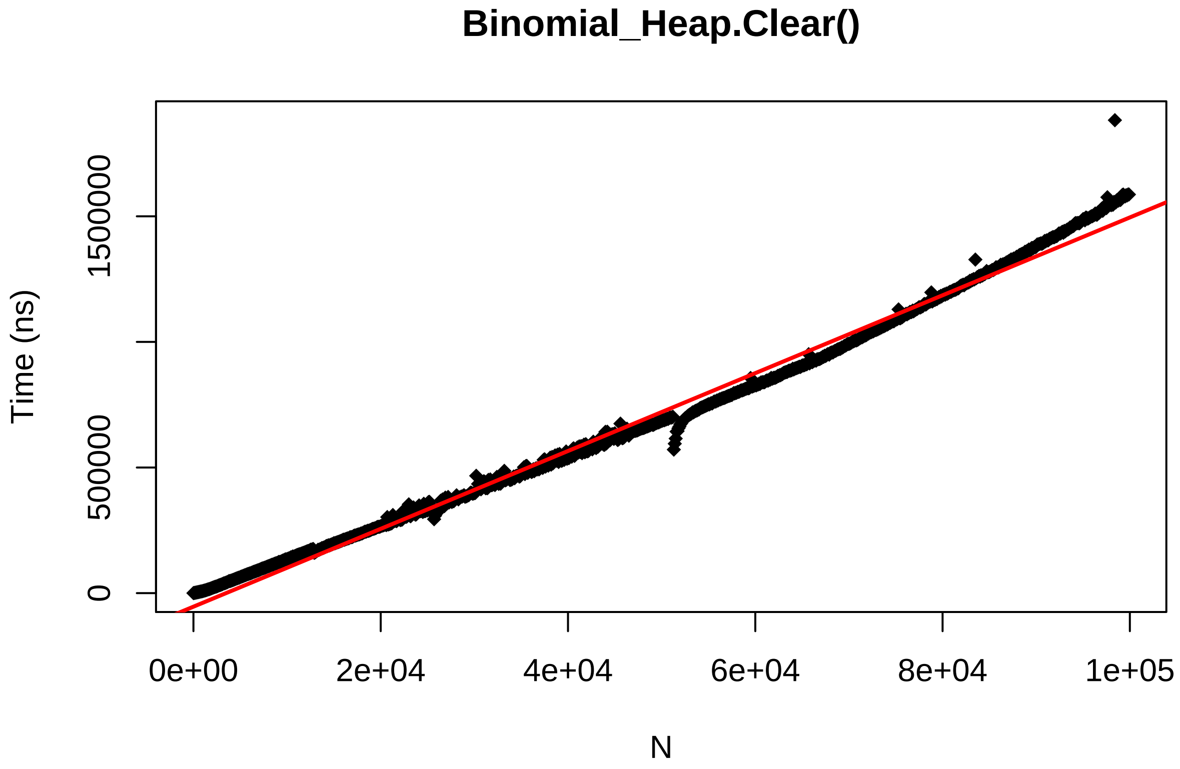
```
# this will not strongly affect our analysis

# Let's see if we can correlate N and T.  It appears to be a linear relationship, as we expect.

cor(N,T) # very strong corelation of 0.995939
```

```
## [1] 0.995939
```

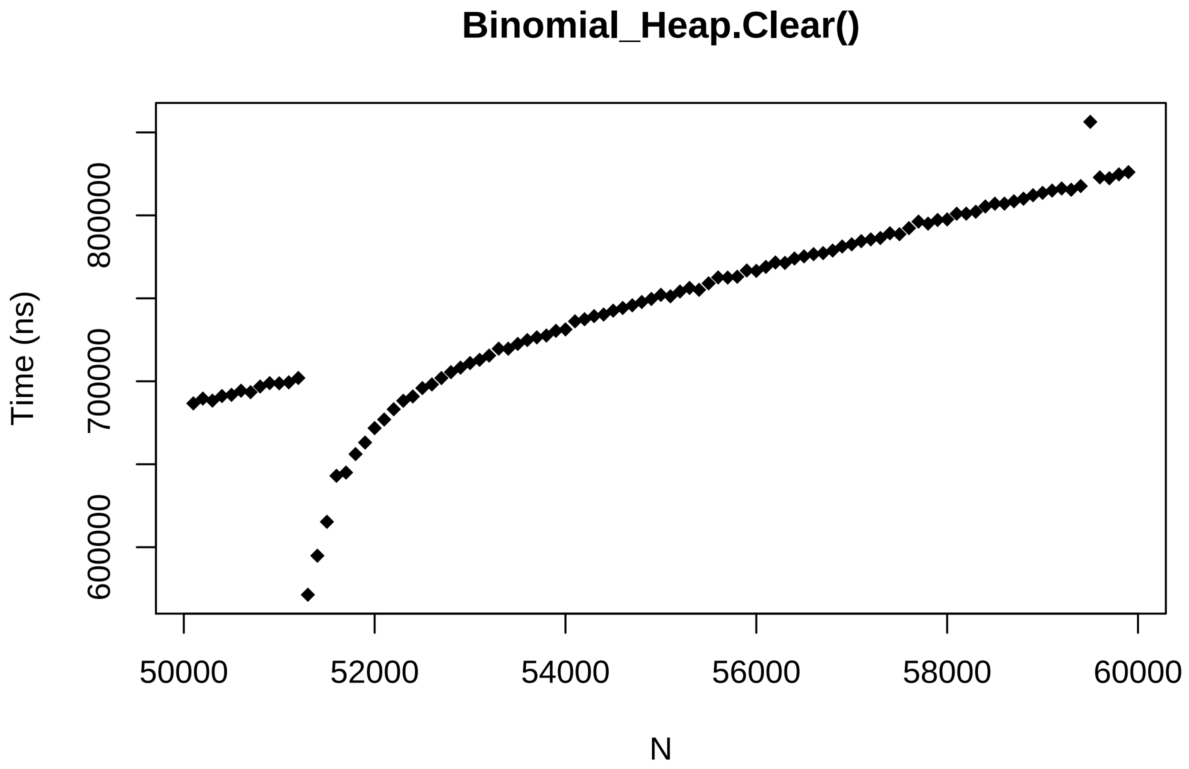
```
model = lm(T~N)
summary(model)
```

```
##
## Call:
## lm(formula = T ~ N)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -169521  -32481   -2790   31718  411858
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept) -5.402e+04  2.558e+03  -21.12   <2e-16 ***
## N            1.549e+01  4.434e-02  349.47   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 40470 on 998 degrees of freedom
## Multiple R-squared:  0.9919, Adjusted R-squared:  0.9919
## F-statistic: 1.221e+05 on 1 and 998 DF,  p-value: < 2.2e-16
```

```
plot(N,T,pch=18,xlab="N",ylab="Time (ns)",main="Binomial_Heap.Clear()")
abline(model,lwd=2,col="red")
```



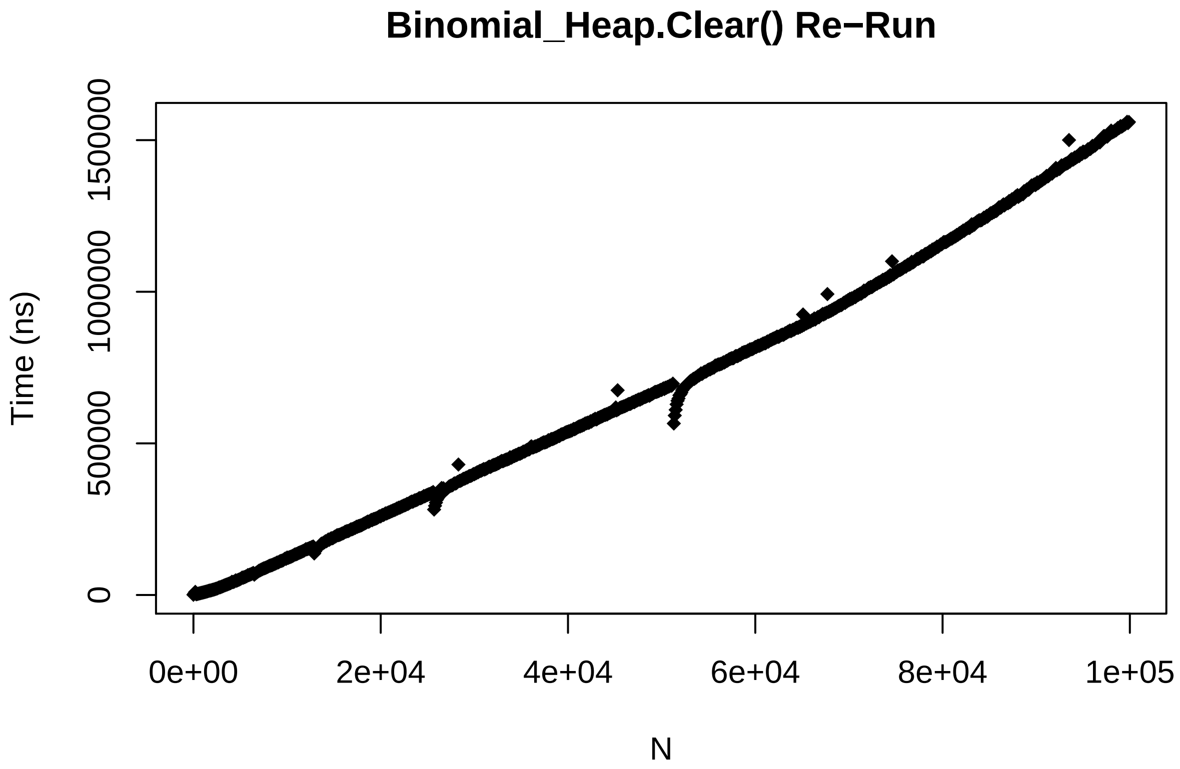
```
# there also appears to be something going on in 50000<n<60000.  Lets's look
plot(N[which(50000<N & N<60000)],T[which(50000<N & N<60000)],pch=18,xlab="N",ylab="Time (ns)",main="Bin
```



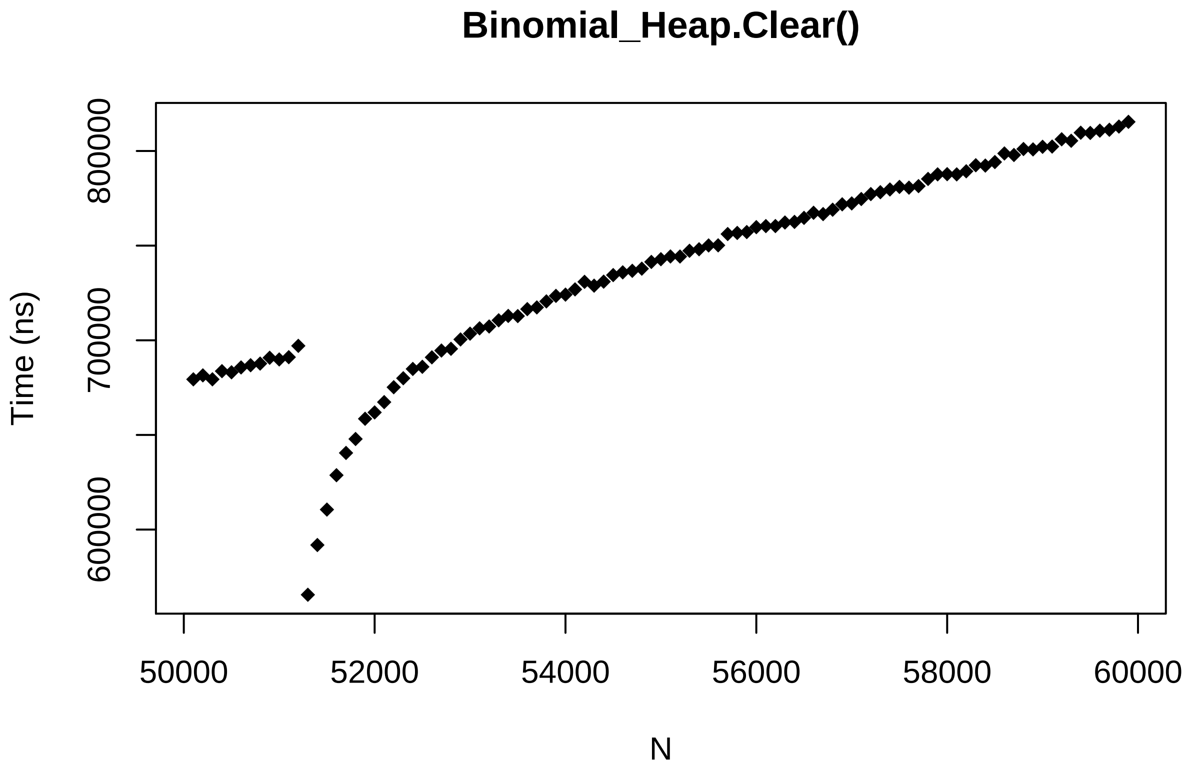
```
# there's a weird dropoff hapening.  Let's re-run the tests and see if this patter repeats

detach(clear_binomial)
clear_rerun_binomial = read.csv("./clear_rerun_binomial.csv")
attach(clear_rerun_binomial)
```

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

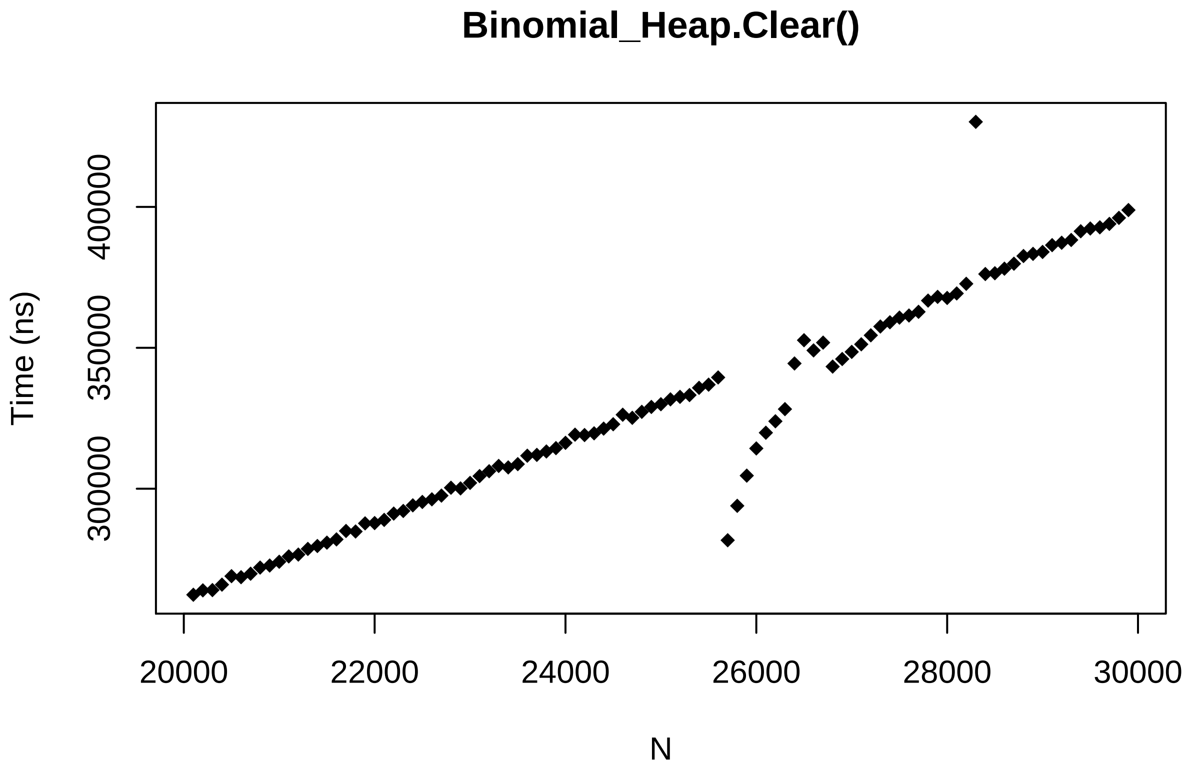
```
plot(N,T,pch=18,xlab="N",ylab="Time (ns)",main="Binomial_Heap.Clear() Re-Run")
```



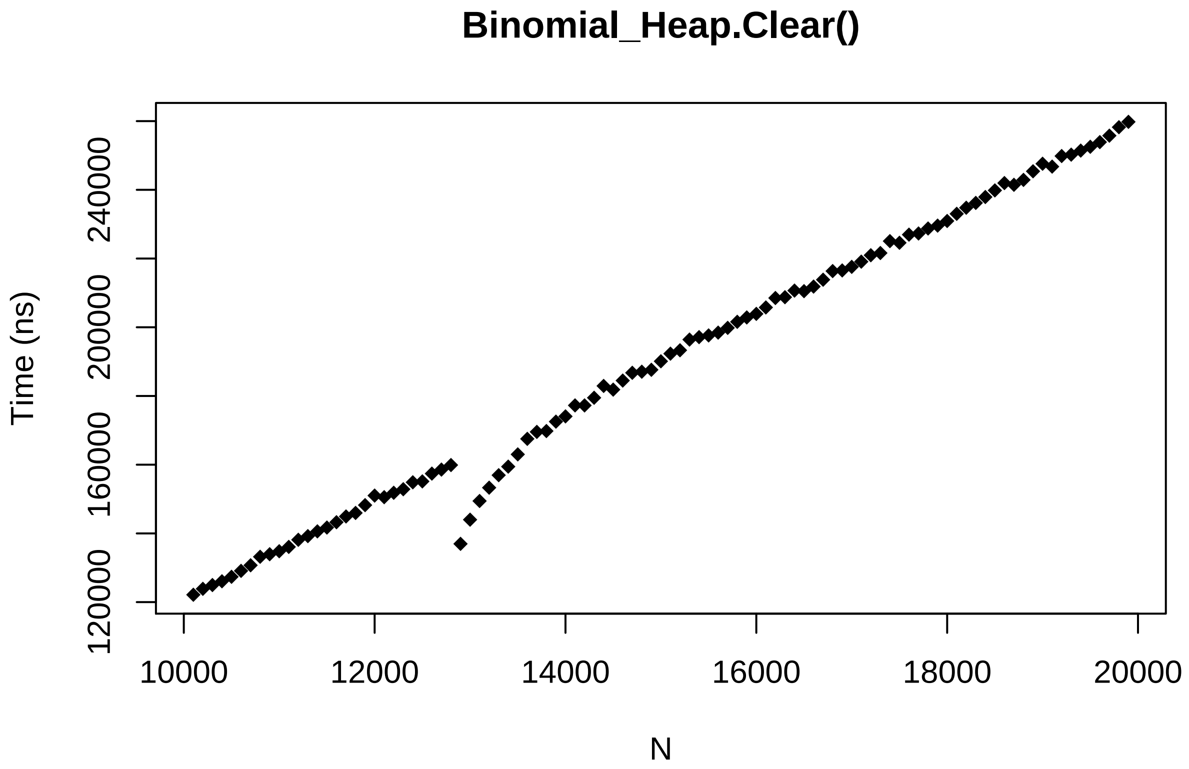
```
# The pattern repeats, and now we see it in a few different places!  Let's look closely.
plot(N[which(50000<N & N<60000)],T[which(50000<N & N<60000)],pch=18,xlab="N",ylab="Time (ns)",main="Bin
```



```
plot(N[which(20000<N & N<30000)],T[which(20000<N & N<30000)],pch=18,xlab="N",ylab="Time (ns)",main="Bin
```



```
plot(N[which(10000<N & N<20000)],T[which(10000<N & N<20000)],pch=18,xlab="N",ylab="Time (ns)",main="Bin
```



```
# The drop off seems to be more exagerated as N increases.

detach(clear_rerun_binomial)

# The data seen for the clear() operation is clearly linear, which is expected given that
# to clear a heap, you must delete each element.
# Our linear regersion model tells us that for each additional element in the heap,
# clearing takes about 15.49 extra nanoseconds.
# As far as that strange dropoff pattern goes, I have no idea why this could be occuring.
# The recursive delete_tree() method runs for every node once clear() is called, so
# I don't see why adding more elements can sometimes cause a dropoff in time.
# Oh well.

# Complexity is O(n)
```