# SortTimes

*Samyak Ahuja*

*August 23, 2018*

## Complexity for different Sorting Algorithms.

### Helper Functions

#### Replicator

```r
replicator <- function(func, size = 1000){
  if(size == 1000){
      ele <- seq(from = 0, to = 1000, by = 50)
  }else{
      ele <- seq(from = 0, to = 10000, by = 250)
  }
  ele <- ele[-1]
  timeElapsed <- c()
  for(n in ele){
    op <- 0
    for(i in 1:10){
        op = op + func(sample(x = 1:100, size = n, replace = TRUE))$operations
    }
    op = op / 10
    timeElapsed <- c(timeElapsed, op)
  }
  return (data.frame(ele,timeElapsed))
}
```

#### Plotter

```r
plotter <- function(df, df_title){
  ggplot(df, aes(ele, timeElapsed, color = timeElapsed)) +
    geom_point(shape = 16, size = 5, show.legend = FALSE, alpha = 0.6) +
    stat_smooth(method="lm", formula=y~poly(x,2), rm = FALSE) +
    theme_minimal() +
    labs(subtitle = "Time vs Size",
       y = "Number of Comparisons (Averaged)",
       x = "Number of Elements",
       title = df_title) +
    scale_color_gradient(low = "#32aeff", high = "#f2aeff") +
    stat_poly_eq(parse=T, aes(label = ..eq.label..), formula=y~poly(x,2))
}
```

### Combined Plotter

```
comb_plotter <- function(df, df_title){
  ggplot(df, aes(ele, value, col = variable)) +
  geom_point(shape = 16, size = 2, alpha = 0.6) +
  stat_smooth(method="lm", formula=y~poly(x,2)) +
  theme_minimal() +
  labs(subtitle = "Time vs Size",
       y = "Number of Comparisons (Averaged)",
       x = "Number of Elements",
       title = df_title) +
  stat_poly_eq(parse=T, aes(label = ..eq.label..), formula=y~poly(x,2))
}
```

## Insertion Sort

### Sorting Algorithm

```
insertionSort <- function(vec){
  n <- length(vec)
  op <- 0
  for(i in 2:n){
    key <- vec[i]
    pos <- i - 1
    while(pos > 0 && vec[pos] > key){
      vec[pos + 1] = vec[pos]
      pos = pos - 1
      op <- op + 1
    }
    vec[pos + 1] <- key
    op <- op + 1
  }
  return (list("vec" = vec, "operations" = op))
}
```

### Proof of concept

```
insertionSort(c(12,-22,13,2,-33,2))
```

```
## $vec
## [1] -33 -22   2   2  12  13
##
## $operations
## [1] 14
```

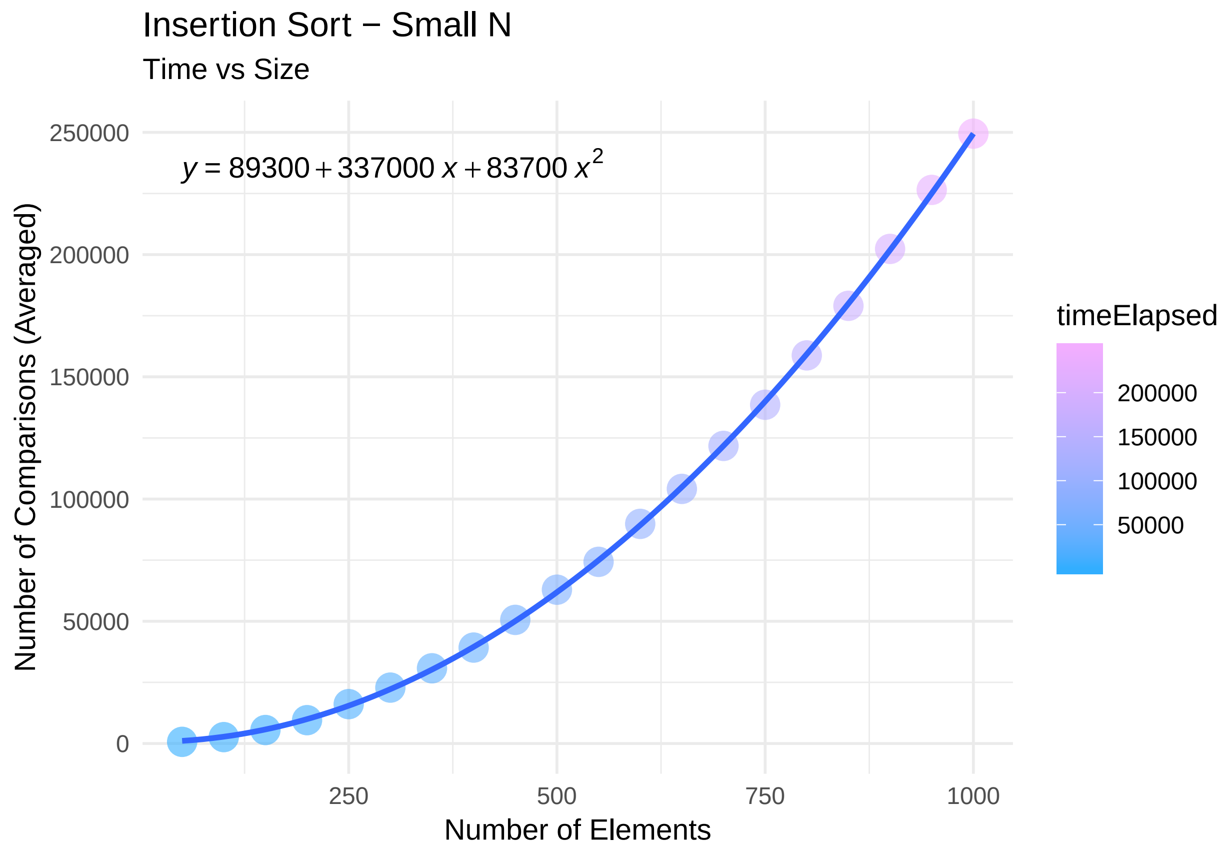
### RunTime and Plot

```
isdf_small <- replicator(insertionSort)
isdf_small
```

```
##      ele timeElapsed
## 1     50       668.4
## 2    100      2608.6
## 3    150      5505.3
## 4    200      9547.1
## 5    250     16106.9
## 6    300     22865.4
## 7    350     30766.5
## 8    400     39237.5
## 9    450     50565.8
## 10   500     62914.3
## 11   550     74306.8
## 12   600     89847.2
## 13   650    104158.7
## 14   700    121741.4
## 15   750    138576.9
## 16   800    158763.7
## 17   850    179051.0
## 18   900    202337.9
## 19   950    226474.3
## 20  1000    249483.6
```

```
plotter(isdf_small, "Insertion Sort - Small N")
```

```
## Warning: Ignoring unknown parameters: rm
```



## Merge Sort

### Sorting Algorithm

```
mergeSort <- function(vec){

  mergeTwo <- function(left,right){
    op <- 0
    res <- c()
    while(length(left) > 0 && length(right) > 0){
      op <- op + 1
      if(left[1] <= right[1]){
        res <- c(res,left[1])
        left <- left[-1]
      }else{
        res <- c(res,right[1])
        right <- right[-1]
      }
    }
    if(length(left) > 0){
      res <- c(res,left)
    }
    if(length(right) > 0){
      res <- c(res,right)
    }
    return (list("vec" = res, "operations" = op))
  }

  op <- 0
  n <- length(vec)
  if(n <= 1) return (list("vec" = vec, "operations" = op))
  else{
    middle <- length(vec) %/% 2 #integer division
    left_list <- mergeSort(vec[1:middle])
    right_list <- mergeSort(vec[(middle + 1):n])
    left <- left_list$vec
    right <- right_list$vec
    res <- mergeTwo(left,right)
    op <- op + left_list$operations + right_list$operations + res$operations
    return (list("vec" = res$vec, "operations" = op))
  }
}
```

### Proof of Concept

```
mergeSort(c(12,-22,13,2,-33,2))
```

```
## $vec
## [1] -33 -22   2   2  12  13
##
## $operations
## [1] 10
```

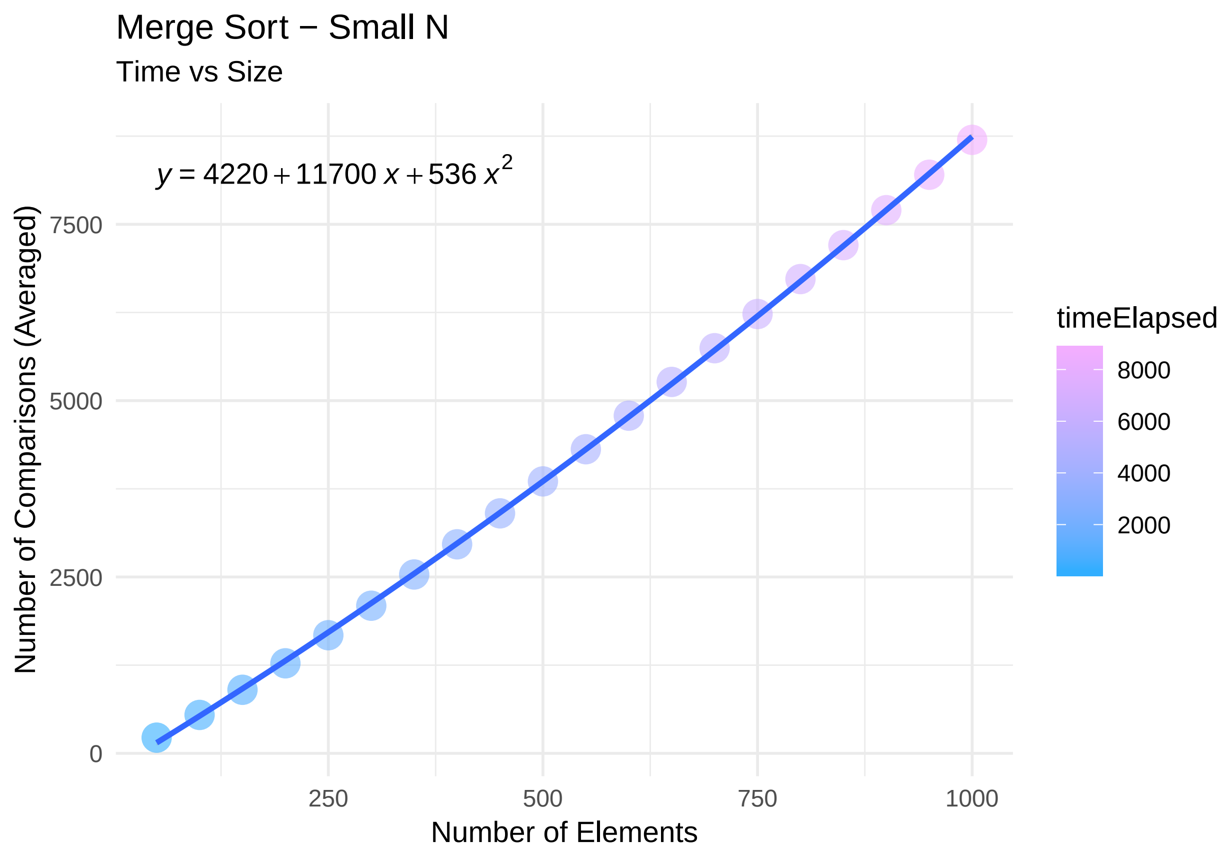
**RunTime and Plot**

```
msdf_small <- replicator(mergeSort)
msdf_small
```

```
##     ele timeElapsed
## 1    50       222.1
## 2   100       544.1
## 3   150       900.8
## 4   200      1276.0
## 5   250      1674.4
## 6   300      2092.5
## 7   350      2535.4
## 8   400      2963.1
## 9   450      3401.9
## 10  500      3855.1
## 11  550      4311.0
## 12  600      4786.9
## 13  650      5265.5
## 14  700      5743.6
## 15  750      6228.0
## 16  800      6723.5
## 17  850      7204.8
## 18  900      7700.7
## 19  950      8202.9
## 20 1000      8697.6
```

```
plotter(msdf_small, "Merge Sort - Small N")
```

```
## Warning: Ignoring unknown parameters: rm
```



## Quick Sort

### Sorting Algorithm

```
quickSort <- function(vec, low = 1, high = length(vec)){

  partition <- function(vec, low, high){
    i = low
    op <- 0
    pivot = vec[high]
    for(j in low:(high - 1)){
      op <- op + 1
      if(vec[j] <= pivot){
        temp = vec[i]
        vec[i] = vec[j]
        vec[j] = temp
        i = i + 1
      }
    }
    temp = vec[i]
    vec[i] = vec[high]
    vec[high] = temp
    return (list("vec" = vec, "operations" = op, "pi" = i))
  }
  op <- 0
  if(low < high){
    pi_list = partition(vec, low, high)
    vec <- pi_list$vec
    pi <- pi_list$pi

    left_list <- quickSort(vec, low, pi - 1)
    vec <- left_list$vec

    right_list <- quickSort(vec, pi + 1, high)
    vec <- right_list$vec

    op <- op + left_list$operations + right_list$operations + pi_list$operations
    return (list("vec" = vec, "operations" = op))
  }else{
    return (list("vec" = vec, "operations" = op))
  }
}
```

### Proof of Concept

```
quickSort(c(12,-22,13,2,-33,2))
```

```
## $vec
## [1] -33 -22   2   2  12  13
##
## $operations
## [1] 9
```

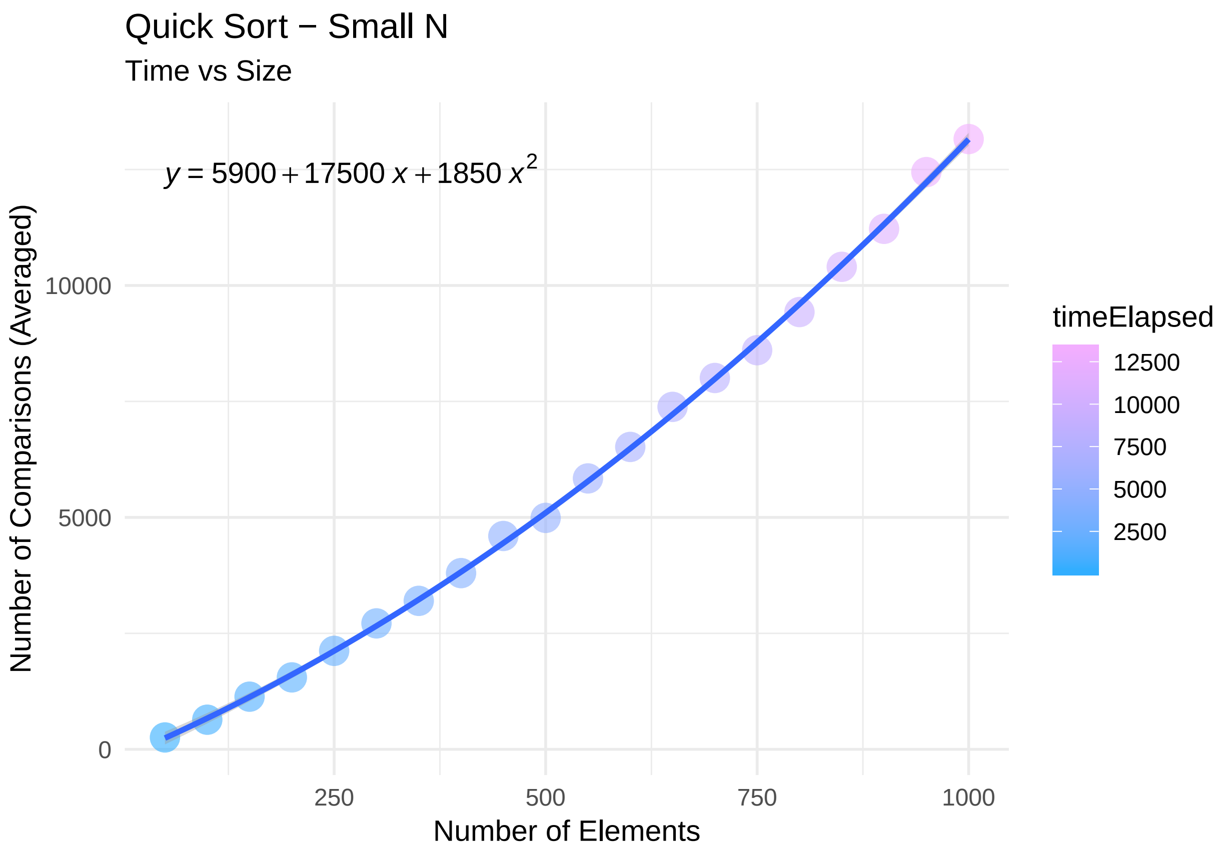
### RunTime and Plot

```
qsdf_small <- replicator(quickSort)
qsdf_small
```

```
##     ele timeElapsed
## 1    50       255.7
## 2   100       638.1
## 3   150      1135.0
## 4   200      1550.7
## 5   250      2122.8
## 6   300      2715.0
## 7   350      3200.7
## 8   400      3798.6
## 9   450      4599.3
## 10  500      4989.2
## 11  550      5840.4
## 12  600      6519.2
## 13  650      7382.9
## 14  700      8005.5
## 15  750      8604.5
## 16  800      9427.3
## 17  850      10402.5
## 18  900      11221.3
## 19  950      12445.1
## 20 1000      13157.5
```

```
plotter(qsdf_small, "Quick Sort - Small N")
```

```
## Warning: Ignoring unknown parameters: rm
```

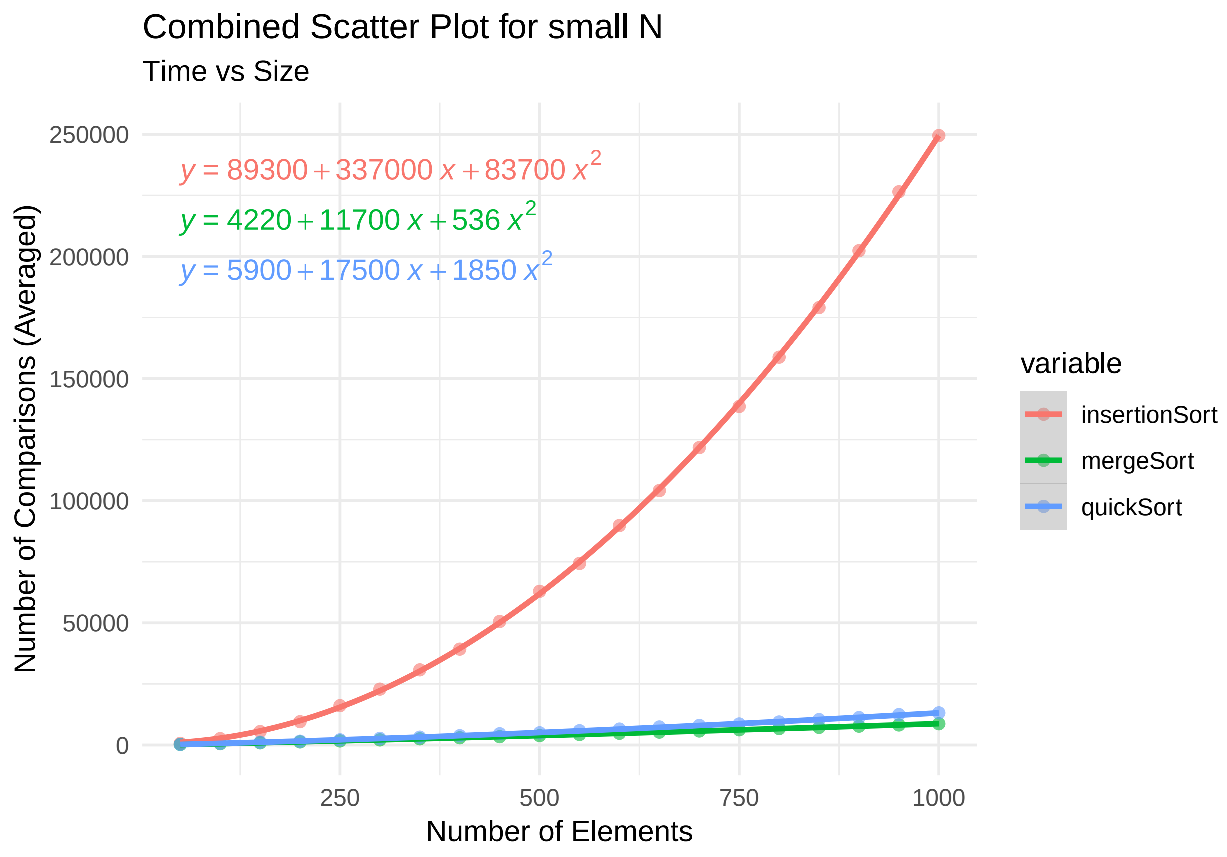


### Combined Plots

```
df_small <- data.frame(ele = msdf_small[[1]],
                 insertionSort = isdf_small[[2]],
                 mergeSort = msdf_small[[2]],
                 quickSort = qsdf_small[[2]])
df_small
```

```
##     ele insertionSort mergeSort quickSort
## 1    50         668.4     222.1     255.7
## 2   100        2608.6     544.1     638.1
## 3   150        5505.3     900.8    1135.0
## 4   200        9547.1    1276.0    1550.7
## 5   250       16106.9    1674.4    2122.8
## 6   300       22865.4    2092.5    2715.0
## 7   350       30766.5    2535.4    3200.7
## 8   400       39237.5    2963.1    3798.6
## 9    450        50565.8    3401.9    4599.3
## 10   500        62914.3    3855.1    4989.2
## 11   550        74306.8    4311.0    5840.4
## 12   600        89847.2    4786.9    6519.2
## 13   650       104158.7    5265.5    7382.9
## 14   700       121741.4    5743.6    8005.5
## 15   750       138576.9    6228.0    8604.5
## 16   800       158763.7    6723.5    9427.3
## 17   850       179051.0    7204.8   10402.5
## 18   900       202337.9    7700.7   11221.3
## 19   950       226474.3    8202.9   12445.1
## 20 1000        249483.6    8697.6   13157.5
```

```
df_small <- melt(df_small, id.vars = "ele")
comb_plotter(df_small, "Combined Scatter Plot for small N")
```


Combined Scatter Plot for small N
Time vs Size
$y = 89300 + 337000\,x + 83700\,x^2$
$y = 4220 + 11700\,x + 536\,x^2$
$y = 5900 + 17500\,x + 1850\,x^2$
Number of Comparisons (Averaged)
Number of Elements
variable
insertionSort
mergeSort
quickSort