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 <- 1
  for(i in 2:n){
    key <- vec[i]
    pos <- i - 1
    op <- op + 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

```
cat(insertionSort(c(1,2,99,-21,2,23,1))$vec, "\n")
```

```
## -21 1 1 2 2 23 99
```

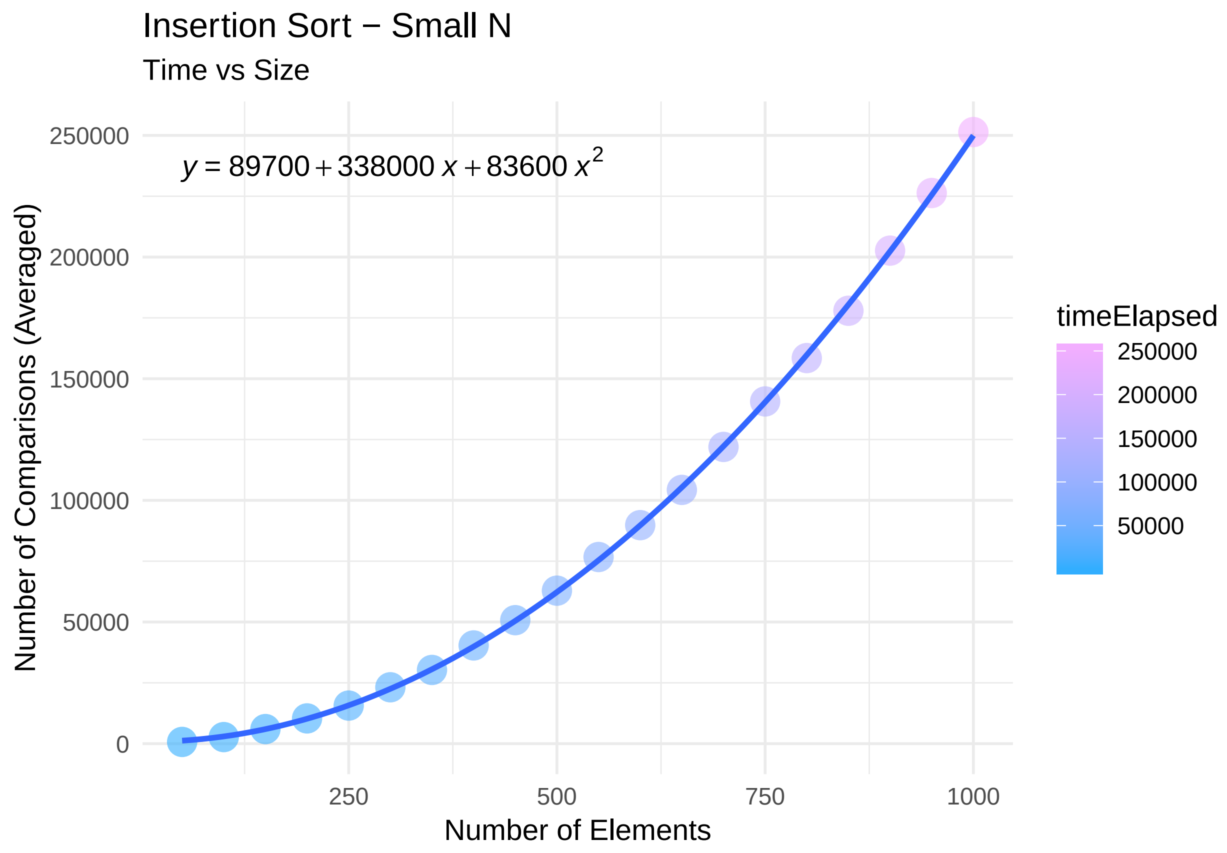
RunTime and Plot

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

```
##     ele timeElapsed
## 1    50       707.0
## 2   100      2700.4
## 3   150      5990.0
## 4   200     10376.5
## 5   250     15645.5
## 6   300     23175.1
## 7   350     30290.3
## 8   400     40368.4
## 9   450     50761.2
## 10  500     62878.0
## 11  550     76712.3
## 12  600     89814.9
## 13  650    104304.8
## 14  700    121954.7
## 15  750    140651.7
## 16  800    158477.2
## 17  850    177918.9
## 18  900    202638.0
## 19  950    226319.5
## 20 1000    251374.9
```

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

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



Merge Sort

Sorting Algorithm

```r
mergeSort <- function(vec){

  mergeTwo <- function(left,right){
    op <- 0
    res <- c()
    while(length(left) > 0 && length(right) > 0){
      op <- op + 3
      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)
    }
    op <- op + 4
    return (list("vec" = res, "operations" = op))
  }

  op <- 0
  n <- length(vec)
  if(n <= 1) return (list("vec" = vec, "operations" = op + 1))
  else{
    op <- op + 1 # 1 added for previous if
    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

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

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

```
## $operations
## [1] 61
```

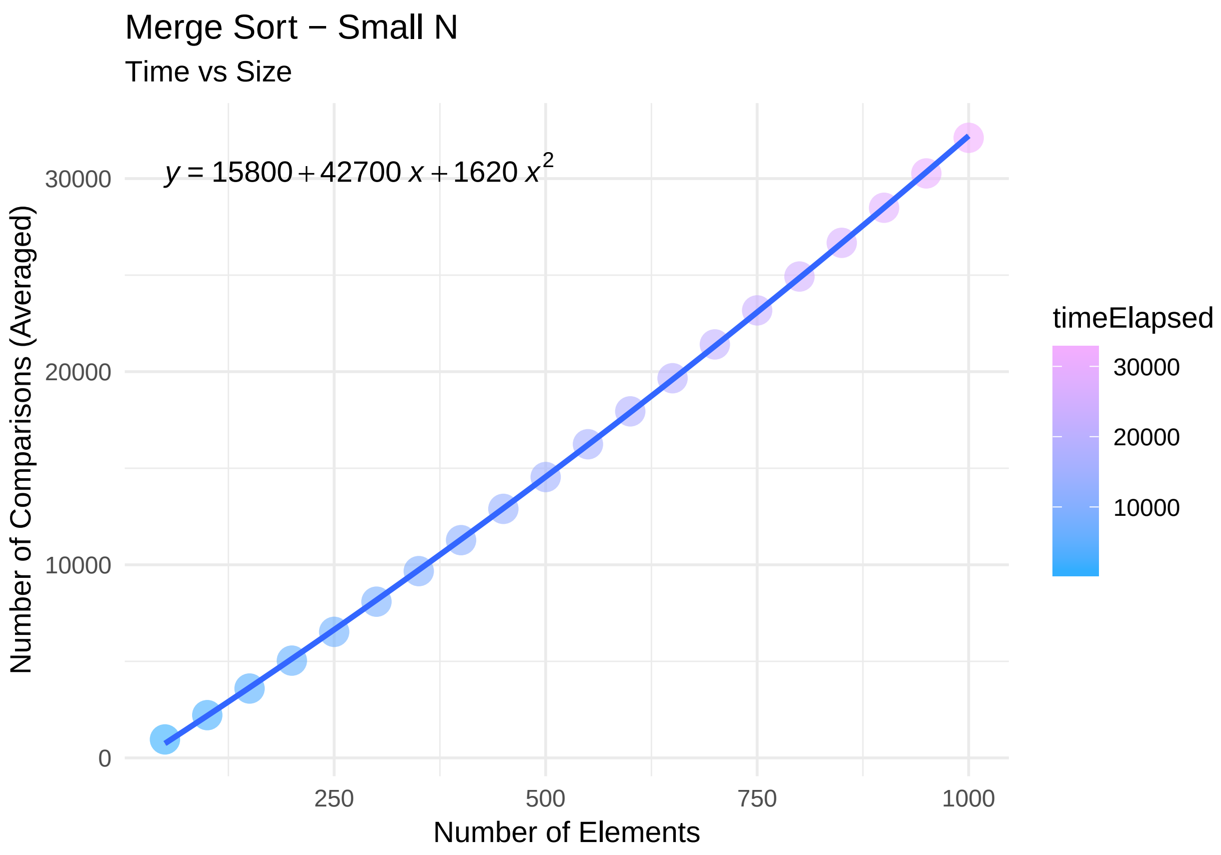
RunTime and Plot

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

```
##     ele timeElapsed
## 1    50       956.8
## 2   100      2214.1
## 3   150      3591.7
## 4   200      5037.7
## 5   250      6526.6
## 6   300      8089.3
## 7   350      9672.4
## 8   400     11276.8
## 9   450     12896.2
## 10  500     14544.7
## 11  550     16243.0
## 12  600     17945.8
## 13  650     19661.2
## 14  700     21412.9
## 15  750     23173.9
## 16  800     24928.3
## 17  850     26673.1
## 18  900     28489.6
## 19  950     30267.4
## 20 1000     32112.4
```

```
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 = 1
    pivot = vec[high]
    for(j in low:(high - 1)){
      op = op + 2
      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 <- 1
  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] 31
```

```
quickSort(c(1,1,-2,2,3,-3))
```

```
## $vec
## [1] -3 -2  1  1  2  3
##
## $operations
## [1] 35
```

```
quickSort(c(-10,-9,10,12))
```

```
## $vec
## [1] -10  -9  10  12
##
## $operations
## [1] 22
```

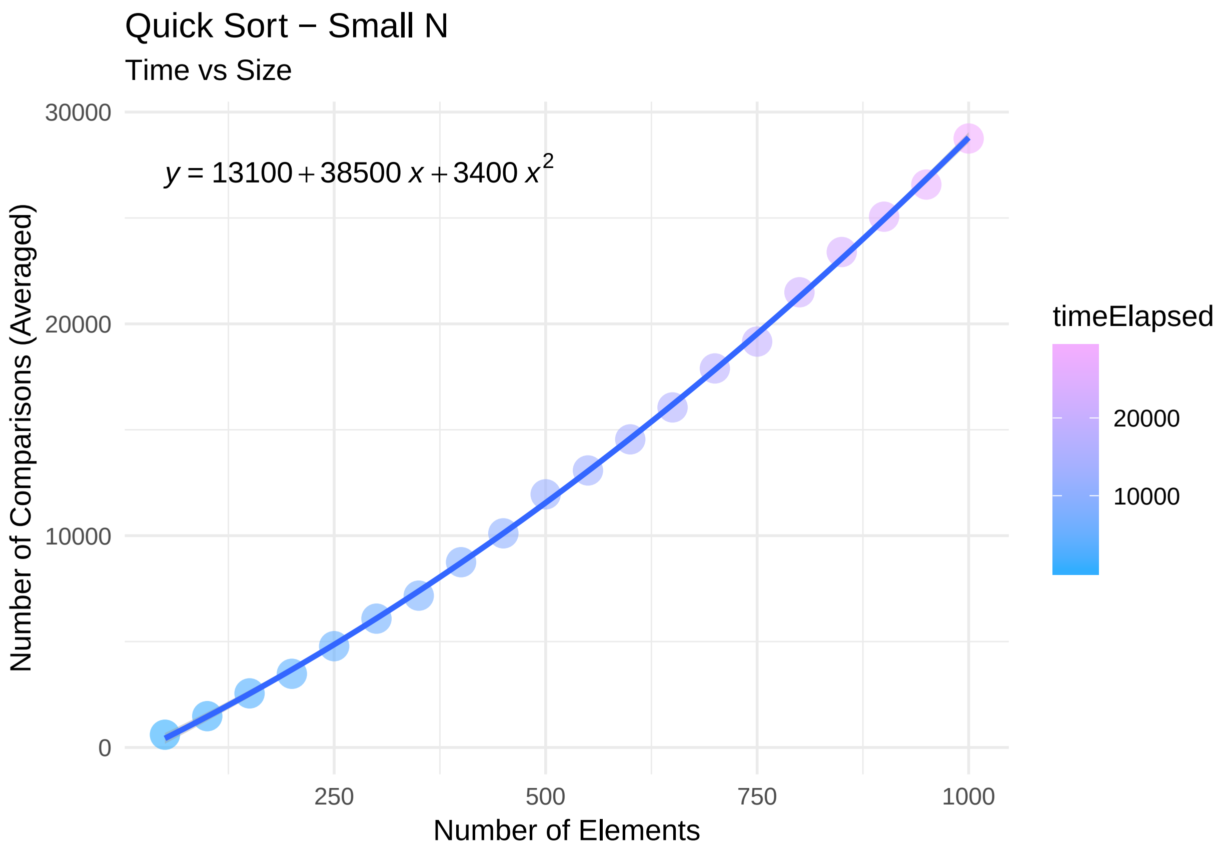
RunTime and Plot

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

```
##     ele timeElapsed
## 1    50       601.4
## 2   100      1477.8
## 3   150      2553.0
## 4   200      3478.0
## 5   250      4781.2
## 6   300      6082.5
## 7   350      7167.8
## 8   400      8746.4
## 9   450     10109.1
## 10  500     11951.6
## 11  550     13077.7
## 12  600     14546.4
## 13  650     16054.7
## 14  700     17895.4
## 15  750     19162.8
## 16  800     21490.8
## 17  850     23391.3
## 18  900     25059.1
## 19  950     26576.1
## 20 1000     28750.3
```

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

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

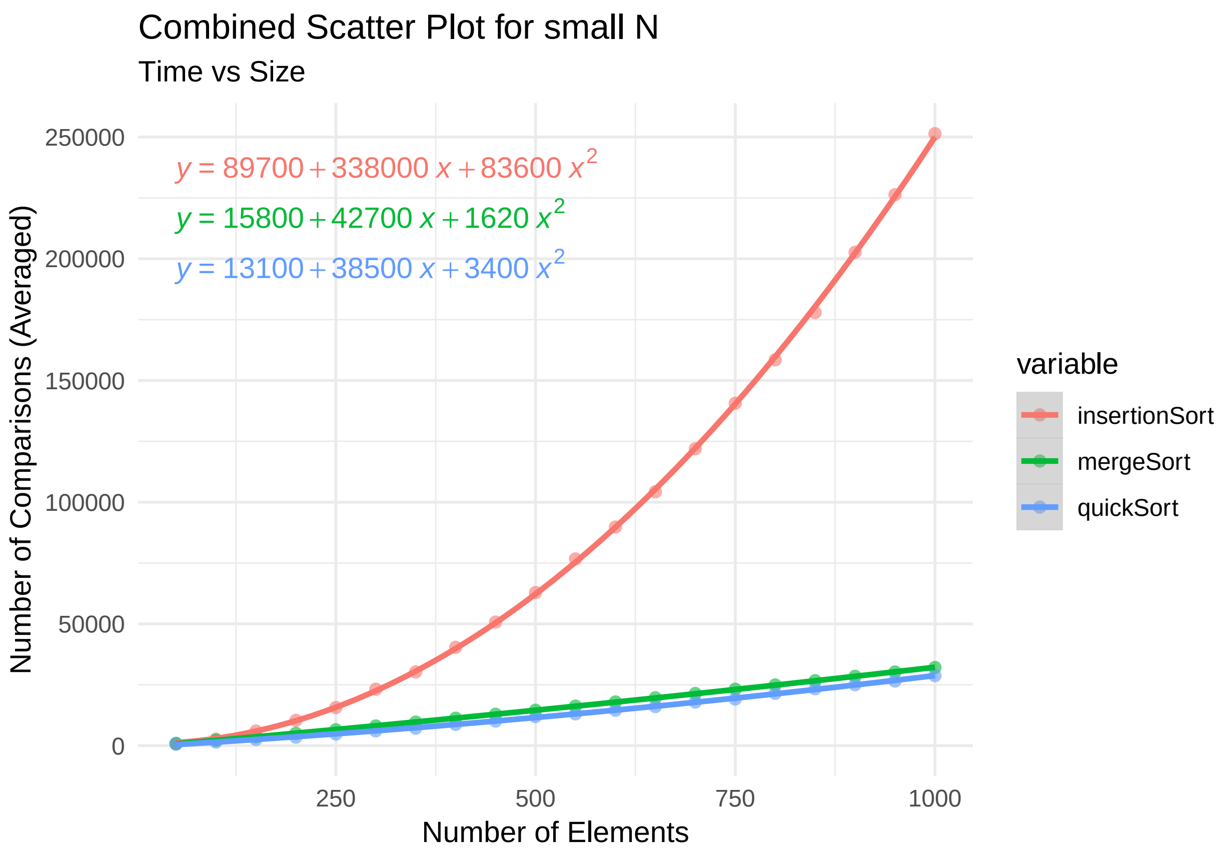


Combined Plots

```
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         707.0     956.8     601.4
## 2   100        2700.4    2214.1    1477.8
## 3   150        5990.0    3591.7    2553.0
## 4   200       10376.5    5037.7    3478.0
## 5   250       15645.5    6526.6    4781.2
## 6   300       23175.1    8089.3    6082.5
## 7   350       30290.3    9672.4    7167.8
## 8   400       40368.4   11276.8    8746.4
## 9   450       50761.2   12896.2   10109.1
## 10  500       62878.0   14544.7   11951.6
## 11  550       76712.3   16243.0   13077.7
## 12  600       89814.9   17945.8   14546.4
## 13  650      104304.8   19661.2   16054.7
## 14  700      121954.7   21412.9   17895.4
## 15  750      140651.7   23173.9   19162.8
## 16  800      158477.2   24928.3   21490.8
## 17  850      177918.9   26673.1   23391.3
## 18  900      202638.0   28489.6   25059.1
## 19  950      226319.5   30267.4   26576.1
## 20 1000      251374.9   32112.4   28750.3
```

```
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
$y = 89700 + 338000\,x + 83600\,x^2$
$y = 15800 + 42700\,x + 1620\,x^2$
$y = 13100 + 38500\,x + 3400\,x^2$
Number of Comparisons (Averaged)
0
50000
100000
150000
200000
250000
250
500
750
1000
Number of Elements
variable
insertionSort
mergeSort
quickSort