results algoritms

Adrian Arimany - 211063

2025-02-10

```r
algorithm_times <- function(data, name) {
  # First plot: Measured Time
  measured_plot <- ggplot(data, aes(x = InputSize)) +
    geom_line(aes(y = TimeNano, color = "Measured Time"), size = 1) +
    geom_point(aes(y = TimeNano, color = "Measured Time"), size = 2) +
    theme_minimal() +
    labs(title = paste("Measured Execution Time for", name),
         x = "Input Size (n)",
         y = "Time (nanoseconds)") +
    scale_color_manual(values = c("Measured Time" = "blue")) +
    theme(legend.position = "bottom")

  # Second plot: Theoretical Time
  theoretical_plot <- ggplot(data, aes(x = InputSize)) +
    geom_line(aes(y = theoretical_time, color = "Theoretical Time"), linetype = "dashed", size = 1) +
    geom_point(aes(y = theoretical_time, color = "Theoretical Time"), size = 2) +
    theme_minimal() +
    labs(title = paste("Theoretical Execution Time for", name),
         x = "Input Size (n)",
         y = "Theoretical Computation") +
    scale_color_manual(values = c("Theoretical Time" = "red")) +
    theme(legend.position = "bottom")

  # Combine both plots stacked vertically
  combined_plot <- measured_plot / theoretical_plot

  # Print the combined plot
  print(combined_plot)
}
```

Merge Sort Method

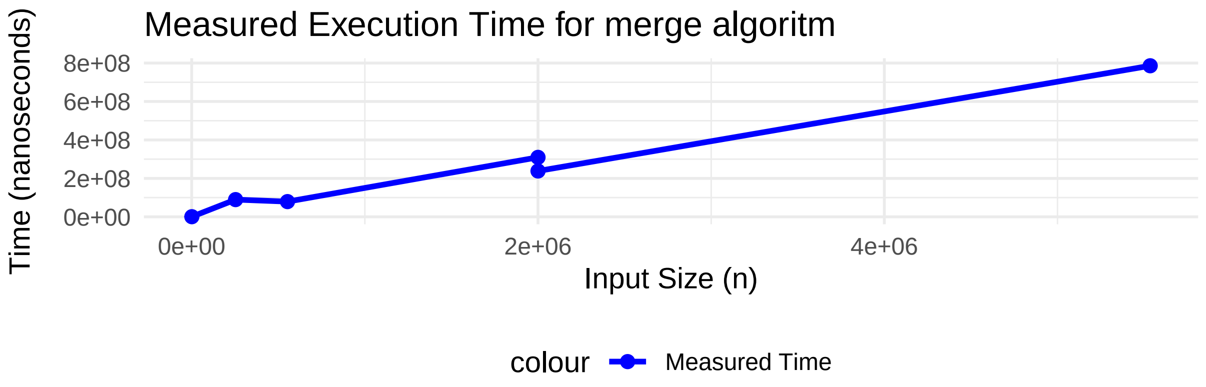
Data:

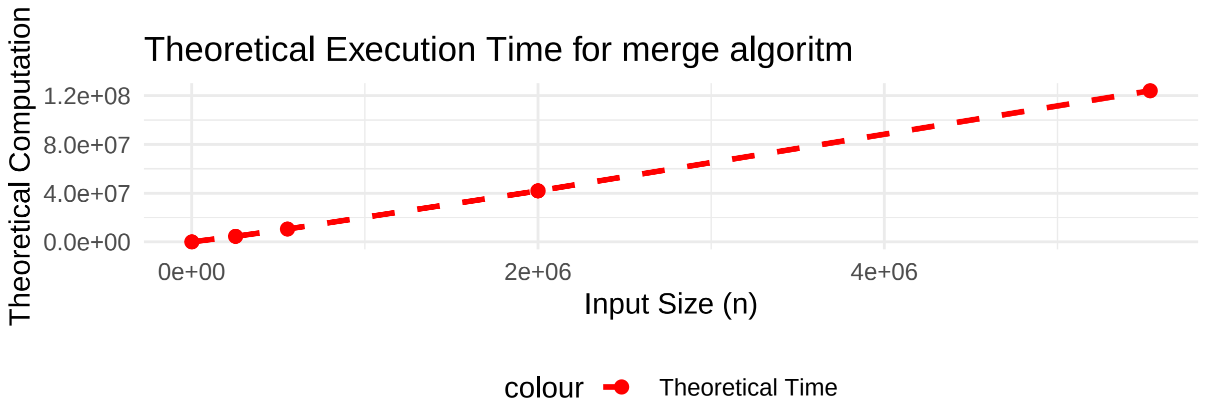
```r
mergedata <- csvPath("merge_sort_times.csv", removeSpace = FALSE)

mergedata <- mergedata %>%
  mutate(
    theoretical_time = mergedata$InputSize * log2(mergedata$InputSize)# Compute O(n log n)
  )
head(mergedata)
```

```
## # A tibble: 6 x 3
##   InputSize  TimeNano theoretical_time
##       <dbl>     <dbl>            <dbl>
## 1      1000    585620            9966.
## 2    253532  89591455         4551358.
## 3    553256  79135783        10554790.
## 4   1999999 309398292        41863115.
## 5   2000000 238434087        41863137.
## 6   5535352 785904546       123993233.
```

Plots:

```
## Warning: Using `size` aesthetic for lines was deprecated in ggplot2 3.4.0.
## i Please use `linewidth` instead.
## This warning is displayed once every 8 hours.
## Call `lifecycle::last_lifecycle_warnings()` to see where this warning was
## generated.
```





Insertion Sort Method

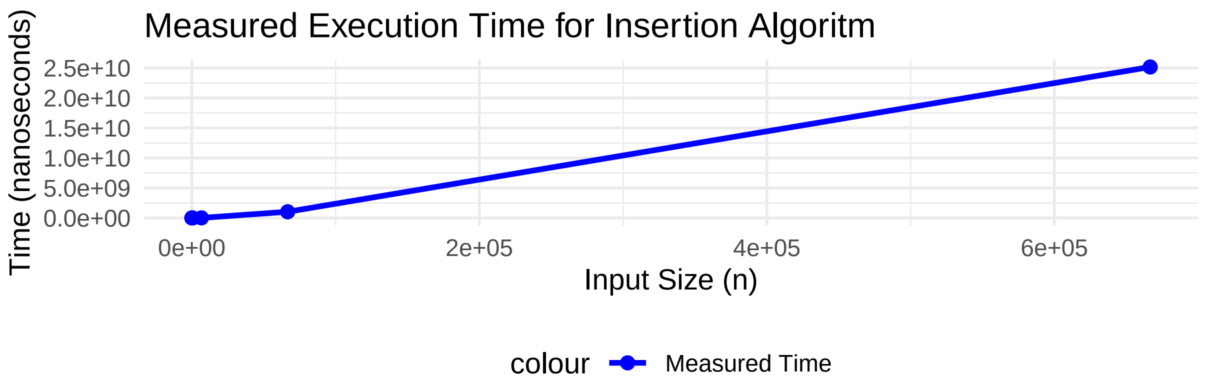
Data:

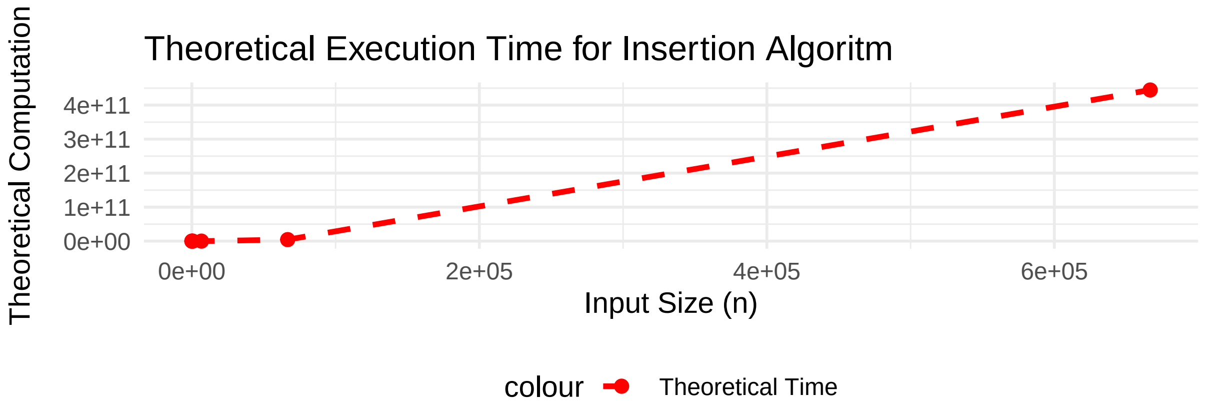
```r
Insertiondata <- csvPath("insertion_sort_times.csv", removeSpace = FALSE)

Insertiondata <- Insertiondata %>%
  mutate(
    theoretical_time = (Insertiondata$InputSize)^2  # O(n^2)
  )
```

```r
head(Insertiondata)
```

```
## # A tibble: 6 x 3
##   InputSize    TimeNano theoretical_time
##       <dbl>       <dbl>            <dbl>
## 1         6        2794               36
## 2        66       25356             4356
## 3       666     1516000           443556
## 4      6666    10616471         44435556
## 5     66666  1028100621       4444355556
## 6    666666 25159229255     444443555556
```

Plots:





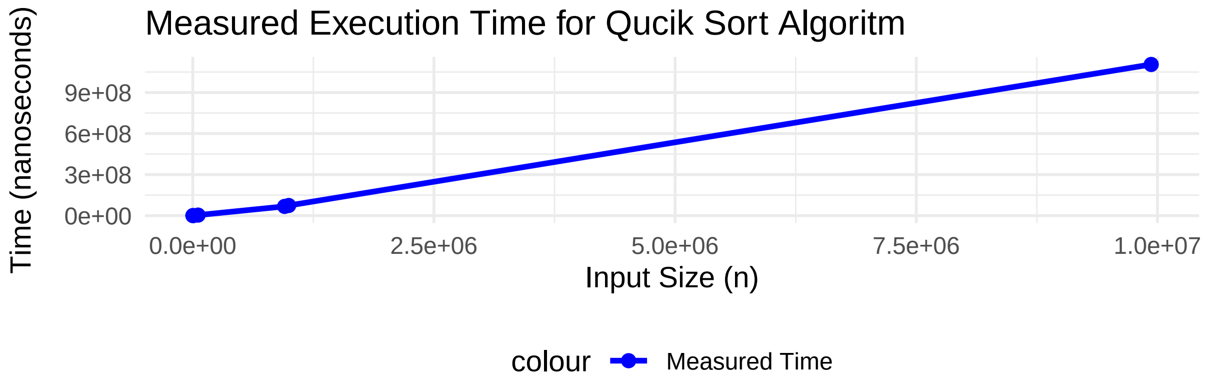
Quick sort

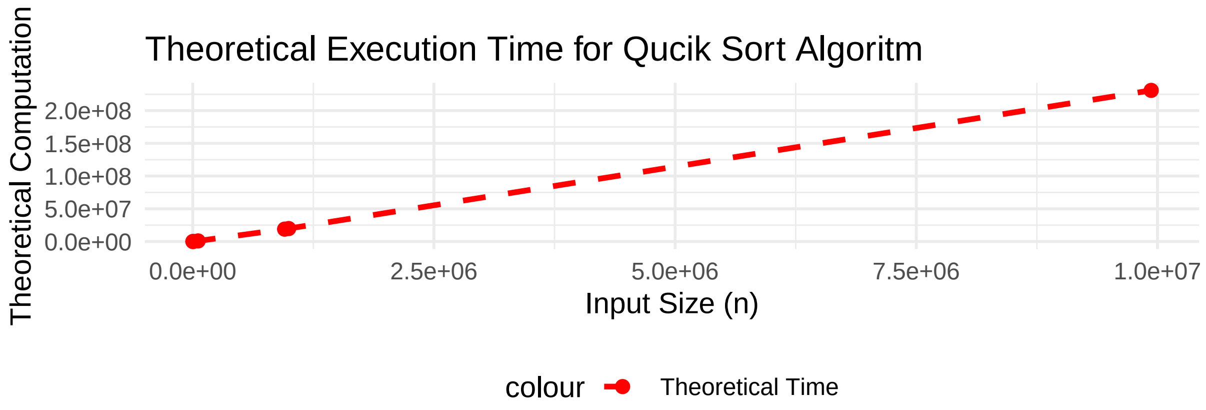
Data:

```r
quicksortData <- csvPath("quick_sort_times.csv", removeSpace = FALSE)
quicksortData <- quicksortData %>%
  mutate(
    theoretical_time = quicksortData$InputSize * log2(quicksortData$InputSize)  # O(n log n)
  )
head(quicksortData)
```

```
## # A tibble: 6 x 3
##   InputSize   TimeNano theoretical_time
##       <dbl>      <dbl>            <dbl>
## 1      1000      313379             9966.
## 2      5053      432667            62167.
## 3     55222     3970486           869910.
## 4    994200    74131031         19807622.
## 5    952220    67502519         18911980.
## 6   9935200 1105116529        230934957.
```

Plots:





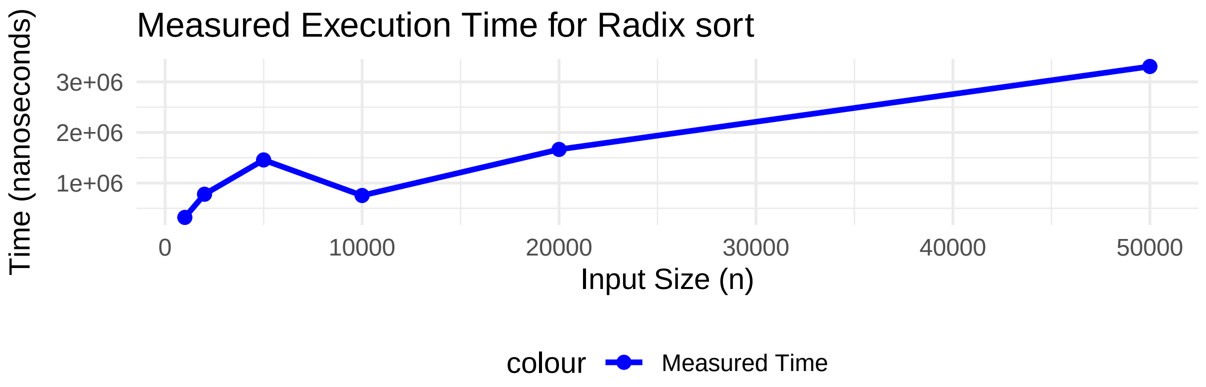
Radix Sort

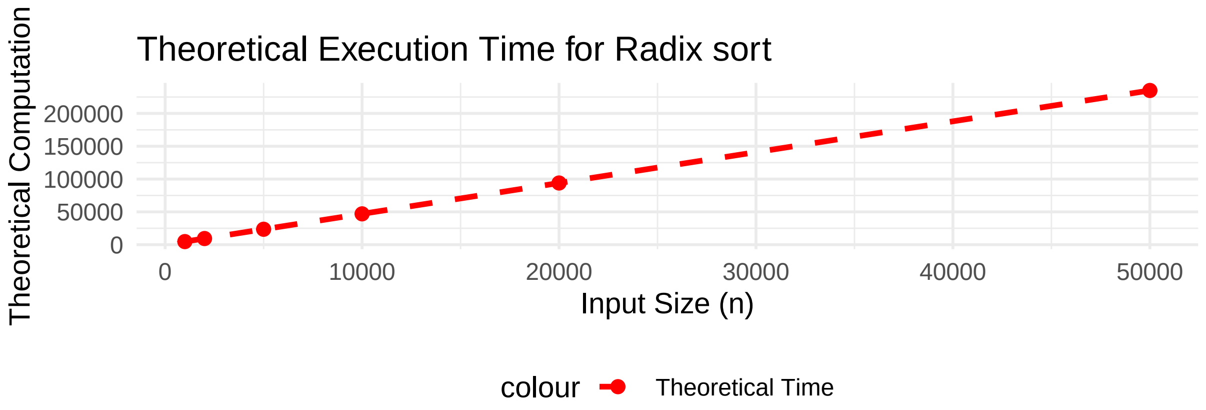
Data:

```
radfixdata <- csvPath("radix_sort_times.csv")
radfixdata <- radfixdata %>%
  mutate(
    theoretical_time = radfixdata$InputSize * log10(max(radfixdata$InputSize))  # O(nk), assuming k = l
  )
head(radfixdata)
```

```
## # A tibble: 6 x 3
##   InputSize TimeNano theoretical_time
##       <dbl>    <dbl>            <dbl>
## 1      1000   320078            4699.
## 2      2000   778368            9398.
## 3      5000  1456447           23495.
## 4     10000   753296           46990.
## 5     20000  1664222           93979.
## 6     50000  3304767          234949.
```

Plots:





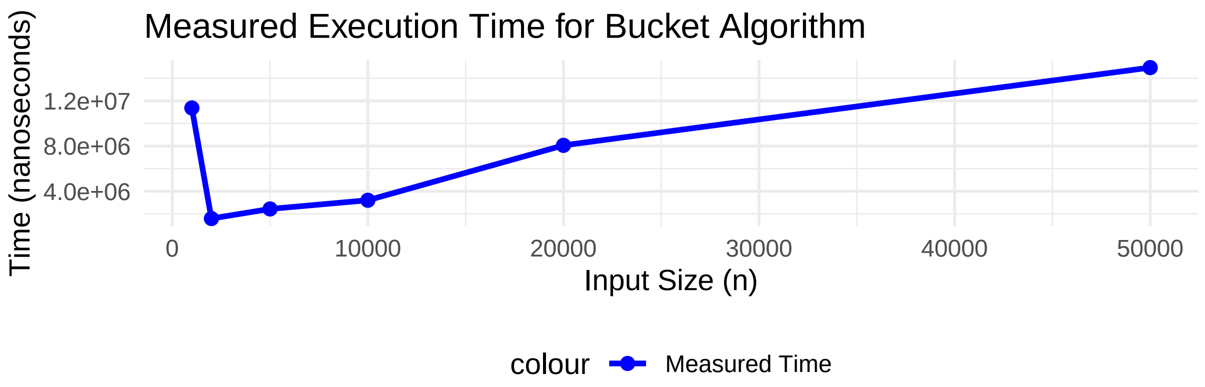
Bucket Sort

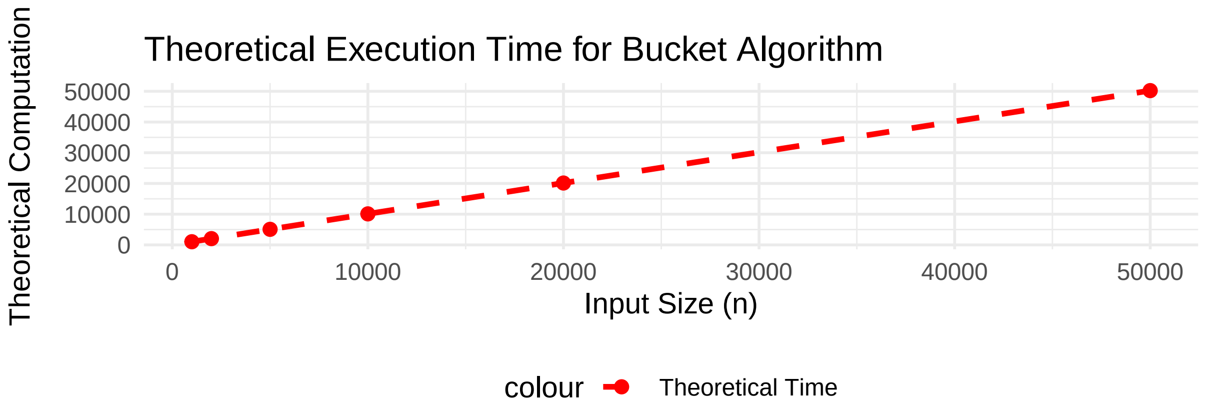
Data:

```
bucketdata <- csvPath("bucket_sort_times.csv")
bucketdata <- bucketdata %>%
  mutate(
    theoretical_time = bucketdata$InputSize + sqrt(bucketdata$InputSize)  # O(n + k), where k = sqrt(n)
  )
head(bucketdata)
```

```
## # A tibble: 6 x 3
##   InputSize TimeNano theoretical_time
##       <dbl>    <dbl>            <dbl>
## 1      1000 11366354            1032.
## 2      2000  1582510            2045.
## 3      5000  2433023            5071.
## 4     10000  3201824           10100
## 5     20000  8060259           20141.
## 6     50000 14938681           50224.
```

Plots:





#

Bogo Sort

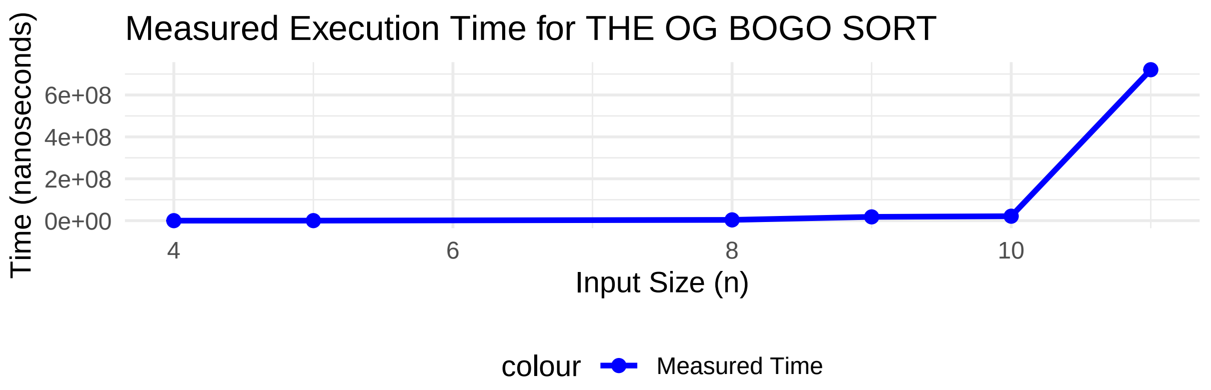
data

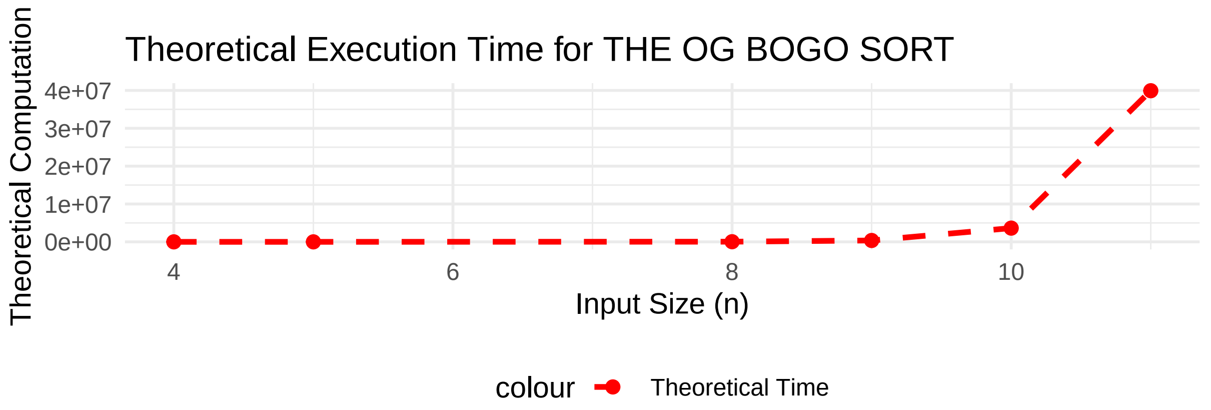
```
bogodata <- csvPath("bogo_sort_times.csv")
bogodata <- bogodata %>%
  mutate(theoretical_time = factorial(bogodata$InputSize))  # O(n!) complexity
head(bogodata)
```

```
## # A tibble: 6 x 3
##   InputSize  TimeNano theoretical_time
##       <dbl>     <dbl>            <dbl>
## 1         4      8381               24
## 2         5    119225              120
## 3         8   3735635            40320
## 4         9  18050829           362880
## 5        10  21252432          3628800
## 6        11 720511812         39916800
```

plot

```
algorithm_times(bogodata, name ="THE OG BOGO SORT")
```





References:

Programiz. (2025). Sorting Algorithm. Programiz: Learn to Code for Free. https://www.programiz.com/dsa/sorting-algorithm

Neto, A. (2023, May 5). Bogosort: The Stupid Sorting Algorithm. DEV Community. http://dev.to/adolfont/bogosort-the-stupid-sorting-algorithm-168f