SortTimes

kaymas

August 3, 2018

Complexity for different Sorting Algorithms.

Pre-requisite Functions

Insertion Sort

Sorting Algorithm

```r
insertionSort <- function(vec, comp=FALSE){
  steps <- 0
  n <- length(vec)
  for(i in 2:n){
    val <- vec[i]
    pos <- which.max(vec[1:i] > val) #returns index of first occurence of TRUE
    if(pos == 1){
      if(val < vec[1]){
        vec <- c(val, vec[-i])
      }
    }
    else{
      vec <- vec[-i]
      vec <- c(vec[1:(pos-1)], val, vec[pos:(n-1)])
    }
  }
  if(comp){
    return (list(vec = vec,steps = steps))
  }
  return (vec)
}
```

Proof of concept

```r
insertionSort(c(1,2,99,-21,2,23,1))
```

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

RunTime

```r
system.time(replicate(10, insertionSort(sample(x = 1:100, size = 10, replace = TRUE)))) / 10
```

```
##    user  system elapsed
##   3e-04   0e+00   2e-04
```

```r
system.time(replicate(10, insertionSort(sample(x = 1:100, size = 100, replace = TRUE)))) / 10
```

```
##    user  system elapsed
##   6e-04   0e+00   7e-04
```

```r
system.time(replicate(10, insertionSort(sample(x = 1:100, size = 1000, replace = TRUE)))) / 10
```

```
##    user  system elapsed
##   0.032   0.000   0.032
```

```r
system.time(replicate(10, insertionSort(sample(x = 1:100, size = 10000, replace = TRUE)))) / 10
```

```
##    user  system elapsed
##  3.2014  0.0036  3.2152
```