Psuedocode

```
PROMPT for filename
GET filename

file <- OPEN(filename)
if error
    PUT error message
    QUIT

data_text <- file.read()
data_JSON <- json.loads(data_text)
data <- data_JSON['array']

# A FOR i <- 0 ... len(data) - 1
# B    i_pivot <- i
# C    FOR j <- i + 1 ... len(data)
         IF data[j] < data[i_pivot]
# D            i_pivot <- j
       IF i_pivot != i
# E        data[i], data[i_pivot] <- data[i_pivot], data[i]

PUT data
```

Program Trace

	i	**i_pivot**	**j**	**data**
A	0	/	/	52, 26, 39, 15
B	0	0	/	52, 26, 39, 15
C	0	0	1	52, 26, 39, 15
D	0	1	1	52, 26, 39, 15
C	0	1	2	52, 26, 39, 15
C	0	1	3	52, 26, 39, 15
D	0	3	3	52, 26, 39, 15
E	0	3	/	15, 26, 39, 52
A	1	3	/	15, 26, 39, 52
B	1	1	/	15, 26, 39, 52
C	1	1	2	15, 26, 39, 52
C	1	1	3	15, 26, 39, 52
A	2	1	/	15, 26, 39, 52
B	2	2	/	15, 26, 39, 52
C	2	2	3	15, 26, 39, 52
A	3	2	/	15, 26, 39, 52
B	3	3	/	15, 26, 39, 52
C	3	3	4	15, 26, 39, 52

Algorithmic Efficiency

This algorithm is textbook O(n^2). To start, there is a FOR loop inside of a FOR loop. One of these loops are nested inside the other. This means that each iteration of the loop, we visit each element in the data set. Increasing the data set by 1 means going through the entire data set again. This can get exceptionally costly with large data sets.