**Code Reflection**

In this project, I implemented two sorting algorithms, selection sort and quicksort, to organize a collection of bids based on their titles. The selection sort has a time complexity of O(n^2), making it slower for large datasets, while quicksort, with its average time complexity of O(n log n), is faster, especially for larger datasets.

One challenge that I encountered was handling the partitioning step in quicksort efficiently, ensuring that the vector is correctly divided into two parts. To overcome this, I implemented the code for partition function that ensures accurate division and recursive sorting.

**Pseudocode for Selection Sort Function**

function selectionSort(bids):

size = length(bids)

for pos = 0 to size - 1:

min = pos

for j = pos + 1 to size:

if bids[j].title < bids[min].title:

min = j

swap(bids[pos], bids[min])

**Pseudocode for Quicksort Function:**

function quickSort(bids, begin, end):

if begin >= end:

return

mid = partition(bids, begin, end)

quickSort(bids, begin, mid)

quickSort(bids, mid + 1, end)

function partition(bids, begin, end):

pivot = bids[(begin + end) / 2].title

low = begin

high = end

while true:

while bids[low].title < pivot:

low++

while bids[high].title > pivot:

high--

if low >= high:

return high

swap(bids[low], bids[high])

low++

high--