Philip Trinh

SNHU/CS-300

3/11/2023

**Reflection and Pseudocode module 2 for Vector sorting**

In this assignment, I am working with information from a municipal government data feed containing bids submitted for auction of property. I was provided an incomplete VectorSorting.cpp program in which I need to make changes to Selection Sort, Partition, and Quick Sort. To tackle this project, I reviewed the ZYBook lessons on Vector Sorting, Partition, and Quick Sort. “The **selection sort** repeatedly selecting the smallest element (or largest) from the unsorted portion of the list and moving it to the sorted portion of the list. The algorithm repeatedly selects the smallest (or largest) element from the unsorted portion of the list and swaps it with the first element of the unsorted portion. This process is repeated for the remaining unsorted portion of the list until the entire list is sorted”. Meanwhile, “QuickSort is a Divide and Conquer algorithm. It picks an element as a pivot and partitions the given array around the picked pivot.” In this assignment, the pivot I pick is from the middle of the index. I enjoy following the instructor provided lecture and read the instruction comment provided from the VectorSorting.ccp so I did not run into any issue beside compile my code and typing error that I found after review my codes a few times prior to submission. I was able to download and run g++ compile with no problem but I run into problem with compile using the CMake program. Download was success but I was not able to run the /.bids with it. May be I will find a way to fix it in future if I need to use this program to compile my C++ project.

REFERENCES:

GeeksforGeeks. (2014, January 7). Access (2023, March 11). *QuickSort - GeeksforGeeks*. GeeksforGeeks; GeeksforGeeks. <https://www.geeksforgeeks.org/quick-sort/>

geeksforgeeks. (2014, January 31). Access (2023, March 11). *Selection Sort - GeeksforGeeks*. GeeksforGeeks. <https://www.geeksforgeeks.org/selection-sort/>

**Pseudocode**

Declare integer for partition (vector, begin, end)

Define variable for begin (low)

Define variable for end (high)

Set the middle element in the index as pivot point [low + (high – low) /2]

BOOL Done is FALSE

WHILE not done

Keep incrementing low index while bids[low] < bids[pivot]

Keep decrementing high index while bids[pivot] < bids[high]

If

there are zero or one elements remaining, Bool Done is TRUE

Return High

Else

Swap low and high bids and increment low while decrementing high

Declare integer for quicksort (vector, begin, end)

Set midpoint equal to zero

If

begin greater or equal to end

return

recursively sort low partition (begin to mid) and high partition (mid +1 to end)

Declare selection sort (vector)

Define integer minimum

Define integer maximum equal to bids size

Set position within the bids that divides sorted/unsorted

For position equal 0, position is less than bids size, increment position.

Set minimum equal to position.

(Move to remainder position on the right by)

For position + 1, position is less than bids size - 1, increment position

If this element is less than the minimum title

Set minimum equal to this position.

Repeat this and swap the current minimum with smaller one found.

Per the menu create case 3 and case 4

Case 3 for selection sort and case 4 for quick sort

For case 3 selection sort input is bids while case 4 quick sort input is (bids, 0, bids.size()-1)

Convert ticks to clock

Call method selection sort / quick sort accordingly

display sorted bids

Calculate elapsed time and display result (current clock ticks minus starting clock ticks)

Display time, ticks, and clock ticks

Display time, ticks times 1.0 divide by clock per second.