Hiep Ha

CS-300

Professor Nathan Lebel

Sept. 23, 2024

2-3 Assignment: Vector Sorting

This program is designed to implement two sorting algorithms, selection sort and quicksort, to efficiently sort bids by their titles. Users can load, display, and organize bid data with this functionality. The selection sort algorithm finds the minimum element from the unsorted part of the list and moves it to the sorted section.

On the other hand, quicksort uses a divide-and-conquer approach by partitioning the data around a pivot element. A challenge I faced was understanding how quicksort works, especially due to its recursive nature. I tackled this by breaking the process down into smaller parts, focusing on partitioning the data and sorting the partitions step by step.

**Task 1: Implement the Selection Sort Algorithm**

* **Function SelectionSort(bids: vector<Bid>)**
* **Define** minIndex as integer
* **Get** size of bids as size\_t
* **For** pos from 0 to size - 2
  + - Set minIndex = pos
    - **For** i from pos + 1 to size - 1
      * **If** bids[i].title < bids[minIndex].title
        + Set minIndex = i
    - **Swap** bids[pos] with bids[minIndex]

**Task 2: Implement the Quick Sort Algorithm**

* **Function QuickSort(bids: vector<Bid>, begin: int, end: int)**
  + **If** begin >= end, return
  + Set mid = Partition(bids, begin, end)
  + **Call** QuickSort(bids, begin, mid)
  + **Call** QuickSort(bids, mid + 1, end)
* **Function Partition(bids: vector<Bid>, begin: int, end: int)**
  + Set pivot = bids[(begin + end) / 2].title
  + Set low = begin
  + Set high = end
  + **While** low <= high
    - **While** bids[low].title < pivot, increment low
    - **While** bids[high].title > pivot, decrement high
    - **If** low <= high
      * **Swap** bids[low] with bids[high]
      * Increment low
      * Decrement high
  + **Return** high