Data Structures - Lab Worksheet

Simple Sorting

Start by downloading the provided coding canvas for Simple Sorting algorithms.

Question 1

Implement `Array.bubbleSort()`

Question 2

Implement `Array.selectionSort()`

Question 3

Implement `Array.insertionSort()`

Question 4

Modify your methods so that they display the total number of comparisons and the total number of swaps the sort required.

For every algorithm, repeat 6 times the process of generating a random array containing 100 elements and sorting it. Compute the average numbers of comparisons and swaps performed by each algorithm.

Q1: What conclusions do you draw from your results?

Result:

######### PERFORMANCE EXPERIMENTS #########

-- #comparisons --

BUB> 4902.5

SEL> 4950.0

INS> 2555.3333333333335

-- #swaps --

BUB> 2461.5

SEL> 94.0

INS> 2461.5

Conclusion:

When the data is large and embodies randomness, in average, Bubble Sort has O(n^2)+O(n^2)= O(n^2) time complexity, Selection Sort has O(n^2)+O(n)= O(n^2) time complexity, Insertion Sort has O(n^2)+O(n^2)= O(n^2) time complexity. In my code, I optimize each sorting a little bit to cut n^2 steps into approximately 1/2 n^2 steps in practice, but still, this does not influence the general time complexity. The case that Insertion Sort takes special advantage is when the array has already been partly sorted, where it can be well-reduced to approximately O(n)+O(n)=O(n) time complexity. Though in extreme case that the array is completely sorted, bubble sort can also reach O(n) by optimization, it does not perform well as Insertion Sort in general in the case of partly-sorted array.

Q2: In the worst case, for an array with N elements, how many comparisons does your code perform? How many swaps?

The worst case in comparisons is for Bubble Sort and Selection Sort, which takes approximately 1/2 N^2 steps. Interestingly, for Insertion Sort, it takes approximately 1/4 N^2 steps, where some mathematical proofs might be needed since it cannot be directly seen from my code.

The worst case in swaps is for Bubble Sort and Insertion Sort, interestingly, which takes approximately 1/4 N^2 steps, where some mathematical proofs might be needed since it cannot be directly seen from my code. For Selection Sort, it takes approximately N steps.

Q3: What kind of initial ordering produces the worst case?

Completely unsorted array, i.e., a non-increasing array.

Q4: What is the overall complexity of each algorithm?

Bubble Sort: O(n^2) [easy to write]

Selection Sort: O(n^2)

Insertion Sort: O(n^2) [well-performed in the case of partly-sorted array]