Task one arrays analysis

This code compares the bubble sort with selection sort and insertion sort by measuring both the execution time of the arrays of many different sizes and types and the number of comparisons to see the fastest algorithm and most effiecent one under various conditions

Types of algorithm in the code :

Bubble sort : its function is the bubble_sort_count(int arr[],int n) this goes through the array many times to compare each pair of elements and swaps them if theyre in wrongorder and counts every comparison and returns number

Selection sort : its function is selection_sort_count(int arr[],int n) this looks for and fins the smallest element in the unsorted part of array and switches it with unsorted element number one and it counts all of the comparisons and returns the total

Insertion sort : its function is insertion_sort_count(int arr[],int n) this takes each element and inputs it to its correct position in the arrays sorted part it finds the correct position while it counts the comparison

Testing :

3 types of arrays were made we have random,sorted and reverse sorted these were made using generate_random_array, generate_sorted_array and generate_reverse_sorted_array

Before we sort we copy the original array to another array using copy_array so that the algorithms work on the same data

The function test_comparison() runs each of the sorting algorithm 30 times and measures the number of conmaprisons and execution time for each run

Resilts are neatly printed in a table in the console and also saved as csv files with the names (comparison.csv and execution_times.csv so that they can be opened on excel

The results were that the bubble sort was the slowest one for reverse sorted arrays its better for sorted arrays and it does a few comparisons to the already sorted arrays while the selection sort does the same number of comparisons always which makes it predictable but it isn't faster for sorted arrays lastly the insertion sort was the faster for the already sorted arrays but slower for the reverse_sorted

The time complexeties of each sort are :

The bubble sort : best is O(n) while the worst id O(n^2) //(this is power two)

The selection sort :its always O(n^2)

The insertion sort: its worst one is O(n^2) while its best one is O(n)

Some alternative algorithms would be merge sort or quick sort for larger arrays as they have O(n log n) but for small arrays this is much easier