U10787

—

LAB 3 - Part B: Algorithms -Sorting and Searching

Author	**Dr. Alexios Louridas**
Last Update	**2023-02-21**
Number of pages	**2**

1 Task - Bubble Sort

Use your program in part A or if you want clone the following: GraphMaker.

Add a button next to your "Create Graph" button and implement a bubble sort on the array. Bubble sort is also called sinking sort because smaller values gradually "bubble" their way to the top of the array (i.e. towards the first element) like air bubbles rising in water, while the larger values sink to the bottom (end) of the array.

Using the lecture notes try to create an implementation of bubble sort. The technique will use nested loops to make several passes through the array. Each pass compares successive overlapping pairs of elements. If a pair is in increasing order (or the values are equal) the bubble sort leaves the values as they are. If a pair is in decreasing order, the bubble sort swaps their values in the array.

The first pass compares the first two elements of the array and swaps them if necessary. It then compares the second and third elements. The end of this pass compares the last two elements in the array and swaps them if necessary. after one pass, the largest element will be in the last position. After two passes, the largest two elements will be in the last two positions.

Every time in your implementation your bubble sort swaps elements update your picture box.

2 Task - Bubble Sort Enhancement

Make the following simple modification to improve the performance of the bubble sort you developed in the previous task:

1. After the first pass, the largest number is guaranteed to be in the highest-number element of the array; after the second pass, the two highest numbers are in place; and so on. Instead of making for example 5 comparisons for a 6 element array on every pass, modify your code to make one less on every pass.
2. The data in the array may already be in the proper order or near-proper order, so why make all passes? Modify your sort algorithm even further by checking at the end of each pass whether any swaps have been made. If none have been made, the data must already be in the proper order, so the app should terminate. If swaps have been made, at least one more pass is needed.

3 Task - Linear Search

A linear search searches each element of an array. Create a method to search an array to find a specific key. Add it to your GraphMaker program to search an integer.

4 Task - Recursive Linear Search

Modify your search to use a recursive method RecursiveLinearSearch to perform a linear search of the array. The method should receive the search key and starting index as arguments. If the search key is found, return its index in the array; otherwise, return –1. Each call to the recursive method should check one index in the array.

5 Task - Recursive Binary Search

Add a search option to perform a binary search of the array making sure you use a recursive method RecursiveBinary. The method should receive the search key, starting index and ending index as arguments. If the search key is found, return its index in the array. If the search key is not found, return –1.