Tutorial 04 & 05 Sort

1. The Bubble Sort algorithm can be optimized by checking if there are any swapping operation in the inner loop. If there is no swapping in any pass, it means that the list is already fully sorted, and we can stop the sorting process. In other words, we can optimize the number of passes required when the list gets sorted before the completion of all passes.

The following code is an incomplete implementation of this optimized Bubble Sort described above:

*# Sorts a sequence in ascending order using the # optimized bubble sort algorithm*

**def** bubbleSort\_optimized( theSeq ): n = len( theSeq )

*# Perform n-1 bubble operations on the sequence*

**for** i **in** range(n - 1, 0, -1):

***# Set boolean variable to check occurrence of swapping # in inner loop***

*# Bubble the largest item to the end*

**for** j **in** range(i):

**if** theSeq[j] > theSeq[j + 1]: *# Swap the j and j+1 items* tmp = theSeq[j]

theSeq[j] = theSeq[j + 1] theSeq[j + 1] = tmp

***# Set boolean variable value if swapping occurred***

***# Exit the loop if no swapping occurred # in the previous pass***

* 1. Complete the implementation by providing the rest of the required code for the optimized Bubble Sort.

***Graphical user interface, text, application

Description automatically generated***

***We set the Boolean variable to check if the argument passed in is sorted. If it is not sorted, then it breaks out of the loop.***

* 1. How many pass(es) is(are) required to sort the following list of numbers with the standard Bubble Sort and the optimized Bubble Sort?
     + [ 1, 2, 3, 5, 4 ]

***Standard Bubble Sort: 4 passes***

***Optimized Bubble Sort: 2 passes***

* 1. How many pass(es) is(are) required to sort a list of numbers that are already sorted with the optimized Bubble Sort?

***1 pass***

***First time it switches the placement of 5 and 4. Then the second time it passes to check if there is any other number to sort.***

***\*\* one advantage that bubble sort has over other sorting algorithms is having an efficient way to check if a list of numbers is already sorted in order.***

1. Given the following list of numbers:

* 12, 7, 9, 24, 7, 29, 5, 3, 11, 7

Using the table format shown below, trace each pass of the sorting process.

* 1. Use the Selection Sort algorithm to sort the above list in descending order.

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Pass** | **12** | **7** | **9** | **24** | **7** | **29** | **5** | **3** | **11** | **7** |
| **1** | ***29*** | ***7*** | ***9*** | ***24*** | ***7*** | ***12*** | ***5*** | ***3*** | ***11*** | ***7*** |
| **2** | ***29*** | ***24*** | ***9*** | ***7*** | ***7*** | ***12*** | ***5*** | ***3*** | ***11*** | ***7*** |
| **3** | ***29*** | ***24*** | ***12*** | ***7*** | ***7*** | ***9*** | ***5*** | ***3*** | ***11*** | ***7*** |
| **4** | ***29*** | ***24*** | ***12*** | ***11*** | ***7*** | ***9*** | ***5*** | ***3*** | ***7*** | ***7*** |
| **5** | ***29*** | ***24*** | ***12*** | ***11*** | ***9*** | ***7*** | ***5*** | ***3*** | ***7*** | ***7*** |
| **6** | ***29*** | ***24*** | ***12*** | ***11*** | ***9*** | ***7*** | ***7*** | ***3*** | ***5*** | ***7*** |
| **7** | ***29*** | ***24*** | ***12*** | ***11*** | ***9*** | ***7*** | ***7*** | ***7*** | ***5*** | ***3*** |

* 1. Use the Insertion Sort algorithm to sort the above list in ascending order.

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Pass** | **12** | **7** | **9** | **24** | **7** | **29** | **5** | **3** | **11** | **7** |
| **1** | ***7*** | ***12*** | ***9*** | ***24*** | ***7*** | ***29*** | ***5*** | ***3*** | ***11*** | ***7*** |
| **2** | ***7*** | ***9*** | ***12*** | ***24*** | ***7*** | ***29*** | ***5*** | ***3*** | ***11*** | ***7*** |
| **3** | ***7*** | ***9*** | ***12*** | ***24*** | ***7*** | ***29*** | ***5*** | ***3*** | ***11*** | ***7*** |
| **4** | ***7*** | ***7*** | ***9*** | ***12*** | ***24*** | ***29*** | ***5*** | ***3*** | ***11*** | ***7*** |
| **5** | ***7*** | ***7*** | ***9*** | ***12*** | ***24*** | ***29*** | ***5*** | ***3*** | ***11*** | ***7*** |
| **6** | ***5*** | ***7*** | ***7*** | ***9*** | ***12*** | ***24*** | ***29*** | ***3*** | ***11*** | ***7*** |
| **7** | ***3*** | ***5*** | ***7*** | ***7*** | ***9*** | ***12*** | ***24*** | ***29*** | ***11*** | ***7*** |
| **8** | ***3*** | ***5*** | ***7*** | ***7*** | ***9*** | ***11*** | ***12*** | ***24*** | ***29*** | ***7*** |
| **9** | ***3*** | ***5*** | ***7*** | ***7*** | ***7*** | ***9*** | ***11*** | ***12*** | ***24*** | ***29*** |

1. State whether each of the following statements are True or False:
   1. Using Selection Sort, after **N** passes through the list, the first **N** elements are in sorted order.

**True**

* 1. Using Selection Sort (ascending order), after **N** passes through the list, the first **N**

elements are the **N** smallest elements in the list.

**True**

* 1. Using Insertion Sort, after **N** passes through the list, the first **N** elements are in sorted order.

**True**

* 1. Using Insertion Sort (ascending order), after **N** passes through the list, the first **N**

elements are the **N** smallest elements in the list.

**False**

***-- End of Tutorial –***